Doctoral Thesis

Resource Allocation in Network Function Virtualization with Workload-Dependent Unavailability

Mengfei Zhu

Graduate School of Informatics, Kyoto University

May 2024

Preface

Network Function Virtualization (NFV) has transformed network services by shifting from dedicated hardware to virtualized functionalities, thus eliminating the constraints and expenses of traditional systems. Through NFV, computational resources can be pooled, offering enhanced scalability, flexibility, and cost-efficiency. Traditional network functions like firewalls and load balancers are virtualized for deployment on virtual machines and containers. Efficient resource allocation is vital for optimizing network performance, ensuring availability, and meeting agreements. Properly managing resources can enhance the overall efficiency and responsiveness of the service, and maintain uninterrupted functionality. Node failures and resource backups impact service reliability and continuity. Resource backups, such as cold and hot backups, enable recovery and minimize downtime. However, challenges arise from many factors like backup strategies, utilization ratio, failure probabilities, recovery strategies, and shared protection, necessitating innovative approaches to balance resource sharing, recovery time, and availability.

This doctoral thesis addresses a series of unique resource allocation challenges in the context of improving service quality. To tackle these issues, the research introduces optimization models and algorithms. Numerical experiments and analysis are conducted to evaluate the performance and effectiveness of the proposed methodologies. More specifically, by considering approximation of the workload-dependent unavailability, recovery priority, expected recovery time guarantees, fault-tolerance, load balance, uncertainty sets, and unsuccessful recovery under shared protection, this thesis offers insights into the trade-offs, and performance improvements achievable through the proposed methodologies. Moreover, this thesis not only focuses on theoretical advancements but also addresses the practical implementation of resource allocation

in containerized environment. By exploring and implementing real-world applications, this thesis aims to bridge the gap between theoretical concepts and their practical realization.

Firstly, this thesis proposes an optimization model to derive a primary and single backup resource allocation considering a workload-dependent failure probability to minimize the maximum expected unavailable time (MEUT). The workload-dependent failure probability is a non-decreasing function which reveals the relationship between the workload and the failure probability. The proposed model adopts hot backup and cold backup strategies to provide protection. The cold backup strategy is a protection strategy, in which the requested loads of backup resources are not activated before failures occur to reduce resource utilization with the cost of longer recovery time. The hot backup strategy is a protection strategy, in which the backup resources are activated and synchronized with the primary resources to recover promptly with the cost of higher workload. The optimization problem is formulated as a mixed integer linear programming (MILP) problem. This work proves that MEUT of the proposed model is equal to the smaller value between the two MEUTs obtained by applying only hot backup and cold backup strategies with the same total requested load. A heuristic algorithm inspired by the water-filling algorithm is developed with the proved theorem. The numerical results show that the proposed model suppresses MEUT compared with the conventional model which does not consider the workload-dependent failure probability. The developed heuristic algorithm is approximately 10^5 times faster than the MILP approach with 10^{-2} performance penalty on MEUT.

Secondly, this thesis proposes a multiple backup resource allocation model to minimize MEUT under a protection priority policy with a workload-dependent failure probability. The proposed model adopts hot backup and cold backup strategies to provide multiple protection. For protection of each function with multiple backup resources, it is required to adopt a suitable priority policy to determine the expected unavailable time. This work analyzes the superiority of the protection priority policy for multiple backup resources in the proposed model and provide the theorems that clarify the influence of policies on MEUT. The optimization problem is formulated as an MILP problem. This work provides a lower bound of the optimal objective value in the proposed model. This

work proves that the decision version of the multiple resource allocation problem in the proposed model is NP-complete. A heuristic algorithm inspired by the water-filling algorithm is developed with providing an upper bound of the expected unavailable time obtained by the algorithm. The numerical results show that the proposed model reduces MEUT compared to baselines. The priority policy adopted in the proposed model suppresses MEUT compared with other priority policies. The developed heuristic algorithm is approximately 10^6 times faster than the MILP approach with 10^{-4} performance penalty on MEUT.

Thirdly, this thesis proposes a primary and backup resource allocation model under preventive recovery priority setting to minimize a weighted value of unavailable probability (W-UP) against multiple failures with workload-dependent failure probability. W-UP considers the probability of unsuccessful recovery and the maximum unavailable probability after recovery among physical nodes. This work introduces a recovery strategy to handle the workload variation which is determined at the operation start time and can be applied for each failure pattern. Once a failure pattern occurs, the recoveries are operated according to the priority setting to promptly recover the functions hosted by failed nodes. This work also discusses an approach to obtain unsuccessful recovery probability with considering the maximum number of arbitrary recoverable functions by a set of available nodes without the priority setting. The optimization problem is formulated as an MILP problem. This work develops a heuristic algorithm to solve larger size problems in a practical time. The developed heuristic algorithm is approximately 729 times faster than the MILP approach with 1.6% performance penalty on W-UP. The numerical results observe that the proposed model reduces W-UP compared with baselines.

Fourthly, this thesis proposes a robust function deployment model against uncertain recovery time with satisfying an expected recovery time guarantee in a cost-efficient manner. The preventively deployed backup resources can recover an unavailable function hosted by a failed node in a period of time, which is related to the backup strategies and protection types. Multiple functions protected by a node can share the backup resources to reduce the number of active nodes to save cost, which also affects the recovery time if the number of unavailable functions is so large that the remaining capacity cannot recover

them and causes a waiting procedure of an unavailable function before being recovered. This work introduces an uncertainty set that considers the upper and lower bounds of the recovery time of a function protected by each node and the upper bound of the average recovery time among nodes. This work considers the expected remaining capacity of a node and the expected number of unavailable functions hosted by the node to approximate the waiting time in the shared protection. The robust optimization technique is applied to obtain the worst-case expected recovery time under an uncertain recovery time set. With this technique, the model is formulated as an MILP problem. To solve the problem in a practical time, a heuristic algorithm is developed. It reduces the number of active nodes while decreasing the worst-case expected recovery time within the uncertainty set by converting the linear programming problem to a graph problem. The numerical results reveal the superiority of the proposed model by considering the recovery time guarantee, uncertainty set, and shared protection.

Fifthly, this thesis proposes a primary and backup resource allocation model under reliability guarantees to minimize the deployment cost with considering the effect of the assigned workload on recovery time. Backup resources are allocated for recovery of an unavailable function in a period, where the time relates to different backup modes and assigned recovery workload. This work considers multiple states of pre-configuration for each function with different degrees of instantiation, initialization, and synchronization and different recovery times. This work considers that the extra-assigned recovery workload can be adopted, which means that the recovery workload can be scaled, to speed up the recovery, while improving the resource efficiency to fully utilize the idle capacity for faster recovery. On the other aspect, the extra-assigned recovery workload may lead to unsuccessful recovery in a specific failure configuration. This work considers two reliability indicators, which are recovery time and the total unsuccessful recovery probability; each reliability indicator is restricted under guarantee while minimizing the number of activated nodes as deployment cost. The numerical results indicate that the deployment cost is saved on average 19% and 9% with considering the proposed model compared to two baselines that do not consider flexible backup modes and extra-assigned recovery workload, respectively.

Sixthly, this thesis designs and implements a custom resource and the corresponding controller in Kubernetes to manage the primary and backup resources. The custom resource is a set of Pods with different types, which includes primary, hot backup, and cold backup Pods. The controller manages the set of Pods and maintains the current state of the different types of Pods to keep the current state consistent with the desired state of each type of Pod. Demonstration validates that the controller automatically manages the primary and backups resources correctly. Moreover, Prompt function deployment and management is a key role to improve the continuity and reliability of network services. Kubernetes is a system to deploy and manage functions automatically. Existing tools in Kubernetes do not provide automatic function deployment and management in a real-time and optimal manner. It does not provide a resource type to manage the migratable resource, either. This thesis designs and implements a two-layer controller structure in Kubernetes to achieve the function deployment in a limited computation time with considering resource migration for allocation optimality. A controller in the lower layer manages the Pods for an intermediate allocation with a model or a heuristic algorithm to respond to requests promptly. A controller in the upper layer manages instances by optimizing resource allocations with considering resource migration; it maintains the Pods by keeping the current state (intermediate allocation) consistent with the desired state (optimal allocation). The demonstration validates that the controller automatically manages the resources promptly and correctly.

This thesis is organized as follows. Chapter 1 introduces the background of resource allocation for higher availability and fault tolerance with workload-dependent unavailability. Chapter 2 investigates the related works in literature. Chapter 3 presents the single backup resource allocation model with workload-dependent failure probability. Chapter 4 presents the multiple backups resource allocation with workload-dependent failure probability and introduces the priority policy. Chapter 5 presents a resource allocation model and recovery priority setting against multiple failures for load balancing and fault tolerance with workload-dependent failure probability. Chapter 6 presents the resource allocation with considering the uncertainty of the recovery time. Chapter 7 presents the resource allocation with considering the effect of the as-

signed workload on the recovery time. Chapter 8 presents two implementations for backup resource allocation and fast implementation. Finally, Chapter 9 concludes this thesis.

Acknowledgments

This thesis is the summary of my study in master's and doctoral courses at the Graduate School of Informatics, Kyoto University, Japan. I would like to extend my gratitude to a number of institutions and organizations and people for their help on this thesis and my future career.

First of all, I am grateful to the Japan Society for the Promotion of Science for their financial support, as well as the Japanese government, Aeon 1%club, and Kyoto University for their scholarships and tuition exemption. This financial assistance has alleviated the burden of living expenses and allowed me to focus on my research without financial constraints. It has been instrumental in completing my thesis and pursuing a career in research.

I would like to thank my supervisor Professor Eiji Oki. He led me into a new world of research and used his experience to help me. Without his help, there would be no basis for this thesis. I would like to thank Professor Hiroshi Harada and Professor Nobuo Yamashita for being part of my judging committee and providing their precious comments to improve my thesis. I would also like to express my best appreciation to my co-authors, Dr. Rui Kang and Dr. Fujun He, whose contributions have been instrumental in completing this comprehensive study. Your collaboration and shared efforts have eased the challenges and difficulties along the way; also thanks for sharing the stresses with me. Additionally, I would like to acknowledge the significant assistance of Associate Professor Takehiro Sato and our secretary, Ms. Mariko Tatsumi, for their invaluable support in navigating the laboratory environment.

I extend my sincere appreciation to Prof. Klaus-Tycho Foerster from TU Dortmund for his exceptional academic accomplishments and his remarkable kindness. He plays a pivotal role in expanding my academic horizons when I was a visiting researcher at TU Dortmund. Furthermore, I would like to

express my gratitude to every individual I had the opportunity to meet during the conferences. Their insightful discussions, words of encouragement, and understanding have been truly valuable.

I would like to express my special heartfelt gratitude to the dean of the department, Prof. Kawahara, and a staff from the office of academic affairs, Ms. Yamagawa, for their invaluable support and assistance. Their unwavering support during a challenging period of being academic plagiarism not only restored my faith in humanity but also played a pivotal role in rescuing me from the depths of depression. I will forever cherish the words of encouragement given to me by Prof. Kawahara, who reminded me that "your academic and your life matter the most." Similarly, the comforting words from Ms. Yamagawa, who assured me that "your efforts are recognized and cannot be undermined by anyone," provided me with the strength to persevere. Without their encouragement, I would not have been able to overcome the obstacles that I faced. I will keep your words in my heart for a lifetime.

Finally, I must express my very profound gratitude to my family members, Ning Zhu, Yanmei Guo, and Maomao, You are always there for me. I love you so much and thank you for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

Contents

List of Figures

List of Tables

Notations

Notation	Description
K	Set of PMs.
V	Set of VMs.
N	Set of nodes, e.g., servers.
F	Set of functions.
T	Recovery time guarantee.
$\mathcal{P}_\Gamma$	Set containing all possible failure patterns, each of which has at most Γ failed nodes.
$\mathbb{C}$	Set; it is $[0, \lvert\mathcal{P}_\Gamma\rvert]$.
$\mathcal{P}_\Gamma^\sigma$	σth element in $\mathcal{P}_\Gamma$, $\sigma \in \mathbb{C}$; when $\sigma = 0$, $\mathcal{P}_\Gamma^\sigma = \emptyset$.
$\mathcal{S}$	Number of steps to approximate workload-dependent failure probability.
P_s	sth value of $\mathcal{S}$-step workload-dependent failure probability, $s \in [1, \mathcal{S}]$.
T_i^s	Given value of workload of node i, where $s \in [1, \mathcal{S}]$. When workload of node i increases from range of $(T_i^{s-1}, T_i^s]$ to $(T_i^s, T_i^{s+1}]$, failure probability increases from P_s to P_{s+1}.
C_i	upper bound of computing capacity which node i can provide for primary and backup allocation.
t_0	Recovery time for CB strategy.
t_1	Recovery time for HB strategy.
t_3	Recovery time of the situation when both primary and all corresponding backup resources are unavailable.
t_j^0	Given recovery time of function j for CB strategy.
t_j^1	Given recovery time of function j for HB strategy.

Notation	Description
t_j^{m}	Given recovery time of function j in the situation that all of the primary and backup resources of the function are unavailable. This time relies on the maintenance systems.
P_{L}	Lower failure probability of node i when its workload satisfies $W_i \leq T_i$ when $S = 2$.
P_{H}	Higher failure probability of node i when its workload satisfies $T_i < W_i \leq C_i$ when $S = 2$.
x_{ij}^{kb}	x_{ij}^{kb} is set to one if VM $j \in V$ is allocated at PM $i \in K$ and is protected by PM $k \in K \backslash \{i\}$ with strategy $b \in B$, and zero otherwise.
l_j^{R}	Requested load of VM j.
W_i	Workload of node i.
R_i	Requested load of node i, which contains primary resources and both HB and CB backup resources.
q_i	Workload-dependent failure probability of node i.
r_j	Expected unavailable time for VM $j \in V$.
r	Maximum expected unavailable time among $r_j, j \in V$.
S_i	Set of backup servers assigned to function $i \in F$.
w_i	Requested load of function $i \in F$.
p_i	Failure probability of function $i \in F$.
T_j	Threshold that the network maintains its normal and efficient functioning.
C_j	Upper bound of computing capacity which server j can provide for resource allocation.
x_{ij}^{b}	x_{ij}^{ks} is set to one if function $i \in F$ is protected by server $j \in N$ with strategy $b \in B$, and zero otherwise.
L_j	Workload of server $j \in N$, which contains the hot backup resources.
O_{ij}	Determined priority score of each j assigned to function $i \in F$.
q_j	Failure probability of server $j \in N$ according to the workload.
u_i	Expected unavailable time for function $i \in F$.
u	Maximum expected unavailable time among $u_i, i \in F$.

Notation	Description
l_j^{u}	Upper bound of computing capacity for function j required for information synchronization and snapshot updating.
$\delta, 1-\delta$	Given weights for $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively, where $0 \leq \delta \leq 1$.
x_{ij}^{kp}	x_{ij}^{kp} is set to one if function j is allocated at node i and is protected by node k whose priority among nodes in $N\backslash\{i\}$ is equal to p, and zero otherwise.
χ_{ij}	Binary variable that equals $\vee_{p\in[1,\lvert N\rvert]} \vee_{k\in N\backslash\{i\}} x_{ij}^{kp}$; it is set to one if function j is allocated to node i, and zero otherwise.
$L_{i\sigma}^{\mathrm{W}}$	Workload of node $i \in N$ under failure pattern $\mathcal{P}_\Gamma^\sigma$. When $\sigma = 0$, L_{i0}^{W} represents the workload of node $i \in N$ before any failure occurs.
$q_{i\sigma}$	Workload-dependent failure probability of node i according to the workload in failure pattern σ. When $\sigma = 0$, q_{i0} represents the failure probability of node i before any failure occurs.
w_σ	Probability of failure pattern σ; it is a weight of the maximum unavailable probability for failure pattern σ.
e_j^σ	Binary variable; it is set to one if the number of nodes in $\mathcal{P}_\Gamma^\sigma$ hosting the primary and backup resources of function j is equal to the number of nodes in N hosting the primary and backup resources of j, and zero otherwise.
g_i^σ	Binary variable; it is set to one if the workload $L_{i\sigma}^{\mathrm{W}}$ of node i exceeds the capacity C_i under failure pattern $\mathcal{P}_\Gamma^\sigma$, and zero otherwise.
Q_σ	Maximum unavailable probability among nodes after failure recovery against failure pattern $\mathcal{P}_\Gamma^\sigma$.
$\mathcal{U}_1$	Total probability of unsuccessful recovery.
$\mathcal{U}_2$	Weighted maximum unavailable probability among nodes after failure recovery.
M_j	Set of backup modes of function $j \in F$.
w_j^{P}	Workload of each running primary function $j \in F$.
w_{jm}^{B}	Workload of pre-configuration of function j under mode $m \in M_j$.

Notation	Description
t^{B}_{jm}	Recovery time of function $j \in F$ protected with backup mode $m \in M_j$.
x^{km}_{ij}	Binary; it is 1 if function $j \in F$ is hosted by node $i \in N$ and protected by node $k \in N \backslash \{i\}$ with backup mode $m \in M_j$, and otherwise 0.
χ_{ij}	Binary; it is 1 if primary resource of function $j \in F$ is allocated to node $i \in N$, and otherwise 0.
ζ^{m}_{jk}	Binary; it is 1 if the backup resources of function j is allocated to node $k \in N$ with backup mode $m \in M_j$, and otherwise 0.
w^{R}_{jk}	Workload in node $k \in N$ that is permitted to be used by function $j \in F$ for recovery.
r^{R}_{jk}	Rate (Speed) of node $k \in N$ for recovery function $j \in F$.
$\mathbb{N}^{b}_{j}$	Subset of N_j; set of nodes that protect function j with strategy b.
l^{w}_{j}	Requested workload of function j.
l^{u}_{j}	Requested load for function j for CB resource for information synchronization and snapshot updating.
O_{jk}	Integer variable to indicate priority of each $k \in N_j$ that protects function $j \in F$.
ξ^{b}_{jk}	Binary variable that equals $\vee_{i \in N \backslash \{k\}} x^{kb}_{ij}$; it is set to one if function j is protected by node k with strategy b, and zero otherwise.
t_j	Expected recovery time of function j.
τ^{0}_{jk}	Recovery time of function j when it is protected by node $k \in \mathbb{N}^{0}_{j}$ with priority O_{jk}.
$\mathcal{T}_j$	Uncertainty set that describes recovery time of function j.
Λ_i	Number of functions whose primary resource is hosted by node i.
$\mathcal{N}_{jk}$	Expected number of unavailable functions when function j is recovered by available node k.
C_{jk}	Expected remaining capacity when function j is recovered by available node k.

Notation	Description
ρ_{jk}	Binary variable; it is set to one if unavailable function j is recovered by node k with priority O_{jk} without any waiting time, and zero otherwise.
ω_{jk}	A coefficient that equals $(\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'}) \mathcal{N}_{jk} - \mathcal{C}_{jk})$; it shows the relationship between the remaining capacity and the number of unavailable functions when function j is recovered by node k.
U_{jk}	Upper bound of τ^0_{jk}; it is equal to $(1-\rho_{jk})\tau^{\mathrm{u}}_j + \rho_{jk}\tau_{\mathrm{w}}\omega_{jk}$.
L_{jk}	Lower bounds of τ^0_{jk}; it is equal to $(1-\rho_{jk})\tau^{\mathrm{l}}_j$.
τ^{E}_j	Maximum average recovery time of function j.

Abbreviations

Abbreviation	Description
API	application programming interface
BFS	brute-force search
BPS	backup Pod set
CB	cold backup
CPU	central processing unit
CRD	custom resource definitions
DPI	deep packet inspection
DPS	determined priority scores
DRAM	Dynamic random-access memory
FIFO	first-in, first-out
FW	firewall
GLB	generalized load balancing
GO	global optimizer
HB	hot backup
HTTP	hypertext transfer protocol
IP	Internet protocol
LB	load balancer
MBRA	multiple backup resource allocation
MEUT	maximum expected unavailable time
MEUT-TW	a model that minimizes a weighted value considering MEUT and the total workload
MILP	mixed integer linear programming
MKP	multiple knapsack problem
MPS	migratable Pod set

Abbreviation	Description
NAT	network address translation
NFV	network function virtualization
PM	physical machine
RAM-P	a primary and backup resource allocation model with preventive recovery priority setting
SA	simulated annealing
SFC	service function chain
TW-GU	a model that minimizes the total workload with guaranteeing the unavailable time
TW-SO	a model that minimizes the objective with adopting the total workload as the secondary objective
VM	virtual machine
VNF	virtual network function
W-UP	weighted value of unavailable probability
WAGA	workload-aware greedy algorithm
WB	warm backup
WOGA	workload-ordered greedy algorithm

Chapter 1

Introduction

1.1 Backgrounds of network function management and orchestration (NFMO)

Network Function Virtualization (NFV) has revolutionized the networking field by enabling the virtualization of network services and functions. Traditionally, functions were tightly coupled with dedicated hardware, resulting in inflexibility for scaling on demand and high costs for updates. Traditional functions also led to idle resources on each dedicated hardware, increasing deployment and maintenance expenses. With NFV technology, computational resources provided by hardware can be shared among different functions, offering increased flexibility, scalability, and cost efficiency. This transformation involves virtualizing traditional network functions such as firewalls, load balancers, and network address translators, which can be softwarized and deployed on virtual machines and containers. The adoption of NFV brings numerous benefits to network management and resource utilization, empowering organizations to meet the evolving demands of modern networking environments.

Network function management and orchestration is a pivotal consideration for efficiently managing, coordinating, and operating diverse virtualized functionalities, enabling smooth service delivery and resource use. It plays a vital role in cloud computing and cloud-native settings, aiding service providers in achieving agile, scalable, and automated networks. In cloud computing, NFMO lets users swiftly create, configure, and manage network functions,

like firewalls and load balancers which simplifies network admin, enhancing resource efficiency and flexibility. In cloud-native development, NFMO seamlessly coordinates and optimizes network resources, ensuring high availability and scalability. By integrating functionalities into service development and deployment, it achieves continuous delivery and rapid deployment.

Efficient resource allocation is a crucial aspect for network function management and orchestration that plays a vital role in optimizing service performance, ensuring high availability, and meeting service-level agreements. Node failures and resource backups are crucial considerations in resource allocation due to their significant impact on service reliability and continuity. Node failures can disrupt the operation of virtualized functionalities, leading to service interruptions and degraded performance. Resource backups, such as cold and hot backups, provide mechanisms to recover from node failures and minimize service downtime. By considering these aspects, service providers can optimize resource utilization, enhance service availability, and maintain uninterrupted service delivery, making node failures and resource backups essential factors in resource allocation.

The recovery time of a failed virtualized network function (VNF) hosted by a node is influenced by backup strategies and protection types, while the failure probability of a node is dependent on its workload. These dynamic factors pose challenges in developing robust resource allocation models that can ensure an expected recovery time and costs. Moreover, the introduction of shared protection, where multiple functions are protected by a node with oversubscribed resources, presents another technical hurdle. The main issue arises from insufficient resources to successfully recover all protected functions, resulting in delays and potential unsuccessful recovery. Effectively managing shared backup for recovery against failures becomes challenging due to the intricate balance required between resource sharing, recovery time, and availability, making it crucial to devise innovative approaches to address these complexities.

1.1.1 Workload-dependent unavailable probability

The stability and reliability of large-scale cluster computing systems hold paramount importance, ensuring uninterrupted service and optimal performance. These systems operate within intricate and dynamic computing landscapes, where the inevitability of system failures and malfunctions can lead to service disruptions and performance deterioration. To address this challenge, harnessing historical fault data, existing system information, and reliability models is essential. Employing techniques like statistical analysis and machine learning allows for the exploration of the intricate relationship between workload patterns and system failures. This exploration, in turn, paves the way for effective system failure prediction and potential maintenance strategies.

In the context of datacenters, applications generate diverse workloads encompassing computing resources such as central processing units (CPUs) and memory [1]. Facebook's data centers exemplify this diversity, hosting an array of applications, including web serving, caching, database management, video and image processing, and messaging routing. Each application category necessitates varying quantities of processor cores and memory capacity [2]. Termed as workload, these resource demands can be equated to the likes of CPU and memory. Workload characterization entails two principal methodologies: trace-based and model-based approaches [3][4][5]. These methodologies contribute to understanding workload behavior, enabling efficient allocation and management of computing resources for optimal system operation.

Higher workload levels are associated with increased performance degradation and node failure probability, highlighting the significant connection between workload intensity and failure likelihood. The works in [6,7] studied that a higher level of workload often leads to more performance degradation. For instance, augmented CPU utilization can result in an unstable packet processing performance, leading to an increased likelihood of packet loss due to the potential inability to process arriving packets within the requisite timeframe, subsequently resulting in packet drops. In a physical node, surpassing a specific CPU utilization threshold [8] may instigate performance degradation, each node could be allocated an assigned threshold indicative of the initiation of performance degradation. Rather than confining workload assessment to mere

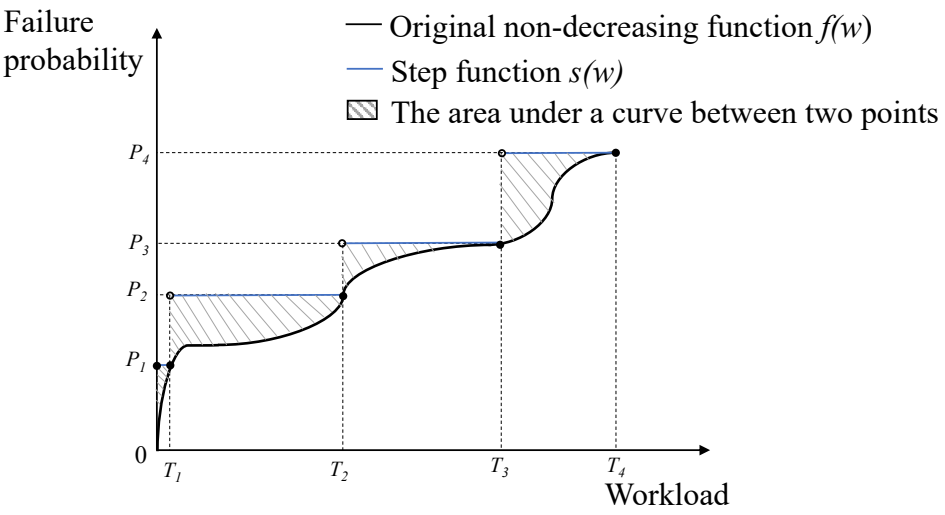


Figure 1.1: Approximation of the workload-dependent unavailable probability.

characterization (e.g., utilization ratio and types), it is imperative to contemplate the connection between workload and the node's failure probability [2]. Numerous studies have scrutinized the influence of workload attributes, including CPU utilization, memory usage, and application type, on node failure rates. For example, Sahoo et al. [9] established a correlation between workload type and failure probability, while Iyer et al. [10] disclosed a direct relationship between workload intensity and the likelihood of failure. The study in [11] introduced the observation that the failure probabilities of dynamic random-access memory (DRAM) modules in servers experiencing intensified utilization, as measured by CPU utilization and allocated memory, tend to be, on average, 4-10 times higher compared to servers operating at lower utilization levels. The workload-related unavailable characteristics of each server can emanate from various factors, such as machine properties and aging dynamics. Every workload-dependent probability of failure unveils an empirical correlation between workload and failure probability; workload characterization can be performed by collecting and analyzing the workload-related failure characteristics of each server. Therefore, the unavailable probability of any node can be assumed to be a non-decreasing function of the node workload.

Given the workload-dependent unavailable probability, which is assumed

to be a non-decreasing function, an S-step function can be used, as shown in Fig. 1.1.1, to conservatively approximate it. Adopting step functions is a common approach for approximating a given curve. Let $f(w)$ be a given non-decreasing unavailable probability, which shows the practical relationship between the unavailable probability and the workload; let $s(w)$ denote an S-step function; w is the pysical machine (PM) workload. Given the step number S with considering the accuracy of fitting the non-decreasing function $f(w)$ in an S-step function and the computation time to solve the proposed model, the unavailable probability can be designed with the threshold of $T_s, s \in \mathcal{S}$, and corresponding unavailable probability P_s. To fit the given curve as accurately as possible, the conservative approximation can have different goals while keeping $s(w) \geq f(w)$, such as minimizing $\int_0^{C_i} (s(w) - f(w))\,\mathrm{d}w$ or $\max_{s\in\mathcal{S}} \left\{ \int_{T_i^{s-1}}^{T_i^s} (s(w) - f(w))\,\mathrm{d}w \right\}$.

1.1.2 Backup strategies in terms of reliability and recovery time

Unavailability affects the service quality and user experience [12]. A failure of network function stops users' services unexpectedly when it occurs; prevention and protection strategies are required. Service providers can use backup resources to improve the availability of functions confronting failures and to avoid the interruption of services [13]. Prevention and protection strategies are common approaches to improve the availability of functions confronting failures, which can be generally classified into dedicated protection and shared protection [14]. Dedicated protection has a one-to-one correspondence between the working and each backup function to ensure higher recovery efficiency. Backup resources are shared by multiple functions in shared protection, which makes the resource efficiency higher than dedicated protection at the cost of a higher risk of unsuccessful recovery and longer recovery time due to the concurrent failures of functions.

One of the key challenges in backup resource allocation is striking the balance between resource availability and cost-efficiency. Allocating excessive backup resources can lead to resource wastage and unnecessary expenses, while insufficient allocation may compromise system reliability and recovery

capabilities. Therefore, it is crucial to develop robust models and algorithms that can optimize the allocation of backup resources based on factors such as system dependencies, failure probabilities, recovery time objectives, and cost considerations.

In dedicated protection, there are two backup strategies to provide protection to network systems: cold backup (CB) and hot backup (HB). The recovery is performed on the trigger basis of failure detection. The CB strategy provides protection to corresponding functions, where the backup resources are only reserved without being active. When a function monitoring function that checks the status of every function periodically detects the failure, the latest snapshot of each function is used to re-establish the failed function in the corresponding backup resource [15]. The HB strategy takes place while the system is still processing before a failure occurs. During the implementation of a service, the backup resource in the corresponding node is always activated. The first returned results of interrupted services caused by failed nodes will be adopted by the backup resource to ensure the successful implementation of service composition [16]. The workload may exceed the threshold caused by higher resource utilization; the CB strategy can be adopted to reduce the workload with the cost of longer recovery time. Since a failed function is replaced with the backup resource immediately, HB reduces the recovery time compared with CB. However, a backup resource is activated and synchronized with the primary resource at the same time, which may cause to increase the failure probabilities of the nodes due to the increased workloads.

1.1.3 Resource sharing

Resource sharing enables multiple functions to share a pool of computing resources, such as storage, processing power, and memory, through virtualization technologies. This allows for efficient resource allocation and scalability, catering to dynamic demands. Shared resources, such as network links or routers, can be dynamically allocated based on traffic patterns and quality of service requirements, ensuring optimal network performance. However, resource sharing can lead to resource contention and competition, which can impact system performance and response time. When multiple tasks or services share the

same resources, resource bottlenecks, and congestion may occur, resulting in performance degradation.

To increase availability, multiple functions protected by a node can share the backup resources to reduce the number of active nodes to save cost, which also affects the recovery time if the number of unavailable functions is so large that the remaining capacity cannot recover them. In shared protection, backup resources are shared by several primary functions. Different from dedicated protection, where each backup resource is specifically allocated and reserved for a primary function. If a primary function fails, it can be recovered by any one of the backup resources as long as it is available, with considering the resource sharing for higher efficiency, the capacity for recovery of each function is not reserved in advance, which indicates that an unavailable function may not be recovered if there are not sufficient computing resources. Resource sharing can increase resource efficiency at the cost of unsuccessful recovery and a longer recovery time. When an unavailable function cannot be recovered by the remaining capacity among available nodes, a waiting procedure is required to be considered. For example, the work in [17] introduced an unavailability-aware backup allocation model with considering the waiting time and repairing time of the failed function in the shared protection.

The advantages and disadvantages of resource sharing manifest in resource utilization and failure recovery. For instance, idle resources can be scaled up to improve processing speed, resulting in higher resource utilization. However, when failures occur, insufficient overall capacity may hinder the recovery process or prolong the recovery time. In cases where a function is protected by multiple nodes, careful consideration must be given to selecting the node that can recover the failed function with minimal resource contention. It is important to strike a balance between resource optimization and failure recovery to achieve efficient and resilient system operation. While resource sharing enhances utilization and scalability, resource allocation and recovery activation during failure scenarios require careful planning to minimize the impact on overall system performance.

Striking a balance between optimizing resources and ensuring effective recovery is crucial for achieving efficient and robust service delivery. While resource sharing improves resource usage and scalability, careful allocation and

coordinated recovery planning in case of failures are essential to minimize any impacts on overall service performance.

1.1.4 Approaches for solving considered problems

In resource allocation models, both mixed integer linear programming (MILP) and heuristic algorithms are commonly used methods. MILP is an optimization technique that formulates resource allocation problems as linear programming problems with integer variables, aiming to maximize or minimize specific objective functions. For example, minimizing unavailability time, minimizing deployment cost, maximizing resource utilization, etc. MILP methods can provide globally optimal solutions but may face challenges of computational and time complexity when dealing with large-scale problems. It ensures precise constraint satisfaction, accommodating resource limitations and quality of service requirements.

On the other aspect, heuristic algorithms offer efficient solutions, particularly in large-scale resource allocation problems where MILP may face computational complexities. Heuristic algorithms are approximate solution methods based on experience and rules. By leveraging problem-specific heuristic information and rules, heuristic algorithms iteratively search the solution space to find suboptimal solutions that satisfy the constraints. Heuristic algorithms are often applied in resource allocation models for large-scale problems or real-time decision-making scenarios due to their lower computational complexity and faster solution times. These algorithms provide approximate solutions that satisfy constraints within a reasonable timeframe. The considered problem can be solved with a greedy approach or different randomized heuristic algorithms. A greedy approach is rapid since it makes only one greedy local optimal decision at each step and does not revisit previous decisions. A randomized heuristic algorithm tries to avoid falling into a local optimal solution as much as possible and improve the accuracy of the solution at the cost of longer computation time. The allocation problem can be solved by randomized heuristic algorithms such as simulated annealing [18–22], a generic algorithm [23, 24], and tabu search [25]. Simulated annealing randomly explores a neighborhood solution and accepts a worse solution with a certain probability. A genetic

algorithm [26] is an iterative optimization algorithm that uses a population-based approach to search for solutions by evolving a population of candidate solutions through selection, crossover, and mutation. In tabu search [25], an initial solution is randomly generated and iteratively improved by exploring the neighborhood solutions while keeping track of previously visited solutions using a tabu list, and the process continues until a stopping criterion is met.

MILP methods excel in small-scale problems by providing optimal solutions and strict constraint satisfaction; heuristic algorithms are well-suited for large-scale problems and real-time decision-making, offering faster solution times and scalability, albeit without guaranteed optimality.

1.1.5 Network function management and orchestration

Network function management and orchestration is a framework that focuses on managing and orchestrating VNFs within a software-defined networking environment. It aims to streamline the deployment, scaling, and management of network functions by leveraging automation and centralized control.

VNF deployment can take two approaches: containerized deployment and virtual machine deployment. In containerized deployment, VNFs are packaged as separate container instances using technologies like Docker, offering lightweight and flexible deployment. Containers include all necessary components and dependencies, ensuring easy management and efficient resource usage; they offer quick startup and shutdown, and low resource usage. On the other hand, virtual machine deployment provides enhanced isolation and security by running each VNF in its own virtual machine. This emulation of complete OS and hardware environments enables the execution of various VNF types and supports different operating systems and application stacks. Virtual machines enable multiple VNFs to run on a single physical server, optimizing resource utilization and ensuring compatibility.

Kubernetes, as an open-source system, offers comprehensive features and functionalities that are specifically tailored for the deployment and management of containerized functions, making it an ideal platform for VNFs. One of the key components in Kubernetes is the Pod, which serves as the fundamental unit of deployment. A Pod encapsulates one or more containers and associated

resources required for the execution of a VNF.

Furthermore, Kubernetes acts as a powerful orchestration framework for VNFs, providing advanced capabilities for managing and controlling their lifecycle. It offers robust mechanisms for load balancing, fault tolerance, and automatic scaling, ensuring the reliability and availability of VNFs in dynamic network environments. Kubernetes controllers play a crucial role in maintaining the desired state of VNFs by continuously monitoring their health, making necessary adjustments, and ensuring adherence to defined policies.

1.2 Problem statements

This thesis studies six specific problems about resource allocation with workload-dependent failure probability in network virtualization, each of which includes one or several questions that have not been addressed and are answered in this thesis.

1.2.1 Resource allocation model with single backup against workload-dependent failure probability

Failures affect the service quality and user experience [12]. A failure of network function stops users' services unexpectedly when it occurs; prevention and protection strategies are required. Service providers can use backup resources to improve the availability of functions confronting failures and to avoid the interruption of services [13]. The recovery is performed on the trigger basis of failure detection. The CB strategy provides protection to corresponding functions, where the backup resources are only reserved without being active. The HB strategy takes place while the system is still processing before a failure occurs. During the implementation of a service, the backup resource in the corresponding node is always activated.

For each function, a shorter unavailable time for recovery can improve the user experience [12]. This thesis takes the maximum expected unavailable time (MEUT) among all functions as an evaluation metric, which is determined by different protection strategies and allocations. Questions arise: how to approximate a workload-dependent failure probability and what is the primary

and backup resource allocation with different protection strategies to minimize MEUT with a workload-dependent failure probability? These question has not been addressed, which is studied in Chapter 3.

1.2.2 Resource allocation model with multiple backups against workload-dependent failure probability

The allocation of multiple backup servers offers potential benefits in enhancing function reliability, but introduces the challenge of optimizing protection policies to manage anticipated unavailable time while considering varying backup strategies. Several unavailability-aware works on the assignment of multiple backup servers have been studied in [27–32]. The works in [31, 32] focused on the reliability of services which consist of both function and link reliability. The works in [27–30] focused on function reliability and allocated backup servers for the functions. The above works mainly considered the reliability based on the failure probabilities regardless of considering the effects of backup strategies on the workload and the recovery time. This thesis takes MEUT as a metric and considers a workload-dependent failure probability. Further, compared with a single backup model in Chapter 3, where each function is protected by only one server, multiple backup servers can be assigned to protect a function with different strategies in the proposed model. Adopting multiple backup resources may increase the reliability of a function. In order to cope with the expected unavailable time for each function with multiple backup servers, adopting a suitable protection policy, which prioritizes available remaining servers protected with the different backup strategies, is required.

A question arises: is there any model that allocates backup resources for middleboxes under a suitable protection priority policy to minimize MEUT with the workload-dependent failure probability and different backup strategies? This question has not been addressed, which is studied in Chapter 4.

1.2.3 Resource allocation model against multiple failures with workload-dependent failure probability

Fault tolerance and load balancing are two key roles in resource allocation against failures. The unsuccessful recovery is related to the protection approaches. The works in Chapters 3 and 4 only considered dedicated protection, where each backup resource is specifically allocated and reserved for a primary function. If a primary function fails, it can be recovered by any one of the backup resources as long as it is available. On the other hand, resource sharing is usually considered in backup resource allocation to utilize backup resources more efficiently, which is named shared protection. In shared protection, backup resources are shared by several primary functions. If a primary function fails, it may not be able to be recovered by an available backup resource due to the concurrent failures of other functions sharing the backup resources.

A question arises: is there any resource allocation model for both dedicated and shared protection considering the unsuccessful recovery probability and the maximum unavailable probability after recovery among nodes with the workload-dependent failure probability? This question has not been addressed, which is studied in Chapter 5.

1.2.4 Robust resource allocation model against uncertain recovery time in different protection types with workload-dependent failure probability

The shared protection benefits in maximizing resource utilization and efficiency, yet the challenge lies in balancing the complexity of resource sharing and function recovery to ensure overall system performance and service availability.

The one-to-one protection correspondence in dedicated protection may not always exist due to the insufficient computing capacity of a physical node in practical scenarios; the collected traces for the recovery time of each unavailable function by a node is not limited to the nodes protected with the dedicated protection. The recovery time in the shared protection is larger than that of

the dedicated protection. This is because the dedicated protection assumes that all unavailable functions can be recovered without a waiting procedure, which is caused by resource sharing when other unavailable functions occupy exclusive computing resources for recovery. Combining these two recovery times in different protection types as one and considering the upper bound of the two recovery times in the collected trace brings too conservative resource usage, which leads to less resource efficiency.

Service providers aim to provide reliable function deployment in a cost-efficient manner with considering robustness against uncertain recovery time caused by resource sharing. The unavailable time of a function for recovery can be adopted to estimate the reliability of a function [12]. A recovery time guarantee is defined as that, for any function that is protected with either dedicated or shared protection, the worst-case expected recovery time under an uncertain recovery time set is guaranteed with no greater than a given value.

To manage the primary and backup resources and recovery of the VNFs to improve the service performance is required. A question arises: is there any function deployment model considering uncertain recovery time in both dedicated and shared protection to minimize the deployment cost while satisfying the expected recovery time guarantee with workload-dependent failure probability? These question has not been addressed, which is studied in Chapter 6.

1.2.5 Resource allocation strategies for accelerating recovery under reliability guarantee with workload-dependent failure probability

Shared protection sparks our imagination regarding resource efficiency, prompting the question of how to further optimize the utilization of these resources to enhance recovery efficiency.

The work in Chapter 5 addressed a resource allocation model to minimize unavailability with considering a deterministic recovery rate with cold backup for recovery. In contrast to the approach in Chapter 5, which did not account for the influence of diverse backup modes and the allocated computing capacity of backup resources on function recovery time, this chapter advances resource

efficiency by considering assigning a more active workload for pre-configuration to a backup resource enables faster takeover of tasks from the primary function, leveraging pre-configuration advantages albeit with a trade-off of increased workload. Similarly, allocating additional computing capacity to a backup resource for recovery expedites the recovery process, at the cost of higher deployment expenses [19].

Service providers intend to cost-efficiently deploy reliable functions with considering reliability guarantees. This work considers a recovery time guarantee as the expected recovery time for any function is assured not to exceed a particular value. An unsuccessful recovery probability guarantee is considered as the unsuccessful recovery probability of any function is assured not to exceed a particular value.

The workload for pre-configuration of backup instances in different modes and the allocated workload of backup resources for recovery affect the recovery time, workload-dependent failure probability, and cost. A question arises: is there any resource allocation model satisfying the reliability guarantees aiming to minimize the deployment cost with workload-dependent failure probability? These question has not been addressed, which is studied in Chapter 7.

1.2.6 Implementation of backup resource management controller for reliable function allocation in Kubernetes

The costs for resource management and maintenance account for a large part of the entire life cycle of network software. Considering that a shorter expected unavailable time for recovery can improve the user experience, the works in Chapters 3 and 4 adopted the two strategies for primary and backup resource allocation. The resource allocation model can be extended to handle the dynamic reconfiguration based on the requirement. Taking a dynamic reconfiguration triggered by a failure as an example, one of the backup resources protecting the failed function needs to be activated for recovery. A new backup resource needs to be allocated to replace the activated one which is used for the recovery of the failed primary function for higher availability. To adapt to the dynamic reconfiguration requirement, backup resource management con-

sidering the dynamic reconfiguration is required.

However, Kubernetes does not provide a resource type to define the backup Pods with considering the different backup strategies. In addition, it does not provide automatic resource management based on user requests for the backup Pods.

1.2.7 Implementation of real-time function deployment with resource migration in Kubernetes

Several studies addressed reliable function allocation models with considering different objectives for the continuity of network services such as unavailable time of functions [33] and continuous servable time [34]. The models are solved by different optimization solvers and heuristic algorithms. The work in [35] introduced an allocation-model-based scheduler in Kubernetes to connect multiple optimization models with the platform. However, the computation for solving an optimization model and deploying the functions with an allocation-model-based scheduler [35] is time-consuming.

In a dynamic scenario of arrivals and releases of requests, resources of arriving functions need to be allocated; function deployment is required to respond to the requests promptly and continuously [36]. Similarly, the deployment after request releasing loses the optimality so that re-allocation of functions for optimality is required to be considered. Function migration is a common approach for handling real-time requests by adjusting the resource allocation to avoid service performance degradation or constraint violations, where simple migration schemes inevitably affect the continuity of the network service and lead to massive reconfiguration costs and service degradation. However, Kubernetes does not provide a tool to deploy the functions in a real-time and optimal manner. In addition, it does not provide automatic resource migration management of functions to maintain the optimal deployment and continuous services for the dynamic scenario.

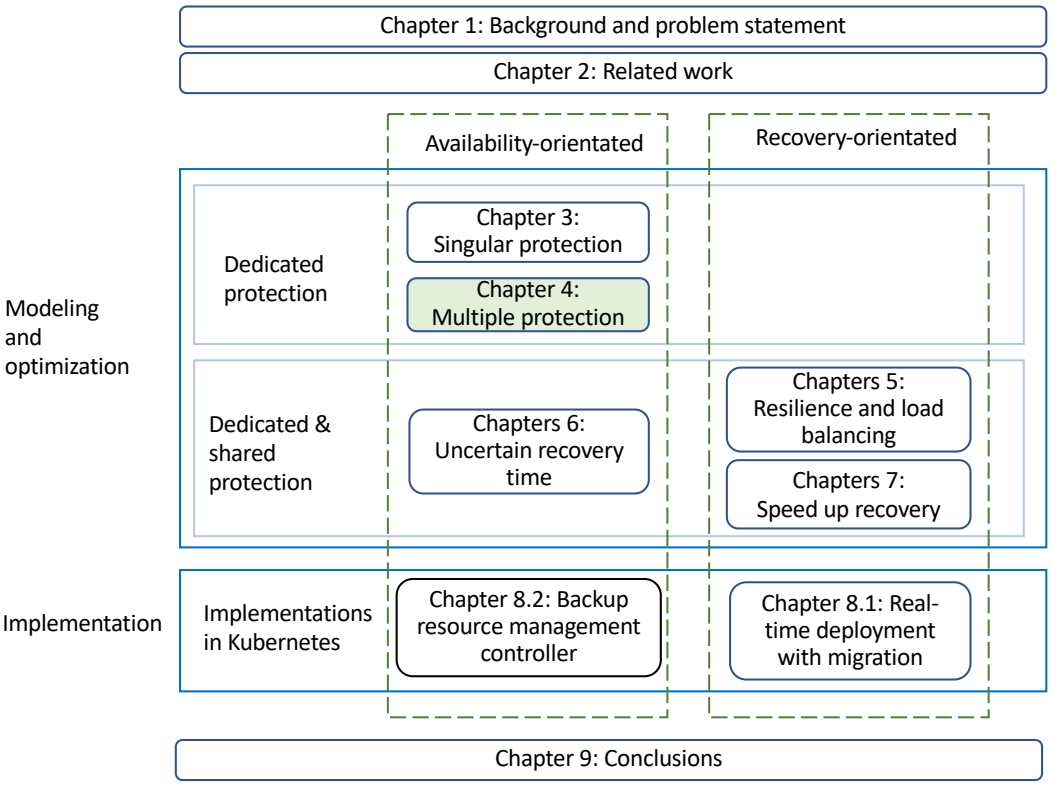


Figure 1.2: Chapter overview.

1.3 Overview and contributions of this thesis

Figure 1.2 shows the chapter overview of this thesis. Chapter 1 introduces the background and the problem statements of this thesis. Chapter 2 surveys the related works in literature.

Chapters 3–7 center around the modeling and optimization of high availability and fault-tolerant resource allocation, categorized into two primary aspects: availability-oriented resource allocation and recovery-oriented resource allocation. This exploration encompasses two primary protection strategies: dedicated protection and shared protection. In the realm of dedicated protection, Chapters 3 and 4 explore the effect of resource allocation on availability by considering the workload-dependent failure probability and different backup modes. Transitioning to the realm of shared protection, which introduces additional complexities in the recovery process, Chapter 6 delves into the impact of uncertain recovery times stemming from shared protection. Chapters 5 and 7 concentrate on specific aspects, including resilience, load balancing, and recovery speed, all intimately related to the recovery process.

To be more specific, Chapters 3 and 4 focus on the dedicated protection; each function is protected by one and multiple nodes in each chapter, respectively. Chapter 3 examines single protection cases and approximates workload-dependent failure probability through a step function. It investigates the numerical relationship between workload-dependent failure probability and Maximum Expected Unavailable Time (MEUT). Chapter 4 focuses on the case that multiple backup servers can be assigned to protect a function with different strategies by adopting a suitable protection policy, which prioritizes available servers protected with the different backup strategies, is discussed. Chapters 5–7 consider the backup resource allocation and recovery in terms of both dedicated and shared protection. Chapter 6 focuses on the high available protection aspect. It considers an uncertain recovery time due to several reasons, such as waiting time due to shared protection; node and function characteristics that lead to different initiation times for hardware and applications, and restoration times for data and process states, etc. It considers an uncertainty set of recovery time and calculates the worst-case recovery time under the uncertainty set. Chapters 5 and 7 concentrate on the

fast recovery aspect, delving into strategies aimed at expediting the recovery process. Chapter 5 considers a deterministic recovery time and focuses on the resilience and load balance after failure. It considers preventive recovery priority setting to minimize a weighted value of the probability of unsuccessful recovery and the maximum unavailable probability after recovery among physical nodes. Chapter 7 considers that the extra-assigned recovery workload can be adopted, which means that the recovery workload can be scaled, to speed up the recovery, while improving the resource efficiency to fully utilize the idle capacity for faster recovery. On the other aspect, the extra-assigned recovery workload may lead to unsuccessful recovery in a specific failure configuration. Chapter 7 handles the availability guarantee with minimizing deployment cost. All the resource allocation models introduced in Chapters 3–7 are solved by both the MILP approach for optimal resource allocation and by the heuristic algorithm to tackle the larger size resource allocation scenarios. Chapters 3–7 focus on the analysis of the theoretical optimization model; Chapter 8 focuses on the implementation of function deployment; it also encompasses two aspects, availability-oriented and recovery-oriented for implementation. It solves the backup resource definition and transformation of different types of the backup resource by controller; it also solves the real-time function deployment by considering resource migration with a two-layer controller. The detail of each chapter is described as follows.

Chapter 3 proposes an optimization model to derive a primary and backup resource allocation considering a workload-dependent failure probability to minimize MEUT. Each node is able to offer capacity for both primary and backup resource allocation. The proposed model adopts both HB and CB strategies, which may lead to different failure probabilities of nodes depending on the workloads and have different recovery time. The workloads determined by different primary and backup resource allocations and different strategies affect the failure probability of node. The proposed model handles the balance between the workload and the recovery time caused by different strategies. This model can conservatively approximate different relationships between the failure probability and the workload by using a non-decreasing step function with an increase of the number of steps. With a step function as a tool, the workload-dependent failure probability can be linearized for simpli-

fication. This work formulates the optimization problem as an MILP problem. Numerical results show that the proposed model reduces MEUT compared with the conventional model which does not consider the workload-dependent failure probability.

Chapter 4 proposes an optimization model to derive a multiple backup resource allocation with the workload-dependent failure probability to minimize MEUT under a priority policy. The proposed model adopts the HB and CB strategies, which require different workloads and recovery time. The workloads determined by different backup resource allocations and different strategies affect the failure probabilities of servers. This chapter derives the theorems that clarify the influence of protection policies on MEUT. This chapter formulates the optimization problem as an MILP problem. The numerical results show that the proposed model reduces MEUT compared with the single backup model [37] in which each function is protected by only one server without protection priority of servers and the conventional model without the workload-dependent failure probability. The priority policy adopted in the proposed model specifying that the server which adopts the HB strategy has higher priority than that with the CB strategy for multiple backup resources suppresses MEUT compared with other priority policies.

Chapter 5 proposes a primary and backup resource allocation model with preventive recovery priority setting (RAM-P) against multiple failures to minimize a weighted value of unavailable probability (W-UP) for both dedicated and shared protection. W-UP consists of two parts of unavailable probabilities; it considers the probability of unsuccessful recovery and the weighted maximum unavailable probability after recovery among nodes. Each node fails with different workload-dependent failure probabilities which are related to the machine property; each failure pattern occurs with a probability. The model considers the unsuccessful recovery probability against multiple failures and the weighted maximum unavailable probability after recovery with the failure probability related to the workload variation after activating the backup resource against failures. This chapter introduces a recovery strategy which is determined at the operation start time. Once failures are detected, the recoveries are operated with the workload variation based on the priority setting. This chapter also discusses an approach to obtain unsuccessful recovery prob-

ability without priority setting for a special case that focuses on unsuccessful recovery at the operation start time and handles the load balancing against failures at run time. This chapter formulates the optimization problem as an MILP problem. This chapter develops a heuristic algorithm to solve larger size problems in a practical time. The numerical results observe that the proposed model, which jointly considers the unsuccessful recovery and load balancing against failures, outperforms the baseline models which consider each type of unavailability separately. The developed heuristic algorithm is approximately 729 times faster than the MILP approach with 1.6% performance penalty on W-UP.

Chapter 6 proposes a robust function deployment model against uncertain recovery time with satisfying an expected recovery time guarantee to minimize the number of active nodes. This chapter considers that each node fails with a workload-dependent failure probability; both HB and CB strategies are considered to provide protection against failures, which have different recovery times and may lead to different failure probabilities of nodes depending on the workloads. The proposed model handles the balance among the workload, deployment cost, and recovery time caused by different strategies and different protection types. Multiple functions protected by a node can share the backup resources to reduce the number of active nodes for cost-efficiency, which also affects the recovery time if the number of unavailable functions is so large that the remaining capacity cannot recover them. The considered problem aims to minimize the number of active nodes to determine the protection types and strategies with satisfying an expected recovery time guarantee. The uncertain recovery time can be represented by an uncertainty set considering the upper bound of the average recovery time among the nodes protecting it and the upper and lower bounds of each recovery time protected by a node collected in each protection type. The model is formulated as an MILP problem with handling the worst-case expected recovery time under an uncertain recovery time. To solve the problem in a practical time, a heuristic algorithm is developed. It reduces the number of active nodes while decreasing the worst-case expected recovery time. The numerical results reveal the superiority of the proposed model with the recovery time guarantee, uncertainty set, and shared protection; This chapter investigates the dependency on the bounds of uncertain

recovery time.

Chapter 7 proposes a resource allocation model under reliability guarantees to minimize the number of activated nodes. The model considers that backup resources can recover an unavailable function in a period of time, which relates to the backup modes and assigned recovery workload. The backup resource of each function has multiple modes with different degrees of pre-configuration and different recovery times. This chapter further considers that the extra-assigned recovery workload can be adopted to speed up the recovery while improving the resource efficiency to fully utilize the idle capacity for faster recovery. On the other hand, the extra-assigned recovery workload may lead to unsuccessful recovery in a specific failure configuration. This chapter considers two reliability indicators, which are recovery time and the total unsuccessful recovery probability; each reliability indicator is restricted under guarantee while minimizing the number of activated nodes. The numerical results reveal that the number of activated nodes is reduced compared to baselines by considering the proposed model.

Chapter 8.1 designs and implements a controller to manage the primary and backup resources of network functions. This chapter introduces a new resource type called *backup Pod set (BPS)*, which is a *custom resource* in Kubernetes [38]. *BPS* includes a certain number of different types of Pods which are the primary, HB, and CB Pods. The transitions of different types of Pods can be customized by cooperating with the allocation-model-based scheduler introduced in [35] or randomly. Demonstrations validate the effectiveness of the controller. Chapter 8.2 designs and implements controllers to deploy network functions in a real-time and reliable manner. This chapter introduces two new resource types called *migratable Pod set* (MPS) and *global optimizer* (GO), each of which is *custom resource* in Kubernetes [38]. To achieve the function deployment in a limited computation time, this chapter introduces two controllers in Kubernetes for a two-stage function deployment. The MPS controller manages the Pods for an intermediate allocation with a model or a heuristic algorithm to respond to the requests promptly. The GO controller manages the MPS instances by optimizing Pod allocations with considering resource migration of network functions to maintain the current state of the Pods to keep the current state consistent with the desired state.

Chapter 2

Related works

2.1 Resource allocation problems in different fields

The work in [39] realized the collaborative resource allocation optimization of mobile edge computing to reduce the delay of edge servers in the transmission process. The work in [40] addressed both node and path resource allocation and introduced a backup computing and transmission resource allocation model for virtual networks with providing a probabilistic protection against multiple facility node failures. The work in [41] investigated a task, spectrum, and transmit power allocation problem for a wireless network to jointly provide computation and communication services to users. The work in [42] introduced distributed resource allocation approaches with jointly considering power allocation, interference management, power and rate allocation, resource allocation, and pricing policies. The work in [43] studied spectrum management for each user to determine the optimal allocation of their transmission power in each one of the bands, where users are offered the option to transmit via licensed and unlicensed bands.

This thesis focuses on resource allocation in the field of virtual function deployment with considering the effect of high workload on the failure probabilities of nodes. Allocating virtual function, such as network services and applications, to suitable physical or virtual resources can yield cost savings, heightened elasticity and adaptability, enhanced fault tolerance, and optimization

of performance parameters encompassing response time, resource utilization, scalability, stability, and dependability. The proposed models in this thesis jointly considered two aspects, which are availability-oriented and recovery-oriented, for resource allocation. This consideration not only offers promising avenues for cost reduction, but also amplifies reliability and stability, fortifying its resilience in the face of potential failures.

2.2 Failure scenarios in NFV and workload-dependent failure probability

VNF operations are susceptible to various failure scenarios. These include VNF crashes due to resource constraints or software bugs, network disruptions causing delays or failures between VNFs, load imbalances, cascading failures, resource exhaustion, dependency issues, data integrity concerns, configuration drift, and the significant impact of physical node failures, rendering all hosted functions unavailable. Addressing these scenarios through monitoring, recovery mechanisms, and resilient resource allocation design ensures reliable VNF performance.

Several studies addressed fault-torelance and tackled the resource allocation problem with different objectives and approaches. The work in [44] minimizes the required total capacity for primary physical machines in a cloud provider by providing probabilistic protection. The work in [28] maximizes the probability for a full recovery or maximizes the expected number of functions that can be recovered simultaneously by providing several properties of an optimal assignment of backups servers to functions. The work in [34] maximizes the continuous available time of service function chains by allocating functions to virtual machines (VMs) in a sequence of time slots. The work in [45] minimizes resource consumption while providing specific latency to ensure the required quality of services based on a flexible resource allocation approach. The resource allocation models presented in the above works determine the allocations without considering a workload-dependent failure probability and different protection strategies; this thesis jointly considers different protection strategies and the relationship between the workload and the failure probabil-

ity.

Several studies addressed the analysis of failure characteristics with computing and storage (workload) in datacenters. The increase of workload affects both CPU utilization and computation time [46], which further causes degradation of cloud service availability [47]. The work in [9] reported a correlation between the type of workload and the failure probability; the work in [10] reported a correlation between the workload intensity and the failure probability. Cloud computing workload characteristics have been studied in [48]. They presented a large-scale analysis of workload resource utilization and a characterization of a cloud datacenter using tracelogs made available by Google. The work in [49] conducted further research on modeling the reliability of cloud datacenters, presenting an analysis of failure patterns and repair time of a large-scale production system. Based on the relationship between the workload and the failure probability, the work in [8] introduced a migration policy which is aware of service level agreements and availability violations at the PM and VM caused by high workload when CPU utilization exceeds a certain threshold. The work in [50] presented a dynamic replication scheduling based on workload statistics, which does not trigger replication if the available CPU usage does not exceed the threshold value. The above works in [2, 6, 7, 9–11, 46, 47] considered the relationship between the workload and the failure probability without approximating it as a numerical function and jointly considering the influence of different protection strategies on the workload. This thesis adopts an S-step function which can conservatively approximate a non-decreasing function to approximately present the relationship between the workload and the failure probability. This thesis presents resource allocation models based on workload-dependent failure probability.

2.3 NFV resilience and backup strategies

Resilience is a critical quality-of-service (QoS) metric in NFV, determining the ability of a service to withstand failures. When a node or link within a service fails, the entire service becomes inoperable, necessitating its cessation. To address this challenge, a common approach is to deploy redundant services, distributing requested VNFs across different nodes (referred to as node-disjoint)

to mitigate node failures and selecting paths that avoid traversing the same link (known as link-disjoint) to mitigate link failures. This redundancy strategy ensures continuous service availability and enhances fault tolerance in the event of failures. Beck et al. [51] propose a recursive heuristic for survivable VNF placement to guarantee an end-to-end VNF protection. Han et al. [52] explore resilient respects of the individual VNF in terms of fault management (e.g., failure detection and automated recovery) or state management. They also discuss existing solutions for these aspects. Herker et al. [53] formulate how to calculate the VNF placement availability in data center topology. They also present an efficient heuristic on how to place VNFs and their backups to satisfy the requested availability. Fan et al. [54] consider how to compute one backup service function chain (SFC) when the primary SFC is given such that the overall availability is satisfied.

To ensure the reliability of network services, it is essential to have recovery strategies in place to address network failures. Protection mechanisms reserves backup resources in advance and reassign traffic to these resources upon failure, enabling prompt recovery. Several works addressed to suppress the effect of unavailability by using backup resources to improve the survival rate of function confronting network failures. Taking this direction, the work in [55] studied several approaches to provide backup and recovery including the HB and CB strategies. The work in [30] presented a backup resource allocation model which adopts the HB strategy for middleboxes to minimize the worst weighted unavailability. The work in [56] studied a virtual network embedding problem by adopting the CB strategy. Different from the above works, this thesis takes the unavailable time as an evaluation metric and considers the different workload and recovery time of different backup strategies. The resource allocation problem introduced in [57] minimizes the latency between the pair of primary and backup functions with considering the redundancy to be used for recovering the failed functions. The work in [58] introduced a cost-efficient redundancy algorithm to enhance the network resilience with less backup cost and operation expenditure while maintaining high requests acceptance ratio by using the importance measure in the context of VNF forwarding graph to decide the backup candidate and the PN hardware it works on. The introduced methods in [59] considers the number of active services as

the system state, estimates the state of idle resources based on it, and treats the number of admitted services as the action. The Viterbi-based Reliable Static Service Placement algorithm evaluates each action, and the VRSSP-based Value Iteration algorithm determines the optimal policy using the value iteration method.

The recovery time against failures relates to different backup modes [60]; the modes have different workloads to provide different pre-configuration states for recovery [61]. If the backup resources are only deployed without being activated as an instance, the recovery time of an unavailable function relates to the activation and instantiation time; the backup mode is called cold backup (CB). If a backup instance is activated and synchronized with the primary function so that the backup instance can take over the task running on the function immediately, the recovery time is only affected by the monitoring mechanism of the availability of physical nodes [62]; the backup mode is called hot backup (HB). In addition to the two typical backup modes, warm backup (WB) is also commonly used for cost-efficient prompt failure recovery [63]. In the warm backup, a backup instance of the function is already resident in memory; it is partially or fully initialized for standby; the backup instance synchronizes the state of services from the processing primary function periodically. When the primary function fails, the backup instance can take over the tasks running on the primary function faster than CB based on the pre-configuration. Compared with the HB mode, where the backup instance is fully pre-configured, backup resources with different degrees of pre-configuration in WB recover the unavailable primary function in a longer or equal time. In other words, the more sufficient pre-configuration (including the instantiation, initialization, and synchronization) a backup instance provides, the faster the recovery procedure can be. This thesis considers that there exist multiple states of pre-configuration for different types of service, i.e., stateful and stateless services [64] with different degrees of instantiation, initialization, and synchronization.

2.3.1 Reliability-aware resource allocation

Several reliability-aware works [31, 32, 65] consider both VNF and link availabilities in SFC provisioning. The work in [31] addressed a server selection

problem to jointly minimize the cost of resources and maximize the reliability of the service. It assumed that the cost of using a server is linearly proportional to the amount of resources utilized. The work in [32] introduced a VNF placement and traffic routing policy that jointly maximizes the achieved respective reliabilities of supported network services and minimizes their respective end-to-end delays. Different from the above works, this thesis focuses on the availability of functions, but not the availability of links to estimate the service availability. Similar to [31], this thesis also considers the resource utilization in the proposed model. Different from taking resource utilization as a part of cost, this thesis takes it as a factor which affects the failure probability. Secondly, focusing on function-level placement, there are mainly two types in terms of the assumptions taken by these works. In one type, the works in[27–29] addressed how to allocate the backup resources with different objectives assuming that the backup servers not to fail [27–29]; only the failures of functions were considered. The other type of assumption was considered in [30], which studied unavailability caused by failures of functions and all the corresponding backup servers. Taking the design goals of these works, the work in [27] presented a model to maximize the survival probability of functions. The work in [28] presented models to maximize the recovery probability of failed functions and the expected number of functions that can be recovered simultaneously. The work in [29] analyzed the maximum number of failed functions which can be fully recovered and the minimum number of backup servers required to guarantee a full recovery. Compared to the assignment aspect studied in their works, this thesis has three main differences. First, multiple failures may occur to the servers, which is diffident from [27–29]. Second, different from focusing on unavailable probability by calculating the product of failure probability of servers [30], our work focuses on the expected unavailable time of the function. We consider each situation of different available remaining servers to protect the function by applying different backup strategies, HB or CB. Third, this thesis adopts a policy-based approach to prioritize available remaining servers among multiple backup resources.

Several works investigated reliable resource allocation and load balancing against failures. Taking the aspect of reliable resource allocation, the work in [40] addressed the resource allocation with guaranteeing that the probability

that the protection from a backup facility node fails due to the insufficient reserved backup computing capacity is within a given value. The work in [66] presented an approach to determine the allocations of VNFs in service function chains (SFCs) aiming to minimize the number of affected SFCs upon a node failure. The work in [67] introduced an optimization model to derive the resilient resource allocation aiming to reduce the recovery time. The addressed allocation satisfies a fault-tolerance assurance. The work in [68] presented a fault-tolerant virtual machine placement method for enhancing service reliability in a cloud environment. The work in [69] presented a backup server assignment to maximize the survival probability of network functions of middleboxes. Compared with the existing studies in [40, 66–68], the proposed model addresses the fault tolerance performance by considering the unsuccessful recovery probability against each failure pattern. Instead of taking the fault tolerance assurance as a constraint, the proposed model in the thesis considers the probability of each failure pattern with the workload-dependent failure probabilities of nodes and takes the weighted unsuccessful recovery probability as a part of the objective value.

2.4 Load balancing against failures

Taking the aspect of load balancing against failures, some literature studied load balancing considering the workload variation against failures. The work in [70] addressed how to allocate slave controllers for switches to minimize the load variance difference after a single controller failure. The work in [71] introduced a preventive priority setting model to handle load balancing against multiple failures. The priority for recovery that can be applied to different failure patterns is determined at the operation start time to minimize the maximum utilization ratio in the worst-case of failure patterns. Once nodes fail, the recovery is operated according to the priority setting for prompt recovery. However, the works in [70,71] merely considered reducing the workload variation or the utilization ratio to avoid unevenly overloading without taking the workload-dependent failure probability into account.

Several works introduced resource allocations considering workload variation in load balancing. One type of workload variation is caused by the service

requests. The work in [72] addressed a VNF placement model, where the variation in the load is handled by dynamically instantiating services. As the load of the request types changes, the number of instantiated services changes. The work in [73] introduced an algorithm to balance the load across multiple types of defined resources, i.e., CPU, storage, and bandwidth, by maximizing the resource utilization. Another type of workload variation is caused by failures. The work in [71] addressed how to allocate slave controllers for switches to minimize the maximum resource utilization against controller failures. The work in [74] introduced an optimization model to preventively minimize the worst network congestion among situations with all possible single link failures at the operation start time by determining link weights. Different from the load balancing models presented in the above works, our work minimizes the maximum unavailable probability with considering a workload-dependent failure probability for each node to reveal the relationship between failure probability and workload.

2.5 Water-filling algorithm

The water filling algorithm is one of the classical algorithms in wireless communications to improve the throughput by power allocation, which can be considered as a greedy algorithm to evenly distribute the resource. A traditional physical understanding is to use the analogy of pouring water over a pool with fluctuating bottom. Taking this direction, the work in [75] adopted the Lagrangian multiplier method [76] to obtain the expression of the optimum power allocation. They developed a low-complexity iterative approximate water filling algorithm, which reduces the number of iterations for convergence compared with the traditional iteratively greedy algorithm, to maximize the spectrum efficiency of the system. The water filling approach not only has applications in communication systems to maximize spectrum efficiency by adjusting the transmitting power but also plays an important role in achieving the load balancing in the network [77] and power grid systems [78]. Since a network puts great emphasis on providing low latency services which naturally require using fast-timescale load balancing, the work in [79] reported distributed water-filling algorithm for load balancing to avoid excessive delay

resulted from overloading any particular network node. This thesis adopts the water-filling algorithm to solve the resource allocation effectively with considering the S-step function to approximate the given non-decreasing workload-dependent failure probability. By dividing the capacity into different ranges by thresholds, our developed water-filling algorithm determines the resource allocation by allocating functions on pysical node iteratively until the workload exceeds each threshold.

2.6 Resource sharing and deployment cost

To reduce the cost of utilizing functions, function sharing was studied [17, 80, 81]. Sharing function instances through multiple service chains and sharing backup resources on a node to protect multiple functions are effective approaches to reduce costs. A dynamic backup scheme introduced in [80] reduces resource consumption with sharing backup resources. A dynamic and flexible approach in [81] addresses rate coordination between the upstream and downstream functions as well as the resource allocation for function sharing. An unavailability-aware backup allocation model with the shared protection introduced in [17] minimizes the maximum unavailability among functions which considers a distribution of recovery following the queuing approach. Compared with shared protection models in [17], our work considers the uncertainty of the recovery time caused by resource sharing, instead of focusing on each recovery procedure of a function by a node that provides shared protection. With multiple concurrent unavailable functions, our work judges whether a waiting procedure occurs by considering the expected number of unavailable functions and expected remaining capacity when a function is protected by a node.

The deployment cost also has several measurements. The VNF placement problem for service chains was studied in [82] for minimizing deployment cost and network traffic cost; the traffic cost of each physical link is calculated with the hop distance and the total allocated bandwidth of virtual links embedded in that physical link. It addressed energy consumption as one of the key sources of deployment costs for the service provider and tackled suppressing the cost by reducing the number of unused nodes. The VNF sharing problem in [83] focuses on deployment cost and introduced an incurred cost, which

consists of a fixed cost paid if the VM is activated and a proportional cost paid for each unit of computational capability. The resource allocation model in [36, 84] minimizes the utilization ratio of nodes and increases the number of acceptable requests while satisfying k-fault-tolerance guarantee. Our work focuses on deployment cost; it considers the active number of nodes as the deployment cost and combines the proportional cost into the expected recovery time with considering the workload-dependent failure probability. Our work considers to minimize the deployment cost while satisfying the recovery time guarantee.

2.7 Robust optimization and uncertain recovery time

Robust optimization is one of the commonly used approaches to handle the uncertainty of parameters. A hose model introduced in [85] represents the uncertainty of the traffic matrix. A probabilistic protection model against failure uncertainty and demand uncertainty was introduced in [86]. The resource allocation problem in [23] considered the uncertainty set of the start time and period of unavailability on each node in the availability schedule.

Uncertainty of parameters commonly exists in an optimization problem and can be tackled by robust techniques. The implementation introduced in [62] demonstrates the HB and CB resources and investigated the recovery time against failures. The recovery time depends on the recovery mechanism, such as the types of failures, the time for failure detection and distinction, the initiation time for hardware and applications, and the restoration time for data and process state [87]. It also relates to the size of the function and the activation time for it, which leads to the uncertainty of the recovery time.

Except for the parameters in the previous studies [23, 85, 86] mentioned above, the recovery time may change in diffident situations and float in a certain range. The recovery time has its uncertainty due to several reasons, which include the node and function characteristics that lead to different initiation times for hardware and applications and restoration times for data and process states. This thesis introduces an uncertainty set that considers the upper and

lower bounds of the recovery time of a function protected by each node and the upper bound of the average recovery time among the nodes. The uncertainty of the recovery time is also affected by the protection types; waiting time due to the concurrent failures of functions needs to be considered. This is because, if a primary function fails, it may not be able to be recovered by an available backup resource due to the concurrent failures of other functions sharing the backup resources; the recovery time also depends on the availability of other functions protected by the same set of nodes. Taking the average value for estimating the recovery time in a different situation may lead to a reliability guarantee violation. In other words, the reliability indicators satisfy each reliability guarantee in the average cases; however, they may not satisfy the reliability guarantee when we consider the worst case among changeable recovery times and probabilities. Thus, to satisfy the reliability guarantees in any recovery time distributions, the uncertainty of recovery time needs to be involved in the considered problem, which is addressed in this thesis.

Chapter 3

Single backup resource allocation with workload-dependent failure probability

This chapter proposes an optimization model to derive a primary and backup resource allocation considering a workload-dependent failure probability to minimize the maximum expected unavailable time (MEUT) [37, 88].

The remainder of the chapter is organized as follows. Section 3.1 describes the proposed model. Section 3.2 discusses different values of MEUT corresponding to different parameters. Section 3.3 presents a heuristic algorithm. Section 3.4 presents the numerical results. Section 3.5 discusses the approximation of non-decreasing function. Section 3.6 summarizes this chapter.

3.1 Optimization Model

3.1.1 Primary and backup resource allocation

This chapter builds a primary and backup resource allocation model with consideration of the workload-dependent failure probabilities. The model considers two backup strategies to provide protection. Each PM has its failure

probability affected by the resource allocation and backup strategies. The model focuses on the primary and backup resource allocation and the backup strategies to minimize MEUT among VMs by balancing the workload and the recovery time of the two strategies. Consider a set of PMs and a set of VMs, which are denoted by K and V, respectively. Each PM is able to offer capacity for both primary allocation and backup resource allocation. When failures occur in some PMs, the workloads in the failed PMs are transferred to the corresponding PMs which provide protection. Note that the primary and backup resources of a VM are allocated in two different PMs.

Consider two backup strategies $b \in B = \{0, 1\}$. $b = 0$ denotes the CB strategy and $b = 1$ denotes the HB strategy. When a PM fails, the backup resource corresponding to each VM allocated in the failed PM provides either HB or CB to recover the VM. In the CB strategy, a VM is allocated and operated in the primary resource of PM while it is protected by the backup resource of another PM. The reserved idle capacity is not taken into account as an active workload in the backup resource allocation. In the HB strategy, the VM protected by the corresponding backup resource which is operated with the VM allocated in the primary resource simultaneously is counted as the active workload, and the active workload affects the failure probability of PM.

Let x_{ij}^{kb}, $i \in K, k \in K\backslash\{i\}, j \in V, b \in B$, denote a binary decision variable; x_{ij}^{kb} is set to one if VM $j \in V$ is allocated at PM $i \in K$ and is protected by PM $k \in K\backslash\{i\}$ with strategy $b \in B$, and zero otherwise. l_j^{R} denotes the requested load of VM $j \in V$, which denotes the computing resources requested by the VM. The failure probability of each VM is related to the workload of corresponding PM. Let W_i denote the workload of PM $i \in K$, which contains the primary resources and hot backup resources. Let R_i denote the requested load of PM $i \in K$, which contains the primary resources and both HB and CB backup resources. W_i and R_i, $i \in K$ are expressed by:

$$W_i = \sum_{j \in V} \sum_{b \in B} \sum_{k \in K\backslash\{i\}} l_j^{\mathrm{R}} x_{ij}^{kb} + \sum_{i' \in K\backslash\{i\}} \sum_{j \in V} l_j^{\mathrm{R}} x_{i'j}^{i1}, \forall i \in K, \tag{3.1a}$$

$$R_i = \sum_{j \in V} \sum_{b \in B} \sum_{k \in K\backslash\{i\}} l_j^{\mathrm{R}} x_{ij}^{kb} + \sum_{b \in B} \sum_{i' \in K\backslash\{i\}} \sum_{j \in V} l_j^{\mathrm{R}} x_{i'j}^{ib}, \forall i \in K. \tag{3.1b}$$

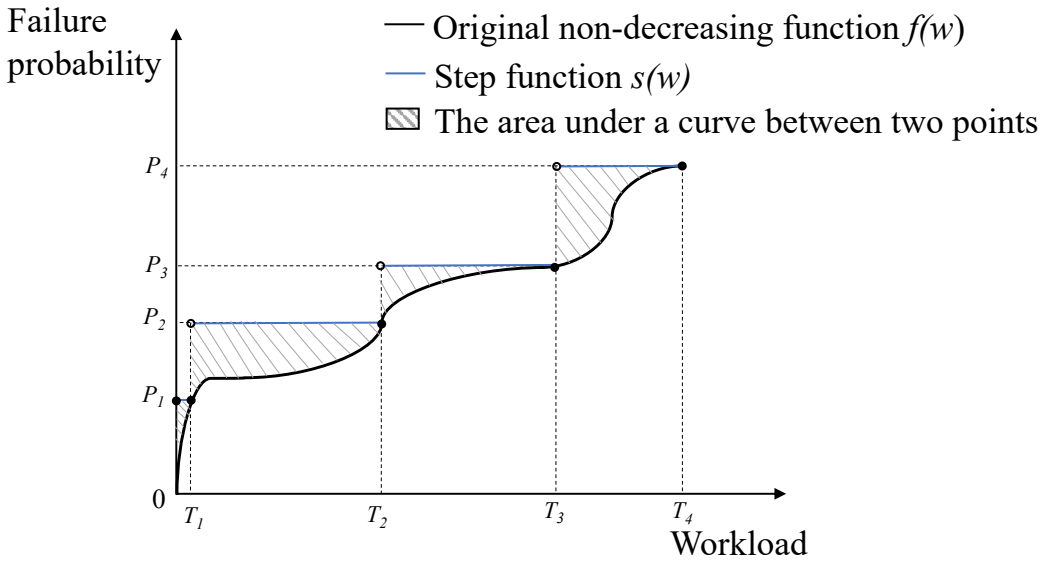
Let C_i denote the upper bound of computing capacity which PM $i \in K$ can provide for primary and backup allocation. The computing resources include multiple types of resources, such as CPU and memory; the workload is one of the computing resources. The units are changed when the measured computing resources are changed. For example, if the CPU usage is measured as a computing resource, the unit can be CPU cores; if the memory usage is measured as a computing resource, the unit can be Byte. Therefore, this work assumes that W_i, R_i, C_i, $i \in K$, and l_j^{R}, $j \in V$, are unitless.

3.1.2 Unavailability with workload-dependent failure probability

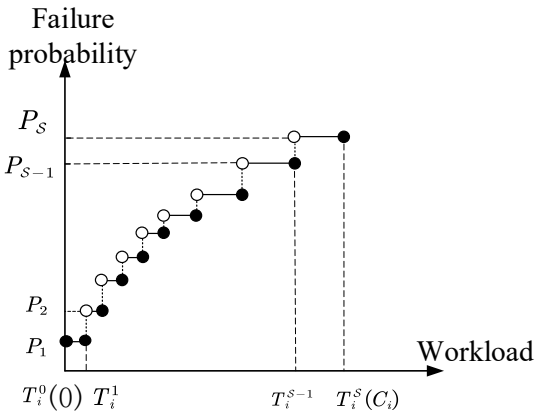
This work considers that multiple simultaneous failures occur in PMs. This work assumes that each PM fails independently. This work assumes that backup resources in PM $k \in K$ can recover all VMs which are protected by PM k simultaneously.

Given the workload-dependent failure probability, which is assumed to be a non-decreasing function, an S-step function can be used, as shown in Fig. 3.1(b), to conservatively approximate it. Adopting step functions is a common approach for approximating a given curve. Let $f(w)$ be a given non-decreasing failure probability, which shows the practical relationship between the failure probability and the workload; let $s(w)$ denote an S-step function; w is the PM workload. Given the step number S with considering the accuracy of fitting the non-decreasing function $f(w)$ in an S-step function and the computation time to solve the proposed model, the failure probability can be designed with the threshold of $T_s, s \in \mathcal{S}$, and corresponding failure probability P_s. To fit the given curve as accurately as possible, the conservative approximation can have different goals while keeping $s(w) \geq f(w)$, such as minimizing $\int_0^{C_i} (s(w) - f(w))\,\mathrm{d}w$ or $\max_{s \in S} \left\{ \int_{T_i^{s-1}}^{T_i^s} (s(w) - f(w))\,\mathrm{d}w \right\}$, as shown in Fig. 3.1(a).

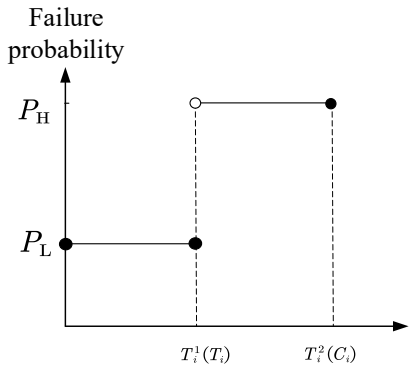
This work considers that the failure probability of PM $i \in K$ according to the PM workload is denoted by $\mathcal{P}_i^{\mathrm{S}}$. $\mathcal{P}_i^S$ is expressed by the following S-step function, which relates the PM workload to the failure probability of the PM. Let $s \in \mathcal{S} = [1, S]$ denote the sth step in the S-step workload-dependent failure



(a) Approximation of non-decreasing failure probability.



(b) S-step failure probability.



(c) Two-step failure probability.

Figure 3.1: Workload-dependent failure probability is expressed by a non-decreasing step function.

probability; S denotes the number of steps in a step function:

$$\mathcal{P}_i^{\mathrm{S}} = \begin{cases} P_1, & T_i^0 \leq W_i \leq T_i^1 \\ P_2, & T_i^1 < W_i \leq T_i^2 \\ \cdots & \cdots \\ P_S, & T_i^{S-1} < W_i \leq T_i^S, \end{cases} \tag{3.2}$$

where $T_i^0 = 0$ and $T_i^S = C_i, i \in K$.

When the workload of a PM increases from the range of $(T_i^{s-1}, T_i^s]$ to $(T_i^s, T_i^{s+1}]$, the failure probability increases from P_{s-1} to P_s. As the workload increases, even though the PM has remaining capacity, the PM becomes fragile, and has a higher failure probability to handle the extra workload.

When a PM fails, it is not able to handle the corresponding computing resources for VMs hosting on it. Each VM hosted on the failed PM needs to be reallocated to another available PM. r_j denotes the expected unavailable time for VM $j \in V$. When a VM becomes unavailable, two situations may hold. One is that VM $j \in V$ fails and becomes unavailable before it is recovered by an available PM hosting its backup resources. Once a failure occurs to a PM, the recovery time t_1 or t_0 which depends on the backup strategies is required to recover the unavailable VM hosted on the failed PM. Let t_1 be a given parameter that denotes the recovery time for the HB strategy. Let t_0 be a given parameter that denotes the recovery time for the CB strategy. The backup resource protecting a VM with the HB strategy is activated and synchronized with the VM, while the backup resource with the CB strategy requires time for activation and information synchronization. The recovery time t_1 for VM is equal to or smaller than t_0. The other one is that both two PMs hosting primary and backup resources of VM fail simultaneously. When the VM is protected by an unavailable PM, there is no available PM that is determined in advance to provide protection to the VM, where the unavailable time of t_3 is required. This work assumes $t_3 \geq t_0 \geq t_1$.

Once a failure of the PM hosting the primary resource of a VM is detected, the hot backup resource of the VM takes over the tasks operating on the primary resource immediately. t_1 is determined by the availability monitoring mechanism of PMs. The backup resource with CB is activated and attached to the virtual storage volume to obtain the latest snapshot of the failed primary

resource [15]. t_0 is determined by multiple factors including the availability monitoring mechanism, the time for activating the backup resource, and the time for attaching the volume. The situation that the two PMs hosting the primary and backup resources of a VM fail simultaneously corresponds to the unavailable time of t_3. There are several maintenance systems to maintain this situation with the unavailable time of t_3 [87,89,90]. VMware vCenter Site Recovery Manager delivers automated orchestration of fail-over and fail-back to minimize downtime [89]. The works in [87, 90] introduced the maintenance mechanisms that can be applied for the systems. The maintenance process may include monitoring and updating the standby system, and initiating the emergency operation. t_3 can be influenced by the interplay of several factors of the maintenance mechanisms, such as the types of failures, the time for failure detection and distinction, the initiation time for hardware and applications, the restoration time for data and process state, and the Internet Protocol (IP) switching time [87].

3.1.3 MILP problem formulation

In order to get the optimal solution of the proposed primary and backup resource allocation problem with the step function that approximates the given workload-dependent failure probability, this work introduces a mathematical model as follows:

$$\min r \tag{3.3a}$$

$$s.t. \sum_{i\in K}\sum_{b\in B}\sum_{k\in K\setminus\{i\}} x_{ij}^{kb} = 1, \forall j \in V, \tag{3.3b}$$

$$\sum_{b\in B}\sum_{k\in K\setminus\{i\}}\sum_{j\in V} l_j^{\mathrm{R}} x_{ij}^{kb} + \sum_{b\in B}\sum_{i'\in K\setminus\{i\}}\sum_{j\in V} l_j^{\mathrm{R}} x_{i'j}^{ib} \le C_i, \forall i \in K, \tag{3.3c}$$

$$\begin{aligned} r_j = {} & t_1 \sum_{i\in K}\sum_{k\in K\setminus\{i\}} x_{ij}^{k1}\mathcal{P}_i^{\mathrm{S}}(1-\mathcal{P}_k^{\mathrm{S}}) + t_0 \sum_{i\in K}\sum_{k\in K\setminus\{i\}} x_{ij}^{k0}\mathcal{P}_i^{\mathrm{S}}(1-\mathcal{P}_k^{\mathrm{S}}) + \\ & t_3 \sum_{i\in K}\sum_{b\in B}\sum_{k\in K\setminus\{i\}} x_{ij}^{kb}\mathcal{P}_i^{\mathrm{S}}\mathcal{P}_k^{\mathrm{S}}, \forall j \in V, \end{aligned} \tag{3.3d}$$

$$r \ge r_j, \forall j \in V, \tag{3.3e}$$

$$x_{ij}^{kb} \in \{0,1\}, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B. \tag{3.3f}$$

Equation (3.3a) minimizes r, which denotes MEUT among $r_j, j \in V$. Equation (3.3b) ensures that each VM $j \in V$ is allocated into only one PM and protected by another PM with selecting one of the protection strategies. Equation (3.3c) imposes the capacity constraint that the requested load for the primary and backup resources on each PM does not exceed its maximum computing capacity. The first term in the right side of (3.3d) is the expected unavailable time for VM $j \in V$ by applying the HB strategy. This term considers a case that VM $j \in V$ fails and is protected by its corresponding available backup resource whose non-failure probability is $1 - \mathcal{P}_k^{\mathrm{S}}$ with the HB strategy. The second term in the right side of (3.3d) is the expected unavailable time for VM $j \in V$ by applying the CB strategy. This term considers a case that VM $j \in V$ fails and is protected by its corresponding available backup resource in PM $k \in K \backslash \{i\}$ whose non-failure probability is $1 - \mathcal{P}_k^{\mathrm{S}}$ with the CB strategy. The third term in the right side of (3.3d) is the expected unavailable time of a VM in the situation that both two PMs hosting the primary and backup resources of the VM fail simultaneously.

The proposed model can be solved by an MILP approach, where the S-step function in (3.2) can be linearized, as described in Appendix A. A service provider can choose a suitable S considering the accuracy of fitting the practical relationship between the failure probability and the workload in an S-step function, and the computation time to solve the proposed model. To simplify the discussion and demonstration of the basic idea of the proposed model, this work considers the case with $S = 2$ in the following discussions, unless otherwise stated. Section 3.4.4 discusses the effect of different S for approximating a given workload-dependent failure probability in the proposed model.

In the case with $S = 2$, this work uses q_i and T_i as simple forms of $\mathcal{P}_i^2$ and T_i^s, respectively. Equation (3.2) in the case with $S = 2$ can be simplified as follows:

$$q_i = \begin{cases} P_{\mathrm{L}} & 0 \le W_i \le T_i \\ P_{\mathrm{H}} & T_i < W_i \le C_i, \end{cases} \tag{3.4}$$

where P_{L} denotes the given failure probability of each PM $i \in K$ with $0 \le W_i \le T_i$, and P_{H} denotes the given failure probability of each PM $i \in K$ with

$T_i < W_i \leq C_i$, as shown in Fig. 3.1(c).

Equation (3.4) can be linearized to (3.5)-(3.8) by introducing binary variable $y_i, i \in K$. y_i is set to one if $0 \leq W_i \leq T_i$, and zero otherwise, (3.4) can be expressed by:

$$q_i = P_{\mathrm{L}} y_i + P_{\mathrm{H}}(1 - y_i), \forall i \in K, \tag{3.5}$$

$$W_i \leq T_i y_i + C_i(1 - y_i), \forall i \in K, \tag{3.6}$$

$$W_i \geq T_i(1 - y_i), \forall i \in K, \tag{3.7}$$

$$y_i \in \{0, 1\}, \forall i \in K. \tag{3.8}$$

In order to linearize (3.3d), this work introduces binary variables $\phi_{ij}^{kb} = y_i x_{ij}^{kb}, \pi_{ij}^{kb} = y_i y_k x_{ij}^{kb}$, and $\theta_{ij}^{kb} = y_k x_{ij}^{kb}$. Since $\phi_{ij}^{k1}, \pi_{ij}^{k1}, \theta_{ij}^{k1}, \phi_{ij}^{k0}, \pi_{ij}^{k0}, \theta_{ij}^{k0}, \delta_{ij}^{kb}, \zeta_{ij}^{kb}, \eta_{ij}^{kb} \in \{0, 1\}$, $\forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B$, (3.3d) can be linearized as:

$$\begin{aligned} r_j = t_1 \sum_{i \in K} \sum_{k \in K \backslash \{i\}} \{ & (P_{\mathrm{L}} - P_{\mathrm{H}})(1 - P_{\mathrm{H}}) \phi_{ij}^{k1} - \\ & (P_{\mathrm{L}} - P_{\mathrm{H}})^2 \pi_{ij}^{k1} + P_{\mathrm{H}}(P_{\mathrm{H}} - P_{\mathrm{L}}) \theta_{ij}^{k1} + P_{\mathrm{H}}(1 - P_{\mathrm{H}}) x_{ij}^{k1} \} + \\ t_0 \sum_{i \in K} \sum_{k \in K \backslash \{i\}} \{ & (P_{\mathrm{L}} - P_{\mathrm{H}})(1 - P_{\mathrm{H}}) \phi_{ij}^{k0} - \\ & (P_{\mathrm{L}} - P_{\mathrm{H}})^2 \pi_{ij}^{k0} + P_{\mathrm{H}}(P_{\mathrm{H}} - P_{\mathrm{L}}) \theta_{ij}^{k0} + P_{\mathrm{H}}(1 - P_{\mathrm{H}}) x_{ij}^{k0} \} + \\ t_3 \sum_{i \in K} \sum_{b \in B} \sum_{k \in K \backslash \{i\}} \{ & P_{\mathrm{H}}(P_{\mathrm{L}} - P_{\mathrm{H}}) \phi_{ij}^{kb} + (P_{\mathrm{L}} - P_{\mathrm{H}})^2 \pi_{ij}^{kb} + \\ & P_{\mathrm{H}}(P_{\mathrm{L}} - P_{\mathrm{H}}) \theta_{ij}^{kb} + (P_{\mathrm{H}})^2 x_{ij}^{kb} \}, \forall j \in V, \end{aligned} \tag{3.9}$$

$$\phi_{ij}^{kb} \leq y_i, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.10}$$

$$\phi_{ij}^{kb} \leq x_{ij}^{kb}, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.11}$$

$$\phi_{ij}^{kb} \geq y_i + x_{ij}^{kb} - 1, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.12}$$

$$\pi_{ij}^{kb} \leq y_i, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.13}$$

$$\pi_{ij}^{kb} \leq y_k, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.14}$$

$$\pi_{ij}^{kb} \leq x_{ij}^{kb}, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.15}$$

$$\pi_{ij}^{kb} \geq y_i + y_k + x_{ij}^{kb} - 2, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.16}$$

$$\theta_{ij}^{kb} \leq y_k, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.17}$$

$$\theta_{ij}^{kb} \leq x_{ij}^{kb}, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B, \tag{3.18}$$

$$\theta_{ij}^{kb} \geq y_k + x_{ij}^{kb} - 1, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B. \tag{3.19}$$

From the above, this work formulates the problem as an MILP problem, which is expressed by:

$$\min r \tag{3.20a}$$

$$s.t. (3.1a),\ (3.3b)\text{-}(3.3c),\ (3.3e), (3.5) - (3.19), \tag{3.20b}$$

$$y_i \in \{0,1\}, \forall i \in K, \tag{3.20c}$$

$$x_{ij}^{kb}, \phi_{ij}^{kb}, \pi_{ij}^{kb}, \theta_{ij}^{kb} \in \{0,1\}, \forall i \in K, j \in V, k \in K \backslash \{i\}, b \in B. \tag{3.20d}$$

The proposed model is designed to preventively determine the initial allocations of VMs in PMs before services run by considering the unavailable time of PMs caused by the workload-dependent failures and different strategies.

The proposed model can be applied to other scenarios, where failures occur in PMs, and requests for a set of VMs arrive and some of existing VMs are released. The primary and backup resource allocation of existing VMs are collected and can be used as the given parameters for each computation after any change of given conditions, such as failures, arriving VMs, and releasing VMs. After each change of given conditions, the proposed model recomputes the primary and backup resource allocation with the collected primary and backup resource allocation and the updated information of the workloads and requested loads caused by the change of given conditions.

3.2 Problem analysis

In this section, this work analyze a special case for the proposed model with $S = 2$. From (3.3d), we observe that multiple values of MEUT in the proposed model exist, each of which corresponds to a situation. In the case with $S = 2$, q_i and q_k denote the failure probabilities of PMs, each of which is either P_{L} or P_{H}, respectively. Either the first term or the second term in the right hand side of (3.3d) is chosen to calculate the values of the expected unavailable time since each VM is allowed to adopt either HB or CB to provide protection. Different situations have different allocations and strategies, which are determined by

Table 3.1: Rank of MEUTs among five values in different conditions.

Condition	Rank of MEUT among the five values
$t_1 \leq v_1$	$t_1P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_1P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H} < t_1P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H} \leq t_0P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_0P_{\rm H}(1-P_{\rm L})+t_3P_{\rm H}P_{\rm L}$.
$v_1 < t_1 \leq v_2$	$t_1P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_1P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H} \leq t_0P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_1P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H} \leq t_0P_{\rm H}(1-P_{\rm L})+t_3P_{\rm H}P_{\rm L}$.
$v_2 < t_1$	$t_1P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_0P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L} < t_1P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H} < t_0P_{\rm H}(1-P_{\rm L})+t_3P_{\rm H}P_{\rm L} < t_1P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H}$.

the given parameters. With considering the given workload-dependent failure probability, MEUT has eight possible values: $t_1P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L}$, $t_1P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H}$, $t_1P_{\rm H}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm H}$, $t_1P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H}$, $t_0P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L}$, $t_0P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H}$, $t_0P_{\rm H}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm H}$, and $t_0P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H}$.

This work considers that the optimal solution of the proposed model is obtained with considering the step function for approximating the non-decreasing failure probability. Following the analysis in Appendix B.1, several values of the above eight values do not exist when MEUT is minimized. This model has five feasible values with optimal consideration left: $t_1P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L}$, $t_1P_{\rm L}(1-P_{\rm H})+t_3P_{\rm L}P_{\rm H}$, $t_0P_{\rm L}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm L}$, $t_0P_{\rm H}(1-P_{\rm L})+t_3P_{\rm L}P_{\rm H}$, and $t_1P_{\rm H}(1-P_{\rm H})+t_3P_{\rm H}P_{\rm H}$.

The relationship among these five feasible values with optimal consideration depends on the values of t_1, t_0, and t_3. When the total requested load is fixed, the proposed model chooses the HB strategy or the CB strategy due to the different relationships among $t_1, t_0, t_3, P_{\rm L}$, and $P_{\rm H}$. Appendix B.2 clarifies two boundary values, which are: $v_1 = \frac{P_{\rm L}(1-P_{\rm L})}{P_{\rm H}(1-P_{\rm H})}t_0 + \frac{P_{\rm L}^2-P_{\rm H}^2}{P_{\rm H}(1-P_{\rm H})}t_3$ and $v_2 = \frac{1-P_{\rm L}}{1-P_{\rm H}}t_0 + \frac{P_{\rm L}-P_{\rm H}}{1-P_{\rm H}}t_3$, where $v_1 < v_2$.

This work lists the rank of MEUT among the five values in different conditions with two boundary values v_1 and v_2 in Table 3.1. The protection strategies can be switched according to the total requested load so that the larger value of MEUT does not exist when MEUT is minimized. This work analyzes

each value of MEUT as the total requested load increases in different conditions in Appendix B.3. Similar to the analysis in Appendix B, this work can list the possible values of MEUT under the S-step function and list the rank of MEUT among the possible values in different conditions with calculated boundary values.

In order to verify the five feasible possible values of MEUT under the optimal consideration for the proposed model, this work considers comparing several cases with the two approaches which adopt only the HB strategy and only the CB strategy, respectively. Approaches 1 and 2 obtain the primary and backup resource allocation of each VM by minimizing MEUT among VMs with adopting only the HB strategy and only the CB strategy to provide protection, respectively.

Both approaches can be formulated as optimization problems similar to (3.3a)-(3.3c) and (3.3e)-(3.3f). r_j in (3.3d) can be rewritten as (3.21) and (3.22) for approaches 1 and 2, respectively. The optimization problems of approaches 1 and 2 are formulated as MILP problems by fixing $x_{ij}^{k0} = 0$ and $x_{ij}^{k1} = 0$, respectively.

$$r_j = t_1 \sum_{i \in K} \sum_{k \in K \setminus \{i\}} x_{ij}^{k1} q_i (1 - q_k) + t_3 \sum_{i \in K} \sum_{k \in K \setminus \{i\}} x_{ij}^{k1} q_i q_k, \forall j \in V. \tag{3.21}$$

$$r_j = t_0 \sum_{i \in K} \sum_{k \in K \setminus \{i\}} x_{ij}^{k0} q_i (1 - q_k) + t_3 \sum_{i \in K} \sum_{k \in K \setminus \{i\}} x_{ij}^{k0} q_i q_k, \forall j \in V. \tag{3.22}$$

Theorem 3.1 *MEUT of the proposed model with the two-step function is equal to the smaller value between the two MEUTs obtained by approaches 1 and 2 with the same total requested load.*

Proof :There are five ranges for MEUT of the proposed model with an increase of the total requested load in each case. In the first range, MEUT is $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$, which is the smallest value among the five feasible values. It corresponds to the situation that the total requested load is so low that all VMs are protected with the HB strategy with failure probability P_{L}. Approach 1 can achieve the same MEUT of $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ by the same allocation in the optimal solution of the proposed model, where all VMs are protected with the HB strategy.

In the second range, when $t_1 \leq v_2$, MEUT in the proposed model is equal to $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. Note that some VMs may adopt the HB strategy, where the expected unavailable time is $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$. Since, based on the optimal allocation of the proposed model, selecting the HB strategy for all backup resources does not change the value of MEUT, approach 1 can achieve this optimal value of $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. When $t_1 > v_2$, MEUT in the proposed model is equal to $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$. Based on the optimal allocation of the proposed model, selecting the CB strategy for all backup resources does not change the value of MEUT. Thus, approach 2 can achieve this optimal value of $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$.

In the third range, when $t_1 \leq v_1$, MEUT in the proposed model is equal to $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. Similar to the second range, some VMs may adopt the HB strategy, where the expected unavailable time is either $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ or $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. Since, based on the optimal allocation of the proposed model, selecting the HB strategy for all backup resources does not change the value of MEUT, approach 1 can achieve this optimal value of $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. When $t_1 > v_1$, MEUT in the proposed model is equal to $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$. The expected unavailable time of other VMs in the proposed model is either $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ or $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. Similar to the condition of $t_1 \leq v_1$, based on the optimal allocation of the proposed model, selecting the CB strategy for all VMs obtains two values of expected unavailable time: $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ and $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. If the second value exists, there is at least one VM whose backup resource is allocated in a PM with P_{H}. Since the primary resources of all VMs are protected with the CB strategy, this PM with P_{H} hosts the primary resource for at least one VM. It contradicts the condition that MEUT in the proposed model is equal to $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ and the failure probabilities of all PMs hosting the primary resources are P_{L}. Hence, the second value does not exist. Selecting the CB strategy for all backup resources does not change the value of MEUT, approach 2 can achieve this optimal value of $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$.

In the fourth range, the total requested load for primary resource is larger than the sum of thresholds of all PMs. As a result, the failure probability of the primary resource which is protected with the CB strategy is P_{H}. When $t_1 \leq v_2$, MEUT in the proposed model is equal to $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. With

the same reason in the third range, approach 1 can achieve this optimal value of $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. When $t_1 > v_2$, MEUT in the proposed model is equal to $t_0 P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3 P_{\mathrm{H}} P_{\mathrm{L}}$. Note that each of other VMs may adopt either CB or HB, where the expected unavailable time is $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ and either $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ or $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$, respectively. Similar to the reason in the second and third ranges, based on the optimal allocation of the proposed model, selecting the CB strategy for all backup resources does not change the value of MEUT, approach 2 can achieve this optimal value of $t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{H}} P_{\mathrm{L}}$.

In the fifth range, the total requested load is so large that the failure probabilities of both primary and backup resources which are protected with the CB strategy are P_{H}. No matter the relationship between $t_1, t_0, t_3, P_{\mathrm{L}}$, and P_{H}, the proposed model adopts the HB strategy instead of the CB strategy to reduce MEUT, which is equal to $t_1 P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. Approach 1 can achieve this lower optimal value of $t_1 P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3 P_{\mathrm{H}} P_{\mathrm{H}}$, instead of $t_0 P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. Note that each of other VMs may adopt either CB or HB, where the expected unavailable time is either $t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{H}} P_{\mathrm{L}}$ or $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$, respectively. Similar to the reason in the second and third ranges, based on the optimal allocation of the proposed model, selecting the HB strategy for all backup resources does not change the value of MEUT, approach 1 can achieve the optimal value of $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$. □

By adopting Theorem 1, this work can reduce the computation time by adopting approaches 1 and 2 to obtain resource allocations and then finding out the minimum value of MEUT calculated based on the resource allocations. Compared to the proposed model, which directly solving (20a)-(20d), separating the primary and backup resource allocation problem to the two sub-problems can reduce the problem size of each. Here, this work defines an HB sub-problem and a CB sub-problem as the primary and backup resource allocation problem by only adopting approaches 1 and 2, respectively.

3.3 Heuristic algorithm

As clarified in [44], the decision version of primary and backup allocation problem is an NP-complete problem by reducing the number partition problem,

which is a well-known NP-complete problem. It indicates that the primary and backup resource allocation problem becomes difficult to solve as the problem size increases. As clarified in section 3.2, the primary and backup resource allocation problem can be simplified by adopting approaches 1 and 2. However, even the two sub-problems remain to be NP-complete. Without the different strategies among the VMs to balance the workload and the recovery time, the problem size of each sub-problem is reduced by only considering the primary and backup resource allocation problem.

This section presents a main idea of heuristic algorithm design which obtains an approximate solution of the resource allocation to minimize MEUT with considering the workload to reduce the computation time. Thus, this work can solve a problem with a faster algorithm than the MILP approach by sacrificing proper optimality.

Gallager et al. in [91] introduced the idea of water filling, which has been applied to various areas in communications and signal processing. The water-filling algorithm is a pseudo-polynomial [92] algorithm to approximately solve the problem. Inspired by the work in [77], which adopts the water-filling algorithm for load balancing to minimize the maximum load of links and nodes, the developed heuristic algorithm determines the resource allocation by allocating VMs on PM $i \in K$ iteratively until the workload for the PM exceeds threshold T_i with the allocation of the next VM. Then, this work continues to allocate VMs to the next PM until the workload exceeds the corresponding threshold with the allocation of the next VM. When the allocation of the next VM causes the workload of every PM to exceed its corresponding threshold, the algorithm allocates VMs to the first visited PM iteratively until the requested load for the PM exceeds the corresponding capacity with allocation of the next VM. Then, this work continues to allocate VMs to the next PM until the requested load for the PM exceeds the corresponding capacity with the allocation of the next VM. If there is any PM with the requested load larger than the capacity, the developed algorithm takes the two sub-problems with given parameters as infeasible.

Let W_i^b and R_i^b denote the workload and the requested load of PM when it is protected and it protects other PMs with strategy $b \in \{0, 1\}$, respectively. By fixing $x_{ij}^{k0} = 0$ and $x_{ij}^{k1} = 0$, respectively, the workload and the requested

load of PM in the HB and CB sub-problems can be expressed by:

$$W_i^1 = \sum_{j\in V}\sum_{k\in K\setminus\{i\}} l_j^{\mathrm{R}} x_{ij}^{k1} + \sum_{i'\in K\setminus\{i\}}\sum_{j\in V} l_j^{\mathrm{R}} x_{i'j}^{i1}, \forall i \in K, \tag{3.23a}$$

$$R_i^1 = \sum_{j\in V}\sum_{k\in K\setminus\{i\}} l_j^{\mathrm{R}} x_{ij}^{k1} + \sum_{i'\in K\setminus\{i\}}\sum_{j\in V} l_j^{\mathrm{R}} x_{i'j}^{i1}, \forall i \in K, \tag{3.23b}$$

$$W_i^0 = \sum_{j\in V}\sum_{k\in K\setminus\{i\}} l_j^{\mathrm{R}} x_{ij}^{k0}, \forall i \in K, \tag{3.23c}$$

$$R_i^0 = \sum_{j\in V}\sum_{k\in K\setminus\{i\}} l_j^{\mathrm{R}} x_{ij}^{k0} + \sum_{i'\in K\setminus\{i\}}\sum_{j\in V} l_j^{\mathrm{R}} x_{i'j}^{i0}, \forall i \in K. \tag{3.23d}$$

Algorithm 3.1 solves both HB and CB sub-problems by fixing $b = 1$ and $b = 0$, respectively, and it sorts i and j by T_i and l_j^{R} decreasingly, respectively, in advance (line 2). Algorithm 3.1 handles two situations. In the first situation, the total requested load is lower than $\sum_{i\in K} T_i/2$ (lines 2-12). A VM is allocated to PM $i = k \bmod |K|+1$ and protected by PM $k = i \bmod |K|+1$, if $W_i^b + l_j^{\mathrm{R}} \le T_i$, $W_k^b + l_j^{\mathrm{R}} \le T_k$, $R_i^b + l_j^{\mathrm{R}} \le C_i$, and $R_k^b + l_j^{\mathrm{R}} \le C_k$ (lines 4-10). Algorithm 3.1 processes until all VMs are allocated and protected. If the next VM causes every PM to exceed the threshold, Algorithm 3.1 traverses the PMs to search for PMs whose remaining capacities are equal to or larger than the requested load of the VM (line 12). Algorithm 3.1 deals with the allocation and backup protection of each VM individually by adopting different PMs in the first situation.

The second situation is when $\sum_{j'\in V} l_{j'}^{\mathrm{R}} \ge \sum_{i\in K} T_i/2$ (lines 13-41). VMs are allocated to PM i and protected by PM $k = (i + 1 + \lfloor \frac{i}{|K|} \rfloor) \bmod |K|$, iteratively, if $W_i^b + l_j^{\mathrm{R}} \le T_i$, $W_k^b + l_j^{\mathrm{R}} \le T_k$,$R_i^b + l_j^{\mathrm{R}} \le C_i$, and $R_k^b + l_j^{\mathrm{R}} \le C_k$ (lines 14-26). Algorithm 3.1 does not change primary and backup resource allocation until the allocation of next VM j causes $W_i^b + l_j^{\mathrm{R}} > T_i$, $W_k^b + l_j^{\mathrm{R}} > T_k$, $W_k^b + l_j^{\mathrm{R}} \le T_k$, or $R_k^b + l_j^{\mathrm{R}} \le C_k$ (lines 32-34).

Since $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}} < t_1 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$, if the next VM causes the workload of every PM to exceed the corresponding threshold, the algorithm allocates the VM to a PM whose capacity is only used to allocate backup resources. This PM works as a backup PM with failure probability P_{H} because of the backup allocation of the VM (line 14). The PM $i = 2$, which is

only used as backup resource for VMs which are allocated to the PM $i = 1$, is used to accept the backup resource of the VMs in the last PM $i = |K|$. As a result, Algorithm 3.1 may avoid the larger MEUT.

Algorithm 3.1 Developed heuristic algorithm

Input: $K, V, T_i, C_i, l_j^{\mathrm{R}}, \forall i \in K, j \in V, s$

Output: x_{ij}^{kb}, f_{flag}

1: $x_{ij}^{kb} = 0$, $f_{\mathrm{flag}} = 1, k = 0$, $j = j_{\mathrm{stop1}} = j_{\mathrm{stop2}} = 1$
2: Sort i and j by T_i and l_j^{R} decreasingly, respectively
3: **if** $\sum_{j' \in V} l_{j'}^{\mathrm{R}} < \sum_{i \in K} T_i/2$ **then**
4: **repeat**
5: $i = k \bmod |K| + 1$
6: $k = i \bmod |K| + 1$
7: Calculate W_i^b, W_k^b, R_i^b, and R_k^b
8: **if** $W_i^b + l_j^{\mathrm{R}} \le T_i$ and $W_k^b + l_j^{\mathrm{R}} \le T_k$ and $R_i^b + l_j^{\mathrm{R}} \le C_i$ and $R_k^b + l_j^{\mathrm{R}} \le C_k$ **then**
9: $x_{ij}^{kb} \leftarrow 1$, $j_{\mathrm{stop1}}, j \leftarrow j + 1$
10: **end if**
11: **until** $j = |V|$ or $W_i^b + l_j^{\mathrm{R}} > T_i, \forall i \in K$
12: TRANSVERSE$(j_{\mathrm{stop1}}, |K|)$
13: **else**
14: **for** $i = 1 \rightarrow |K|$ **do**
15: $k = (i + 1 + \lfloor \frac{i}{|K|} \rfloor) \bmod |K|$
16: **for** $j = j_{\mathrm{stop2}} \rightarrow |V|$ **do**
17: Calculate W_i^b, W_k^b, R_i^b, and R_k^b
18: **if** $W_i^b + l_j^{\mathrm{R}} > T_i$ or $R_i^b + l_j^{\mathrm{R}} > C_i$ or $W_k^b + l_j^{\mathrm{R}} > T_i$ or $R_k^b + l_j^{\mathrm{R}} > C_i$ **then**
19: Break
20: **end if**
21: $x_{ij}^{kb} \leftarrow 1$, $j_{\mathrm{stop2}} \leftarrow j + 1$
22: **if** $j = |V|$ **then**
23: Exit
24: **end if**
25: **end for**
26: **end for**

```
    REALLOCATION(j_stop2, |K|)
    for i = 1 → |K| − 1 do
        k = i + 1
        for j = j_stop2 → |V| do
            Calculate R_i^b
            if R_i^b + l_j^R > C_i or R_k^b + l_j^R > C_k then
                Break
            end if
            x_ij^kb ← 1, j_stop2 ← j + 1
            if j = |V| then
                Exit
            end if
        end for
    end for
    TRANSVERSE(j_stop2, |K|)
end if
```

Algorithm 3.2 Reallocation

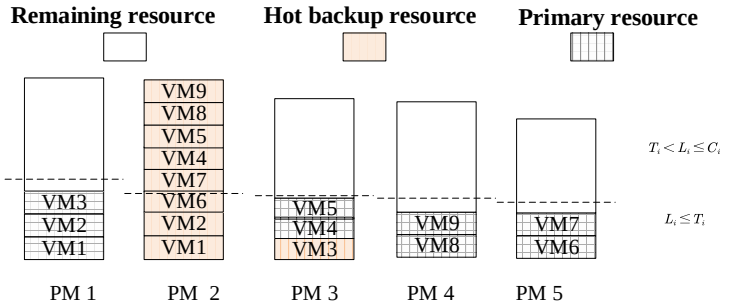
```
function REALLOCATION(j_stop, K)
    K_E ← set of PMs with even numbers in K in an increasing order by
    their labels
    K'_E ← set of PMs with even numbers in K in a decreasing order by
    their labels
    for k' ∈ K'_E do
        Define N' as the set of backuped VMs in PM k',
        j'_stop ← 1
        for k ∈ {k|k ∈ K_E, k < k'} do
            for j' = j'_stop → |N'| do
                if W_k + l_j'^R > C_k then
                    Break
                end if
                VM j' which is protected by PM k'
                 moves from PM k' to PM k,
                W_k ← W_k + l_j'^R, j'_stop ← j' + 1
                if W_k + l_j_stop^R > C_k then
```

14: Break
15: **end if**
16: $x^{kb}_{k'j_{\text{stop}}} \leftarrow 1, j_{\text{stop}} \leftarrow j_{\text{stop}} + 1$
17: **end for**
18: **end for**
19: **end for**
20: **return** x^{kb}_{ij}, j_{stop}
21: **end function**

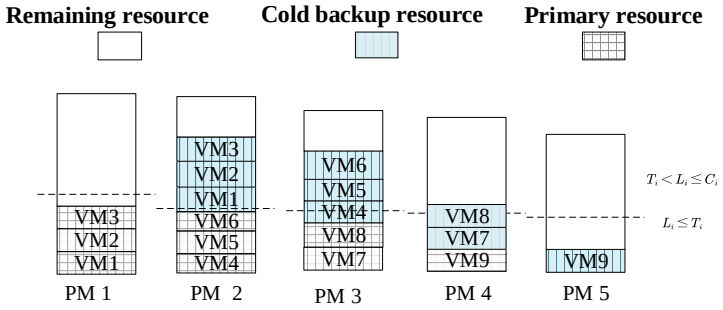
Algorithm 3.3 Transverse

1: **function** TRANSVERSE($j_{\text{stop}}, |K|$)
2: **for** $i = 1 \rightarrow |K| - 1$ **do**
3: **for** $k = i + 1 \rightarrow |K|$ **do**
4: **if** $C_i - R^b_i \geq l^{\text{R}}_{j_{\text{stop}}}$ and $C_k - R^b_k \geq l^{\text{R}}_{j_{\text{stop}}}$ **then**
5: $x^{kb}_{ij_{\text{stop}}} \leftarrow 1$, $j_{\text{stop}} \leftarrow j_{\text{stop}} + 1$
6: **end if**
7: **end for**
8: **end for**
9: **if** $j_{\text{stop}} \neq |V| + 1$ **then**
10: $f_{\text{flag}} = 0$
11: **end if**
12: **return** $x^{kb}_{ij}, f_{\text{flag}}$
13: **end function**

When the allocation of a VM causes the workload of every PM to exceed its threshold and further leads to failure probability P_{H} of a PM, allocating the VM to a PM whose capacity is only used to allocate backup resources is better than allocating the VM to a PM whose capacity is used to allocate both backup and primary resources since $t_1 P_{\text{H}}(1-P_{\text{L}}) + t_3 P_{\text{L}} P_{\text{H}} > t_1 P_{\text{L}}(1-P_{\text{H}}) + t_3 P_{\text{L}} P_{\text{H}}$. It is hard for the developed water-filling algorithm to preset PMs with a suitable number to allocate only backup resources, so this work defines Algorithm 3.2 as a reallocation function and define a set of PMs $i = 2n, n \in [1, \lfloor |K|/2 \rfloor]$, which are PMs with even numbers in K. Let K_{E} and K'_{E} denote sets of even numbers in K in an increasing order and a decreasing order by their labels, respectively (lines 2-3). Algorithm 3.2 moves the VMs protected by PM in K'_{E}



(a) Example for sub-problem only adopting HB strategy.



(b) Example for sub-problem only adopting CB strategy.

Figure 3.2: Examples of developed heuristic algorithm.

to PMs in K_{E} when the allocation of the next VM causes the workload of each PM to exceed the threshold (lines 4-12). After the reallocation, PM $k' \in K'_{\mathrm{E}}$ has enough capacity for primary resource allocation of new VMs, VMs are allocated for the primary and backup resources on PMs $k' \in K'_{\mathrm{E}}$ and $k \in K_{\mathrm{E}}$, respectively (lines 13-17). As a result, the VMs can be protected iteractively by the PM with failure probability P_{H} while primary resources can be allocated to the PM with failure probability P_{L}.

Similar to lines 14-26 in Algorithm 3.1, if the next VM causes the workload of every PM to exceed the corresponding threshold, Algorithm 3.1 allocates the primary and backup resources of each VM iteratively on the same PMs until the allocation of the next VM j causes $R_i^b + l_j^{\mathrm{R}} > C_i$ or $R_k^b + l_j^{\mathrm{R}} > C_k$ (lines 28-40).

If remaining VMs are waiting to be allocated after iterations of allocation, Algorithm 3.3 traverses PMs for all remaining VMs to search for PMs, each remaining capacity of which is equal to or larger than the requested load of

a remaining VM until there is a VM that cannot find a feasible allocation (lines 2-8). If at least a VM cannot find a feasible allocation, the solution of sub-problem is infeasible (lines 9-11).

The computation time complexities of sorting $|K|$ PMs and sorting $|V|$ VMs are $O(|K|\log|K|)$ and $O(|V|\log|V|)$, respectively. The computation time complexities of Algorithms 3.2 and 3.3 are $O(|V||K|^2)$ and $O(|K|^2)$, respectively. The computation time complexities of lines 14-26 and lines 28-40 in Algorithm 3.1 are $O(|K|^2)$ and $O(|V||K|)$, respectively. Calling Algorithms 3.2 and 3.3 in Algorithm 3.1, the overall computation time complexity is $O(|V||K|^2 + |V|\log|V|)$.

Figure 3.3 shows examples of the developed heuristic algorithm by focusing both HB and CB sub-problems with five PMs sorted by T_i decreasingly. Figure 3.2(a) shows the example for the HB sub-problem. VMs 1, 2, and 3 are allocated to PM 1 until VM 4 causes the exceeding of T_1. VMs 1 and 2 are protected by PM 2 until VM 3 causes the exceeding of t_0. For PM 3, VM 3 is protected by PM 3 because of the limited threshold for PM 2; VMs 4 and 5 are allocated to PM 3 until VM 4 causes the exceeding of T_4. VMs 4 and 5 are protected by PM 4 until VM 6 causes the exceeding of T_4. VMs 6 and 7 are allocated by PM 5 and protected by PM 2 until VM 8 causes the exceeding of T_5. When the allocation of VM 8 causes the workload of every PM to exceed its threshold, Algorithm 2 moves VMs 4 and 5, which are protected by PM 4 originally, to PM 2. As the result of the reallocation, VMs 8 and 9 are allocated for the primary and backup resources on PMs 4 and 2, respectively.

Similarly, Fig. 3.2(b) shows the example for the CB sub-problem. VMs 1, 2, and 3 are allocated to PM 1 until VM 4 causes the exceeding of T_1. Since resources for CB strategy are not being activated, PM 2 still has remaining resource for primary resources. VMs 4, 5, and 6 are allocated to PM 2, iteratively, until VM 7 causes the exceeding of t_0. For PM 3, VMs 4, 5, and 6 are protected by PM 3; VMs 7 and 8 are allocated to PM 3 until the allocation of VM 9 causes the exceeding of C_3. For PM 4, VM 7 and 8 is protected by PM 4; VM 9 is allocated to PM 4 while protected by PM 5.

Algorithm 3.1 consider the case with $S = 2$ to simply the clarifications. In order to increase the accuracy of the step function to approximate the given workload-dependent failure probability, an S-step function, where $S > 2$,

Table 3.2: Expected unavailable time and primary backup resource allocation of each VM.

	VM1	VM2	VM3	VM4
r_j [s]	0.01725	0.01725	0.01725	0.0754
(i,k,b)	(3,1,1)	(3,1,1)	(2,1,1)	(3,2,0)
	VM5	VM6	VM7	VM8
r_j [s]	0.0754	0.0754	0.01725	0.01725
(i,k,b)	(2,3,0)	(2,3,0)	(3,1,1)	(3,1,1)

can be extended from the two-step workload-dependent failure probability to conservatively approximate a non-decreasing function of the workload. By processing lines 27-41 in Algorithm 3.1 repeatedly for S times, the heuristic algorithm can be extended to solve the proposed model with an S-step failure probability.

3.4 Numerical evaluations

First, this work shows a demonstration of the proposed model to observe the basic characteristics. Second, this work investigates the dependency of MEUT on the recovery time. Third, this work evaluates the computation time of the MILP approach and the heuristic algorithm and the accuracy of the latter. Fourth, this work investigates the effect of the different numbers of steps for approximating a given workload-dependent failure probability. The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.7.1 [93]. Both MILP problems and heuristic algorithm are coded by Python 3.7, using Intel Core i7-7700 3.60 GHz 4-core CPU with 32 GB memory.

3.4.1 Demonstration

This work presents a demonstration of the proposed model to observe the basic characteristics. As shown in Fig. 3.4.1, this work has three PMs whose capacities are 15, 20, and 25, and whose thresholds are 9, 12, and 15, respectively. This work considers eight VMs, with requested loads of 2, 2, 3, 3, 4, 4, 4, and

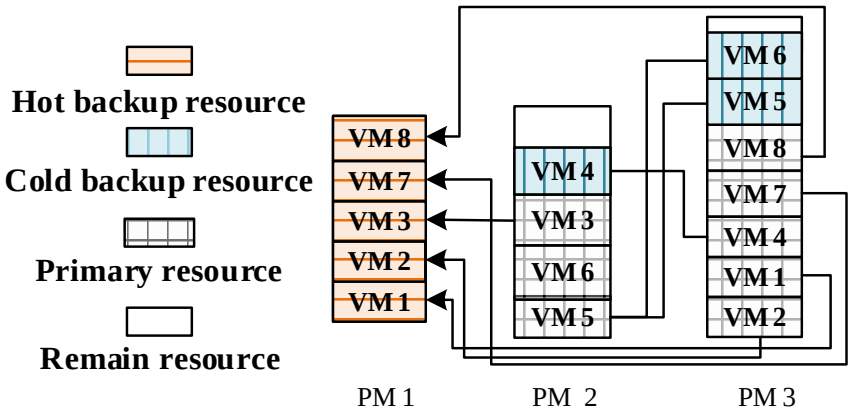


Figure 3.3: Demonstration of primary and backup resource allocation.

4, respectively. This work sets P_{L} and P_{H} to 0.0025 and 0.02, and unavailable time t_1, t_0, and t_3 to 5 [s], 30 [s], and 100 [s], respectively.

Table 3.2 shows the expected unavailable time and the primary and backup resource allocation yielded by the proposed model. Triplet (i, k, b) of each VM $j \in V$ denotes the resource allocation result obtained by the proposed model. It represents that VM $j \in V$ is allocated at PM $i \in K$ and protected by PM $k \in K \backslash \{i\}$ with strategy $b \in B$. Expected unavailable time r_j for each VM $j \in V$ is shown in Table 3.2 and MEUT is 0.0754 [s]. Figure 3 shows the primary and backup resource allocation for three PMs with $L_1 = 15$, $L_2 = 11$, and $L_3 = 15$. The primary and backup resources are allocated into three PMs and protected with either the CB or the HB strategy. The computation time to solve the MILP problem in (3.20b)- (3.20d) is 0.65 [s].

Then, this work shows the effectiveness of proposed model by comparing it with the conventional model. Each backup resource of a VM is allowed to adopt either HB or CB to provide protection in the conventional model. Regardless of a workload-dependent failure probability, failure probability in the conventional model is set to P_{C} since this work cannot judge the failure probabilities with different workloads. With the constant failure probability, (3.3b)-(3.3f) are in the linear forms. The optimization problem of conventional

model is formulated as an MILP problem:

$$\min\ r \tag{3.24a}$$

$$s.t.\ q_i = P_{\mathrm{C}}, \forall i \in K, \tag{3.24b}$$

$$\text{(3.3b)-(3.3f).} \tag{3.24c}$$

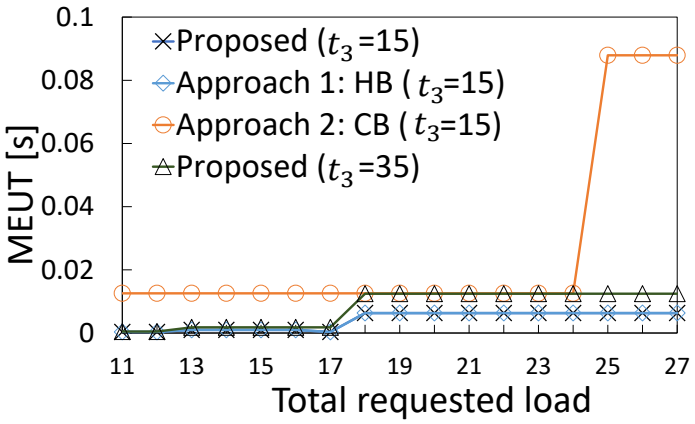
In the conventional model, adopting the CB strategy increases unavailable time without decreasing of the failure probability. Therefore, the conventional model adopts the HB strategy no matter what the exact value of P_{C} is. Solving the MILP problem in (3.24a)-(3.24c), the workloads of PMs obtained by the conventional model are $L_1 = 9$, $L_2 = 19$, and $L_3 = 24$ and MEUT is 0.137 [s]. Compared with the conventional model, the proposed model balances the recovery time and the workload by considering the workload-dependent failure probability.

3.4.2 Dependency of MEUT on recovery time

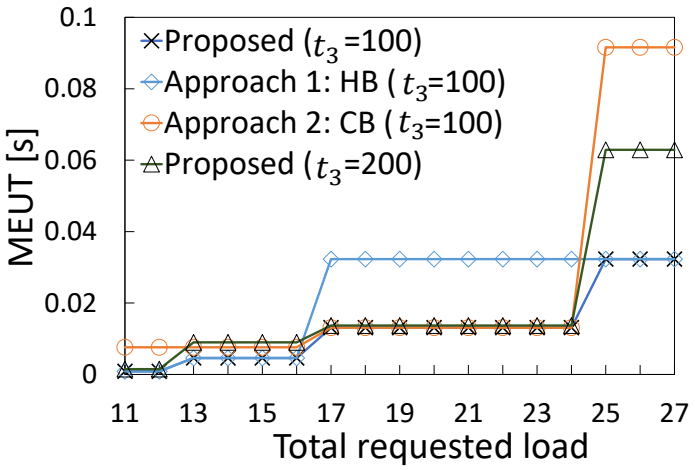
As shown in (3.3d), t_1, t_0, and t_3 affect MEUT. In this subsection, this work evaluates the dependency of MEUT on the recovery time and present a detailed analysis of the results.

As described in Appendix B.2, there are two boundary values revealing the relationship of t_1, t_0, and t_3 and MEUT. Each boundary value divides the rank of MEUTs into different situations, as shown in Table II. There are two boundary values for t_3, $\omega_1 = \frac{P_{\mathrm{H}}(1-P_{\mathrm{H}})}{P_{\mathrm{L}}^2-P_{\mathrm{H}}^2}t_1 - \frac{P_{\mathrm{L}}(1-P_{\mathrm{L}})}{P_{\mathrm{L}}^2-P_{\mathrm{H}}^2}t_0$, $\omega_2 = \frac{1-P_{\mathrm{H}}}{P_{\mathrm{L}}-P_{\mathrm{H}}}t_1 + \frac{1-P_{\mathrm{L}}}{P_{\mathrm{L}}-P_{\mathrm{H}}}t_0$. When $t_3 \leq \omega_1$, this work calls the situation Case 1. Similarly, Case 2 corresponds to the situation with $\omega_1 < t_3 \leq \omega_2$; Case 3 corresponds to the situation with $\omega_2 < t_3$.

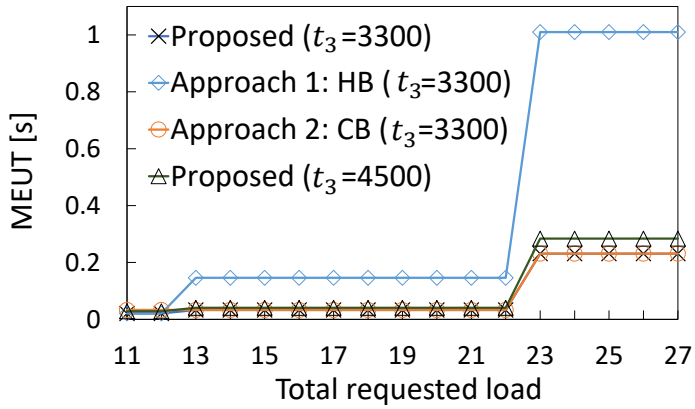
This work shows the relationship among t_1, t_0, and t_3 on MEUT in different cases; P_{H} and P_{L} are set to 0.0175 and 0.0025, respectively, as an example. First, the two boundary values of t_3 are related to t_1 and t_0, for example, when $t_0 = 10t_1$, $\omega_1 \approx 2.5t_0$ and $\omega_2 \approx 60t_0$. Second, with the assumption of $t_3 \geq t_0$, the existence of Case 1 is also related to $\frac{t_0}{t_1}$. If $\frac{t_0}{t_1} < \frac{P_{\mathrm{H}}(1-P_{\mathrm{H}})}{P_{\mathrm{L}}-P_{\mathrm{H}}^2}$, Case 1 does not exist. Similar to that, if $t_3 < \frac{(1-P_{\mathrm{L}})t_0-(1-P_{\mathrm{H}})t_1}{P_{\mathrm{H}}-P_{\mathrm{L}}}$, Case 3 does not exist. Inserting the values of P_{H} and P_{L} into the boundary values, we observe that, unless t_0



(a) Case 1 ($5 \leq t_3 \leq 35.8$).



(b) Case 2 ($35.8 < t_3 \leq 3259.5$).



(c) Case 3 ($3259.5 < t_3$).

Figure 3.4: Comparison between proposed and two approaches with different t_3 [s] with t_1=0.1 [s], t_0=5 [s].

is at least 7.84 times larger than t_1, Case 1 does not exist with the assumption of $t_3 \geq t_0$; unless t_3 is larger than $65.5(t_0 - t_1)$, Case 3 does not exist.

This work considers three PMs whose capacities and thresholds are set to 20 and 8, respectively. Both models are adopted with eight VMs, whose requested loads are uniformly distributed over the range of [1, 4]. This work sets $P_{\mathrm{H}} = 0.0175$ and $P_{\mathrm{L}} = 0.0025$ for all cases. Cases 1, 2, and 3 have the same unavailable time t_1 and t_0, which are set to 0.1 [s] and 5 [s], respectively. t_3 in each case is set to two values which are in the range of [5, 35.8] [s], (35.8, 3259.5] [s], and (3259.5,10000] [s] for cases 1, 2, and 3, respectively.

Figure 3.4 shows MEUTs obtained by the proposed model and the two approaches for different values of the total requested load in the three cases corresponding to the two boundary values with different t_3. Figure 3.4 observes that MEUT yielded by the proposed model is equal to the smaller value between the two MEUTs obtained by approaches 1 and 2 in each case. We observe that there are several values of MEUT for the proposed model. As shown in Figs. 3.4, when the total requested load is smaller than 12, MEUT is $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$, which is the smallest value among the five feasible values described in Appendix. B.1. In the second range, there are two values of MEUT in the proposed model, which are $t_1 P_{\mathrm{L}}(1-P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$ in Figs. 3.4(a) and (b) and $t_0 P_{\mathrm{L}}(1-P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ in Fig. 3.4(c). In the third range, there are two values of MEUT in the proposed model, which are $t_1 P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3 P_{\mathrm{H}} P_{\mathrm{H}}$ in Figs. 3.4(a) and (c) and $t_0 P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3 P_{\mathrm{L}} P_{\mathrm{L}}$ in Fig. 3.4(c). In the fourth range, the total requested load for primary resource is larger than the threshold. As a result, the failure probability of the primary resource which is protected with the CB strategy is P_{H}. The values of MEUT in Figs. 3.4(a) and (c) do not change, which are the same with those in the third range, respectively. The value of MEUT in Fig. 3.4(b) is $t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{H}} P_{\mathrm{L}}$.

MEUTs derived by the larger value of t_3 are on average 104.80%, 49.44%, and 24.38% larger than those of the smaller value of t_3 in the three cases among all the total requested loads, respectively. In Fig. 3.4(a), the difference of MEUTs obtained by the larger and smaller values of t_3 in the third range is larger than those of the first and the second ranges. This is because coefficient of t_3 in the third range is $P_{\mathrm{H}} P_{\mathrm{H}}$, while those in the first and second ranges are $P_{\mathrm{L}} P_{\mathrm{H}}$ and $P_{\mathrm{L}} P_{\mathrm{H}}$, respectively. Similar to that, in Fig. 3.4(b), the difference

Table 3.3: MEUTs and computation time of MILP approach and heuristic algorithm.

Situation	Test	MEUT (s)		Computation time (s)				τ_{MILP}	τ_{sub}
		MILP	Heuristic	MILP (multiple strategies)	MILP (HB-sub)	MILP (CB-sub)	Heuristic		
$t_1 \le v_1$	1	0.00025	0.00025	13.41	4.76	3.64	0.001	1.34×10^4	8.40×10^3
	2	0.00099	0.00099	120.15	39.81	8.03	0.002	6.01×10^4	2.39×10^4
	3	0.00690	0.00690	250.31	149.15	10.04	0.004	6.26×10^5	3.98×10^4
	4	0.00099	0.0069	2234.53	1584.07	14.19	0.008	2.79×10^6	1.99×10^5
	5	0.00690	0.00690	489654.61	332687.67	25.10	0.010	4.89×10^7	3.32×10^7
$v_1 < t_1 \le v_2$	1	0.00137	0.00137	18.83	3.52	3.69	0.001	1.88×10^4	7.21×10^3
	2	0.00210	0.00210	210.47	107.07	7.28	0.002	1.05×10^5	5.71×10^4
	3	0.01470	0.01470	6431.36	836.86	16.55	0.004	1.60×10^6	2.13×10^5
	4	0.00761	0.00761	79495.58	20279.01	58.50	0.006	1.32×10^7	3.39×10^6
	5	0.00761	0.00761	603793.13	120655.24	133.68	0.010	6.04×10^7	1.21×10^7
$v_2 < t_1$	1	0.00710	0.00710	7.87	1.17	5.31	0.001	7.87×10^3	6.48×10^3
	2	0.00761	0.00761	84.81	15.99	5.44	0.003	2.82×10^4	7.14×10^3
	3	0.00761	0.00761	193.35	64.22	9.84	0.005	3.87×10^4	1.48×10^4
	4	0.00761	0.00761	310.32	120.18	11.88	0.008	3.88×10^4	1.65×10^4
	5	0.00761	0.00761	681.58	139.25	26.18	0.010	4.87×10^4	1.66×10^4

of MEUTs obtained by the larger and smaller values of t_3 in the fourth range is larger than those of the first, second, and third ranges. In Fig. 3.4(c), the difference of MEUTs obtained by the larger and smaller values of t_3 in the third range is larger than those of the first and second ranges.

3.4.3 Computation time and accuracy

This work shows the comparison on MEUT and computation time between the MILP approach and the heuristic algorithm among all the total requested loads and for different numbers of VMs.

Consider three PMs whose capacities and thresholds are set to 20 and 8, respectively. The proposed models are adopted with VMs whose requested loads are randomly distributed over the range of [1, 4]. This work sets $P_{\text{H}} = 0.0175$ and $P_{\text{L}} = 0.0025$ for all cases. Cases 4, 5, and 6 have the same unavailable time t_0 and t_3, which are set to 3 and 20, respectively. t_1 is set to 0.05 [s], 0.5 [s], and 2.8 [s] for Cases 4, 5, and 6, respectively. This work solves the MILP problem in (3.20b)-(3.20d) and solve the two sub-problems by minimizing (3.21) and (3.22), separately. This work also solves the same problem by

using Algorithm 3.1. We observe that with different computation time, the results from the heuristic algorithm are equal to those from the MILP approach, approximately. We observe that the heuristic algorithm reduces the computation time by 95.35% compared with the MILP approach on average.

Consider that the primary and backup resource allocation solved by the MILP approach introduced in (3.20b)-(3.20d) cannot be solved in a practical time when the number of PMs and VMs become large. This evaluation performs the tests for different sizes of VMs to investigate the computation time for the heuristic algorithm compared with the MILP approach to solve the proposed model by applying multiple strategies, only the HB strategy, and only the CB strategy, respectively.

t_1 is set to 0.05 [s], 0.5 [s], and 2.8 [s] for the three situations, respectively. Tests 1-5 for each situation corresponds to 10 VMs, 20 VMs, 30 VMs, 40 VMs, 50 VMs, respectively. The requested loads of VMs are randomly distributed over the range of [1, 4]. This evaluation considers six PMs whose capacities and thresholds are set to 20 and 8, 40 and 16, 60 and 24, 80 and 32, and 100 and 40, respectively, in each test.

Table 3.3 shows the average computation time for the MILP approach and the heuristic algorithm for each situation. There are four columns in computation time which indicate those to solve the proposed model by applying the MILP approach and the heuristic algorithm. The first column is the computation time to solve (3.20b)-(3.20d) with multiple strategies; the second and third columns are the computation time to solve (3.20b)-(3.20d), by fixing $x_{ij}^{k0} = 0$ and $x_{ij}^{k1} = 0$, respectively. r_j in (3.3d) can be rewritten as (3.21) and (3.22) for the HB and CB sub-problems, respectively. The fourth column is the computation time to solve the proposed model with the heuristic algorithm. τ_{MILP} denotes the ratio of the computation time of the proposed model with the MILP approach to that of the heuristic algorithm; τ_{sub} denotes the ratio of the sum of computation time to solve the HB and CB sub-problems with the MILP approach to that of the heuristic algorithm.

We observe that the total computation time to solve the HB and CB sub-problems by the MILP approach separately is smaller than that with applying multiple strategies to solve (3.20b)-(3.20d). We observe that the computation time of the heuristic algorithm is 2.69×10^{-5} times smaller than that of the

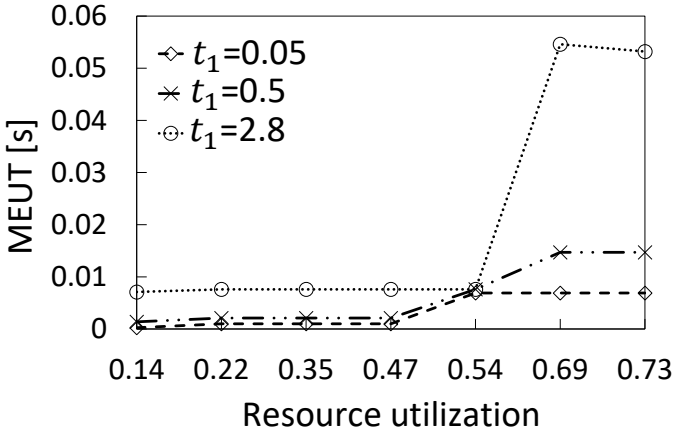


Figure 3.5: Dependency of MEUT on resource utilization.

MILP approach on average; as the problem size increases, the computation time of the MILP approach increases significantly. The larger the problem size is, the more the computation time of the heuristic algorithm is reduced compared with that of the MILP approach. MEUT calculated by the MILP approach ia 5.71% smaller than the result derived by the heuristic algorithm on average among 15 tests.

In order to evaluate the performance of the heuristic algorithm for a larger-size problem, this evaluation considers that there are 200 PMs hosting 1000 VMs. The capacity of 200 PMs are randomly distributed over the range of [50, 100]. The ratios of the threshold to the capacity of a PM are randomly distributed over the range of [0.4, 0.8]. The requested loads of 1000 VMs are randomly distributed over the range of [1, 20].

Resource utilization denotes the ratio of the total requested load of VMs to the total capacity for all PMs. Figure 3.5 shows the relationship between MEUTs and resource utilization under three situations by applying the heuristic algorithm. The computation time to solve the problem by the heuristic algorithm among different resource capacities and thresholds utilizations under different situations are almost the same and are approximately 105 [s].

3.4.4 *S*-step failure probability

The step function can be extended to be an S-step function, where $S \geq 2$, to conservatively approximate a non-decreasing function of the workload with higher accuracy. This evaluation investigates the difference between the two-step and the S-step failure probabilities, where $S > 2$.

Given a workload-dependent failure probability, this work uses step functions with different numbers of steps to approximate it. To conservatively approximate with the workload-dependent failure probability with the two-step failure probability, the capacities and thresholds of PMs in the proposed model are 50 and 25, respectively. The failure probabilities corresponding to the two ranges of capacity are 0.0025 and 0.0175. To improve the accuracy of fitting the given workload-dependent failure probability with the step function, this evaluation sets $S = 4$ and 8 for the S-step failure probabilities in addition to $S = 2$ so that a failure probability with S_1 can conservatively approximate that of S_2, where $S_1 < S_2$. To approximate the workload-dependent failure probability by a four-step failure probability, T_i^1 to T_i^4 of PMs in the proposed model are set to 12, 25, 37, and 50, respectively; each range corresponds to the failure probability of 0.00125, 0.0025, 0.01, and 0.0175, respectively. To approximate the workload-dependent failure probability by a eight-step function failure probability, T_i^1 to T_i^8 of PMs in the proposed model are set to 6, 12, 18, 25, 31, 37, 43, and 50, respectively; each range corresponds to the failure probability of 0.000625, 0.00125, 0.001875, 0.0025, 0.00625, 0.01, 0.01375, and 0.0175, respectively. This work considers that there are 20 PMs hosting 100 VMs. The requested loads of 100 VMs are randomly distributed over the range of [1, 20]. Figure 3.6 shows the comparison of MEUTs between a two-step failure probability with an S-step failure probability, where S=4 and 8. We observe a more detailed and smooth increasing tendency of MEUT with an increase of resource utilization since PMs have more ranges of capacity which corresponds to different failure probabilities to allocate the primary and backup resources with the S-step workload-dependent failure probability. MEUTs obtained by the proposed model based on the workload-dependent failure probability with S=4 and S=8 are on average 42.38% and 59.77%, respectively, smaller than those of S=2, among the values of resource utilization. By adjusting the width

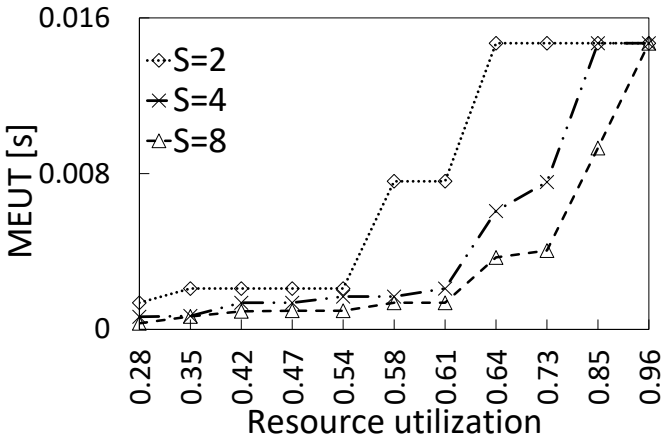


Figure 3.6: Comparison of MEUTs obtained with S-step workload-dependent failure probabilities, where S= 2, 4, and 8.

of the S-step failure probability, the proposed model can modify different failure patterns to meet the requirement of service providers with comprehensively considering the influence of system resource utilization, accumulation of errors on failure probability and other factors which affect the failure patterns.

3.5 Discussion on impact of approximation of non-decreasing function

3.5.1 Baseline model

Given the non-decreasing workload-dependent failure probability $f(w)$, which can be performed by collecting and analyzing the workload-related failure characteristics of each server, this work considers that step function $s(w)$ fits the curve of $f(w)$ with minimizing $\int_0^{C_i} (s(w) - f(w))\,\mathrm{d}w$.

The proposed model obtains the resource allocation and MEUT with considering step function $s(w)$. This section considers that MEUT of the proposed model is recalculated with $f(w)$ based on the obtained allocation, instead of the objective value of the proposed model with consdiering $s(w)$, unless otherwise stated. The baseline model directly obtains the resource allocation and MEUT with $f(w)$, regardless of the step function.

3.5.2 Solving approaches for baseline model

The proposed model adopts a step function to approximate the non-decreasing workload-dependent failure probability so that the considered problem can be simplified and solved by an MILP approach. Since the failure probability in the baseline model may not be in a linear form, the model cannot be formulated as an MILP problem. Brute-force search (BFS) can be adopted for a smaller size problem of the baseline model to obtain the optimal solution by listing all possible resource allocations and calculating each corresponding MEUT. Algorithm 3.4 shows the procedure of BFS to solve the baseline model; the optimal solution of the baseline corresponds to the minimum MEUT among all possible resource allocations.

Algorithm 3.4 Brute-force search

Input: $K, V, f(w), C_i, l_j^{\mathrm{R}}, t_1, t_0, t_3$

Output: x_{ij}^{kb}, minimum MEUT among all possible resource allocations

1: List all the possible resource allocations for all $i \in K, j \in V, k \in K \backslash \{i\}, b \in \{0, 1\}$.
2: Calculate W_i and $R_i, i \in K$ in each allocation pattern.
3: Remove the allocation patterns if any node satisfies $R_i > C_i$ in the pattern.
4: Substitute W_i into $f(w)$ and calculate the corresponding failure probability for all nodes in each allocation pattern.
5: Calculate MEUT by (3d) and (3e) based on the failure probability $f(w)$ for each possible resource allocation.
6: Return the minimum MEUT among all possible resource allocations

As the problem size increases, both MILP approach to solve the proposed model and BFS to solve the baseline model become difficult in a practical time. The developed water-filling algorithm with considering a step function is introduced to solve the proposed model. Similarly, a heuristic algorithm can be adopted for the baseline model. This work provides two heuristic algorithms to solve it.

Algorithm 3.5 Workload-ordered greedy algorithm (WOGA)

Input: $K, V, f(w), C_i, l_j^{\mathrm{R}}, t_1, t_0, t_3$

Output: x_{ij}^{kb}, MEUT

Sort j by l_j^{R} decreasingly.
for $b = 0 \rightarrow 1$ **do**
    $W_i \leftarrow 0$, $q_i \leftarrow 0$
    for $j = 1 \rightarrow |V|$ **do**
        Sort i by q_i increasingly as a set K'.
        Delete node i from K' if $R_i + l_j^{\mathrm{R}} > C_i$.
        if $K' = \emptyset$ **then**
            Return infeasible
        else
            Allocate j to the first node in K' as primary resource and the second node in K' as backup resource.
            Update W_i and calculate q_i with $f(w)$.
        end if
    end for
end for
Select the smaller value between two MEUTs obtained with $b = 0$ and 1; the corresponding resource allocation is the solution.
Return x_{ij}^{kb}, MEUT

Algorithm 3.6 Simulated Annealing

Input: $K, V, f(w), C_i, l_j^{\mathrm{R}}, t_1, t_0, t_3, T_{\min}, T_{\mathrm{init}}, \rho$
Output: x_{ij}^{kb}, MEUT

$T \leftarrow T_{\mathrm{init}}$; initialize K_j, which is a set of node hosting function j.
Randomly generate a resource allocation $\mathbf{x} = x_{ij}^{kb}$ and calculate MEUT r with $f(w)$.
while $T \geq T_{\min}$ **do**
    Switch both primary and backup resource allocation of two randomly chosen functions; update K_j.
    Switch the protection strategies of a randomly chosen function.
    Reallocate a random function to a randomly chosen node in $K \backslash K_j$.
    Update W_i and failure probability q_i calculated with $f(w)$; sort i by q_i increasingly as a set K'. Allocate j to the first node in K' as primary resource and the second node in K' as backup resource.
    while $R_i > C_i$ **do**

9: Release a random function hosted by node i.
10: **end while**
11: **if** $|K_j| < 2$ for any j **then**
12: Allocate function j to a random node in $K \backslash K_j$.
13: **end if**
14: Calculate MEUT r' with (3d) and (3e) by using updated resource allocation $\mathbf{x}'$.
15: $r \leftarrow r'$ and $\mathbf{x} \leftarrow \mathbf{x}'$ with probability of $min(1, u)$, where $u = -\frac{r'-r}{T}$.
16: $T = T \cdot \rho$
17: **end while**
18: **if** $|K_j| \neq 2$ for all functions **then**
19: Return infeasible
20: **else**
21: Return $\mathbf{x}$ and MEUT
22: **end if**

Algorithm 3.5 shows a workload-ordered greedy algorithm (WOGA) for solving the baseline model. Each function is allocated to the two nodes with the minimum failure probability calculated with $f(w)$. Algorithm 3.6 shows simulated annealing (SA), which is a randomized heuristic algorithm [94], to minimize MEUT based on $f(w)$. In Algorithm 3.6, T_{init} is the initial temperature; ρ is a given decreasing rate of the temperature. In each iteration of decreasing the temperature until T_{min}, the existing solution $\mathbf{x}$ changes randomly to generate a new solution and calculate a new MEUT with $f(w)$ (lines 3-13). SA accepts the solution with a worse objective value than the existing solution with a certain probability; the higher temperature is, the higher probability to accept a worse solution is (lines 14-16). As a result, SA can jump out of a local optimal solution.

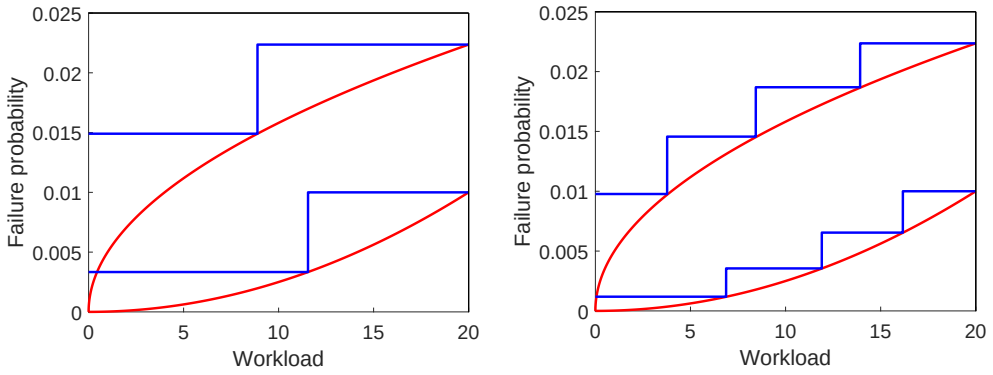
3.5.3 Case study

This work considers that the given non-decreasing workload-dependent failure probability can be expressed by $f_1(w) = (0.005w)^2$ as a convex function or $f_2(w) = 0.005\sqrt{w}$ as a concave function. Assuming that the capacities of nodes are 20, this work obtains step function $s_\mu(w)$, where $\mu = 1, 2$, fitting the curve

Table 3.4: Comparisons between proposed model with $s(w)$ and baseline model with considering $f(w)$.

			$f_1(w) = (0.005w)^2$		$f_2(w) = 0.005\sqrt{w}$	
Problem size	Model	Solving approach	MEUT (s)	Computation time (s)	MEUT (s)	Computation time (s)
Smaller	Proposed model (S=2)	MILP	0.051 (0.068)‡	0.61	0.192 (0.210)‡	0.35
	Proposed model (S=4)	MILP	0.039 (0.045)‡	1.20	0.181 (0.194)‡	1.72
	Baseline	BFS	0.039	679.96	0.181	577.76
	Baseline	WOGA (Greedy)	0.051	0.26	0.196	0.17
Larger	Proposed model (S=2)	Water-filling	0.014 (0.019)‡	0.98	0.120 (0.139)‡	1.15
	Proposed model (S=4)	Water-filling	0.006 (0.008)‡	1.01	0.116 (0.126)‡	1.06
	Proposed model (S=8)	Water-filling	0.005 (0.006)‡	1.10	0.107 (0.115)‡	1.17
	Proposed model (S=16)	Water-filling	0.004 (0.005)‡	1.20	0.094 (0.100)‡	1.10
	Baseline	WOGA (Greedy)	0.021	0.35	0.145	0.51
	Baseline	SA (T_{init} = 100)	0.008	1635.98	0.119	1832.25
	Baseline	SA (T_{init} = 1000)	0.005	2276.94	0.090	2368.24

$r(r')$‡: r represents MEUT calculated with $f(w)$; r' represents the value calculated with $s(w)$ as a reference value.



(a) Two-step functions to fit $f_1(w)$ and $f_2(w)$.

(b) Four-step functions to fit $f_1(w)$ and $f_2(w)$.

Figure 3.7: Step functions fitting $f_1(w) = (0.005w)^2$ and $f_2(w) = 0.005\sqrt{w}$.

of $f_\mu(w)$ to minimize $\int_0^{20}(s_\mu(w) - f_\mu(w))\mathrm{d}w$, with constraints $s_\mu(w) \geq f_\mu(w)$ and $T_s \geq T_{s-1}, \forall s \in \mathcal{S}$. The approximation is implemented with Matlab 2021a; the results are shown in Fig. 3.7. The computation time for fitting each curve is 0.074 [s] and 0.077 [s] for two-step and four-step approximations, respectively.

For $f_1(w) = (0.005w)^2$, this work has T_1=11.55, $t_0 = C$ =20.00; $P_1 = 0.0033$, P_2 =0.0100 for two-step approximation, and T_1=6.87, t_0=11.90, T_3=16.17, $T_4 = C = 20.00$; $P_1 = 0.0012$, $P_2 = 0.0036$, $P_3 = 0.0065$, $P_4 = 0.0100$ for four-step approximation.

For $f_2(w) = 0.005\sqrt{w}$, this work has T_1 =8.89, $t_0 = C$ =20.00; P_1= 0.0149, P_2= 0.0224 for two-step approximation, and T_1=3.77, t_0=8.43, T_3=13.92, $T_4 = C$ =20.00; $P_1 = 0.0098$, $P_2 = 0.0146$, $P_3 = 0.0187$, $P_4 = 0.0224$ for four-step approximation.

This work conducts two tests with different problem sizes. In the smaller problem size, this work considers three PMs, whose capacities are set to 20, hosting five VMs with requested loads of 5. In a larger problem size, this work considers 60 PMs, with capacities of 20, hosting 100 VMs, whose requested loads are randomly distributed over the range of [0.1, 4.0]. In both tests, unavailable time t_1, t_0, and t_3 are set to 5 [s], 20 [s], and 200 [s], respectively. This work sets T_{init} = 100 and 1000 for two cases, and $\rho = 0.98$ for SA.

Table 3.4 shows the comparisons between the proposed model with $s(w)$ and the baseline model with considering $f(w)$. Each MEUT obtained with

the proposed model shown in the table consists of two values. This work uses the values calculated with $f(w)$ for comparison with the baseline models, and the values calculated with $s(w)$ for reference, where each reference value is described inside parentheses.

Smaller problem size

In the test with the smaller problem size, this work compares the solutions obtained by the MILP approach with $s(w)$ and BFS with $f(w)$. This work considers that the proposed model substitutes the allocation obtained by the MILP approach and calculates a new MEUT with $f(w)$. We observe that MEUT of the proposed model after recalculation is smaller than the objective value calculated with $s(w)$. Similar with the objective value, MEUT after recalculation decreases as S increases. A better performance of the proposed model on MEUT can be observed as the increase of S. When this work considers $f_1(w)$, MEUTs obtained by the baseline model are on average 23.53% and 0%, respectively, smaller than those of the proposed model with two-step and four-step approximations. However, since BFS searches for all the possible allocation patterns, even for the tested small size problem, there exists $(|i| \times |k| \times |b|)^{|j|} = (3 \times (3-1) \times 2)^5 = 12^5 = 248832$ allocation patterns. The computation time for solutions with BFS is about 8×10^2 times longer than the proposed model with step approximations, which can be linearized and simply solved by the MILP approach. Then this work compares the proposed model with the baseline model when it is solved by the greedy algorithm. MEUTs obtained by the baseline model with WOGA are on average 0% and 30.77%, respectively, larger than those of the proposed model with $S = 2$ and $S = 4$.

Similarly, when this work considers $f_2(w)$, MEUTs obtained by the baseline model are on average 5.72% and 0%, respectively, smaller than those of the proposed model with two-step and four-step approximations. The computation time for solutions with BFS is about 10^3 times longer than the proposed model. MEUTs obtained by the baseline model with WOGA are on average 2.08% and 8.29%, respectively, larger than those of the proposed model with $S = 2$ and $S = 4$. This work compares convex function $f_1(w)$ with concave function $f_2(w)$. Since the workloads of nodes in the tests are larger than 15, the gradient of

$f_1(w)$ is larger than $f_2(w)$ in the range. The performance of model with $s_2(w)$ to approximate $f_2(w)$ is sightly better than that of $f_1(w)$.

The proposed model with the step approximation reduces the computation time significantly with almost the same objective value, compared with BFS to solve the baseline model. As the problem size increases, the time for solving BFS significantly increases; it cannot be solved in a practical time. Simplifying $f(w)$ by step approximation as $s(w)$ enables the model to be solved in a relatively larger size of problem by the MILP approach, as shown in Section V.C.

Larger problem size

In the test with the larger problem size, this work compares the solutions obtained by the water-filling algorithm with considering $s(w)$, and WOGA and SA with $f(w)$. To investigate the dependency of MEUT on S and compare the proposed model with the baseline model solved by heuristic algorithms, this work further considers larger S for approximation. For $f_1(w)$, this work has the thresholds of 3.00, 6.03, 8.38, 10.62, 12.88, 15.22, 17.56, and 20.00 for eight-step approximation with minimizing $\int_0^{C_i} (s(w) - f(w))\,\mathrm{d}w$, and those of 2.02, 3.42, 4.55, 5.47, 6.40, 7.36, 8.41, 9.60, 10.94, 12.35, 13.75, 15.11, 16.31, 17.55, 18.76, and 20.00 for 16-step approximation. For $f_2(w)$, this work has the thresholds of 1.07, 2.64, 4.59, 6.85, 9.46, 11.27, 12.32, and 15.83 for eight-step approximation, and those of 0.67, 1.70, 2.89, 4.29, 5.39, 6.45, 7.52, 8.61, 9.69, 10.76, 11.84, 12.94, 14.14, 15.67, 17.50, and 20.00 for 16-step approximation.

We observe that MEUT of the proposed model calculated with $f(w)$ reduces as S increases as well as that calculated with $s(w)$. When this work considers $f_1(w)$, MEUTs of the proposed model with $S = 2$, 4, 8, and 16 obtained by the water-filling algorithm are 33.33%, 71.43%, 76.19%, and 80.95%, respectively, smaller than those of the baseline model obtained by WOGA with about 2.84 times longer computation time. Similarly, when this work considers $f_2(w)$, MEUTs of the proposed model with $S = 2$, 4, 8, and 16 obtained by the water-filling algorithm are 17.24%, 20.00%, 26.21%, and 35.17%, respectively, smaller than those of the baseline model obtained by WOGA.

The water-filling algorithm is also one of greedy algorithms. The algorithm considering thresholds related to the step function enables the VMs to

be protected iteractively by the PM with failure probabilityP_{s_2}, while primary resources can be allocated to the PM with failure probability P_{s_1}, where $P_{s_2} \geq P_{s_1}$. The water-filling algorithm considering the step approximation outperforms WOGA which only considers $f(w)$, with similar computation time. Besides, without step approximation, Theorem 3.1 does not work for the baseline model. The selection of the different protection strategies may lead to a smaller MEUT for the baseline model; it can be achieved by SA.

This work firstly considers the situation under $T_{\text{init}} = 100$ for SA. When this work considers $f_1(w)$, MEUT obtained by the baseline model with SA is 42.86% smaller than that of the proposed model with $S = 2$; MEUTs obtained by the baseline model with SA are 33.33%, 60.00%, and 100.00% larger than those of the proposed model with $S = 4$, 8, and 16, respectively, with about 1.5×10^3 times longer computation time. When this work considers $f_2(w)$, MEUT obtained by the baseline model with SA is 8.3% smaller than that of the proposed model with $S = 2$; MEUTs obtained by the baseline model with SA are 2.59%, 11.21%, and 26.60% larger than those of the proposed model with $S = 4$, 8, and 16, respectively.

This work secondly considers the situation under $T_{\text{init}} = 1000$, with a longer computation time than that of $T_{\text{init}} = 100$. When this work considers $f_1(w)$, MEUTs obtained by the baseline model with SA are 64.29%, 16.67%, and 0% smaller than those of the proposed model with $S = 2$, 4, and 8, respectively; the computation time of SA is at least 1.36×10^3 longer than that of the proposed model. MEUT obtained by the baseline model with SA is 25% larger than that of the proposed model with $S = 16$. When this work considers $f_2(w)$, MEUTs obtained by the baseline model with SA are 25.00%, 22.41%, 15.89%, and 4.26% smaller than those of the proposed model with $S = 2, 4, 8$, and 16, respectively. The designed SA searches a larger solution space with a longer computation time by involving randomly generated solutions. SA, as a randomized algorithm, may outperform slightly the water-filling algorithm, at the cost of longer computation time. As S increases, both MEUTs of the proposed model calculated with $s(w)$ and $f(w)$ decrease in the tested case. The accuracy of the approximation for the non-deceasing failure probability is improved as S increases. The difference between MEUTs obtained by the water-filling algorithm and SA decreases as S increases.

In conclusion, MEUTs of the proposed model calculated with $f(w)$ have the same tendency with MEUTs calculated with $s(w)$ as the increase of S. The proposed model with considering the step approximation outperforms WOGA; it becomes comparable with SA as the increase of S. As S increases, a better performance of the proposed model in terms of MEUT can be observed.

3.6 Summary

This work proposed a primary and backup resource allocation model with considering a workload-dependent failure probability aiming to minimize the maximum expected unavailable time. The workload-dependent failure probability and the consideration of different backup strategies cause different recovery time and lead to a nonlinear programming problem to calculate unavailable time for each VM. With step functions to approximate the given non-decreasing workload-dependent failure portability, this work formulated the problem as an MILP problem to obtain the primary and backup resource allocation of each VM in PMs, where the expected unavailable time is suppressed. This work proved that MEUT of the proposed model is equal to the smaller value between the two MEUTs obtained by applying only HB and CB strategies with the same total requested load by comprehensively discussing different values of MEUT corresponding to different parameters. The heuristic algorithm inspired by the water-filling algorithm was developed based on the proved theorem. The numerical results showed that the proposed model suppresses MEUT compared with the conventional model which does not consider the workload-dependent failure probability. The developed heuristic algorithm is approximately 10^5 times faster than the MILP approach with 10^{-2} performance penalty on MEUT. This work discussed and implemented the approximation of step function for a non-decreasing function with a goal; this work investigated the performance of approximation for different problem sizes.

Chapter 4

Prioritized multiple backups resource allocation with workload-dependent failure probability

This chapter proposes an optimization model to derive a multiple backup resource allocation with the workload-dependent failure probability to minimize MEUT under a priority policy [33, 95].

The remainder of the chapter is organized as follows. Section 4.1 presents the motivation of the proposed model. Section 4.2 describes the proposed model. Section 4.3 introduces a heuristic algorithm. Section 4.4 presents numerical results that show the performance of the proposed model in different cases. Section 4.5 discusses the considerations related to the network services, workload-independent failures, and practical situations. Section 4.6 summarizes this chapter.

4.1 Motivation and use case

4.1.1 Motivation

Middleboxes fail due to hardware failures and overloads. In order to improve fault tolerance, service providers usually deploy virtual instances as backup resources of the network functions across multiple servers with different backup strategies [55, 96]. With considering the workload of servers, different backup strategies, i.e., HB and CB, lead to different computation resource utilization of the servers and recovery time for the function. Unavailable time for a function for recovery can be adopted to estimate the availability of a function [12]. A shorter expected unavailable time for recovery can improve the user experience. MEUT is related to the backup strategies and the S-step function, which conservatively approximates a non-decreasing function of workload according to the requirement of users. The S-step function can be different for each server. With considering the workload-dependent failure probability, a service provider is able to provide each service with the strategy selection and resources allocation by minimizing MEUT among functions.

The proposed model is designed to determine the initial backup resource allocations of functions to multiple servers before services run by considering the unavailable time of functions caused by the workload-dependent failure and different strategies. Minimizing MEUT among all functions preventively before services run can improve the fault tolerance; when a failure occurs, the recovery time among functions is minimized based on the initial backup resource allocation.

4.1.2 Use case of proposed model

NFV is applicable across a wide range of network functions. For example, a simple service function model for hypertext transfer protocol (HTTP) traffic over TCP port 80 includes load balancers (LB), network address translation (NAT), firewall (FW), deep packet inspection (DPI), and some other network functions. LB splits HTTP over transmission control protocol (TCP) port 80 away from the rest of Internet traffic; FW protects the carrier network from the outside; NAT maps the private Internet protocol (IP) address space dedicated

to user equipment to a public IP address. DPI inspects the performances of traffic, identification of applications, and policy enforcement. The unavailable time of each function affects the unavailability of the service [65].

This work presents an allocation example to show the features of proposed model and compare it with two conventional models that do not consider the balancing of the unavailable time and unavailable probability [30, 56]. In the example, this work considers the workload-dependent failure probability for three cases. Case 1 minimizes the unavailable time; only the HB strategy is adopted to provide protection [30]. Case 2 minimizes the unavailable probability; only the CB strategy is adopted to provide protection [30]. Case 3, which is the proposed model, considers both HB and CB strategies to balance the unavailable time and unavailable probability to minimize MEUT.

This work considers that four functions and three servers are given in advance. This work assumes that the maximum computing capacity of each server is 1 CPU core. Each function requests amounts of computing resources of the server for providing the protection with the HB strategy and the CB strategy. The four functions protected with the HB strategy require the computing resources, which are 0.2, 0.2, 0.3, and 0.3 CPU cores, respectively. The functions protected with the CB strategy do not require computing resources. The thresholds of the three servers are 0.4, 0.6, and 0.8 CPU cores, respectively. This work assumes that there are two failure probabilities corresponding to different workloads, which are 0.1 and 0.2, respectively. The recovery time for different strategies are determined by the synchronization frequency, the synchronization content size, and the synchronization strategy. The unavailable time of a function when all of the backup servers fail simultaneously depends on the recovery mechanism of the physical server. This work assumes that the recovery time for the HB strategy, the CB strategy, and the physical server are 0.5 [s], 2 [s], and 100 [s], respectively.

In case 1, each function is protected by three servers with the HB strategy. The workload of each server is 1 CPU, so that the failure probability is 0.2. The unavailable time for each function is $0.5 \times (1 - 0.2) + 0.5 \times 0.2(1 - 0.2) + 0.5 \times 0.2^2(1 - 0.2) + 100 \times 0.2^3 \approx 1.926$ [s].

In case 2, each function is protected by three servers with the CB strategy. The computing resource of the CB strategy is assume to be 0, so that the

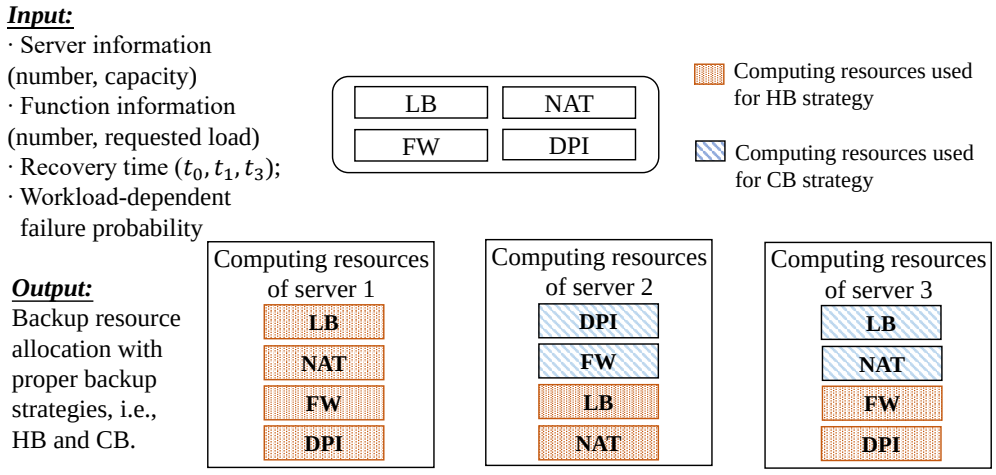


Figure 4.1: Use case to show applicability in practical situation.

failure probability of each server is 0.1. The unavailable time for each function is $2 \times (1 - 0.1) + 2 \times 0.1(1 - 0.1) + 2 \times 0.1^2(1 - 0.1) + 100 \times 0.1^3 \approx 2.100$ [s].

In case 3, the functions are protected with the strategies of HB and CB flexibly. Based on the backup strategies and resource allocations of the functions as shown in Fig. 4.1, the workloads of the servers are calculated, which are 1, 0.3, and 0.7, respectively. Comparing the workload and the threshold of each server, the failure probabilities of servers are obtained correspondingly, which are 0.2, 0.1, and 0.1. The unavailable time for each function is $0.5\times(1-0.2)+0.5\times0.1(1-0.2)+2\times0.1(1-0.1)(1-0.2)+100\times0.1^2\times0.2 \approx 0.726$ [s].

By balancing the unavailable time and the unavailable probability with two backup strategies and the workload-dependent failure probability, MEUT in case 3 is smaller than those of cases 1 and 2. MEUT among functions in a HTTP traffic service, which includes four functions, i.e., LB, NAT, FW, and DPI, is suppressed; user experience is improved confronting server failures.

4.2 Optimization Model

This work builds a multiple backup resource allocation model with consideration of determined priority scores (DPS). This work considers that a function

can be protected by multiple servers. DPS is defined as the priority score of the set of backup servers which protects the same function. DPS is determined by the workload and strategies of each server which protects the function. In this model, each backup resource of a function is allowed to adopt either HB or CB to provide protection. Each server has its failure probability affected by the resource allocation and backup strategies. This work focuses on the assignment to minimize the maximum expected unavailable time of each function. Let F and N represent a set of functions and a set of backup servers, respectively, where $|F|$ and $|N|$ denote the numbers of functions and backup servers, respectively. Consider two kinds of backup strategies $b \in B := \{0, 1\}$. $b = 0$ denotes the CB strategy and $b = 1$ denotes the HB strategy. A function can be protected by one or more servers. When a function fails, one of the corresponding servers provides either HB or CB to recover the function. The reserved idle capacity of the function protected by the server with the CB strategy is not taken into account as the active workload in the server. The requested load of the function protected by the server protected with the HB strategy are counted as the active workload.

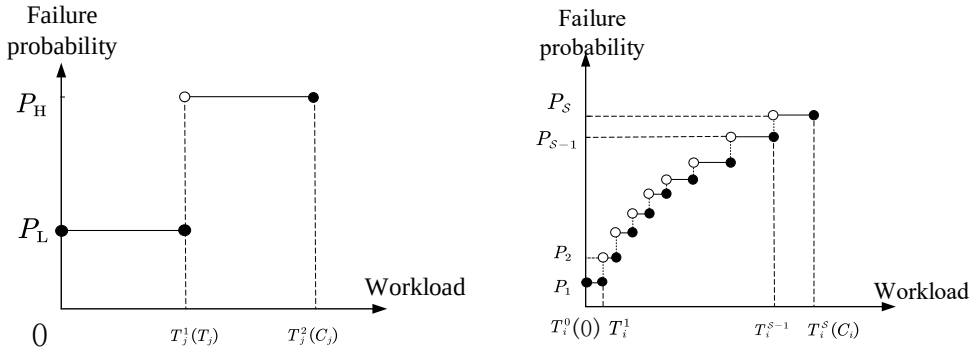
Let x_{ij}^b, $i \in F, j \in N, b \in B$, denote a binary decision variable; x_{ij}^b is set to one if function $i \in F$ is protected by server $j \in N$ with strategy $b \in B$, and zero otherwise. w_i denotes the requested load of each function $i \in F$.

A Failure probability of each server is related to the workload of corresponding function and strategies. Let L_j denote the workload of server $j \in N$, which is expressed by:

$$L_j = \sum_{i \in F} w_i x_{ij}^1, \forall j \in N. \tag{4.1}$$

Let C_j denote the upper bound of computing capacity which server $j \in N$ can provide for backup allocation. T_j is a given parameter of servers $j \in N$, which denotes the threshold that the network maintains its normal and efficient functioning. P_{L} denotes the given failure probability of each server $j \in N$ with $0 \leq L_j \leq T_j$, while P_{H} denotes the given failure probability of each server $j \in N$ with $T_j < L_j \leq C_j$.

This work assumes that there is a non-decreasing relationship between the workload and the failure probability. The failure probability of any server is an



(a) Two-step failure probability. (b) S-step failure probability.

Figure 4.2: Workload-dependent failure probability is expressed by a non-decreasing step function.

non-decreasing function of the workload of itself. Without loss of generality, given the workload-dependent failure probability, which is assumed to be a non-decreasing function, this work can use an S-step function, as shown in Fig. 4.2, to conservatively approximate it. $T_j^s, s \in [1, |S|]$, in the horizontal axis in Fig. 4.2 is a given value of workload, which can indicate a ratio of a workload to capacity. When the workload of a server increases from the range of $(T_j^{s-1}, T_j^s]$ to $(T_j^s, T_j^{s+1}]$, the failure probability increases from P_{s-1} to P_s. As shown in Fig. 4.2(b), the step function can be extended to be an S-step function, where $S \geq 2$, to conservatively approximate a general non-decreasing function of workload.

According to [8], this work assumes that when the workload of server j does not exceed T_j, a network maintains its normal and efficient functioning. Each server has a low failure probability P_{L}, and has capacity to handle the extra workload. When the workload of server j is larger than T_j, the failure probability increases because of higher workload, where the part or whole network cannot maintain its normal and efficient functioning. Even though each server has capacity, the server becomes fragile, and each server has a higher failure probability P_{H} to handle the extra workload. Thus, how to balance the workload becomes a fundamental task for operating NFV clusters [97].

The failure probability of server $j \in N$ according to the workload is denoted by q_j. Given the workload-dependent failure probability, which is assumed to be a non-decreasing function, this work can use a two-step function, as shown

in Fig. 4.2(a), to conservatively approximate it. q_j is expressed by the following step function, which relates the workload of server to the failure probability:

$$q_j = \begin{cases} P_{\mathrm{L}} & L_j \leq T_j, \\ P_{\mathrm{H}} & T_j < L_j \leq C_j. \end{cases} \tag{4.2}$$

Equation (4.2) can be extended by expressing an S-step failure probability, which can conservatively approximate a general non-decreasing function of workload with higher accuracy according to the requirement of users.

This work considers that multiple simultaneous failures occur in functions and servers due to their workload. This work assumes that all servers fail independently. Let t_1 be a given parameter that denotes the recovery time for the HB strategy. Let t_0 be a given parameter that denotes the recovery time for the CB strategy. u_i denotes the expected unavailable time for each function $i \in F$. There are two situations that function is unavailable. One is that function $i \in F$ fails and becomes unavailable before it is recovered by an available server. One of the corresponding available backup servers $j \in N$ determined by DPS, requires the recovery time which depends on the backup strategies and the assignment of backup resource. The other one is that function $i \in F$ and all of its corresponding backup servers protecting it fail simultaneously. Let t_3 be a given parameter that denotes the recovery time of the situation when both function and corresponding backup servers are unavailable. This work assumes $t_3 \geq t_0 \geq t_1$.

Let N_i denote the set of backup servers assigned to function $i \in F$. Let O_{ij} be an integer which denotes DPS of each $j \in N_i$ assigned to function $i \in F$, and relates to strategies of each server. O_{ij} is expressed by:

$$O_{ij} = \sum_{b \in B} (j + n|N|) x_{ij}^b, \forall i \in F, j \in N. \tag{4.3}$$

This work formulates the multiple backup resource allocation problem with considering DPS as the following optimization problem:

$$\min u \tag{4.4a}$$

$$s.t. \sum_{j \in N} \sum_{b \in B} x_{ij}^b \geq 1, \forall i \in F, \tag{4.4b}$$

$$\sum_{b\in B} x_{ij}^{b} \leq 1, \forall i \in F, j \in N, \tag{4.4c}$$

$$\sum_{b\in B}\sum_{i\in F} w_i x_{ij}^{b} \leq C_j, \forall j \in N, \tag{4.4d}$$

$$u_i = \sum_{b\in B}\sum_{j\in N_i} t_b x_{ij}^{b} p_i \prod_{O_{ij'}>O_{ij}:j'\in N_j} q_{j'}(1-q_j) + t_3 \prod_{j\in N_i}\sum_{b\in B} x_{ij}^{b} p_i q_j, \forall i \in F, \tag{4.4e}$$

$$u \geq u_i, \forall i \in F, \tag{4.4f}$$

$$x_{ij}^{b} \in \{0,1\}, \forall i \in F, j \in N, b \in B. \tag{4.4g}$$

Equation (4.4a) minimizes u, which denotes the maximum expected unavailable time among $u_i, i \in F$. Equation (4.4b) ensures that each function $i \in F$ can be protected by one or more servers with selecting one of the backup strategies for each server. Equation (4.4c) ensures that each function $i \in F$ can be protected with either HB or CB for each server $j \in N$. Equation (4.4d) indicates the capacity constraint that the required space on each server for the backup resource allocation does not exceed its maximum computing capacity. The first term in the right side of (4.4e) is the expected unavailable time for server $j \in N$ by applying either HB or CB. This term considers the case that function $i \in F$ fails and is protected by one of its corresponding available backup servers. This work considers that the order of servers $j \in N_i$ to provide protection follows O_{ij}, the larger O_{ij} of server $j \in N_i$ is, the higher priority of $j \in N_i$ to protect function $i \in F$ is. The failed function is protected by an available server whose non-failure probability is $1 - q_j$ and DPS is O_{ij} when other servers $j' \in N_i$ which have higher DPS than server $j \in N_i \setminus \{j'\}$ fail at failure probability $\prod_{O_{ij'}>O_{ij}:j'\in N_i} q_{j'}$ simultaneously. The second term in the right side of (4.4e) is the expected unavailable time if function $i \in F$ fails and all of the corresponding backup servers in N_i fail at failure probability q_j simultaneously. The expected unavailable time for a server is related to the backup strategies, the workload-dependent failure probability, and the priority policy. The selection of the HB and CB strategies affects the coefficient t_b in the first term of (4.4e) and the workload of each server, as shown in (4.1); the workload may further affect the failure probability q_j, as shown in (4.2).

Theorem 4.1 *When servers $j \in N_i$ adopt the same strategy, either HB or*

CB, the priority does not influence the expected unavailable time u_i.

Proof : When all servers $j \in N_i$ adopt the same strategy, either HB or CB, the expected unavailable time of functions $i \in F$ are only affected by the allocation and the corresponding failure probability instead of the priority of servers. Since $(1 - q_{j'}) + q_{j'}(1 - q_j) = (1 - q_j) + q_j(1 - q_{j'})$, the right side of (4.4e) can be expanded and simplified as:

$$u_i = t_b p_i (1 - \prod_{j \in N_i} q_j) + t_3 \prod_{j \in N_i} p_i q_j, \forall i \in F, \tag{4.5}$$

where b is the adopted strategy. The first term in the right side of (4.5) considers the case that function $i \in F$ fails and is protected by one of its corresponding available backup server. The second term in the right side of (4.5) is the expected unavailable time if function $i \in F$ fails and all of the corresponding backup servers in N_i fail at failure probability q_j simultaneously. The expression of situation in (4.4e) can be simplified to (4.5) by combining the product terms of $q_j q_{j'} \cdots q_{j''}$ and subtracting from each other. Equation (4.5) shows that the relationship of the failure probability of each server $j \in N_i$ does not affect the value of the expected unavailable time, no matter what the relationship between $q_{j'}$ and q_j is. □

Theorem 4.2 *When servers* $j \in N_i$ *adopt the different strategies, setting the server with the HB strategy to higher priority has the lower expected unavailable time than that with the CB strategy.*

Proof : There are two situations for the priority; the first one is that the server with the CB strategy has the higher priority and the second one is that the server with the HB strategy has the higher priority. Assume that one server which is protected with the CB strategy has the failure probability $q_{j'}$ and the other server which is protected with the HB strategy has the failure probability q_j. The right side of (4.4e) of the first and second situations of priority can be simplified as:

$$u_i' = t_0 p_i (1 - q_{j'}) + t_1 p_i q_{j'} (1 - q_j) + t_3 p_i q_j q_{j'}, \tag{4.6a}$$

$$u_i'' = t_1 p_i (1 - q_j) + t_0 p_i q_j (1 - q_{j'}) + t_3 p_i q_j q_{j'}, \tag{4.6b}$$

respectively.

This work subtracts both sides of (4.6b) from those of (4.6a) to compare the value of expected unavailable time of two situations. There is $u'_i - u''_i = t_1(q_{j'} - 1)(1 - q_j) + t_0(1 - q_{j'})(1 - q_j) = (1 - q_j)(1 - q_{j'})(t_0 - t_1) \geq 0$, where $t_0 \geq t_1$. Thus, setting the server with the HB strategy to higher priority has the lower expected unavailable time than that with the CB strategy. □

This work defines the priority policy as follows. If servers $j \in N_i$ adopt the different strategies, the server which adopts the HB strategy has higher priority (see Theorem 4.2). Otherwise, when servers $j \in N_i$ adopt the same strategies, either HB or CB, the priority of servers is arbitrary (see Theorem 4.2). Here, for convenience, this work assumes that servers $j \in N_i$ with larger index j. Here, for convenience, this work assumes that servers $j \in N_i$ with larger index j have higher priority.

This work linearizes (4.2) by introducing binary variable $y_j, j \in N$. y_j is set to one if $L_j \leq T_j$, and zero otherwise, (4.2) can be expressed by:

$$q_j = P_{\mathrm{L}} y_j + P_{\mathrm{H}}(1 - y_j), \forall j \in N, \tag{4.7a}$$

$$L_j \leq T_j y_j + C_j(1 - y_j), \forall j \in N, \tag{4.7b}$$

$$L_j \geq T_j(1 - y_j), \forall j \in N, \tag{4.7c}$$

$$y_j \in \{0, 1\}, \forall j \in N. \tag{4.7d}$$

Let $e^i_{jj'}$, $i \in F, j \in N, j' \in \mathcal{N}_j \coloneqq S \setminus \{j\}$ be a binary variable: it is set to one if $O_{ij'} > O_{ij}$, and zero otherwise. $e^i_{jj'}$ can be expressed as a linear form as:

$$O_{ij'} - O_{ij} \leq e^i_{jj'} A, \forall i \in F, j \in N, j' \in \mathcal{N}_j, \tag{4.8a}$$

$$O_{ij'} - O_{ij} \geq (e^i_{jj'} - 1)A, \forall i \in F, j \in N, j' \in \mathcal{N}_j, \tag{4.8b}$$

$$e^i_{jj'} + e^i_{j'j} = 1, \forall i \in F, j \in N, \mathcal{N}_j, \tag{4.8c}$$

$$e^i_{jj'} \in \{0, 1\}, \forall i \in F, j \in N, j' \in \mathcal{N}_j, \tag{4.8d}$$

where A is a sufficiently large number that satisfies $A \geq 2|N|$.

By introducing variables $r^i_{jj'}$, $i \in F, j \in N, j' \in \mathcal{N}_j$, which takes two values: 1 and $q_{j'}$. This work defines that $r^i_{jj'} = (q_{j'} - 1)e^i_{jj'} + 1$. The first term in the

right side of (4.4e) is transformed to:

$$\sum_{b\in B}\sum_{j\in N} t_b x_{ij}^b p_i(1-q_j)\prod_{j'\in S\setminus\{j\}} x_{ij'}^b r_{jj'}^i, \forall i \in F. \tag{4.9}$$

In order to linearize (4.9), this work introduces binary variables $\omega_{mij}^{\mathrm{L}}$ and $\omega_{mij}^{\mathrm{H}}$, where $m \in [0,|S|]$ and $i \in F, j \in N$. $\omega_{mij}^{\mathrm{L}}$ is set to 1 if $\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} e_{jj'}^i x_{ij'}^b y_{j'}$ is equal to m for $i \in F, j \in N$, and 0 otherwise. $\omega_{mij}^{\mathrm{H}}$ is set to 1 if $\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} e_{jj'}^i x_{ij'}^b(1-y_{j'})$ is equal to m for $i \in F, j \in N$, and 0 otherwise. This work can express $\omega_{mij}^{\mathrm{L}}$ and $\omega_{mij}^{\mathrm{H}}$ in the form of the following constraints:

$$\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} e_{jj'}^i x_{ij'}^b y_{j'} = \sum_{m\in[0,|S|]} m\omega_{mij}^{\mathrm{L}}, \forall i \in F, j \in N, \tag{4.10a}$$

$$\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} e_{jj'}^i x_{ij'}^b(1-y_{j'}) = \sum_{m\in[0,|S|]} m\omega_{mij}^{\mathrm{H}}, \forall i \in F, j \in N, \tag{4.10b}$$

$$\sum_{m\in[0,|S|]} \omega_{mij}^{\mathrm{L}} = 1, \forall i \in F, j \in N, \tag{4.10c}$$

$$\sum_{m\in[0,|S|]} \omega_{mij}^{\mathrm{H}} = 1, \forall i \in F, j \in N, \tag{4.10d}$$

$$\omega_{mij}^{\mathrm{L}}, \omega_{mij}^{\mathrm{H}} \in \{0,1\}, \forall m \in [0,|S|], i \in F, j \in N. \tag{4.10e}$$

Similarly, let z_{mi}^{L}, z_{mi}^{H} be binary variables, where $m \in [0,|S|]$ and $i \in F$ to linearize the second term of the right side of (4.4e). z_{mi}^{L} is set to 1 if $\sum_{b\in B}\sum_{j\in N} x_{ij}^b y_j$ is equal to m for $i \in F$, and 0 otherwise. z_{mi}^{H} is set to 1 if $\sum_{b\in B}\sum_{j\in N} x_{ij}^b(1-y_j)$ is equal to m for $i \in F$, and 0 otherwise. This work can express z_{mi}^{L}, z_{mi}^{H} in the form of the following constraints:

$$\sum_{b\in B}\sum_{j\in N} x_{ij}^b y_j = \sum_{m\in[0,|S|]} m z_{mi}^{\mathrm{L}}, \forall i \in F, \tag{4.11a}$$

$$\sum_{b\in B}\sum_{j\in N} x_{ij}^b(1-y_j) = \sum_{m\in[0,|S|]} m z_{mi}^{\mathrm{H}}, \forall i \in F, \tag{4.11b}$$

$$\sum_{m\in[0,|S|]} z_{mi}^{\mathrm{L}} = 1, \forall i \in F, \tag{4.11c}$$

$$\sum_{m\in[0,|S|]} z_{mi}^{\mathrm{H}} = 1, \forall i \in F, \tag{4.11d}$$

$$z_{mi}^{\mathrm{L}}, z_{mi}^{\mathrm{H}} \in \{0,1\}, \forall m \in [0,|S|], i \in F. \tag{4.11e}$$

By introducing binary variables $\phi^b_{ijj'} = e^i_{jj'} x^b_{ij} y_{j'}$, $\pi^b_{ijj'} = e^i_{jj'} x^b_{ij}$ where $i \in F, j \in N, j' \in \mathcal{N}_j, b \in B$, and $\zeta^b_{ij} = x^b_{ij} y_j$, where $i \in F, j \in N, b \in B$, (4.10a), (4.10b), (4.11a), and (4.11b) can be linearized as:

$$\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} \phi^b_{ijj'} = \sum_{m\in[0,|S|]} m\omega^{\mathrm{L}}_{mij}, \forall i \in F, j \in N, \tag{4.12a}$$

$$\sum_{b\in B}\sum_{j'\in\mathcal{N}_j} (\pi^b_{ijj'} - \phi^b_{ijj'}) = \sum_{m\in[0,|S|]} m\omega^{\mathrm{H}}_{mij}, \forall i \in F, j \in N, \tag{4.12b}$$

$$\sum_{b\in B}\sum_{j\in N} \zeta^b_{ij} = \sum_{m\in[0,|S|]} m z^{\mathrm{L}}_{mi}, \forall i \in F, \tag{4.12c}$$

$$\sum_{b\in B}\sum_{j\in N} (x^b_{ij} - \zeta^b_{ij}) = \sum_{m\in[0,|S|]} m z^{\mathrm{H}}_{mi}, \forall i \in F. \tag{4.12d}$$

With $\omega^{\mathrm{L}}_{mij}$, $\omega^{\mathrm{H}}_{mij}$, z^{L}_{mi}, and z^{H}_{mi}, (4.4e) can be transformed to:

$$\begin{aligned} u_i = &\sum_{b\in B}\sum_{j\in N}\{t_b x^b_{ij} p_i (1-q_j) \sum_{m\in[0,|S|]}\sum_{m'\in[0,|S|]} ((P_{\mathrm{L}})^m (P_{\mathrm{H}})^{m'} \omega^{\mathrm{L}}_{mij}\omega^{\mathrm{H}}_{m'ij})\} + \\ &t_3 p_i \sum_{m\in[0,|S|]}\sum_{m'\in[0,|S|]} \{(P_{\mathrm{L}})^m (P_{\mathrm{H}})^{m'} z^{\mathrm{L}}_{mi} z^{\mathrm{H}}_{m'i}\}, \forall i \in F. \end{aligned} \tag{4.13}$$

According to the linearization process in Appendix C, (4.11a)-(4.11b), (4.12a)-(4.12d), and (4.13) can be linearized.

From the above, this work formulates the problem as an MILP problem, which is expressed by:

$$\begin{aligned} &\min\ u \\ &s.t. (4.1), (4.3), (4.4b) - (4.4d), (4.4f) - (4.4g), (4.7a) - (4.7d), \\ &\quad (4.8a) - (4.8d), (4.10c) - (4.10e), (4.11c) - (4.11e), \\ &\quad (4.12a) - (4.12d), (C.1a) - (C.1l), (C.2a) - (C.2m). \end{aligned} \tag{4.14}$$

This work provides a lower bound of the optimal objective value in the proposed model by analyzing the maximum number of servers that the function which corresponds to MEUT is protected by, which is derived in Lemma D.1 and Theorems D.1 and D.2 in Appendix D.

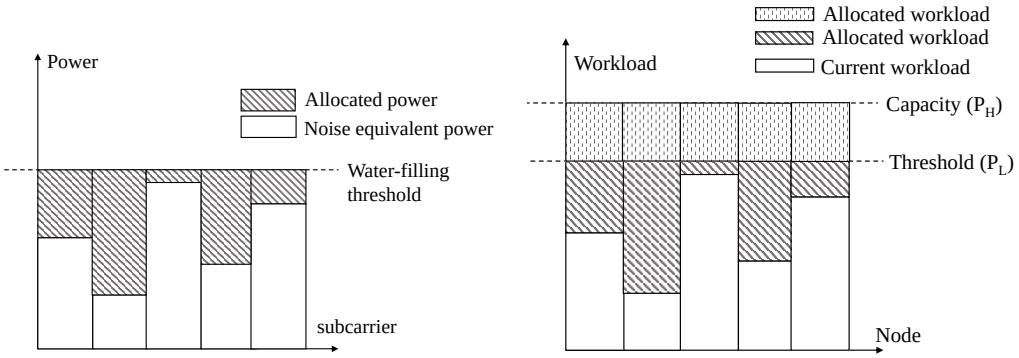
4.3 Heuristic algorithm

Since Appendix E shows that the decision version of the multiple resource allocation problem is NP-complete, the backup resource allocation problem becomes difficult to solve in a practical time as the problem size increases. This section presents a main idea of heuristic algorithm design which obtains an approximate solution of the resource allocation to minimize MEUT with considering workload to solve the problem with a larger size. This work introduces a greedy algorithm to solve the multiple backup resource allocation problem, which aims to reduce the computation time compared with the MILP approach with a limited performance penalty.

This work presents the developed heuristic algorithm based on the idea of water-filling [98], since the algorithm with two stages fits the allocation with workload-dependent failure probability approximated by a step function. As this work considers a two-step function to approximate the workload-dependent failure probability, the capacity of each server can be divided into two parts, each of which corresponds to a failure probability, as shown in Fig. 4.3(b). The traditional water-filling algorithm allocates the functions to each server iteratively with considering the current workload until the workload of the server exceeds the corresponding capacity. The developed algorithm based on the water-filling algorithm allocates the functions to servers iteratively for each part of capacity until the workload exceeds each corresponding capacity, as shown in Fig. 4.3(a).

Other heuristic algorithms such as randomized ones, which include a genetic algorithm and a simulated annealing algorithm, can also be used for solving the proposed model. The accuracy of solution in a randomized heuristic algorithm can be improved with the cost of longer computation time. Compared with randomized heuristic algorithms, the developed algorithm takes advantage of the special structure for the server capacity with the workload-dependent failure profitability in the proposed model; it cannot adjust the computation time and the number of trials to improve the accuracy.

The developed heuristic algorithm determines an initial allocation on multiple backup resources by allocating functions on each server $j \in N$ iteratively by applying the HB strategy, where $b = 1$. The algorithm determines the



(a) Traditinal water-filiing algorithm. (b) Developed water-filiing algorithm.

Figure 4.3: Examples of traditional and developed water-filling algorithms.

resource allocation of functions on each of the servers until the workload for the server exceeds threshold T_j with the allocation of next function. Then, the algorithm searches for a server with enough remaining capacity to allocate functions from the first server and allocates the functions to the server until the workload exceeds the corresponding threshold with the allocation of next function. Before each allocation of function i to server j, the algorithm checks whether there is function i for any previous round is allocated to server j. When the allocation of next function causes the workload of every server to exceed its corresponding threshold, the algorithm searches for a server with enough remaining capacity compared with C_j to allocate the functions iteratively until the workload for the server exceeds the corresponding capacity with the allocation of next function. Then, this algorithm continues to search for a server with enough remaining capacity to allocate functions until the workload for the server exceeds corresponding capacity with the allocation of next function.

The developed heuristic algorithm determines the number of servers which protect each function $i \in F$ according to the ratio of the sum of capacity to the total request load. Let $\mathcal{N}$ denote the upper bound of the number of servers to protect each function, which is expressed by $\min\{|N|, \lfloor \frac{\sum_{j \in N} C_j}{\sum_{i \in F} w_i} \rfloor\}$ and calculated in advance. Let W_j^1 and R_j^1 denote the workload and the requested load of server when it protects other servers with strategy $b = 1$ by fixing $x_{ij}^0 = 0$. W_j^1 and R_j^1 are equal and can be expressed by (4.1).

Algorithm 4.1 Developed heuristic algorithm

Input: $F, S, T_j, C_j, w_i, \forall i \in F, j \in N$

Output: x_{ij}^b

$x_{ij}^b = 0, y \leftarrow \emptyset, flag_{\text{infeasible}} \leftarrow 0$
Sort i by w_i decreasingly; sort j by T_j increasingly
for $n = 1 \rightarrow \mathcal{N}$ **do**
    for $i = 1 \rightarrow |F|$ **do**
        for $j = 1 \rightarrow |N|$ **do**
            Calculate W_j^1
            if $W_j^1 + w_i > T_j$ or i is protected by j **then**
                if $j = |N|$ **then**
                    Append $[n, i]$ to y
                end if
                Continue
            else
                $x_{ij}^1 \leftarrow 1$
                Break
            end if
        end for
    end for
end for
Initialize a full-zero matrix f, whose size is $|N| \times |F|$
for Each $[n, i] \in y$ **do**
    Calculate W_j^1
    for $j = 1 \rightarrow |N|$ **do**
        if $R_j^1 + w_i > C_j$ or i is protected by j **then**
            Continue
        else
            $x_{ij}^1 \leftarrow 1$
            $u_1 \leftarrow$Calculate u_i based on current x_{ij}^b
            $x_{ij}^1 \leftarrow 0, x_{ij}^0 \leftarrow 1$
            $u_2 \leftarrow$Calculate u_i based on current x_{ij}^b
            if $u_1 < u_2$ **then**
                $x_{ij}^1 \leftarrow 1, x_{ij}^0 \leftarrow 0$

32: **end if**
33: Break
34: **end if**
35: **end for**
36: **end for**

Algorithm 4.1 starts to solve the problem by fixing $b = 1$ and it sorts i by w_i decreasingly; sort j by T_j increasingly (line 2). The allocation in Algorithm 1 is divided into two parts by threshold T_j (lines 3-18) and capacity C_j (lines 19-36). Algorithm 4.1 allocates the backup resources of each function iteratively on the same server until the allocation of next function i causes $W_j^1 + w_i > T_j$ and $W_j^1 + w_i > C_j$ as the increase of workload, respectively. Before each allocation of function i to server j, the algorithm the algorithm checks whether there is function i for any previous round is allocated to server j (lines 7 and 23). Let $n \in N$ denote the round of allocation for each function to $\mathcal{N}$ servers. For each round of allocation $n \in \mathcal{N}$, functions are allocated to server j iteratively if $W_j^1 + w_i \leq T_j$. Algorithm 4.1 does not change backup resource allocation until the allocation of next function i causes $W_j^1 + w_i > T_j$ (lines 3-18).

Similar to lines 3-18 in Algorithm 4.1, if the next function causes the workload of every server to exceed the corresponding threshold, Algorithm 4.1 searches for a server with the enough remaining capacity from the first one and allocates the backup resources of each function iteratively on the same server until the allocation of next function i causes $W_j^1 + w_i > C_j$ or there is the same i for any previous round is allocated to j (lines 20-35). $[n, i]$ denotes function i in round n, which is protected by each server whose requested load causes the workload to exceed the corresponding threshold. By adopting the CB strategy for these functions, the workload of each server in Algorithm 4.1 does not exceed the threshold, which indicates the failure probability of P_{L}. Based on the obtained initial multiple backup resource allocation (lines 3-18), Algorithm 4.1 calculates two values u_1 and u_2 of the expected unavailable time for each $[n, i]$ by the allocation with the HB and CB strategies, respectively, and chooses the smaller one in u_1 and u_2 (lines 26-29). Algorithm 4.1 traverses and determines the strategies for each $[n, i]$ to obtain the smaller expected unavailable time, which is the best solution of each function i in the round.

The computation time complexities of sorting $|F|$ functions and sorting $|N|$

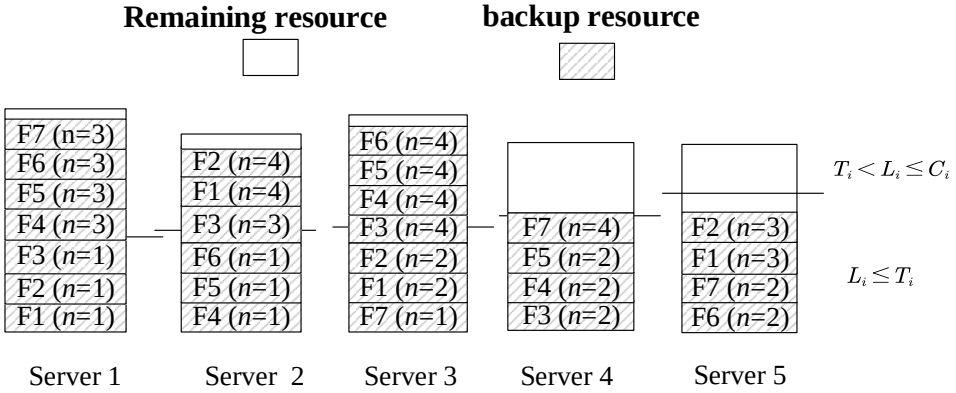


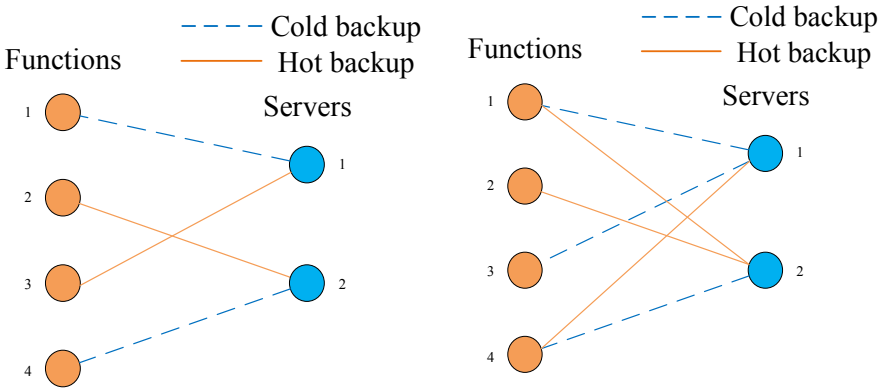
Figure 4.4: Examples of developed heuristic algorithm

.

backup servers are $O(|F| \log |F|)$ and $O(|N| \log |N|)$, respectively. The overall computation time complexity of Algorithm 4.1 is $O(|F||N||N| + |F| \log |F| + |N| \log |N|)$.

This work provides an upper bound of the expected unavailable time obtained by the algorithm by analyzing the lower bound of the minimum size of $N_i, i \in F$, which is derived in Lemma D.1 and Theorem D.3 in Appendix D.

Figure 4.4 shows an example of the algorithm to obtain the initial solution with five servers sorted by T_j increasingly. In this example, this work sets $|F| = 7$, $|N| = 5$, $\sum_{i \in F} w_i = 28$, and $\sum_{j \in N} C_j = 120$. This work calculates the value of $\min\{|N|, \lfloor \frac{\sum_{j \in N} C_j}{\sum_{i \in F} w_i} \rfloor\}$, which equals to 4, in advance. For $n = 1$, functions 1, 2, and 3 are allocated to server 1 until function 4 causes the exceeding of T_1; functions 4, 5, and 6 are allocated to server 2 until function 7 causes the exceeding of T_2 For server 3, function 7 is allocated to server 3; functions 1 and 2 for $n = 2$ are allocated to server 3 until function 3 causes the exceeding of T_3. Functions 3, 4 and 5 are protected by server 4 until function 6 causes the exceeding of T_4. Functions 6 and 7 for $n = 2$ and functions 1 and 2 for $n = 3$ are allocated to server 5 until function 3 for $n = 3$ causes the exceeding of T_5. Function 3 is allocated to server 2 since it is allocated to server 1 for $n = 1$; at this time function 4 cannot be allocated to server 2 since function 4 for $n = 1$ is allocated to server 2. Similarly, functions 4, 5, 6 and 7 are allocated to server 1 for $n = 3$. For $n = 4$, functions 1 and 2 are allocated to server 2 since they are allocated to server 1 for $n = 1$. Similarly, functions 3,



(a) SB model. (b) Proposed model.

Figure 4.5: Example of special and general cases.

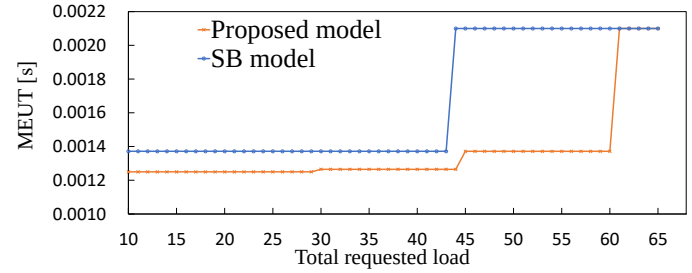
4, 5, and 6 are allocated to server 3 and function 7 is allocated to server 2.

4.4 Numerical evaluations

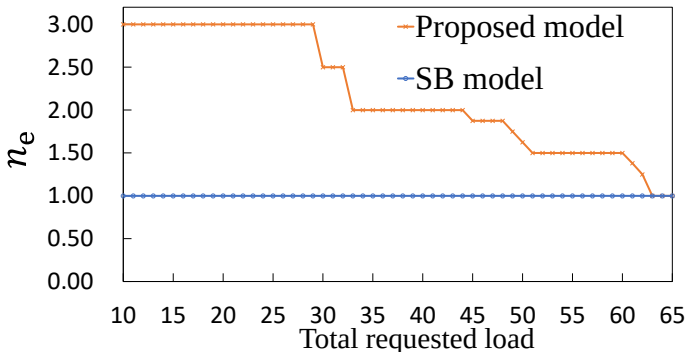
The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.7.1 [93], which is implemented by Python 3.7, using Intel Core i7-7700 3.60 GHz 4-core CPU, 32 GB memory.

4.4.1 Comparison with baseline

Let n_e denote the expected number of servers which provide protection among function $i \in F$, where $n_e = \frac{\sum_{i \in F} n_i}{|F|}$. Figure 4.5(a) shows the SB model with $n_i = 1, \forall i \in F$, which considers the backup resource allocation of each function to minimize MEUT among functions by adopting both HB and CB strategies to provide protection. Chapter 3 discuss the backup resource allocation problem with a special case that a function is only protected by one server, i.e., $n_i = 1, \forall i \in F$. It indicates that the information of a function is synchronized and recovered by only one backup server $j \in N$. The resource allocation, as shown in Figure 4.5(b), is compared with the proposed model with the SB model to investigate effectiveness of multiple protections in the proposed model. Without considering multiple protections and DPS in the SB model,



(a) Comparison on MEUT.



(b) Comparison on n_e.

Figure 4.6: Comparison between proposed and SB models.

an optimization problem can be formulated as follows:

$$\min\ r \tag{4.15a}$$

$$s.t. \sum_{b\in B}\sum_{j\in N} x_{ij}^{b} = 1, \forall i \in F, \tag{4.15b}$$

$$\sum_{b\in B}\sum_{i\in F} q_i x_{ij}^{b} \le C_j, \forall j \in N, \tag{4.15c}$$

$$r \ge r_i, \forall i \in F, \tag{4.15d}$$

$$r_i = \sum_{b\in B} t_b \sum_{j\in N} x_{ij}^{b} p_i (1-q_j) + t_3 \sum_{j\in N}\sum_{b\in B} x_{ij}^{b} p_i q_j, \forall i \in F. \tag{4.15e}$$

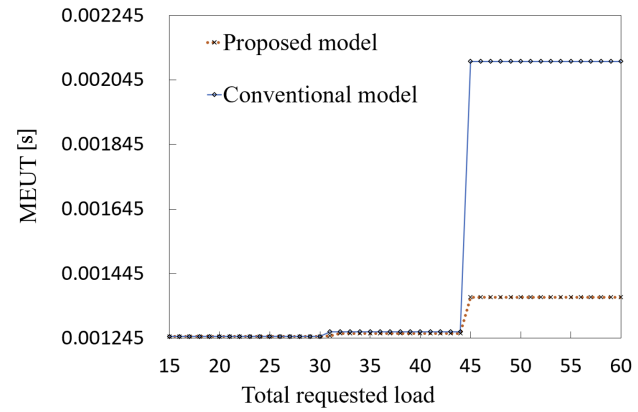
Consider three servers whose capacities and thresholds are set to 30 and 15, respectively. Both models are adopted with eight functions, whose requested loads are randomly distributed over the range of [1, 8]. This work considers that the average value of P_{L} and P_{H} for each server is no larger than 0.01 based on [99]. The work in [11] draw a conclusion that the failure probabilities

of DRAM modules in servers with higher level of utilization, as measured by CPU utilization and the amount of memory allocated, are on average 4–10 times higher than that of lower level of utilization. This work assumes that $P_{\mathrm{H}} = 7P_{\mathrm{L}}$ and $\frac{P_{\mathrm{H}}+P_{\mathrm{L}}}{2} = 0.01$. This work sets $P_{\mathrm{H}} = 0.0175$ and $P_{\mathrm{L}} = 0.0025$. Unavailable time t_1, t_0, and t_3 is set to 0.5, 3, and 20, respectively. Each server in the SB model can provide protection for only one function while each server $j \in N$ in the proposed model can provide protection for at most eight functions.

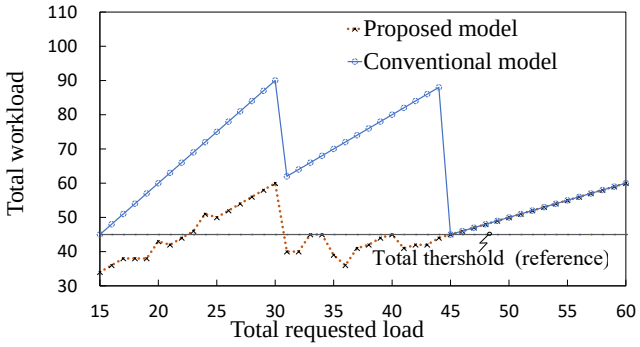
Figure 4.6 shows MEUTs and n_{e} derived by the proposed and SB models for different values of the total requested load. Figure 4.6(a) observes that the proposed model yields smaller MEUT than the SB model until the total requested load is so large that at least one function is protected by only one server whose probability is P_{H} due to the limited capacity of the corresponding server. In the simulation with given parameters, when n_{e} is smaller than 1.5 (see Fig. 4.6(b)), MEUT derived by the proposed model is equal to that of the SB model with the same total requested load. We observe that MEUTs of both models increase as the total requested load of functions increases. MEUT in the proposed model is in average 21.9% reduced compared with that of the SB model among the values of total requested load. Figure 4.6(b) observes that n_{e} in the proposed model decreases as the increase of the total requested load, while MEUTs of the proposed model increase. n_{e} in the proposed model decreases from 3 to 1 due to the limited capacity of servers as the increase of the total requested load while that of the SB model remains to be 1. As the total requested load of functions increases, n_{e} is smaller than two, which indicates that the total requested load is so large that at least one function is protected by only one server due to the limited capacity of the other servers.

4.4.2 Comparison with conventional model that ignores workload-dependent failure probabilities of functions and backup servers

The conventional model considers the multiple backup resource allocation of each function to minimize MEUT among functions to provide protection, regardless of a workload-dependent failure probability. Each function is allowed



(a) MEUT.



(b) Total workload.

Figure 4.7: Comparison of proposed model with conventional model without considering workload-dependent failure probability.

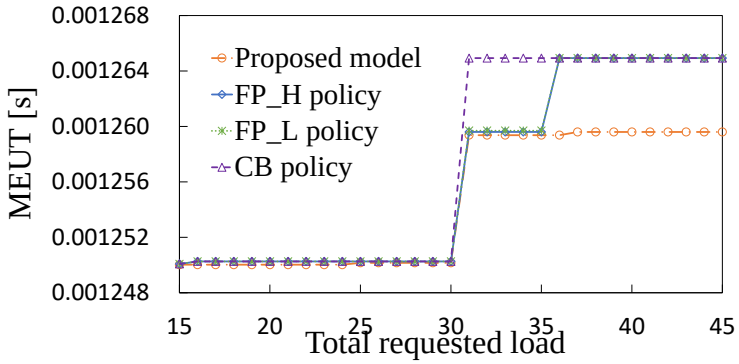


Figure 4.8: Comparison of multiple backup resource allocation model with different priority policies.

to adopt either HB or CB. This work compares the proposed model with the conventional model to investigate effectiveness of the proposed model.

Without considering the workload-dependent failure probability, the failure probability in the conventional model is set to P_{C}, since this work cannot judge the failure probability with different workloads. Since the conventional model adopting the CB strategy increases the expected unavailable time without decreasing of workload-dependent failure probability, the conventional model with the constant failure probability P_{C} adopts the HB strategy no matter what the exact value of P_{C} is. Theorem 4.1 indicates that, when each server protects the function $i \in F$ with the HB strategy, u_i can be simplified as $p_i(t_1 + (t_3 - t_1) \prod_{j \in N_i} q_j)$. Therefore, the optimization problem of conventional model is formulated as an MILP problem, expressed by:

$$\min u \tag{4.16a}$$

$$s.t. (4.4a) - (4.4d), (4.4f) - (4.4g), (4.11c) - (4.11e), (C.2i) - (C.2k), (C.2m), \tag{4.16b}$$

$$q_j = P_{\mathrm{C}}, \forall j \in N. \tag{4.16c}$$

This work obtains the multiple backup resource allocation of both models by solving the MILP problems presented in Section 4.2 and (4.16a)-(4.16c), respectively. The exact values of MEUT of both models are calculated by (4.4d) and (4.4e) by using the obtained allocations.

In order to observe the difference between the proposed and conventional models, this work sets the thresholds of three servers to 10, 15 and 20, respectively. This work sets $P_{\mathrm{H}} = 0.0175$, $P_{\mathrm{L}} = 0.0025$, and $P_{\mathrm{C}} = \frac{P_{\mathrm{L}}+P_{\mathrm{H}}}{2}$. As shown in Fig. 4.7(a), MEUT of the proposed model is in average 12.32% reduced compared with that of the conventional model, which ignores the workload-dependent failure probability among the values of total requested load by adopting both HB and CB strategies. In Fig. 4.7(b), a reference value of total threshold, which is defined by the sum of thresholds over all servers, i.e., $\sum_{j \in N} T_j$, is indicated. As shown in Fig. 4.7(b), the total workload of the proposed model is in average 24.05% reduced compared with that of the conventional model, since functions in the proposed model can adopt the CB strategy to reduce the workload and further reduce failure probability from P_{H} to P_{L}. Inspired by Theorem D.1, the expected unavailable time of function

$i \in F$ is a non-increasing function of n_i in the conventional model with the constant failure probability P_C. Therefore, each function is protected by as many servers as possible as long as the constraints are satisfied; MEUT of the conventional model is determined by the function with the smallest $n_i, i \in F$. Different from the flexible allocations of the proposed model by adopting the CB strategy to reduce the workload, the total workload obtained by the conventional model increases to keep each function being protected by as many servers as possible as long as the constraints are satisfied. When the total requested load is between 30 and 45 and the total capacity of servers is 90, the total workload decreases due to the limited capacities of servers which cannot afford more protection to the functions.

4.4.3 Comparison on different priority policies

This work evaluates the performance of multiple backup resource allocation model with different priority policies and analyze the results.

First, this work considers a multiple backup resource allocation model that sets a server with the CB strategy to the higher priority than that with the HB strategy, which this work calls a CB policy. Similar to the proposed model, the model with CB policy considers the case that function $i \in F$ fails and is protected by one of its corresponding available backup servers which have higher DPS than those which fail at failure probability $\prod_{O_{ij'} > O_{ij}: j' \in N_i} q_{j'}, j \in N_i$ simultaneously. O_{ij} in the CB policy is expressed by:

$$O_{ij} = \sum_{b \in B} (j + (1-n)|N|) x_{ij}^b, \forall i \in F, j \in N. \tag{4.17}$$

Second, this work investigates the influence of priority policy related to failure probability on MEUT. This work considers two multiple backup resource allocation models; one model sets a server whose failure probability is higher than other servers to the higher priority than other servers, which this work calls a FP_H policy; the other one sets a server whose failure probability is lower than other servers to the higher priority than other servers, which this work calls a FP_L policy. This work adopts both FP_L and FP_H policies to investigate the influence of priority policy related to failure probability

on MEUT. Similar to the model with CB policy, O_{ij} in the FP_L and FP_H policies are expressed by:

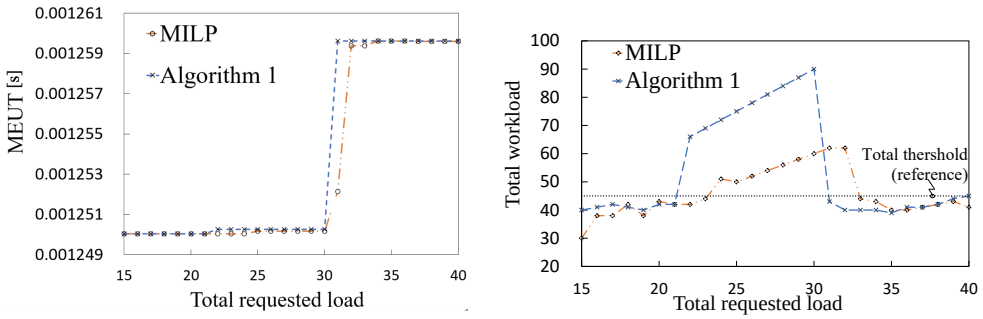
$$O_{ij} = \sum_{b \in B} (j + (1 - q_j)) x_{ij}^b, \forall i \in F, j \in N, \tag{4.18a}$$

$$O_{ij} = \sum_{b \in B} (j + q_j) x_{ij}^b, \forall i \in F, j \in N. \tag{4.18b}$$

In order to observe the difference between the proposed model and the models with different policies clearly, this work modifies unavailable time t_0 from 3 to 2, the thresholds of three servers are set to 10, 15 and 20, respectively. As shown in Fig. 4.8, MEUT in the proposed model is in average 0.22% reduced compared with that of the model with the CB policy among the values of total requested load by adopting both HB and CB strategies. MEUT in the proposed model is in average 0.148% and 0.146% reduced compared with those in the model with the FP_L and FP_H policies, respectively, among the values of total requested load.

Since setting the server with the HB strategy to the higher priority has the lower expected unavailable time than that with the CB strategy (see Theorem 2), the CB policy tends to lead to allocation results that every server in the CB policy adopts the HB strategy to protect functions with given parameters. Note that Theorem 4.2 only claims the superiority of the HB strategy on priority instead of allocation. It is possible that resources allocation and strategies may change due to the different relationships between $t_0, t_1, t_3, P_{\mathrm{L}}$ and P_{H}.

Different from the CB policy, the FP_L and FP_H policies adopt both HB and CB strategies, which reduce MEUT compared with the CB policy. As the total requested load increases, the proposed model adopts both HB and CB strategies to keep the failure probability of each server to P_{L}; the models with FP_L and FP_H policies adopt only the HB strategy. Since both FP_L and FP_H policies only focus on the failure probability, it is possible that server which is protected with the CB strategy, no matter higher failure probability or lower failure probability compared with other servers, has higher priority, which further leads to higher expected unavailable time. The proposed model is more flexible to allocate the backup resource aiming to balance the



(a) MEUT.

(b) Total workload.

Figure 4.9: Comparison of MEUTs and workloads obtained by proposed model of MILP approach and heuristic algorithm.

failure probability and strategies while setting the server which adopts the CB strategy to the lower priority than other servers with the HB strategy, which suppresses MEUT compared with the FP_L and FP_H policies.

4.4.4 computation time and accuracy

This work shows the comparisons on MEUT, workloads, and computation time between the MILP approach and the heuristic algorithm for different numbers of servers to evaluate the heuristic algorithm for different sizes of problem.

Firstly, this work shows MEUTs and workloads derived by the MILP approach and heuristic algorithm for different values of the total requested load. δ denotes the accuracy of the heuristic algorithm, which is defined by the ratio of the difference between MEUTs of the proposed model obtained by the heuristic algorithm and the MILP approach to MEUT of the MILP approach in each test. The experiment continues to apply the same given parameter in Section 4.4.3. Fig. 4.9(a) shows that MEUTs derived by Algorithm 4.1 are in average 0.0289% larger than those in the MILP approach among all the total requested load. The accuracy of the heuristic algorithm δ is 2.89×10^{-4} in average among the given total requested loads. In Fig. 4.9(b), a reference value of total threshold, which is defined by the sum of thresholds over all servers, i.e., $\sum_{j \in N} T_j$, is indicated. Fig. 4.9(b) shows the total workload obtained by the MILP approach is in average 16.82% reduced compared with that obtained by the heuristic algorithm. According to the allocation in Algorithm 4.1, the func-

Table 4.1: MEUTs, Workloads and computation time of MILP approach and heuristic algorithm.

$\|N\|$	$\|F\|$	MEUT (s)		Workload		Computation times (s)	
		MILP	Heuristic	MILP	Heuristic	MILP	Heuristic
3	8	1.2521328	1.2596562	42	36	54.75	0.006
3	12	1.2500234	1.2500241	50	54	331.94	0.008
3	16	1.2501689	1.2502612	78	117	1196.62	0.017
3	20	‡	1.2596562	‡	90	‡	0.010
4	8	1.2500241	1.2500241	48	48	892.73	0.005
4	12	1.2500004	1.2500042	75	100	3239.64	0.010
4	16	1.2500241	1.2500241	96	94	45496.05	0.010
4	20	‡	1.2593757	‡	120	‡	0.015

‡: The results cannot be obtained because of the memory limitation.

tions adopt different strategies to reduce failure probabilities. Different from the flexible allocations of the MILP approach, the total workload obtained by Algorithm 4.1 is either close to the total threshold to keep the failure probability P_{L} or much higher than the total threshold to keep each function being protected by as many servers as possible as long as the constraints are satisfied.

Secondly, this work evaluates the computation time and accuracy of MEUT for different size of problem. This work considers two situations with three servers and four servers, respectively, tests 1-8 for each situation correspond to 8 functions, 12 functions, 16 functions, 20 functions, respectively. The requested loads of functions are randomly distributed over the range of [1, 6]. This work considers servers whose capacities and thresholds are set to 20 and 12, 30 and 18, 40 and 24, 50 and 30, 60 and 24, respectively, in each test. Unavailable time t_1, t_0, and t_3 is set to 500, 2000, and 20000, respectively, for easier observation.

Table 4.1 shows the average computation time for the MILP approach and the heuristic algorithm for each situation. We observe that the computation

Table 4.2: Lower bound of optimal objective value in proposed model and upper bound of expected unavailable time obtained by heuristic algorithm.

$\|N\|$	Lower bound	Upper bound
4	3.87×10^{-12}	0.01469
5	9.68×10^{-15}	0.01451
6	2.42×10^{-17}	0.00838
7	6.05×10^{-20}	0.00823
8	1.51×10^{-22}	0.00823
9	3.78×10^{-25}	0.00799
10	9.45×10^{-28}	0.00784

time of the heuristic algorithm is 8.70×10^{-6} times smaller than that of the MILP approach in average; as the problem size increases, the computation time of the MILP approach increases remarkably. The larger the problem size is, the more the computation time of the heuristic algorithm is reduced compared with that of the MILP approach. MEUTs derived by the heuristic algorithm is 0.122% larger than the results calculated by the MILP approach in average.

Thirdly, in order to evaluate the dependency of MEUT obtained by the developed heuristic algorithm on $|N|$ when $|N|$ increases, this work considers that there are $|N|$ servers hosting 30 functions, where $|N|$ is set to 4, 5, 6, 7, 8, 9, and 10. The capacities of $|N|$ servers are randomly distributed over the range of [50, 100]. The ratios of the threshold to the capacity of a server are randomly distributed over the range of [0.4, 0.8]. The requested loads of 30 functions are randomly distributed over the range of [1, 10]. Since the MILP problem becomes difficult to solve in a practical time as the problem size increases, let $\boldsymbol{T}$ denote the admissible computation time (seconds) in the experiment. This work considers the best feasible solution obtained by solving the MILP problem within $\boldsymbol{T}$, where $\boldsymbol{T}$ is set large enough for running any heuristic algorithm.

Figure 4.10 shows the relationship between MEUTs and $|N|$ by applying the MILP approach and the heuristic algorithm. In Fig. 4.10, two reference values,

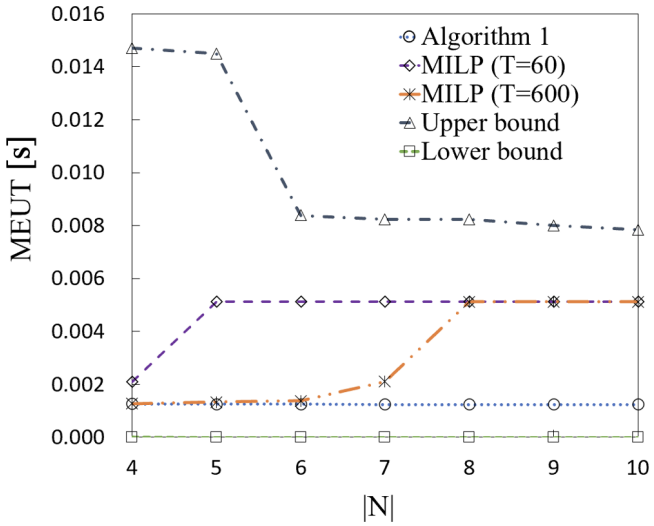


Figure 4.10: Dependency of MEUT on $|N|$.

which are the lower bound of the optimal objective value in the proposed model and the upper bound of the expected unavailable time obtained by the algorithm (see Theorems D.1 and D.3), are indicated. The calculated values of the bounds are shown in Table 4.2. The lower bound decreases and the upper bound increases as $|N|$ increases. Within T = 60 and 600 [s], the heuristic algorithm always outperforms the MILP approach in terms of MEUT. As the value of T increases, MEUT in the best feasible solution obtained by solving the MILP problem decreases. The computation time to solve the problem by the heuristic algorithm among different resource utilization under different $|N|$ are almost the same and are approximately 0.1 [s].

4.4.5 Larger-size problem

In order to evaluate the performance of the proposed model for a larger-size problem, this work considers a larger-size network with 100 middleboxes, or $|F|$ = 100. This work uses Google cluster usage traces dataset [100], where each function is accompanied by a set of resource requirements used for scheduling tasks onto the appropriate servers. Google dataset describes particular information on machines and their attributes, tasks events, service events, and resource usage. The requested load for each function represents the maximum

Table 4.3: MEUTs for larger-size problem obtained by heuristic algorithm under different failure probabilities

(P_L, P_H) \ $\lvert N\rvert$	20	30	40	50
Test 1 (0.0025,0.0175)	1.372×10^{-3}	1.259×10^{-3}	1.251×10^{-3}	1.250×10^{-3}
Test 2 (0.01,0.1)	2.180×10^{-2}	5.168×10^{-3}	5.150×10^{-3}	5.001×10^{-3}
Test 3 (0.05,0.2)	7.375×10^{-2}	3.100×10^{-2}	2.886×10^{-2}	2.520×10^{-2}

amount of CPU that an instance is permitted to use are described in [100]. The capacities of the servers are also obtained by [100]. The measurements have been normalized, where the normalization is a scaling relative to the largest capacity of the resource on any machine in the trace. The ratios of the threshold to the capacity of a server are randomly distributed over the range of [0.4, 0.8]. This work conducts three tests with different failure probabilities of P_L and P_H, where P_L is set to 0.0025, 0.01, and 0.05, respectively; P_H is set to 0.0175, 0.1, and 0.2, respectively.

The failure probabilities in three tests have the relationship of $P_H = 7P_L$, $P_H = 10P_L$, and $P_H = 4P_L$, respectively. MEUTs derived by the heuristic algorithm increase as $|N|$ increases for different values of P_L and P_H; MEUTs increase as the values of P_L and P_H increase for different sizes of the problem. MEUTs with $|N| = 50$ are 9.8%, 335.9%, and 192.7% larger than those of with $|N| = 20$ in Tests 1, 2, and 3, respectively. This result is positively related to the ratio of P_H to P_L.

When relatively higher-reliable servers are considered in Test 1, MEUT with $|N| = 50$ is 0.1% smaller than that of $|N| = 30$; when relatively lower-reliable servers are considered, MEUTs with $|N| = 50$ are 3.4% and 23.0% smaller than those of $|N| = 30$ in Tests 2 and 3, respectively. Thus, we observe that MEUT reduces remarkably as $|N|$ increases in relatively lower-reliable servers compared with the relatively higher-reliable servers. The average computation time among the three tests for different sizes of the problem are 0.5 [s], 2.7 [s], 6.9 [s], and 13.7 [s], respectively.

4.5 Discussions

4.5.1 Considerations related to network service

A network service consists of one or more multiple middlebox functions, and transmits network elements including switches, routers, and links. The service-level unavailability is caused by the unavailability of functions and network elements used in a network service. Similar to this work, the works in [28–30, 56] focus on the availability of functions, but not the availability of link that is used to transmit traffic among VNFs. The work in [65] introduced that the availability of requests is guaranteed by both physical network layer components (links) and application layer components (NFs). In practical applications, physical links provided in the electrical or optical domain may also fail. In literature, several approaches can be adopted to protect logical links against physical link failures. In [101–103], VNF chaining with path or link protection was addressed. For example, this work can allocate two disjoint paths, where one of them works as a primary path and the other is a backup path, for each logical link against any single physical link failure [103]. This work assumes that the network links are provided by reliable manufacturers with the approaches introduced in the literatures. This work focuses on function-level protection for functions with considering the failures of functions and servers. The primary and backup path allocation problem can be incorporated in the optimization model to determine the path allocation at the same time. This work can be used for estimating the service-level unavailability by using the function-level unavailability [65].

4.5.2 Considerations related to workload-independent failures

The workload-independent failures occur in a practical situation, which are not related with workload of servers. There are several kinds of workload-independent failures, such as security breaches, misconfigurations, and human errors. Some approaches can handle some kinds of workload-independent failures, for example, syslog-based failure diagnosis or prediction usually employs machine learning techniques to discover patterns from historical failures [104].

The work in [105] investigated datacenter placement and dynamic content management through proactive and reactive approaches.

The workload-dependent failures are highly related to the utilization of computing resources [8–10, 50, 60, 106, 107]. This work introduces the relationship between the workload and the failure probability to perform the backup resource allocation [37,95]. In a practical situation, both workload-independent and workload-dependent failures may occur simultaneously with complex reasons. In this situation, the model in this work can be independent of the existing works focusing on different types of workload-independent failures, so that workload-independent failures and workload-dependent failures can be jointly handled by different approaches to achieve a higher availability.

4.5.3 Considerations related to practical situation

In practical deployment, this model can be used for VNF orchestration. VNFs are usually packaged in VMs or containers, which is placed to a node determined by an orchestration management platform such as Kubernetes [108]. This work introduces an example to implement the proposed model in a practical situation.

Step 1 (input collection): the inputs of the proposed model include the requested computing resources for each function and the maximum computing capacity for each server. The computing resource information can be derived by tracing applications [3, 4], or a monitoring cloud application, e.g., Prometheus [109]. The applications allow capturing the runtime behavior of the system, enabling a thorough analysis of the collected data, which include resource utilization and available status.

Step 2 (optimization for backup resource allocation): the allocation results are obtained by the proposed model with the collected data to minimize MEUT. The calculation results obtained by the proposed model are stored in a database.

Step 3 (resource binding and Pod creation): a Pod is the smallest deployable unit of computing that can be created and managed in Kubernetes. Functions can be bound to backup servers and Pods can be created manually (Step 3-1) or automatically [35] (Step 3-2) according to the optimization results in Step 2.

Step 3-1 (manually): the service provider manually binds the functions to the corresponding server and creates the Pods according to the calculation results in Step 2.

Step 3-2 (automatically): the proposed model can be implemented based on the tool introduced in [35]. The tool in [35] enables the allocation results obtained by a general model with different objectives and designs to be used to allocate the VNFs to nodes in Kubernetes. In automatic orchestration management, this model serves as a part of function scheduler. The scheduler can allocate functions according to the calculation results obtained by Step 2 based on the scheduler framework in Kubernetes. Adopting the proposed model as a part of scheduler enables the scheduler to evaluate the score of each server according to the suitability for each function by minimizing MEUT. The scheduler in the platform [35] binds the functions to the corresponding server and creates the Pods.

Step 4 (backup Pod label attachment): HB and CB Pods are distinguished by the labels, each of which is attached to the Pod when a service is created. Each Pod with a HB label is activated without being exposed to the service; the Pod takes over the tasks of primary Pod once a failure is detected. When a Pod with a CB label is requested to be created, this work adds an *init* container [110] in the CB Pod. Init containers run before the other containers in the same Pod. Each init container must be completed successfully before the activation of other containers in the same Pod; it is used for waiting for the activation message.

Step 5 (initialization): the Pods start for initialization. The orchestration platform pulls the images of the functions from the repositories to the locations of the functions for both HB and CB Pods. The HB Pods are activated and synchronized with the initial data while the CB Pods are not. After the function is initialized and the service begins to run, the HB Pods synchronize the data from the primary functions periodically; the images of the CB pods are updated by the function orchestration platform periodically.

A demonstration was conducted for a real situation with cooperating the work in [35]. The demonstration runs on a four-node Kubernetes cluster, one master node and three worker nodes. There are four functions given in advance. The required time is measured for implementing the backup resource

allocation model. The time for the proposed model to obtain an optimal solution is 1.42 [s]. The time for binding a function to the server according to the resource allocation obtained by the proposed model is 1.05 [s]. It depends on the number of servers and functions and processing ability of Scheduler. The time for pulling a nginx image from docker.io is 12.43 [s]. The time for the hot backup resources of the functions for synchronizing the initial data, which are default web pages, from the primary resource is 0.17 [s]. It depends on the synchronization content size and the bandwidth between the primary resource (source) and the backup resource (destination).

4.5.4 Considerations related to resource usage

Service providers may also need to consider the resource usage in addition to the unavailable time. There are several works jointly considered the reliability and the resource usage. The work in [19] provided a probabilistic protection guarantee for virtual machines against multiple failures of physical machines to minimize the required total capacity. The work in [58] introduced a cost-aware redundancy scheme to maximize the cost-efficiency ratio of the backup plan with backup cost. The work in [111] introduced a deployment of backup instances, which uses an improved breadth first search algorithm to reduce the consumption of backup bandwidth resources during the deployment of backup instances. Different from the above works, the proposed model does not consider the cost of resource usage in the objective function; all the given resources can be used without cost. The capacity of each server is given as a constraint in the proposed model. The proposed model minimizes MEUT with different strategies only considering the failure probability depending on the workload, regardless of the cost on resource usage. The resources can be fully utilized in terms of MEUT.

This work provides two directions to extend the proposed model with considering resource usage. In the first direction, MEUT and the total workload among servers can be jointly considered in the objective value. Firstly, in (4.4a), the total workload, i.e., $\sum_{j \in F} W_j$, can be added as the secondary objective. The complete objective value can be expressed by $u + \epsilon_1 \sum_{j \in F} W_j$, where u is MEUT; the small number, ϵ_1, is multiplied to the second term to prioritize

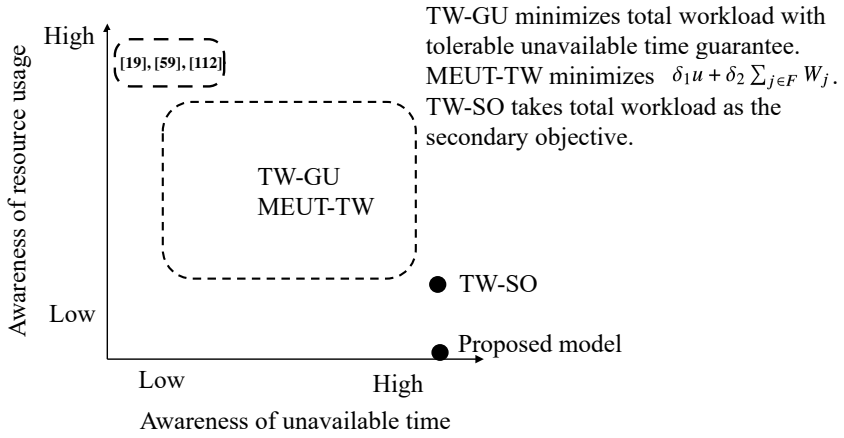


Figure 4.11: Comparison among models with different objectives.

the first term over the second term. The solution that minimizes the total workload, $\sum_{j \in F} W_j$, is chosen when there are multiple solutions that minimize MEUT. Similarly, the two objective values, i.e., MEUT and the total workload, can be incorporated with weights δ_1 and δ_2, respectively, each of which depends on the importance of each term designed based on the requirement of the provider. The goal of the model is to minimize a weighted value considering MEUT and the total workload, i.e., $\delta_1 u + \delta_2 \sum_{j \in F} W_j$. In the second direction, this work can provide a guarantee of tolerable unavailable time for MEUT, i.e., this work can impose that the value of MEUT must be under a certain value as a constraint, and minimize the total workload, $\sum_{j \in F} W_j$.

This work calls a model that minimizes the objective with adopting the total workload as the secondary objective TW-SO. This work calls a model that minimizes a weighted value considering MEUT and the total workload MEUT-TW. This work calls a model that minimizes the total workload with guaranteeing the unavailable time TW-GU.

Figure 4.11 shows the comparison among models on their features of resource usage and unavailable time. Each model with a different objective considers the unavailable time and resource usage in a different degree. As shown in Fig. 4.11, the proposed model is aware of unavailable time without

considering the cost of resource usage. The works in [19, 58, 111] are aware of both resource usage cost and reliability; they are not explicitly aware of unavailable time. TW-SO maintains the awareness of unavailable time compared with the proposed model while it increases the awareness on resource usage with adopting the total workload as secondary objective. The awareness of the unavailable time and the resource usage can be changed depending on parameter settings on weights δ_1 and δ_2 for MEUT-TW. For example, when $\delta_1 = 0$, the modified model focuses only on the resource usage; when $\delta_2 = 0$, the modified model focuses only on the unavailable time. For TW-GU, as the tolerable unavailable time constraint relaxes, the model can have a larger searching space to minimize the total workload. For example, when the tolerable unavailable time becomes large enough so that the time constraint can be easily satisfied, the model can have a larger feasible region, which can lead to obtaining a smaller objective value, and vice versa.

4.5.5 Considerations related to resource re-allocation in dynamic scenarios

The proposed model provides the multiple backup resource allocation under a static scenario, where all the requested functions are given at the operation start time. The proposed model can be used in some real application scenarios, such as initial optimal deployment for network services before volume or phased roll-out [112] and periodic optimal re-deployment for maintenance or upgrading.

In the dynamic scenarios of arrivals of requested functions and releases of existing functions, resources of arriving functions need to be allocated; the resources of existing functions may not need to be re-allocated immediately. Only the allocation of the arriving requested functions and the releasing of terminated functions are considered at each request arrival and release. A triggered re-optimization procedure including the existing resources performs at a certain interval to reduce MEUT; the interval can be set considering the cost of re-allocation and optimality. One option is to perform re-allocation periodically, where an appropriate re-allocation interval is required to be determined. The other option is to set a threshold for the total workload; the

re-allocation is triggered when the total workload exceeds the given threshold.

4.5.6 Considerations related to multiple types of computing resources

Since multiple types of computing resources can become a key role for services or applications at the same time in actual clouds [113], the backup resource allocation with considering multiple types of computing resources, e.g., CPU and memory, is required to be considered. In the view of multiple resource types, different computing resources may have different thresholds and different workload-probability curves. To handle the multiple types of resources with the proposed model, since the concept of the workload in the proposed model represents only one computing resource, the utilization of multiple types of resource needs to be redefined jointly as one type of new resource, so that it can be directly handled by the proposed model. To combine the utilization of multiple resources as one, this work can calculate the average or maximum failure probability among multiple resource types to approximate the workload-dependent failure probability with multiple resource types. The approximation to obtain the joint probability of CPU-related and memory-related failure rate by directly combining the two types of resources as one may lead to inaccuracy, since different types of resources may have different workload-probability curves and the failures of multiple resources may not be independent.

Thus, this work discusses a direction to extend the proposed model with considering the multiple resource types. The workload can be extended to be a multi-dimension concept. In the proposed model, this work considers that the workload is one type of the computing resources with one dimension. The non-decreasing workload-dependent failure probability is expressed with a two-dimensional curve, i.e., the failure probability and the workload. If this work extends the workload concept from one dimension to α dimensions, i.e., this work considers α types of computing resources, a non-decreasing workload probability can be expressed with an $(\alpha+1)$-dimensional curve. The proposed model can be extended to handle a non-decreasing workload probability that is expressed with an $(\alpha+1)$-dimensional curve.

4.6 Summary

This chapter proposed a multiple backup resource allocation model with the workload-dependent failure probability to minimize MEUT under a priority policy. This chapter analyzed the superiority of the protection priority policy to express the expected unavailable time for each function protected by multiple backup resources in the proposed model. This chapter derived the theorems that clarify the influence of policies on MEUT. This chapter formulated the optimization problem as an MILP problem. A lower bound of the optimal objective value in the proposed model was derived. This chapter proved that the decision version of the multiple resource allocation problem in the proposed model is NP-complete. A heuristic algorithm inspired by the water-filling algorithm was developed with providing an upper bound of the expected unavailable time obtained by the algorithm. The numerical results showed that the proposed model reduces MEUT compared with the single backup model in which each function is protected by only one server without protection priority of servers and the conventional model without the workload-dependent failure probability. The priority policy adopted in the proposed model specifying that the server which adopts the HB strategy has higher priority than that with the CB strategy for multiple backup resources suppresses MEUT compared with other priority policies. The developed heuristic algorithm is approximately 10^6 times faster than the MILP approach with 10^{-4} performance penalty on MEUT.

Chapter 5

Resource allocation model with priority setting against multiple failures with workload-dependent failure probability

This chapter proposes a primary and backup resource allocation model with preventive recovery priority setting to minimize a weighted value of unavailable probability (W-UP) against multiple failures. W-UP considers the probability of unsuccessful recovery and the maximum unavailable probability after recovery among physical nodes [114, 115].

The remainder of the chapter is organized as follows. Section 5.1 describes the proposed model. Section 5.2 presents a heuristic algorithm. Section 5.3 discusses an approach to obtain the unsuccessful recovery probability without priority setting Section 5.4 presents the numerical results. Section 5.5 summarizes this chapter.

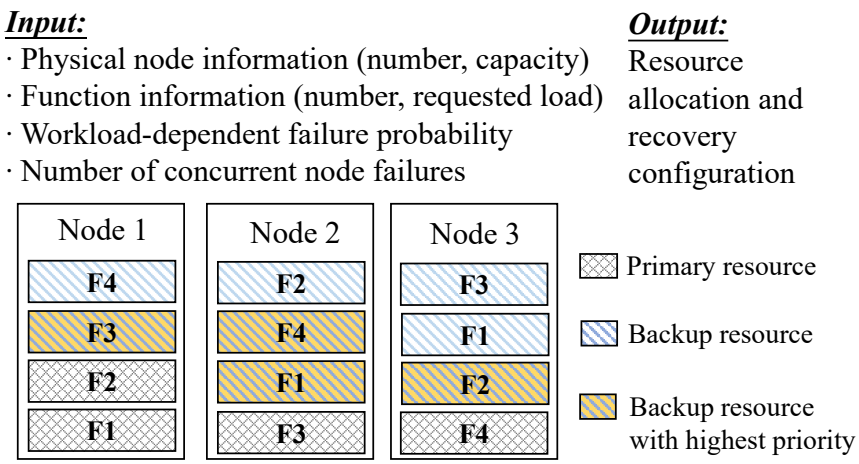


Figure 5.1: Example of resource allocation problem.

5.1 Model and problem definition

5.1.1 Problem definition

This work determines a primary and backup resource allocation model with recovery priority setting against multiple failures preventively with considering different workload-dependent failure probabilities for each node. The goal of the model is to minimize a weighted value considering the unsuccessful recovery probability and the weighted maximum unavailable probability among nodes against all failure patterns.

RAM-P problem is defined as follows:

Problem 5.1 *Given the sets of functions and nodes, the requested load for each function, the maximum capacity for each node, the maximum number of concurrent node failures, and the workload-dependent failure probability for each node, the goal is to find the primary and backup resource allocation and the priority for recovery, aiming to minimize W-UP.*

Figure 5.1 shows an example of the considered problem, where the primary and backup resource allocation with priority setting against node failures is presented. The recovery configuration against any failure pattern is determined at the operation start time with minimizing W-UP.

5.1.2 Model description

Let F and N represent a set of functions and a set of nodes, respectively, where $|F|$ and $|N|$ denote the numbers of functions and nodes, respectively. Consider $N_j \subseteq N$, where $|N_j| \geq 1$, as a set of nodes which are assigned to allocate backup resources of function $j \in F$. This work considers that each node is able to offer capacity for both primary and backup resource allocation; the backup resources of a function are allocated in $|N_j| \geq 1$ different nodes. Let C_i denote the upper bound of computing capacity provided by node $i \in N$ for operating the primary and backup resources.

When a failure occurs in a node, the functions hosted by the failed nodes fail concurrently, the workload of the primary resource of the function is transferred to one of its backup resources in available nodes. Each node in N_j is associated with an integer of $p \in [1, |N|]$, which indicates the priority of the backup resource in a node to recover unavailable function j. The backup resource in node $k \in N_j$ with a greater value of p has a higher priority to recover unavailable function j.

Let x_{ij}^{kp}, $i \in N, k \in N\backslash\{i\}, j \in F, p \in [1, |N|]$, denote a binary variable; x_{ij}^{kp} is set to one if function j is allocated at node i and is protected by node k whose priority among nodes in $N\backslash\{i\}$ is equal to p, and zero otherwise. Each function protected by node $k \in N$ is allocated to at most one node in $N\backslash\{k\}$ associated with priority p; each function protected with a priority is allocated to at most one node in N and protected by another node in N.

Let η_j^* be an integer variable which denotes the maximum priority among nodes in N to recover function $j \in F$; it is expressed by:

$$\eta_j^* = \max_{p \in [1,|N|]} \max_{k \in N} \{\sum_{i \in N} p x_{ij}^{kp}\}, \forall j \in F. \tag{5.1}$$

Let $\rho_j^p, j \in F, p \in [1, |N|]$, denote a binary variable; it is set to one if priority p equals $p = \eta_j^*$, and zero otherwise. Let $\theta_{ij}^{kp} = x_{ij}^{kp} \rho_j^p, i \in N, j \in F, k \in N\backslash\{i\}, p \in [1, |N|]$, denote a binary variable which is set to one if node k has the highest priority among nodes in $N\backslash\{i\}$ to recover function j that is allocated at node i, and zero otherwise.

Let $\chi_{ij}, i \in N, j \in F$, represent a binary variable that equals $\vee_{p \in [1,|N|]} \vee_{k \in N\backslash\{i\}} x_{ij}^{kp}$; it is set to one if function j is allocated to node i, and zero otherwise,

where $\vee$ expresses a binary OR. Each function in F is allocated into one node in N with $\sum_{i \in N} \chi_{ij} = 1, \forall j \in F$. Let $\xi_{jk}, j \in F, k \in N$, denote a binary variable that equals $\vee_{p \in [1,|N|]} \vee_{i \in N \setminus \{k\}} x_{ij}^{kp}$; it is set to one if function j is protected by node k, and zero otherwise.

Multiple concurrent failures among nodes in N are considered. This work assumes that each node failure occurs independently with its corresponding probability. Since a service provider may allow a certain number of concurrent node failures, where the total unavailable probability of nodes is controlled under a certain degree [71], this work considers that there are at most $\Gamma \in [0, |N|]$ failed nodes at the same time. The larger Γ is, the more the considered possible failure patterns of nodes are. Let $\mathcal{P}_\Gamma$ denote a set containing all possible failure patterns, each of which has at most Γ failed nodes. Let $\mathcal{P}_\Gamma^\sigma, \sigma \in \mathbb{C} := [0, |\mathcal{P}_\Gamma|]$, denote the σth element in $\mathcal{P}_\Gamma$; when $\sigma = 0$, $\mathcal{P}_\Gamma^\sigma = \emptyset$. Let $f_i^\sigma \in \{0, 1\}, i \in N, \sigma \in \mathbb{C}$, represent whether node i fails; $f_i^\sigma = 1$ if node i fails, i.e., $i \in \mathcal{P}_\Gamma^\sigma$, and zero otherwise.

The failure probability of each node is related to the corresponding workload. Before any failure occurs, each function is protected by one or more nodes and the reserved idle capacity is not taken into account as active workload in the nodes until the activation of backup resources caused by the failures. When a failure occurs to node i, the functions hosted by failed nodes need to be recovered by an available node; the backup resource in node $k \in N \setminus \{i\}$ with the highest priority is activated. Let $L_{i\sigma}^{\mathrm{W}}, i \in N, \sigma \in \mathbb{C}$, denote the workload of node i under failure pattern $\mathcal{P}_\Gamma^\sigma$. When $\sigma = 0$, L_{i0}^{W} represents the workload of node i before any failure occurs. Let L_i^{R} denote total requested load of node i. This work assumes that the upper bound of computing capacity for each function required for information synchronization and snapshot updating is $l_j^{\mathrm{u}}, j \in F$; l_j^{R} denotes the requested load of each function. $L_{i\sigma}^{\mathrm{W}}$ and L_i^{R} are expressed by:

$$L_{i\sigma}^{\mathrm{W}} = \sum_{j \in F} \{l_j^{\mathrm{R}} \chi_{ij} + l_j^{\mathrm{u}} \xi_{ji}\} + \sum_{j \in F} \sum_{i' \in N \setminus \{i\}} \sum_{p \in [1,|N|]} (l_j^{\mathrm{R}} - l_j^{\mathrm{u}}) \theta_{i'j}^{ip} f_{i'}^\sigma \chi_{i'j}, \forall i \in N,$$
$$\sigma \in \mathbb{C}, \tag{5.2a}$$

$$L_i^{\mathrm{R}} = \sum_{j \in F} \{l_j^{\mathrm{R}} \chi_{ij} + l_j^{\mathrm{R}} \xi_{ji}\}, \forall i \in N, \tag{5.2b}$$

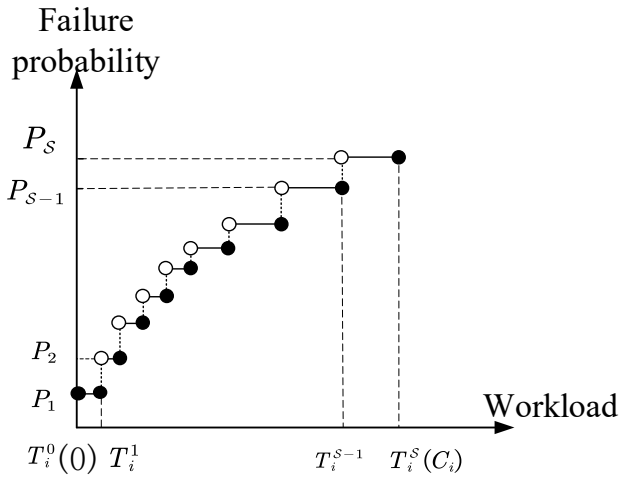


Figure 5.2: Workload-dependent failure probability is expressed by a monotone increasing $\mathcal{S}$-step function.

where $\sum_{j\in F} l_j^{\mathrm{R}} \chi_{ij}$ is the workload for the primary resource; $\sum_{j\in F} l_j^{\mathrm{u}} \xi_{ji}$ is the total requested capacity for functions protected by node i for updating procedure. The functions hosted by failed nodes fail concurrently, $\sum_{i'\in N\setminus\{i\}} \sum_{j\in F} \sum_{p\in[1,|N|]} (l_j^{\mathrm{R}} - l_j^{\mathrm{u}}) \theta_{i'j}^{ip} f_{i'}^{\sigma} \chi_{i'j}$ is the workload for the activated function caused by failure $f_{i'}^{\sigma}$ since node i has the highest priority among nodes in N for which provide protection for function j; $\sum_{j\in F} l_j^{\mathrm{R}} \xi_{ji}$ is the total requested capacity for functions protected by node i for recovery procedure which is reserved in advance.

Equation (5.1), χ_{ij}, ξ_{jk}, and $\theta_{ij}^{kp}, i \in N, j \in F, k \in N\setminus\{i\}, p \in \mathbb{P}$, are linearized to (F.1a)-(F.1f), (F.2a)-(F.2f), and (F.3a)-(F.3g).

The workload-dependent failure probability of node in failure pattern σ is denoted by $q_{i\sigma}$, $i \in N, \sigma \in \mathbb{C}$. When $\sigma = 0$, q_{i0} represents the failure probability of node $i \in N$ before any failure occurs. Given the workload-dependent failure probability, which is assumed to be a non-decreasing function, this work can use an $\mathcal{S}$-step function, where $\mathcal{S} \geq 2$, to conservatively approximate a non-decreasing function of the node workload, as shown in Fig. 5.2; $\mathcal{S}$ denotes the number of steps in a step function. $q_{i\sigma}$ is expressed by the following $\mathcal{S}$-step function; let $s \in \mathbb{S} = [1, \mathcal{S}]$ denote the sth step in $\mathcal{S}$-step workload-dependent failure probability:

$$q_{i\sigma} = \begin{cases} P_1, & T_i^0 \leq L_{i\sigma}^{\mathrm{W}} \leq T_i^1 \\ P_2, & T_i^1 < L_{i\sigma}^{\mathrm{W}} \leq T_i^2 \\ \quad \vdots & \quad \vdots \\ P_{\mathcal{S}}, & T_i^{\mathcal{S}-1} < L_{i\sigma}^{\mathrm{W}} \leq T_i^{\mathcal{S}}, \end{cases} \tag{5.3}$$

where $T_i^0 = 0$ and $T_i^{\mathcal{S}} = C_i, i \in N$. When the workload of a node increases from the range of $(T_i^{s-1}, T_i^s]$ to $(T_i^s, T_i^{s+1}]$, the failure probability increases from P_s to P_{s+1}. As the workload increases, even though the node has remaining capacity, the node becomes fragile, and has a higher failure probability to handle the extra workload. This work considers the workload-dependent failure probability $q_{i\sigma}$ of each physical node so that both failure probabilities of primary and backup resources of a function hosted by the node are the same as the node failure probability. By introducing binary variable $z_{is}^{\sigma}, i \in N, s \in \mathbb{S}, \sigma \in \mathbb{C}$, which is set to one if $T_i^{s-1} < L_{i\sigma}^{\mathrm{W}} \leq T_i^s$, and zero otherwise, (5.3) is linearized to (11a)-(11e) in [114] with several auxiliary variables.

5.1.3 Objective value

Probability of each failure pattern

Since each node fails with the workload-dependent failure probability, failure pattern P_{Γ}^{σ} occurs with a certain probability $w_{\sigma}, \sigma \in \mathbb{C}$, as a weight of the unavailable probability for each failure pattern. w_{σ} is affected by the failure probability of each node before any failure occurs; it can be expressed by:

$$w_{\sigma} = \prod_{i \in \mathcal{P}_{\Gamma}^{\sigma}} q_{i0} \prod_{i \in N \setminus \mathcal{P}_{\Gamma}^{\sigma}} (1 - q_{i0}), \forall \sigma \in \mathbb{C}, \tag{5.4}$$

where q_{i0} is the failure probability of node i before any failure occurs.

Unsuccessful recovery probability

Since all the functions hosted by failed nodes may not be recovered simultaneously by the remaining available nodes, this work considers the probability of unsuccessful recovery. The unavailable functions can be recovered by the corresponding backup resource as long as one of the backup resources protecting the function does not fail and the workload of any node after recovery does not

exceed the capacity under a failure pattern. Thus, a function hosted by failed nodes may not be recovered, if all the nodes hosting the primary and backup resources of the function fail concurrently. Let $e_j^\sigma, j \in F, \sigma \in \mathbb{C}$, be a binary variable; it is set to one if the number of nodes in $\mathcal{P}_\Gamma^\sigma$ hosting the primary and backup resources of function j is equal to the number of nodes in N hosting the primary and backup resources of j, and zero otherwise. In a mathematical expression, e_j^σ is set to one if

$$\sum_{i \in N} (\chi_{ij} + \xi_{ji}) = \sum_{i \in \mathcal{P}_\Gamma^\sigma} (\chi_{ij} + \xi_{ji}), j \in F, \sigma \in \mathbb{C}. \tag{5.5}$$

Further, if the workload of any node after it recovers the unavailable functions hosted by the failed node exceeds the corresponding capacity, at least one function hosted by failed nodes may not be recovered. Let $g_i^\sigma, i \in N, \sigma \in \mathbb{C}$, be a binary variable; it is set to one if the workload $L_{i\sigma}^{\mathrm{W}}$ of node i exceeds the capacity C_i under failure pattern $\mathcal{P}_\Gamma^\sigma$, and zero otherwise.

Let $r_\sigma, \sigma \in \mathbb{C}$, denote a binary variable; it is set to one if at least one function hosted by failed nodes under failure pattern $\mathcal{P}_\Gamma^\sigma$ cannot be recovered, and zero otherwise. The total unsuccessful recovery probability among all failure patterns can be expressed by:

$$\mathcal{U}_1 = \sum_{\sigma \in \mathbb{C}} w_\sigma r_\sigma = \sum_{\sigma \in \mathbb{C}} w_\sigma \left(\vee_{j \in F} e_j^\sigma \vee_{i \in N} g_i^\sigma \right). \tag{5.6}$$

Maximum unavailable probability after recovery

With the workload variation after failure, against each failure pattern, this work considers the maximum unavailable probability among nodes after recovering the functions hosted by failed nodes. The maximum unavailable probability is weighted by the probability of each failure pattern. Let $Q_\sigma, \sigma \in \mathbb{C}$, denote the maximum unavailable probability of node $i \in N \backslash \mathcal{P}_\Gamma^\sigma$ after the available backup resource with the highest priority is activated. The maximum unavailable probability among nodes is given by $Q_\sigma = \max_{i \in N \backslash \mathcal{P}_\Gamma^\sigma} q_{i\sigma}$, where Q_σ is related to the workloads of nodes, which is affected by the determined priorities in given failure pattern $\mathcal{P}_\Gamma^\sigma$.

Against all failure patterns, this work considers the weighted maximum unavailable probability among nodes after failure recovery, which can be ex-

pressed by:

$$\mathcal{U}_2 = \sum_{\sigma \in \mathbb{C}} w_\sigma Q_\sigma. \tag{5.7}$$

5.1.4 Optimization problem

The optimization problem to allocate the primary and backup resources, RAM-P, is formulated as follows:

$$\min \ \delta \mathcal{U}_1 + (1-\delta)\mathcal{U}_2 \tag{5.8a}$$

$$\sum_{p \in [1,|N|]} \sum_{i \in N \setminus \{k\}} x_{ij}^{kp} \leq 1, \forall j \in F, k \in N, \tag{5.8b}$$

$$\sum_{i \in N} \sum_{k \in N \setminus \{i\}} x_{ij}^{kp} \leq 1, \forall j \in F, p \in [1, |N|], \tag{5.8c}$$

$$\sum_{i \in N} \chi_{ij} = 1, \forall j \in F, \tag{5.8d}$$

$$(5.1), (5.2a), (5.3), (5.4) - (5.7), \tag{5.8e}$$

$$x_{ij}^{kp} \in \{0,1\}, \forall i \in N, k \in N \setminus \{i\}, j \in F, p \in [1, |N|]. \tag{5.8f}$$

Equation (5.8a) minimizes W-UP. $\mathcal{U}_1$ and $\mathcal{U}_2$ are two objective values with weights δ and $1-\delta$, respectively, where $0 \leq \delta \leq 1$. Each weight depends on the importance of each term designed based on the requirement of the provider. Equation (5.8b) ensures that each function protected by node $k \in N$ is allocated to at most one node in $N \setminus \{k\}$ associated with priority p. Equation (5.8c) ensures that each function protected with a priority is allocated to at most one node in N and protected by another node in N. Equation (5.8d) ensures that each function in F is allocated into one node in N.

According to the linearization process in (F.4a)-(F.4e), (F.5a)-(F.5f), (F.6a)-(F.6f), (F.7a)-(F.7e) of Appendix F and in [114], this work formulates the problem as an MILP problem.

The proposed model is designed to preventively determine the initial resource allocations and recovery configuration against multiple node failures before services run by considering the probability of each failure pattern.

5.2 Heuristic algorithm

Similar to [71], when $\Gamma = 0$, the decision version of RAM-P is an NP-complete problem by reducing the generalized load balancing (GLB) problem [116], which is a well-known NP-complete problem. It indicates that the RAM-P becomes difficult to solve in a practical time as the problem size increases. This section presents an idea of heuristic algorithm which can obtain an approximate solution of RAM-P with a larger problem size in a practical time.

5.2.1 Developed heuristic algorithm

This work presents an initial workload-aware greedy algorithm (WAGA). The algorithm determines an initial primary and backup resource allocation with considering the workload-dependent failure probability. It avoids uneven failure probabilities among all nodes by evenly utilizing a part of capacity in a node with corresponding different failure probabilities.

In addition to considering the ordered failure probability of nodes for allocating the functions, the algorithm considers three aspects for allocation. Firstly, it evenly distributes the primary resource of each function to nodes so that the number of the functions hosted by each node can be almost the same. The uneven number of concurrent function failures caused by the failed nodes among all failure patterns may be avoided. Secondly, the nodes hosting any of the primary or backup resources of a function are not allowed to host another backup resource of the function. Thirdly, the backup resources of the functions whose primary resources are hosted by the same node, are distributed to different nodes with the best.

Algorithm 5.1 starts to solve the problem by sorting $j \in F'$ by l_j^{R} decreasingly (line 2). Algorithm 5.1 is divided into two parts for allocating the primary resource (lines 4-12) and the backup resources (lines 13-29). Algorithm 5.1 allocates the primary resource of each function to the nodes (lines 4-12) with jointly considering the ordered failure probability for nodes and evenly distributing the functions to nodes so that the number of the functions hosted by each node can be almost the same (line 5). Each function is allocated to only one node to satisfy constraints (5.8b) and (5.8d). When failure $\mathcal{P}_\Gamma^\sigma$ occurs, the number of concurrent failures of function are almost the same.

Table 5.1: List of frequently used notations in algorithm.

Parameter	Description
F'	Set of sorted functions. The functions are sorted by the number of nodes hosting the function increasingly.
N'	Set of sorted nodes. The nodes are sorted by q_i increasing firstly, the nodes that have the same q_i are sorted by the number of hosting functions increasing secondly.
N''_j	Set of sorted nodes except for the nodes that host function j. The nodes are sorted by q_i increasing firstly, the nodes with the same q_i are sorted to distribute the backup resources of the functions hosted by the same node.

Thus, the uneven number of concurrent function failures may not occur in a failure pattern.

Algorithm 5.2 is a function for resource allocation called in Algorithm 5.1. Since any backup resource of each function is not allowed to be allocated to the same nodes that host the primary resource and backup resources of the function for any previous round to satisfy constraints (5.8b)-(5.8d), the algorithm only considers nodes in set N''_j for function j (lines 2-3 in Algorithm 5.2). Since function j is allocated to the first node in sorted set N', N''_j is set by deleting the first node in N'. Based on the current resource allocation and input parameters, the node workload can be calculated with (5.2a) and the failure probability can be calculated with (5.3).

Algorithm 5.1 Workload-aware greedy algorithm (WAGA)

Input: $F, N, C_i, l_j^{\mathrm{R}}, l_j^{\mathrm{u}}, s_i(w), \forall i \in N, j \in F,$

Output: x_{ij}^{kp}

1: $x_{ij}^{kp} = 0, i = j = 1, N''_j = N * |F|, N' = N.$

2: Sort j by l_j^{R} decreasingly as F'.

3: $W_i \leftarrow 0$, $q_i \leftarrow 0$

4: **for** $j = 1 \rightarrow |N|$ **do**

Sort i by q_i increasingly as a set N'. Sort the elements in N' that have same q_i according to the number of allocated functions on each element increasingly.
Delete node i from N' if $L_i^{\mathrm{R}} + l_j^{\mathrm{R}} > C_i$.
if $N' = \emptyset$ **then**
Return infeasible
else
Update the primary resource allocation obtained by Resource_allocation (F', N')
end if
end for
for $n = 1 \rightarrow \Gamma$ **do**
$F' = F, k = 1, j = 1$.
while $F' \neq \emptyset$ **do**
Delete node i from N_j'' if $L_i^{\mathrm{R}} + l_j^{\mathrm{R}} > C_i$.
Sort functions in F' by the number of nodes hosting the function increasingly.
Sort i in N_j'', $j \in F'$, by q_i increasingly. Sort nodes in N_j'' that have same q_i to distribute the backup resources of the functions whose primary resources are hosted by the same node.
Update the backup resource allocation obtained by Resource_allocation (F', N_1'')
Set the node with the lowest failure probability among nodes hosting the backup resource j to the highest priority.
if $N_j'' = \emptyset$ for all functions **then**
if there exists at least one function that cannot be allocated to any node as backup resource **then**
Return infeasible
end if
else
Return x_{ij}^{kp}
end if
end while
end for

Algorithm 5.2 Resource allocation

1: **function** RESOURCE_ALLOCATION(F', N')
2: Allocate the first function in F' to the first node in N'.
3: Record $N''_j = N \backslash \{\text{first element in } N'\}$, where j is the first function in F'.
4: Delete allocated function from F'.
5: Update W_i with $l_j^{\mathrm{R}}, l_j^{\mathrm{u}}$ and current allocation by (5.2a) and calculate q_i with $s_i(w)$ based on W_i.
6: **return** Resource allocation, $N''_j, \forall j \in F$, $q_i, \forall i \in N$
7: **end function**

Algorithm 5.1 allocates the backup resources of each function to the nodes in lines 13-29. The algorithm considers that each function can be allocated to at most Γ nodes for Γ-fault-tolerance with enough remaining capacity. Firstly, since the backup resource of each function is not allowed to be allocated to the same nodes that host the primary resource and backup resources of the function for any previous round, the algorithm only considers nodes in set N''_j for function j (lines 2-3 in Algorithm 5.2). Secondly, the functions hosted by the same node are protected by different nodes with the best. It is achieved by sorting nodes in N'. For function j whose primary resource is allocated to node i, a node i' that hosts the backup resource of a function whose primary resource is also allocated to node i is moved to the later position of array N' (line 18). Thus, the backup resource of a function j is allocated to a node in the ahead position of N' which may not host the backup resource of other functions that share the same node with j for the primary resource. Thirdly, the algorithm sorts elements in F' by the number of nodes hosting the function increasingly (line 17). The functions hosted by less nodes are in the ahead position of F'; the first function in F' is allocated to a node in N''_j prior to the functions hosted by more nodes which are in the latter part of F'. After each allocation of the first function in F', the allocated function is deleted from F' and the elements in N''_j are updated, where $j = 1$ (line 19 in Algorithm 5.1 and lines 2-4 in Algorithm 5.2). Since the algorithm always allocates the first element in F' in each round, N''_j with $j = 1$ is used to specify the nodes permitted to host the function being waited to be allocated in line 19 of Algorithm 5.1.

Thus, the functions can be evenly allocated to nodes in N''_j until all the functions cannot be allocated to any node or all the functions are protected by Γ nodes (lines 21-26). Then the node with the lowest failure probability among nodes hosting the backup resource j is set to the highest priority (line 20) with satisfying (5.8b) and (5.8c). This work summarizes the descriptions of three sorted sets used in Algorithm 5.1 in Table 5.1.

If there exists at least one function that cannot be allocated to any node as the primary resource, the algorithm considers that a function cannot find any feasible allocation (lines 7-8). When failure occurs, one of the backup resources of the functions hosted by a failed node is required to recover the unavailable functions. If there exists at least one function that cannot be allocated to any node as a backup resource, the algorithm also considers the situation as infeasible (lines 22-24) to satisfy (5.8b)-(5.8d).

Algorithm 5.3 Simulated Annealing

Input: x_{ij}^{kp}, $F, N, T_i^n, P_n, C_i, l_j^{\mathrm{R}}, l_j^{\mathrm{u}}$, $T_{\min}$, T_{init}.

Output: $x_{ij}^{kp}, \mathcal{U}$

1: Calculate an initial objective value $\mathcal{U} = \delta\mathcal{U}_1 + (1-\delta)\mathcal{U}_2$ by using initial solution $\mathbf{x}$ obtained by Algorithm 5.1.
2: $T \leftarrow T_{\mathrm{init}}$
3: **while** $T \geq T_{\min}$ **do**
4: Update N''_j.
5: Re-allocate j with random number from a randomly chosen node in $N \backslash N''_j$ to a randomly chosen node in N''_j.
6: Release a randomly chosen backup resource of a randomly chosen function.
7: Allocate a new backup resource of a randomly chosen function to a randomly chosen node in N''_j.
8: Sort $N \backslash N''_j$ by q_i increasingly. Sort the elements in $N \backslash N''_j$ that have the same q_i by the number of allocated functions on each element increasingly. Impose that the first element has the highest priority to recover function j.
9: **while** the workload of a node exceeds the capacity **do**
10: Switch the priority of two backup resources of a function for recovery

11: **end while**
12: Calculate a objective value $\mathcal{U}' = \delta\mathcal{U}'_1 + (1-\delta)\mathcal{U}'_2$ by using updated solution $\mathbf{x}'$.
13: $\mathcal{U} \leftarrow \mathcal{U}'$ and $\mathbf{x} \leftarrow \mathbf{x}'$ with probability of $\min(1, q)$, where $q = -\frac{\mathcal{U}'-\mathcal{U}}{T}$.
14: $T = T \cdot$ a given decreasing rate
15: **end while**
16: Return $\mathbf{x}$ and $\mathcal{U}$

This work uses simulated annealing (SA) [94] to minimize $\delta\mathcal{U}_1 + (1-\delta)\mathcal{U}_2$ calculated with (5.4)-(5.7) based on the initial backup resource allocation obtained by Algorithm 5.1. The accuracy of solutions by adding SA based on Algorithm 5.1 can be improved with the cost of longer computation time. SA is inspired by the principle of solid annealing with a gradually decreasing process of temperature from given initial temperature T_{init} to given minimum value of temperature $T_{\min}$. In each iteration of decreasing the temperature, the existing solution changes randomly to generate a new solution and calculates a new objective function with (5.4)-(5.7) (lines 5-11). SA accepts the solution with a worse objective value than the existing solution with a certain probability; the higher temperature is, the higher probability to accept a worse solution is (line 13). As a result, SA can jump out of a local optimal solution.

5.2.2 Computational complexity and example

The computational time complexities of sorting $|F|$ functions and sorting $|N|$ backup servers are $O(|F|\log|F|)$ and $O(|N|\log|N|)$, respectively. The computational time complexities of lines 4-12 and lines 13-29 are $O(|N|^2\log|N|)$ and $O(\Gamma|F|(|F|\log|F|+|N|\log|N|))$, respectively. The overall computational time complexity of Algorithm 5.1 is $O(|N|^2\log|N|+\Gamma|F|^2\log|F|+\Gamma|F||N|\log|N|)$; the space complexity is $O(|N|^3|F|)$.

Fig. 5.3 shows an example of Algorithms 5.1 and 5.2 to obtain the initial solution with three nodes sorted by q_i increasingly and four functions sorted by l_j^{R} increasingly. The algorithms firstly allocate each primary resource for the functions. F1 is allocated to node 1, which is the first element in the sorted nodes set. After the allocation, node 1 is deleted from N''_1, since the nodes hosting any of the primary or backup resources of a function are not

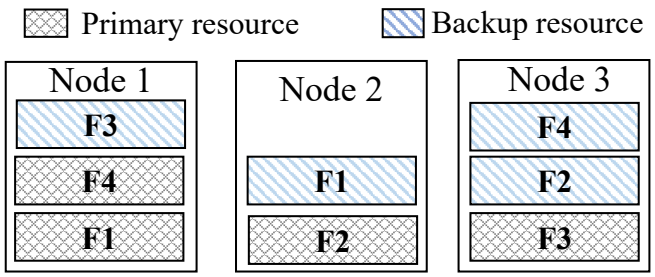


Figure 5.3: Examples of developed heuristic algorithm.

allowed to host another backup resource of the function. After the node set is updated with q_i and the number of allocated functions hosted by each node, F2 is allocated to node 2, which is the first element in the updated node set. Similar to the allocation of F1 to node 1, the deleting and updating procedures are repeated. Similarly, F3 and F4 are allocated to nodes 3 and 1, respectively. With satisfying (5.8d), each function is allocated to one node. Then the algorithms allocate the backup resources of each function. With setting $\Gamma = 1$, the backup resource of each function j is allocated to a node in N_j'' if there exist sufficient computing resources. The node set is sorted to distribute the backup resources of the functions whose primary resources are hosted by the same node. With this rule, the backup resource of F1, F2, and F3 are allocated to nodes 2, 3, and 1, respectively. For F4, whose primary resource is hosted by the same node with F1, the backup resource of F4 is allocated to node 3 to avoid the situation that failure of node 1 causes the unavailability of F1 and F4, which may not be able to be recovered by node 2 simultaneously due to insufficient capacity.

5.3 Discussion: approach to obtain unsuccessful recovery probability without priority setting

The proposed model considers weights δ and $1 - \delta$, where $0 \leq \delta \leq 1$, for $\mathcal{U}_1$ and $\mathcal{U}_2$, respectively, which can be adjusted to prioritize the one term

over the other term. With $\delta = 0$, several works introduced the load balancing model with various objectives, which is summarized in Section 5.4. With $\delta = 1$, the proposed model focuses on fault tolerance. This work defines that Γ-fault-tolerance guarantees that any function hosted by a failed node can be recovered by available nodes if there are at most Γ node failures; Γ is named as fault tolerance level. A priority among multiple backup resources for each function determines the recovery configuration of the unavailable function. Obtaining the unsuccessful recovery probability and fault tolerance level by preparing the recovery configurations and traversing all the failure patterns is more complicated than that without preparing the recovery configuration. The unsuccessful recovery probability and fault tolerance level can be obtained by simplifying the model without considering any recovery configuration to reduce the complexity. Thus,this section discusses an approach to obtain the unsuccessful recovery probability and fault tolerance level with an initial resource allocation without considering all the failure patterns and corresponding recovery configurations. The recovery configuration can be obtained based on the initial allocation after any detection of failure.

5.3.1 Examples of initial allocation without considering failures and recovery configuration

Without appraising the failures and recovery configurations in the initial resource allocation, the workload after failure recovery cannot be used for describing the recovery situation. When a function may be protected by multiple backup resources with dedicated or shared protection against failures, the functions hosted by failed nodes against a failure pattern may not be recovered simultaneously by the remaining available nodes with insufficient capacity for recovery.

This work views an instance of assignment between functions $j \in F$ and nodes $k \in N$ in the model as a bipartite graph; edge (j, k) between function j and node k indicates that node k protects function j. Let d_k denote the degree of node k; the number of protected functions by node k is d_k. n_k is the maximum number of functions that a node k can recover simultaneously.

This work considers an example for unsuccessful recovery in Fig. 5.4(a).

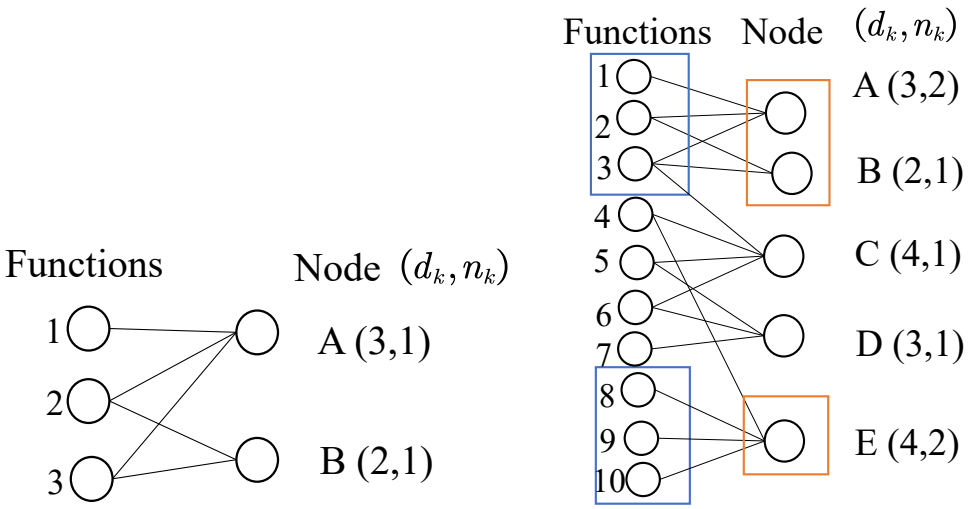
When function 3 fails, it cannot be successfully recovered in the following two situations. In the first situation, nodes A and B, which protect function 3, fail concurrently. In the second situation, multiple functions fail concurrently and cannot be recovered by nodes A and B. For example, node B protects functions 2 and 3 but can only recover one of them. Based on the allocations shown in Fig. 5.4(a), functions 2 and 3 can be recovered by nodes A and B, respectively. However, it also depends on the availability of function 1. If function 1 fails, all the failed functions cannot be recovered by nodes A and B.

In Fig. 5.4(b), as an example, arbitrary three failed functions of functions 1, 2, and 3 can be recovered by nodes A and B; arbitrary two failed functions of functions 3, 4, 5, 6, and 7 can be recovered by nodes C and D; arbitrary two failed functions of functions 4, 8, 9, and 10 can be recovered by node E. Since this work focuses on the arbitrary recoverable function by the nodes, the minimum value among the three values of arbitrary recoverable functions should be considered. Therefore, the maximum number of arbitrary recoverable functions is $\min\{3, 2, 2\} = 2$, arbitrary two failed functions of the ten functions can be recovered by the five nodes.

5.3.2 Maximum arbitrary recoverable functions for nodes in connected component

The example shown in 5.3.1 indicates that the unsuccessful recovery probability can be obtained by comparing the number of recoverable functions and the number of concurrent failed functions without appraising the failures and recovery configurations. Thus, this work discusses the maximum arbitrary recoverable functions for these available nodes in this subsection.

Consider that a bipartite graph consists of functions and nodes, where each edge in the graph between a function and a node indicates that the node protects the function. F and N in the model may consist of multiple connected components in the bipartite graph based on the protection relationship and the availability of nodes. In this subsection, this work only focuses on the available nodes which can recover unavailable functions in one connected component to discuss the maximum arbitrary recoverable functions for these available nodes. The availability of nodes in multiple connected components will be discussed



(a) Unavailability of functions in connected component.

(b) Maximum arbitrary recoverable functions.

Figure 5.4: Examples of unsuccessful recovery without considering failures and priority setting.

in Section 5.3.3 for obtaining the unsuccessful recovery probability.

There is a matching containing any set of arbitrary m failed functions in the connected component. Matching M, representing a subset of the edges of the bipartite that corresponds to the assignment, describes the possibility to recover the $|M|$ functions in F, such that a function can be recovered by the backup resource in its matched node [28]. A matching of size $|M| = |F|$ is enabled to recover all the failed functions. Since a node can recover one or more functions simultaneously in the proposed model, when this work considers the matchings between the functions and the nodes, node $k \in N$ can be transformed into a set of nodes D_k where $|D_k| = n_k$; each node $k' \in D_k$ has the same connection between each function and node k; each node $k' \in D_k$ can recover only one function simultaneously, i.e., $n_{k'} = 1$, as shown in Fig. 5.5. Taking node A with $n_A = 2$ in Fig. 5.5(a) as an example, node A can be transformed into two (n_A) nodes, nodes A1 and A2. Each node of A1 and A2 has the same connection between each function and node A and can only recover one function simultaneously. Considering the matching, this work uses the transformed nodes in $D_k, k \in N$, each of which can recover only one func-

tion, to replace node $k \in N$ that can recover more than one function, unless otherwise stated.

Let $G' = (L \cup R, E')$ be a connected component in bipartite graph $G = (F \cup N, E)$ with $|R|$ nodes and $|L|$ functions, where $R \subseteq N$, $L \subseteq F$, and $E' \subseteq E$. Node $k \in R$ protects d_k functions and can recover at most n_k functions.

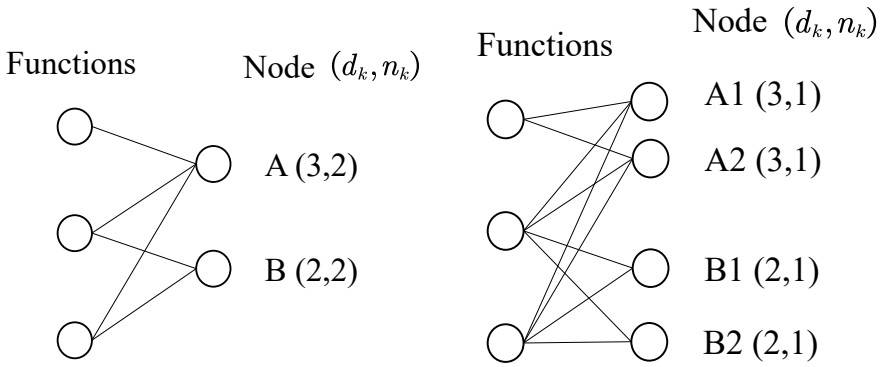
Let $F^{\text{full}} \subseteq L$ denote a set of functions with the maximum number of functions in L that are able to be fully recovered under any failure pattern by a set of directly connected nodes in $R^{\text{full}} \subseteq R$; the nodes in R^{full} directly connect only to the functions in F^{full}. This work considers that a sub-graph is constructed with a set of nodes $R' \subseteq R$, a set of edges connecting with nodes in R', and a set of functions connecting with the edges. F^{full} and R^{full} construct the sub-graph of G' that maximizes $|F^{\text{full}}|$, where the size of maximum matching equals $|F^{\text{full}}|$. There is always a matching between the functions in F^{full} and the nodes in R^{full}.

Let R_{u} denote a subset of R, where the number of functions in $F_{\text{u}} \subseteq F$ that are protected only by node $k \in R_{\text{u}}$ is larger than n_k; node $k \in R_{\text{u}}$ cannot recover all the functions protected only by itself. Each function in F_{u} is protected by one backup resource; the remaining functions in $L \backslash F_{\text{u}}$ are protected by multiple backup resources, each of which corresponds to either dedication protection or shared protection.

A maximum matching is a matching with the maximum number of edges. In a maximum matching, if any edge is added to it, it is no longer a matching. There can be more than one maximum matching for a given bipartite graph.

Lemma 5.1 *Let $\mathcal{I}$ denote a set of functions which are the intersection of functions in all maximum matchings for $G = (L \cup R, E')$. The functions in $\mathcal{I}$ are directly connected to $R^+ \subseteq R$ and $R^* \subseteq R$, where R^+ is directly connected only to the functions in $\mathcal{I}$; R^* is directly connected to the functions both in $\mathcal{I}$ and not $\mathcal{I}$; $R^+ \cap R^* = \emptyset$. The functions in $\mathcal{I}$ can be fully recovered by the nodes in R^+.*

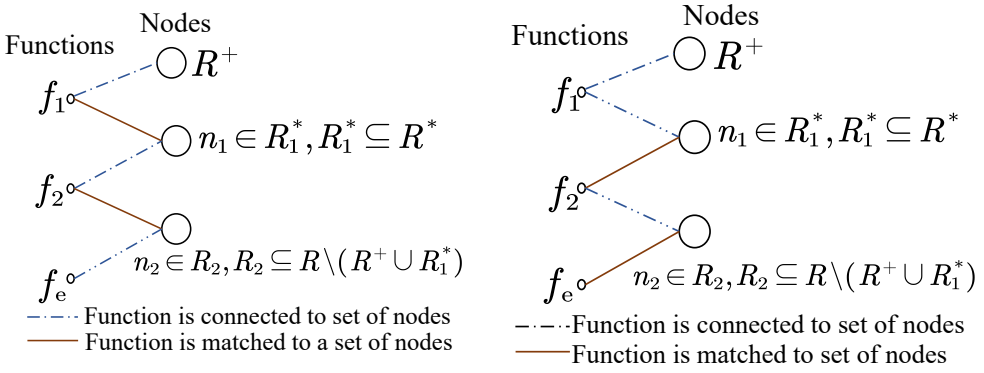
Proof : The proof can be done by contradiction. Suppose that this proposition is false; nodes in R^+ cannot recover all functions in $\mathcal{I}$ without nodes in R^*, which indicates that at least one function in $\mathcal{I}$ is matched to a node in R^*



(a) Node k in original bipartite graph can recover $\min\{d_k, n_k\}$ nodes.

(b) Each node in node set D_k transformed from node k in the original bipartite graph only recovers one node.

Figure 5.5: Example of bipartite-graph transformation in matching consideration.



(a) An unmatched function can be always found, f_e, as long as $\bar{\mathcal{I}}^* \neq \emptyset$.

(b) Withdraw matching of $f_1 \in \mathcal{I}$ that is matched to R^*, a matching for f_e can be added.

Figure 5.6: Example of Lemma 5.1 to show contradiction.

in each maximum matching. Let $\bar{\mathcal{I}}^*$ denote a set of functions disjoint with $\mathcal{I}$ that are directly connected to R^*.

Firstly, according to the definition of R^*, each node in R^* directly connects to functions not in the intersection, it is obvious that there exists a maximum matching that a function, $f_1 \in \mathcal{I}$, that directly connects to a set of nodes $R_1^* \subseteq R^*$ is matched to a node $n_1 \in R_1^*$, and at least one function, $f_2 \in \bar{\mathcal{I}}^*$, that directly connects to R_1^* is not matched to R_1^* in the maximum matching.

There are two situations for f_2. In the first situation, f_2 is not connected to any node in $R \backslash (R_1^* \cup R^+)$, this proof swaps function $f_1 \in \mathcal{I}$ and that matched to R^* and $f_2 \in \bar{\mathcal{I}}^*$that is not matched in the maximum matching so that f_2 is matched to R^* and f_1 is not matched. After swapping, f_2 replaces f_1 in the maximum matching. It contradicts the definition of $\mathcal{I}$, which is the intersection of functions in all maximum matchings.

In the second situation, f_2 is connected to a node in $R \backslash (R_1^* \cup R^+)$, there exists a maximum matching that f_2 is matched to a node $n_2 \in R_2^*, R_2^* \subseteq R \backslash (R_1^* \cup R^+)$, and at least one function, f_3, connected to R_2^* is not matched to R_2^* according to the definition of $\mathcal{I}$ and R^+; otherwise f_3 should be in $\mathcal{I}$.

Since R_2^* is a subset of $R \backslash (R_1^* \cup R^+)$, by changing the size of R_2^* and combination of nodes in R_2^*, all nodes in $R \backslash R^+$ are visited, this proof can always find an endpoint function, $f_e \in R \backslash (R_1^* \cup R^+)$, that directly connects only to the nodes that have been visited as long as $\bar{\mathcal{I}}^* \neq \emptyset$.

Next, as shown in Fig. 5.6, based on this claim, if $\bar{\mathcal{I}}^* \neq \emptyset$, an unmatched function can be always found, f_e, where, if the proof withdraws the matching of $f_1 \in \mathcal{I}$ that is matched to R^*, a matching for f_e can be added. After the original matching is changed, f_e replaces f_1 in the maximum matching. It contradicts the definition of $\mathcal{I}$, which is the intersection of functions in all maximum matchings.

Therefore, the functions in the intersection of functions in all maximum matchings can be fully recovered by the nodes in R^+ without R^*. □

Theorem 5.1 *The intersection of functions in all maximum matchings for* $G = (L \cup R, E')$ *is equal to* F^{full}.

Proof : The proof is done by contradiction. Suppose that this proposition is false. Firstly, the proof supposes that the intersection of functions in all

maximum matchings for $G = (L \cup R, E')$ is smaller than F^{full}. It indicates that at least one function in F^{full} can be recovered by the directed connected node; the matching is not in the maximum matching. It is a contradiction the maximum matching is a matching with the maximum number of edges. Secondly, the proof supposes that the intersection of functions in all maximum matchings for $G = (L \cup R, E')$ is larger than F^{full}, which indicates that a function in each maximum matching is not in F^{full}. With Lemma 5.1, the functions in the intersection of every maximum matching can be recovered by the nodes that directly connect only to the nodes without R^*. It is a contradiction since F^{full} is a set of functions with the maximum number of functions that are able to be fully recovered by a set of directly connected nodes that are directly connected only to F^{full}.

Therefore, the intersection of functions in all maximum matchings for $G = (L \cup R, E')$ is equal to F^{full}. □

Let R_{u} denote a subset of R, i.e., $R_{\text{u}} \subseteq R$, where the number of functions that are protected only by node $k \in R_{\text{u}}$ is larger than n_k; node $k \in R_{\text{u}}$ cannot recover all the functions that are protected only by itself.

Since $R^{\text{full}}, R_{\text{u}}$, and $R \backslash (R^{\text{full}} \cup R_{\text{u}})$ may not always be a non-empty set in $G = (L \cup R, E')$, if any of the sets of nodes is $\emptyset$, it cannot recover any function protected by it. The maximum number of arbitrary recoverable functions is determined by the remaining non-empty set among $R^{\text{full}}, R_{\text{u}}$, and $R \backslash (R^{\text{full}} \cup R_{\text{u}})$. When all the sets among $R^{\text{full}}, R_{\text{u}}$, and $R \backslash (R^{\text{full}} \cup R_{\text{u}})$ are $\emptyset$, they cannot recover any function. Thus, this work defines functions $\Lambda(\alpha)$ and $\Theta(\beta)$, where α and β are non-negative integers, in the following. If α is zero, $\Lambda(\alpha) = \infty$, and otherwise $\Lambda(\alpha) = \alpha$. If $\beta > |F|$, $\Theta(\beta) = 0$, and otherwise $\Theta(\beta) = \beta$.

Theorem 5.2 *If* $\Theta(\min\{\Lambda(|F^{\text{full}}|), \Lambda(\min_{k \in R_u} \{d_k, n_k\}), \Lambda(\sum_{k \in R \backslash (R^{\text{full}} \cup R_u)} \min \{d_k, n_k\})\}) \geq m$*, a set of arbitrary* m *failed functions can be recovered by nodes in* R*, i.e., there is a matching containing any set of arbitrary* m *failed functions.*

Proof : If $d_k \leq n_k, \forall k \in R$, node $k \in R$ is able to recover all functions in L; arbitrary failed functions can be recovered by nodes in R. Otherwise, if there is at least one node that satisfies $d_k > n_k, k \in R$, nodes in R may not be able to recover all the failed functions in L. This work now shows the maximum

number of arbitrary functions that nodes in R can recover.

There are three situations for the recovery. In the first situation, all functions in F^{full} can be fully recovered by a set of nodes in R^{full}, which is the intersection of functions in all maximum matchings for $G' = (L \cup R, E')$.

In the second situation, a set of nodes R_{u} connect to a set of functions $F^{\text{u}} \subseteq F$, each of which is protected only by node $k \in R_{\text{u}}$. The number of functions in F^{u} is larger than n_k for each node in R_{u}. Nodes in R_{u} can recover at most $\min_{k \in R_{\text{u}}}\{d_k, n_k\}$ arbitrary directly connected functions in L. In other words, the number of functions in F^{full} is smaller than or equal to the number of functions that nodes in R^{full} can recover, i.e., $|F^{\text{full}}| \leq \sum_{k \in R^{\text{full}}} n_k$.

In the third situation, if the number of remaining functions from L subtracted by F^{full} and F^{u} is larger than the maximum number of nodes in $R \backslash (R^{\text{full}} \cup R_{\text{u}})$ that can recover, i.e., $|L| - |F^{\text{full}}| - |F^{\text{u}}| > \sum_{k \in R \backslash (R^{\text{full}} \cup R_{\text{u}})} n_k$, the nodes that protect the functions cannot fully recover functions in $L \backslash (F^{\text{full}} \cup F^{\text{u}})$. The maximum number of arbitrary functions in $L \backslash (F^{\text{full}} \cup F^{\text{u}})$ that nodes may recover is $\sum_{k \in R \backslash (R^{\text{full}} \cup R_{\text{u}})} \min\{d_k, n_k\}$.

In the above three situations, if any set among $R^{\text{full}}, R_{\text{u}}$, and $R \backslash (R^{\text{full}} \cup R_{\text{u}})$ is $\emptyset$, the corresponding value of $\Lambda(\alpha)$ is equal to ∞. The maximum number of arbitrary recoverable functions is determined by the remaining non-empty set. When all the sets of $R^{\text{full}}, R_{\text{u}}$, and $R \backslash (R^{\text{full}} \cup R_{\text{u}})$ are $\emptyset$, i.e., the three values are equal to ∞, $\Theta(\min\{\Lambda(|F^{\text{full}}|), \Lambda(\min_{k \in R_{\text{u}}}\{d_k, n_k\}), \Lambda(\sum_{k \in R \backslash (R^{\text{full}} \cup R_{\text{u}})} \min\{d_k, n_k\})\}) = 0$, i.e, there is no function that can be recovered.

Therefore, the maximum number of arbitrary failed functions that nodes in R can recover is $\Theta(\min\{\Lambda(|F^{\text{full}}|), \Lambda(\min_{k \in R_{\text{u}}}\{d_k, n_k\}), \Lambda(\sum_{k \in R \backslash (R^{\text{full}} \cup R_{\text{u}})} \min\{d_k, n_k\})\})$. □

In Fig. 5.4(b), as an example, functions 1, 2, and 3 are the set of functions in F^{full}, where all the functions in F^{full} can be fully recovered by nodes A and B, which is illustrated by the below two boxes. Functions 1, 2, and 3 are the intersection of functions in all maximum matchings; $|F^{\text{full}}|$=3. Node E, i.e., R_{u}, connects to functions 8, 9, and 10, each of which is protected only by node E. The number of functions 8, 9, and 10 is larger than $\min\{d_E, n_E\}$, i.e., $3 > 2$. Node E can recover at most $\min\{d_E, n_E\}$, i.e., 2, arbitrary functions among 8, 9, and 10, which is illustrated by the below two boxes. Nodes in $R \backslash (R^{\text{full}} \cup R_{\text{u}})$, i.e., nodes C and D, can recover at most arbitrary $\min\{d_C, n_C\} + \min\{d_D, n_D\}$,

i.e., 1+1 = 2, functions among functions 4, 5, 6, and 7. Therefore, the maximum number of arbitrary recoverable functions is $\min\{3, 2, 2\} = 2$; arbitrary two failed functions of the ten functions can be recovered by the five nodes.

The unsuccessful recovery probability can be upper-bounded by the probability that the upper bound of the number of concurrent failed functions exceeds the lower bound of the number of recoverable functions.

5.3.3 Unsuccessful recovery probability against failures

The bipartite graph of functions in F and nodes in N in the model consists of one or more connected components. Let H denote the number of the connected component in $G = (F \cup N, E)$, where each edge in E between function and node indicates that node protects function. Let $G''_h = (F^{\mathrm{c}}_h \cup N^{\mathrm{c}}_h, E^{\mathrm{c}}_h), h \in [1, H]$, be the hth connected component in G, where $F^{\mathrm{c}}_h \subseteq F$ is the set of functions in G''_h; $N^{\mathrm{c}}_h \subseteq N$ is the set of nodes in G''_h that protect all functions in F^{c}_h. The concurrent failures of functions in F^{c}_h may come into a collision so that the unavailable functions cannot be recovered, since the unavailable function are protected by the same set of nodes in N^{c}_h.

A node failure may cause the concurrent failures of the functions hosted by the node. Let $\vartheta_{ih}, i \in N, h \in [1, H]$, denote the number of the primary resources of functions in F^{c}_h hosted by node i. When node $i \in N$ fails, ϑ_{ih} functions in F^{c}_h fail concurrently; ϑ_{ih} is equal to $\sum_{j \in F^{\mathrm{c}}_h} \chi_{ij}$. Let $\mathcal{P}_E, E \in [0, \Gamma]$ represent the failure patterns with exact E node failures; $\mathcal{P}^e_E, e \in [1, |\mathcal{P}_E|]$ represents eth element in $\mathcal{P}_E$. Focusing only on the initial allocation without workload variation, the weight for each failure pattern in $\mathcal{P}_E$ is the same, which can be represented by w^{E}. Among failure patterns in $\mathcal{P}_E$, the maximum number of unavailable function that fail concurrently is $\max_{e \in [1, |\mathcal{P}_E|]} \sum_{i \in \mathcal{P}^e_E} \vartheta_{ih}$.

The failures of nodes affect the unsuccessful recovery probability of the functions in F^{c}_h when the remaining available nodes in N^{c}_h cannot recover functions in F^{c}_h hosted by failed nodes. Let $N^{\mathrm{fail}}_{he}, h \in [1, H], e \in E$, denote a set of failed nodes in N^{c}_h in failure pattern $\mathcal{P}^e_E$. Let $N^{\mathrm{av}}_{he} = N^{\mathrm{c}}_h \backslash N^{\mathrm{fail}}_{he}$ denote the set of available nodes in N^{c}_h against failure pattern $\mathcal{P}^e_E$.

Let $d_k, k \in N$, be an integer variable which indicates the number of functions in F^{c}_h that are protected by node k. n_k is an integer variable which

indicates the minimum number of functions in F_h^{c} that node k can recover simultaneously. $n_k \geq \frac{C_k - W_{k0}}{\max_{j \in F_h^{\mathrm{c}}}(w_j - r_j^{\mathrm{u}})}$, where $\max_{j \in F_h^{\mathrm{c}}}(w_j - r_j^{\mathrm{u}})$ is the maximum requested load among functions in $F_h^{\mathrm{c}}, h \in [1, H]$, for recovery procedure with the updated information.

Against each failure pattern, only nodes in N_{he}^{av} can recover functions. This work considers a sub-graph of G_h'' with only $N_{he}^{\mathrm{av}} \subseteq N_h^{\mathrm{c}}$, i.e., $G_{he}^{\mathrm{av}} = (F_h^{\mathrm{c}} \cup N_{he}^{\mathrm{av}}, E_h^{\mathrm{c}\prime})$. Based on Theorem 5.2, $F_{he}^{\mathrm{full}}, h \in [1, H], e \in E$, denotes a set of functions in F_h^{c} with the maximum number of functions in F_h^{c} that can be fully recovered by a set of directly connected nodes in $N_{he}^{\mathrm{full}} \subseteq N_{he}^{\mathrm{av}}$; F_{he}^{full} is the intersection of functions in all maximum matchings for G_{he}^{av}. N_{he}^{u} denotes a subset of N_{he}^{av}, where the number of functions that are protected only by node $k \in N_{he}^{\mathrm{u}}$ is larger than corresponding n_k; node $k \in N_{he}^{\mathrm{u}}$ cannot recover all the functions that are protected only by itself. The maximum number of arbitrary functions that nodes in N_{he}^{av} can recover simultaneously, which is denoted by $\mathcal{M}_{he}$, is given by:

$$\begin{aligned}\mathcal{M}_{he} = \Theta(&\min\{\Lambda(|F_{he}^{\mathrm{full}}|), \Lambda(\min_{k \in N_{\mathrm{u}}}\{d_k, n_k\}),\\ &\Lambda(\sum_{k \in N_{he}^{\mathrm{av}} \setminus (N_{he}^{\mathrm{full}} \cup N_{he}^{\mathrm{u}})} \min\{d_k, n_k\})\}), \forall h \in [1, H], e \in E. \end{aligned} \tag{5.9}$$

Let $\nu_h^E, h \in [1, H], E \in [0, \Gamma]$ denote a binary variable. It indicates whether the maximum unavailable functions due to the failure of nodes in $\mathcal{P}_E^e$ can be recovered by nodes in N_{he}^{av}. ν_h^E is set to one if the maximum unavailable functions in F_h^{c} due to the failed nodes in $\mathcal{P}_E^e$ cannot be recovered by nodes in N_{he}^{av}, and zero otherwise. In a mathematical expression, ν_h^E is set to one if $\max_{e \in [1, |\mathcal{P}_E|]} \sum_{i \in \mathcal{P}_E^e} \vartheta_{ih} > \min_{e \in [1, |\mathcal{P}_E|]} \mathcal{M}_{he}$, and zero otherwise. Let r'_E denote a binary variable; it is set to one if the functions in at least one connected component cannot be recovered under failure pattern $\mathcal{P}_E^e$, and zero otherwise, i.e., $r'_E = \vee_{h \in [1,H]} \nu_h^E$. The unsuccessful recovery probability can be expressed by $\mathcal{U}_1' = \sum_{E \in [0,\Gamma]} \binom{|N|}{E} w^{\mathrm{E}} r'_E$.

This work can obtain the initial resource allocation with minimizing $\mathcal{U}_1'$. After detecting a failure pattern, the recovery configuration results can be obtained by minimizing $\mathcal{U}_2$ introduced in Section 5.1.3. The fault tolerance level can also be obtained by comparing the maximum number of arbitrary recoverable functions by the $|N| - \Gamma$ available nodes and the maximum number

of concurrent unavailable functions hosted by the Γ failed nodes under a failure pattern; the maximum Γ satisfying that the former value is larger than or equal to the latter value under any failure pattern with at most Γ node failures is the fault tolerance level.

5.4 Numerical evaluations

This section first introduces four baselines with different objectives and describe the experiment settings. Second, this section investigates the model characteristics with smaller-size problems in Sections 5.4.3 and 5.4.4 in terms of the effect of considering workload and the dependency of W-UP on weight δ. The smaller-size problems are solved by the MILP approach. Third, in Section 5.4.5, this work evaluates the computation times of the MILP approach and the heuristic algorithm and the accuracy of the latter. In Section 5.4.6, this work considers larger-size problems with 20 nodes and more than 100 functions to evaluate the performance of the proposed model, which are solved by the developed heuristic algorithm.

The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.7.1 [93] which is implemented by Python 3.7, using Intel Core i7-7700 3.60 GHz 4-core CPU with 32 GB memory.

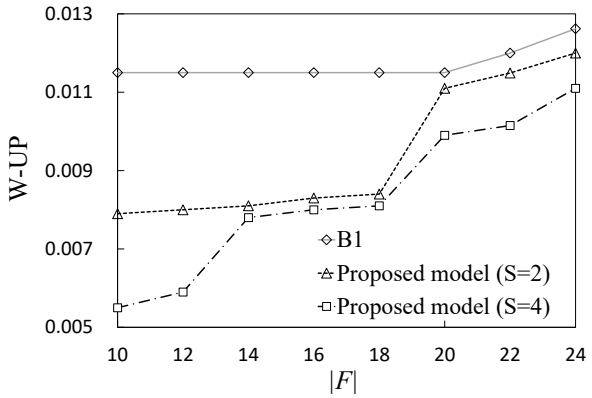
5.4.1 Baselines

Table 5.2 summarizes the features of the proposed model and four baselines. Baseline 1 (B1) is a conventional model that minimizes W-UP without considering a workload-dependent failure probability. Baseline 2 (B2) is a load balancing model introduced in [71] that minimizes the maximum utilization ratio among nodes without considering a workload-dependent failure probability. The recovery priority of each function that can be applied for all failure patterns is determined at the operation starting time. Since W-UP considers two unavailability situations: unsuccessful recovery and workload-related unavailability after recovery, baseline 3 (B3) and baseline 4 (B4) consider, respectively, each unavailable situation addressed for W-UP with the workload-dependent failure probability.

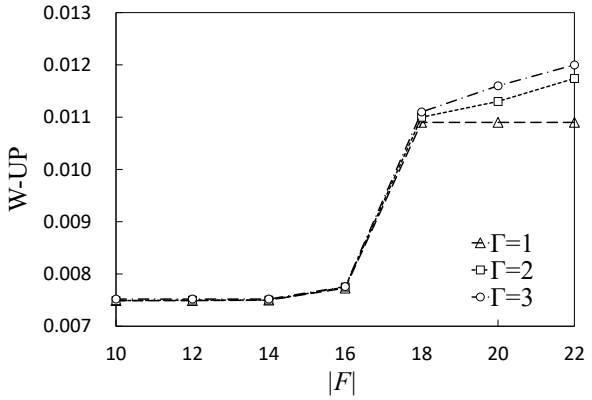
Table 5.2: Features of proposed model and baselines.

Model	Objective (Minimize)	Failure probability†	Workload‡	Recovery
Proposed model	W-UP	Yes	Yes	Determined in advance
B1	W-UP	No	Yes	Determined in advance
B2	Maximum utilization ratio	No	Yes	Determined in advance
B3	Weighted maximum unavailable probability after recovery among nodes ($\delta = 0$)	Yes	Yes	Determined in advance
B4	Unsuccessful recovery probability discussed in Section 5.3 ($\delta = 1$)	Yes	No	Obtained by recalculation

Failure probability†: Is the workload-dependent failure probability adopted in the model?
Workload‡: Is the workload variation after detecting the failure and activating the backup resource taken into consideration?



(a) Comparison between proposed model and B1 with different step number $\mathcal{S}$.



(b) Comparison among proposed posed model with different Γ.

Figure 5.7: Effect of considering workload on W-UP.

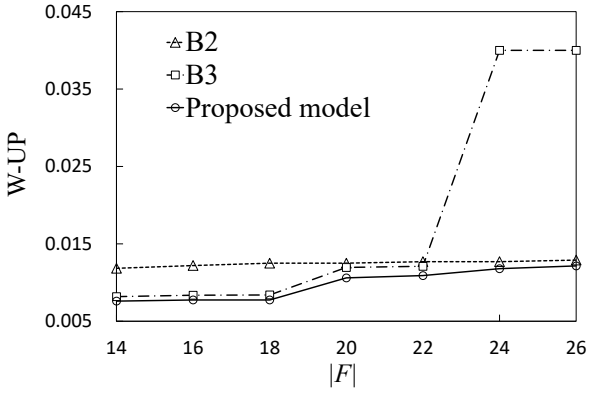


Figure 5.8: Comparison among proposed model, B2, and B3 with $\delta = 0$.

In detail, regardless of a workload-dependent failure probability in B1, the failure probability in B1 is set to $P_{\mathrm{C}} = \frac{\sum_{s \in \mathcal{S}} P_s}{|\mathcal{S}|}$ since this work cannot judge the failure probabilities with different workloads, where P_{C} represents a constant failure probability adopted in B1. The objective value of B1 is $\mathcal{U} = \delta \mathcal{U}_1 + (1 - \delta)\mathcal{U}_2$. B2 handles a load balancing model [71] to minimize the maximum utilization ratio among nodes with considering the activation of a backup resource after failures occur, regardless of a workload-dependent failure probability. The utilization ratio of node $i \in N$ is expressed by $\frac{L^{\mathrm{W}}_{i\sigma}}{C_i}, i \in N, \sigma \in \mathbb{C}$. The optimization problem of B2 is expressed by:

$$\min \max_{\sigma \in \mathbb{C}} \max_{i \in N} \frac{L^{\mathrm{W}}_{i\sigma}}{C_i} \tag{5.10a}$$

$$s.t. (5.1), (5.2a), (5.4) - (5.7), (5.8b) - (5.8f). \tag{5.10b}$$

B3 and B4 consider $\mathcal{U}_1$ and $\mathcal{U}_2$ as objective values, respectively. B4 merely considers the unsuccessful recovery probability without priority setting in advance, which is discussed in Section 5.2. The recovery configuration after failure is required, which can be obtained by minimizing $\mathcal{U}_2$. The computation of reallocation after failure detection in B4, which may affect the recovery time, is required. In the recalculation of B4, this work fixes the resource allocations of available functions to avoid introducing any interruption of these functions.

5.4.2 Experiment settings

In the smaller-size problems investigated in Section 5.4.3 and 5.4.4, consider five nodes whose capacities are set to 20; the requested loads of functions are randomly distributed over the range of [1, 4]; the loads required for information synchronization and snapshot updating are randomly distributed over the range of [0.1, 0.4], unless stated otherwise. To investigate the effect of workload on W-UP, this work only considers the tests in which the total requested load among less number of functions is smaller than that of more number of functions. Let $f(w)$ be a given non-decreasing failure probability, which shows the practical relationship between the failure probability and the workload; let $s(w)$ denote an $\mathcal{S}$-step function; w is a node workload. This work considers that the failure probability of a physical node is approximate to 0.01 based on [99].

Taking $f(w) = 0.005\sqrt{w}$ as an example, which is one of the workload-dependent failure probabilities adopted in the case study of Chapter 3, this work obtains step function $s(w)$ fitting the curve of $f(w)$ to minimize $\int_0^{C_i} (s(w) - f(w))\,\mathrm{d}w$. This work has $P_1 = 0.015$ and $P_2 = 0.022$; $T_1 = 9$ and $T_2 = C = 20$ for two-step approximation and $P_1 = 0.098$, $P_2 = 0.015$, $P_3 = 0.019$, and $P_4 = 0.022$; $T_1 = 4$, $T_2 = 9$, $T_3 = 14$, and $T_4 = C = 20$, unless stated otherwise. The work in Chapter 3 discussed the procedure of the approximation of non-decreasing function in detail, where two examples of step functions fitting the $f(w)$ curve were presented. The evaluation sets weight $\delta = 0.5$ and $\Gamma = 2$, unless stated otherwise. When this work compares W-UPs obtained by the proposed and baseline models, the exact values of W-UP of both models are calculated by (5.4)-(5.7) by using the obtained allocations. In order to evaluate the performance of the proposed model compared with the baselines, this work conducts different trials, in each of which a parameter is set to a value randomly selected from the considered range, to compute the average value for each result. This work performs such sufficient trials that the 95% confidence interval is no greater than 5% of the reported average value, unless stated otherwise.

In the larger-size problems, this work considers problems with 20 nodes and more than 100 functions. this work uses a dataset of Google cluster usage traces [100] as a case study, which describes the maximum amount of CPU that a function is permitted to use and the capacities of the physical nodes. The measurements have been normalized, where the normalization is a scaling relative to the largest capacity of the resource on any machine in the trace. The developed algorithm is adopted to solve larger-size problems; this work sets $T_{\text{init}} = 100$ and the temperature decreasing rate is set to 0.95 for SA. In the larger-size problems, this work focuses on the failure patterns with up to five node failures. Even more failures may occur concurrently, the probability of concurrent failures of more than five nodes is less than 10^{-10}, which is considered as improbable scenarios. The weight for such failure patterns is less than 10^{-10}; the term with 10^{-10} as a coefficient can be ignored in calculating W-UP. This is because the term with 10^{-10} as a coefficient is much smaller than the terms with 1 and 10^{-2} as coefficients for failure patterns with zero and one node failure, respectively, where the node failure probability is approximate to 0.01.

Table 5.3: Examples for $\mathcal{U}_1$ and $\mathcal{U}_2$ of proposed model with different values of weight δ and $|F|$ with $|N| = 3$.

$\|F\|$	δ=0.5			δ=0.1			δ=0.01			δ=0.001		
	r_σ	$\mathcal{U}_1$	$\mathcal{U}_2$	r_σ	$\mathcal{U}_1$	$\mathcal{U}_2$	r_σ	$\mathcal{U}_1$	$\mathcal{U}_2$	r_σ	$\mathcal{U}_1$	$\mathcal{U}_2$
6	[0, 0, 0, 0, 0, 0, 0]	0	0.015207	[0, 0, 0, 0, 0, 0, 0]	0	0.015002	[0, 0, 0, 0, 0, 1, 0]	0.00022	0.015001	[0, 0, 0, 0, 1, 1, 1]	0.00066	0.014999
8	[0, 0, 0, 0, 0, 0, 0]	0	0.015207	[0, 0, 0, 0, 1, 0, 1]	0.00044	0.015002	[0, 0, 0, 0, 0, 1, 1]	0.00044	0.015002	[0, 0, 0, 0, 1, 1, 1]	0.00066	0.015001
10	[0, 0, 0, 0, 0, 0, 0]	0	0.015310	[0, 0, 0, 0, 1, 1, 1]	0.00066	0.015309	[0, 0, 0, 0, 1, 1, 1]	0.00066	0.015221	[0, 0, 0, 0, 1, 1, 1]	0.00066	0.015221

5.4.3 Effect of considering workload

This work compares the proposed model with B1 and investigate the dependency on $\mathbf{\Gamma}$s of the proposed model to show the effect of considering workload in Fig. 5.7.

Figure 5.7(a) shows W-UPs obtained by the proposed model and B1 for different values of $|F|$. Since the total node workload for the initial allocation increases as $|F|$ increases, it leads to the increase of the workload-dependent failure probability; the weight for each failure pattern increases. Similarly, as $|F|$ increases, the number of functions hosted by the same node increases, each failure pattern may cause a larger number of concurrent unavailable functions with a larger workload required to be recovered. Both unsuccessful recovery probability and unavailable probability after recovery may increase. Thus, an increasing tendency of W-UP with the increase of $|F|$ in the proposed model can be observed.

Then this work compares W-UPs between the proposed model and B1 in Fig. 5.7(a). With a constant value of failure probability regardless of considering the effect of node workload on the failure probability, B1 is not sensitive to both initial workload for primary resource and workload variation caused by node failures. W-UPs obtained by B1 are on average 29% and 52%, respectively, larger than those of the proposed model with two-step and four-step approximations, among the values of $|F|$. When $|F| \leq 20$ in the tested cases where the total requested load among $|F|$ functions is so small that it does not lead to the unsuccessful recovery, W-UP only depends on the weighted maximum unavailable probability among nodes after failure recovery, i.e., $\mathcal{U}_2$. W-UPs obtained by B1 remain to be constant. Both weights of failure pat-

terns and maximum unavailable probability among nodes in B1 are larger than those of the proposed model. W-UPs obtained by B1 are on average 38% and 64%, respectively, larger than those of the proposed model with two-step and four-step approximations, among the values of $|F|$. As the workload increases, the failure probability in the proposed model gets close to that of B1. When $|F| > 20$ in Fig. 5.7(a), the total requested load of function is so large that the concurrent unavailable functions cannot be recovered simultaneously due to insufficient capacity. Both W-UPs of the proposed model and B1 increase; a decreasing tendency of the difference of W-UPs between B1 and the proposed model can be observed in the increase of $|F|$.

Then this work considers the workload variation caused by at most Γ node failures. As shown in Fig. 5.7(b), W-UPs obtained by the proposed model with $\Gamma = 1$ is 2.3% smaller than that of $\Gamma = 2$; it is 3.4% smaller than that of $\Gamma = 3$. The average availability among different $|F|$ obtained with 1-(W-UP) are 0.9911, 0.9908, and 0.9907 for $\Gamma = 1$, 2, and 3, respectively. The weight of failure pattern with exact three node failures is 6.6×10^{-6}, which is 2651 times smaller than that of exact one node failure and 515 times smaller than that of exact two node failures. Since the weight of a failure pattern with more failed nodes is smaller than that of less failed nodes, the difference between W-UPs of Γ and $\Gamma + 1$ decreases as Γ increases. The difference between W-UPs of Γ and $\Gamma + 1$ increases as $|F|$ increases. When the primary resources hosted by the same node increase, the increase of the number of concurrent unavailable functions may lead to a higher probability of unsuccessful recovery.

5.4.4 Dependency of W-UP on weight δ

Dependency of $\mathcal{U}_1$ and $\mathcal{U}_2$ on weight

δ is the weight for unsuccessful recovery probability, $\mathcal{U}_1$, of W-UP. Since $r_\sigma, \sigma \in \mathbb{C}$, is a binary variable and $0 < Q_\sigma \approx 0.02 < 1$ under the same σ, this work investigates the tests with $\delta = 0.5$, 0.1, 0.01, and 0.001 so that the ratio of δr_σ to $(1-\delta)Q_\sigma$ approximates to 50, 5, 0.5, and 0.05, respectively. $r_\sigma = 1, \sigma \in \mathbb{C}$, represents that at least one unavailable function under failure pattern $\mathcal{P}_\Gamma^\sigma \in \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}\{1,3\}, \{2,3\}\}$ cannot be recovered. When the total workload of functions is so small that the unsuccessful recovery situa-

tion due to the insufficient capacity does not exist against any failure pattern, i.e., $g_i^\sigma = 0$, $\mathcal{U}_1$ only relates to the unavailable situation that all of the primary and backup resources fail in a failure pattern, i.e., $e_j^\sigma = 1$. Table 5.3 shows examples for $\mathcal{U}_1$ and $\mathcal{U}_2$ with different values of weight δ and $|F|$ with $|N| = 3$. As δ decreases, the proposed model shows a tendency to reduce $\mathcal{U}_2$ at the cost of a larger unsuccessful recovery probability, $\mathcal{U}_1$. This is because the functions are allocated to nodes with a number less than Γ to reduce the workload-dependent failure probability in $\mathcal{U}_2$, at the cost of the unavailable situation of insufficient primary and backup resources against failures. When δ is small enough, $\mathcal{U}_2$ does not decrease since it is the smallest value with conservative workload-dependent failure probability.

$\delta = 0$

When $\delta = 0$, only the weighted maximum unavailable probability is considered in the proposed model; whether concurrent unavailable functions in a failure pattern can be recovered is not considered. This work compares the proposed model with B2 and B3, each of the baselines focuses only on the recovery configuration against failures, regardless of the unsuccessful recovery. This work obtains the resource allocation of baselines by solving the MILP problems presented in (5.8a)-(5.8f) with $\delta = 0$ and (5.10a)-(5.10b), respectively. To differentiate the workload-dependent failure probability among nodes, this work considers that the ratios of the threshold to the capacity of a node are randomly distributed over the range of [0.4, 0.8].

As shown in Fig. 5.8, W-UPs obtained by B2 and B3 are on average 22% and 31%, respectively, larger than those of the proposed model, among the values of $|F|$. In comparison between the proposed model and B2, when the total workload is so small that the proposed model can adapt the resource allocation to reduce the workload-dependent failure probability, B2 merely minimizes the utilization ratio without considering the different workload-related failure characteristics of nodes. As $|F|$ increases, the proposed model cannot suppress the failure probabilities of nodes to be smaller than $P_{\mathcal{S}}$; the failure probability in B2 gets close to that of the proposed model. Thus, the difference between W-UPs in the proposed model and B2 decreases as $|F|$ increases.

Table 5.4: Comparison between proposed model and B4 in terms of average Q_σ among failure patterns with exact Γ failures with different $|F|$.

Model	δ	Γ	$\|F\|$				
			10	12	14	16	18
Proposed	0.5	1	0.0150	0.0150	0.0150	0.0220	0.0220
Proposed	0.5	2	0.0164	0.0185	0.0206	0.0220	0.0220
B4	1	1	0.0150	0.0178	0.0220	0.0220	0.0220
B4	1	2	0.0178	0.0199	0.0220	0.0220	0.0220

Focusing on B3, since it does not consider the unsuccessful recovery due to the insufficient capacity, this work adds $L^{\mathrm{W}}_{i\sigma} \leq C_i, \forall i \in N, \sigma \in \mathbb{C}$, as a constraint to reduce the unsuccessful recovery probability. When the total workload among functions is so large that the workload after recovery against at least one failure pattern exceeds the capacity and B3 cannot find any feasible solution, this work relaxes the constraint as $L^{\mathrm{W}}_{i0} \leq C_i, \forall i \in N$, to only suppress the initial workload not to exceed the capacity. When $|F| < 20$ in Fig. 5.8 where the total workload among functions is so small that the workload of a node can be suppressed to be smaller than the threshold, B3 shows better performance than B2, since B3 considers the workload-dependent failure probability. W-UPs in B3 are 7% larger than that of the proposed model, since the unsuccessful recovery against Γ failures is not considered in B3. When $|F| \geq 22$ in the tested case, B3 relaxes the capacity constraint since it cannot find any feasible solution with $L^{\mathrm{W}}_{i\sigma} \leq C_i$, which further leads to the increase of unsuccessful recovery probability. Thus, W-UPs of B3 are larger than both B2 and the proposed model with $|F| \geq 22$.

$\delta = 1$

When $\delta = 1$, the proposed model focuses on the unsuccessful recovery probability. However, the workload after recovery is also required to be considered, since it is related to the unsuccessful recovery due to insufficient capacity. As

Table 5.5: Maximum concurrent unavailable functions and maximum arbitrary recoverable functions against Γ node failures

$\|F\|$	Γ	Maximum concurrent unavailable functions hosted by Γ failed nodes	Maximum arbitrary recoverable functions
10	1	2	2
	2	4	4
	3	6	6
15	1	3	3
	2	6	6
	3	9	4
20	1	4	4
	2	8	3
	3	12	2

discussed in Section 5.3, the unsuccessful recovery probability with $\delta = 1$ can be obtained by simplifying the model without considering any recovery configuration to reduce the complexity. When a failure pattern is detected, the proposed model recovers the unavailable functions based on the determined priority; B4 determines a recovery configuration for the detected failure pattern in operation run time.

Table 5.4 shows the average Q_σ among failure patterns with exact Γ failures of the proposed model and B4. Each Q_σ is obtained with the priority in the proposed model; it is obtained with the run-time calculation for the recovery configuration against each failure pattern in B4. Average Q_σ obtained by the proposed model is 9% and 4% smaller than that of B4 with exact one failure and two failures, respectively. Q_σ is a value of the workload-dependent failure probability approximated by the step function. Focusing on average Q_σ among failure patterns, when it is equal to the maximum value of the failure probability, i.e., 0.022, as shown in Table VI, it does not increase as $|F|$ increases. Since the initial allocation obtained by B4 with $\delta = 1$ does not consider workload after recovery, it does not distribute the backup resource of

functions whose primary resources are hosted by the same nodes into different nodes against node failures. Thus, the run-time obtained recovery configuration may not the best one even against each failure pattern; the proposed model can reduce W-UP compared with B4. The computation time of B4 to determine recovery configuration after failures is almost the same among failure patterns, which is 0.03 [s]. The average workload-related availability after recovery among different $|F|$ of the proposed model are 0.9822 and 0.9801 for $\Gamma = 1$ and 2, respectively; those of B4 are 0.9802 and 0.9793.

Then this work considers the fault tolerance level of the proposed model, which can be obtained with the discussion in Section 5.3 without recovery configuration. The discussion provides the maximum number of concurrent unavailable functions hosted by the failed nodes and the maximum number of arbitrary recoverable functions by the remaining available nodes. Fault tolerance levels can be obtained by comparing the two numbers, as shown in Table 5.5. The number of concurrent unavailable functions hosted by Γ failed nodes can be obtained with an even primary resource allocation, which is equal to $\Gamma \times \lfloor \frac{|F|}{|N|} \rfloor$. The maximum arbitrary recoverable functions can be obtained with analyzing the graph structure of the connected component in the graph corresponding to backup resource allocation. Since $\mathcal{M}_{he} = \Theta(\min\{\Lambda(|F_{he}^{\text{full}}|), \Lambda(\min_{k \in N_{\text{u}}}\{d_k, n_k\}), \Lambda(\sum_{k \in N_{he}^{\text{av}} \setminus (N_{he}^{\text{full}} \cup N_{he}^{\text{u}})} \min\{d_k, n_k\})\})$, this work allocates all functions in one same set of nodes among N_{he}^{full}, N_{he}^{u}, and $N \setminus (N_{he}^{\text{full}} \cup N_{he}^{\text{u}})$ with the best, so that $\mathcal{M}_{he}$ can be increased. From Table 5.5, we can observe that an allocation with 10 functions is 3-fault-tolerance; an allocation with 15 functions is 2-fault-tolerance; an allocation with 20 functions is 1-fault-tolerance. There is no more Γ-fault-tolerance guarantee with $k \geq 1$ for $|F| > 20$ in the considered case with five nodes whose capacities are 20.

5.4.5 Competitive evaluation on computation time and accuracy

Table 5.6 shows W-UPs and computation times for the MILP approach and the heuristic algorithm. We observe that the computation time of the heuristic algorithm is 729 times smaller than that of the MILP approach on average; as the problem size increases, the computation time of the MILP approach

Table 5.6: W-UPs and computation times of MILP approach and heuristic algorithm.

$\|N\|$	$\|F\|$	Γ	W-UP		computation times (s)	
			MILP	Heuristic	MILP	Heuristic
5	10	2	0.00750	0.00750	15.98	11.70
5	15	2	0.00775	0.00775	339.98	14.25
5	20	2	0.01099	0.01110	2108.95	21.84
5	25	2	0.01099	0.01167	4602.74	26.22
6	10	3	0.00750	0.00750	138.24	28.03
6	15	3	0.00750	0.00766	1760.25	32.72
6	20	3	0.00776	0.00780	63022.89	40.53
6	25	3	0.01099	0.01129	182170.30	47.44

Table 5.7: W-UPs, availability, and computation times of larger size problems with different $|F|$ obtained with heuristic algorithm.

	$\|F\|$				
	100	150	200	250	300
W-UP	0.00769	0.00820	0.00879	0.01088	0.01363
Availability	0.99231	0.99181	0.99121	0.98912	0.98637
computation times (s)	454.6	736.1	3381.2	6907.5	15303.3

increases. The larger the problem size is, the more the computation time of the heuristic algorithm is reduced compared with that of the MILP approach. W-UPs derived by the heuristic algorithm is 1.6% larger than that of the MILP approach on average among eight tests. The average difference among tests between W-UPs derived by the heuristic algorithm and the MILP approach is

0.0001625.

5.4.6 Larger-size problem

First, this work investigates the performance of the developed heuristic algorithm to solve larger-size problems. Table 5.7 shows W-UPs, availability, and computation times for larger-size problems solved with the developed heuristic algorithm. Similar to Section 5.4.4, the ratios of the threshold to the capacity of a node are randomly distributed over the range of [0.4, 0.8]. W-UPs in Table 5.7 show the same tendency with the smaller size problem; the computation time increases as the problem size increases. As shown in Table 5.7, the difference between the availability with $|F| = 300$ and that of $|F| = 100$ is 0.00595. Since the proposed model is designed to preventively determine the initial resource allocations and recovery configuration against failures before services run by considering the probability of each failure pattern, instead of determining the recovery configuration in service run-time after any failure occurs, the computation time of the developed heuristic algorithm to solve the problem is acceptable in this application scenario. It is validated that the proposed model can obtain the primary and backup resource allocation with priority setting for larger-size problems in practical time.

Second, this work compares the performance of the developed heuristic algorithm and the other heuristic algorithms introduced in literature [77, 117]. The work in [117] addressed an SA-based load balancing algorithm with being aware of workload. It transforms the solutions in the algorithm by re-allocating a random chosen function from the node with the highest workload to the node with the lowest workload. The SA-based algorithm accepts the solution with a worse objective value than the existing solution with a certain probability. This work sets the same T_{init} and temperature decreasing rate for the SA-based and developed algorithm. The work in [77] addressed a water-filling-based load balancing algorithm, which is a greedy algorithm, to minimize the maximum utilization among nodes. It determines the maximum water level (utilization ratio) in advance and allocates the resources of each function iteratively until the maximum water level. This work compares the performance of the algorithms in three different scenarios. Scenario 1 considers a stable running status

Table 5.8: Comparisons of W-UPs and computation times of different algorithms in different scenarios with $\Gamma = 5$.

Solving Approaches		Scenarios		
		1	2	3
Developed heuristic algorithm	W-UP	0.0078	0.0123	0.0158
	time (s)	741.4	80787.0	167.3
SA-based algorithm [117]	W-UP	0.4750	0.5011	0.5049
	time (s)	278.5	354.1	132.0
Water-filing-based algorithm [77]	W-UP	0.0102	0.0259	Infeasible
	time (s)	2.1	3.0	Infeasible

with 20 nodes and 100 functions; scenario 2 considers an increasing number of resource requests with 500 functions; scenario 3 considers resource-hungry services with 10 nodes and 100 functions, where random ten of these functions request 10 times of the computing resources compared with their stable running status. Based on the analysis of Section 5.4, in scenarios 2 and 3 with insufficient resource capacity, each node may host more functions while there is a less remaining capacity that can be used for unavailable function recovery. In other words, d_k and n_k for each node are larger than and smaller than those with stable running status, scenario 1, respectively, which leads to more severe unsuccessful recovery. Similarly, since each node hosts more functions and more functions become unavailable due to the node failures, $\mathcal{U}_2$ increases in scenarios 2 and 3.

Table 5.8 shows the comparisons between the developed heuristic algorithm and different algorithms in the three scenarios. First, focusing on the comparison between the developed algorithm and the water-filling-based greedy algorithm in [77], the greedy algorithm saves much computation time. The accuracy of solutions obtained by the developed algorithm with adopting SA is improved at the cost of longer computation time. W-UPs obtained by the developed algorithm are 24% and 53% smaller than those from the greedy al-

gorithm in scenarios 1 and 2, respectively. The greedy algorithm only focuses on the workload without considering the unsuccessful recovery probability; the developed algorithm distributes the backup resources of the functions whose primary resources are hosted by the same node to different nodes. When this work considers scenario 2 with limited computing resources and a higher probability of the unsuccessful recovery, the greedy algorithm shows worse performance in comparison with the developed algorithm than that of scenario 1. In scenario 3, when some functions request tremendous amounts of computing resources, the greedy algorithm cannot find a feasible solution in the case study; parts of resources cannot be efficiently utilized due to the iterative allocation of functions to a node. The developed algorithm with adding SA considers different random transformations of the solutions; it shows superiority to handle the resource-hungry services compared with the greedy algorithm.

Second, this work focuses on the comparison between the developed algorithm and the SA-based algorithm in [117]. Compared with the developed algorithm, the SA-based algorithm in [117] does not take the result of a greedy algorithm as an initial allocation; instead, it adopts a random generated initial allocation. The SA-based algorithm saves computation time for calculating the initial allocation with WAGA. We observe that it is hard for the SA-based algorithm to jump out of a local optimal solution in the case study. The SA-based algorithm only focuses on the workload; it transforms the solutions by re-allocating the functions hosted by the node with the highest workload to the node with the lowest workload; it does not consider random transformations to improve the unsuccessful recovery probability and recovery priority against failures.

5.5 Summary

This chapter proposed a primary and backup resource allocation model with preventive recovery priority setting against multiple failures to minimize W-UP for both dedicated and shared protection. Each node fails with different workload-dependent failure probabilities; each failure pattern has its corresponding weight. This chapter introduced a recovery strategy which is determined at the operation start time and can be applied for each failure pattern.

Once failures are detected, the recoveries are operated with the workload variation according to the priority setting. Besides, this chapter introduced an approach to obtain unsuccessful recovery probability without priority setting. The numerical results observed that the proposed model reduces W-UP compared with baselines. The proposed model, which jointly considers the unsuccessful recovery and load balancing against failures, outperforms the baseline models which consider each type of unavailability separately. The developed heuristic algorithm is approximately 729 times faster than the MILP approach with 1.6% performance penalty on W-UP.

Chapter 6

Robust resource allocation model under uncertain recovery time with workload-dependent failure probability

This chapter proposes a robust function deployment model against uncertain recovery time with satisfying an expected recovery time guarantee to minimize the number of active nodes [118, 119].

The remainder of the chapter is organized as follows. Section 6.1 describes the proposed model. Section 6.2 presents a heuristic algorithm. Section 6.3 presents the numerical results. Section 6.4 summarizes this chapter.

6.1 Model and problem definition

6.1.1 Overall of the considered problem

Figure 6.1 shows the overall of the problem considered in this chapter. As shown in the left hand side of Fig. 6.1, VNFs are hosted by physical nodes deployed on networks and each physical node is connected to the network via a router or switch. The work in [120] introduced a prototype system for connecting VNFs to the network. This system used the OVN [121] as a CNI [122]

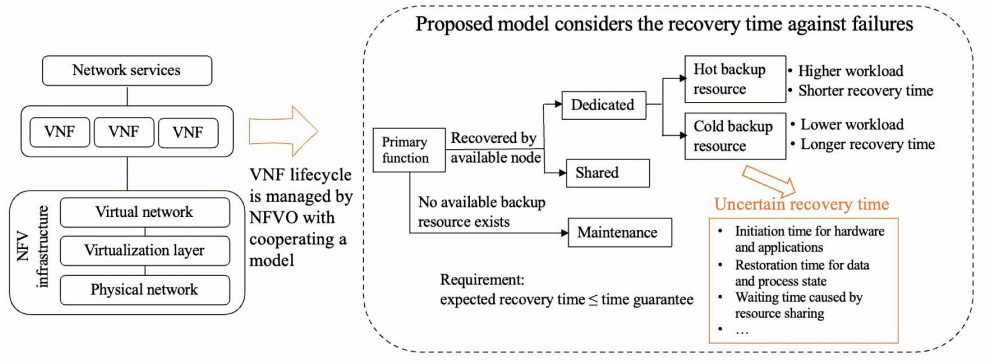


Figure 6.1: Overall of the considered problem in Chapter 6.

to steer the traffic among VNFs in the network. VNFs that run over the NFV infrastructure can be constructed and managed dynamically by NFVO to provide services. NFVO is responsible for managing and orchestrating VNFs. The embedding of VNs on the PNs has been widely studied in previous works. This chapter focuses on the failures of functions with assuming that reliable network resources are provided; it does not include any specific assumption of networking aspects. The proposed model considers VNF placement with satisfying recovery time guarantee and can be cooperated with NFVO. This model considers that the preventively deployed backup resources can recover an unavailable function hosted by a failed node in a period of time, which is related to the backup strategies and protection types, as described in the right hand side of Fig. 6.1. The recovery time depends on the recovery mechanism, such as the types of failures, the time for failure detection and distinction, the initiation time for hardware and applications, and the restoration time for data and process state. It also relates to the size of the function and the activation time for it, which leads to the uncertainty of the recovery time. This section formulates the worst-case of expected recovery time among uncertain recovery time of each function and describe the proposed model to minimize the deployment cost with satisfying the recovery time guarantee.

6.1.2 Model description

Let F and N represent a set of functions and a set of nodes, respectively. Consider $N_j \subseteq N$, where $|N_j| \geq 1$, as a set of nodes which are assigned to

allocate backup resources of function $j \in F$. This model considers that each node is able to offer capacity for both primary and backup resource allocation; let C_i denote the upper bound of computing capacity provided by node $i \in N$. Consider two backup strategies $b \in B := \{0, 1\}$. $b = 0$ denotes the CB strategy and $b = 1$ denotes the HB strategy. A function can be protected by one or more nodes. Let x_{ij}^{kb}, $i \in N, k \in N \backslash \{i\}, j \in F, b \in B$, denote a binary variable; x_{ij}^{kb} is set to one if function j is allocated at node i and is protected by node k with backup strategy b, and zero otherwise.

Let $\chi_{ij}, i \in N, j \in F$, represent a binary variable that equals $\vee_{b \in B} \vee_{k \in N \backslash \{i\}} x_{ij}^{kb}$; it is set to one if function j is allocated to node i, and zero otherwise, where $\vee$ expresses a binary OR. Let $\xi_{jk}^{b}, j \in F, k \in N, b \in B$, denote a binary variable that equals $\vee_{i \in N \backslash \{k\}} x_{ij}^{kb}$; it is set to one if function j is protected by node k with strategy b, and zero otherwise.

Let $W_i, i \in N$, denote the workload of node i. This work assumes that the upper bound of computing capacity for each function protected with CB required for information synchronization and snapshot updating is $r_j^{\mathrm{u}}, j \in F$; l_j^{w} denotes requested workload of function j. W_i is expressed by:

$$W_i = \sum_{j \in F} \{ l_j^{\mathrm{w}} (\chi_{ij} + \xi_{ji}^{1}) + l_j^{\mathrm{u}} \xi_{ji}^{0} \}, \forall i \in F, \tag{6.1}$$

where $\sum_{j \in F} l_j^{\mathrm{w}} \chi_{ij}$ is the workload for the primary resource; $\sum_{j \in F} l_j^{\mathrm{w}} \xi_{ji}^{1}$ is the total requested capacity for functions with the HB resource; $\sum_{j \in F} l_j^{\mathrm{u}} \xi_{ji}^{0}$ is the total requested capacity for functions protected by node i for updating procedure with the CB resource. Let q_i, $i \in N$, denote the workload-dependent failure probability of node i according to the workload.

The functions become unavailable and are required to be recovered when the node hosting them fails. The expected recovery time is related to the backup strategies and node failure probabilities. There are two backup strategies with different recovery times and workloads: cold backup (CB) and hot backup (HB). The CB strategy provides protection to corresponding primary resources, where the backup resources are only reserved without being active; the recovery time is related to the activation time. The backup resources protected with the HB strategy are activated and synchronized with the primary resource so that it can recover the unavailable function immediately. HB re-

duces the recovery time compared with CB at the cost of a higher workload due to the synchronization with the primary resource [62]. Let $t_j^1, j \in F$, be a given recovery time for the HB strategy, which is mainly determined by the availability monitoring mechanism of physical nodes. For instance, t_j^1 is related to the interval of heartbeat packets or the alert thresholds for a monitoring mechanism. Let t_j^0 be a given recovery time for the CB strategy, which is related to the monitoring mechanism, resource activation and synchronization time, and remaining capacity on nodes that can be used for recovery. Let t_j^{m} be a given recovery time of the situation that all nodes hosting the primary and backup resources of a function fail. This situation relies on the maintenance systems, e.g., VMware vCenter Site Recovery Manager, which delivers automated orchestration of fail-over and fail-back to minimize downtime [89].

Let O_{jk} be an integer variable that represents the priority of backup resource in each $k \in N_j$ to recover the unavailable function j.

$$O_{jk} = \sum_{p \in B} (k + p|N|) \xi_{jk}^p, \forall k \in N, j \in F. \tag{6.2}$$

The larger O_{jk} is, the higher priority node k has to recover function j; $O_{jk} = 0, k \in N \backslash N_j$. The expected recovery time of unavailable function j hosted by a failed node can be expressed by:

$$t_j = \sum_{b \in B} \sum_{i \in N} \sum_{k \in N_j} t_j^b x_{ij}^{kb} q_i \prod_{O_{jk'} > O_{jk}: k' \in N_j} q_{k'}(1 - q_k) + t_j^{\mathrm{m}} \prod_{k \in N_j} \sum_{b \in B} \sum_{i \in N} x_{ij}^{kb} q_i q_k, \quad \forall j \in F. \tag{6.3}$$

This work formulates the function deployment problem with deterministic workload and recovery time as the following optimization problem:

$$\min \sum_{i \in N} \vee_{j \in F} (\chi_{ij} \vee \xi_{ji}^0 \vee \xi_{ji}^1) \tag{6.4a}$$

$$\text{s.t.} \sum_{i \in N} \chi_{ij} = 1, \forall j \in F, \tag{6.4b}$$

$$\xi_{jk}^0 + \xi_{jk}^1 \leq 1, \forall j \in F, k \in N, \tag{6.4c}$$

$$t_j \leq T, \forall j \in F, \tag{6.4d}$$

$$W_i \leq C_i, \forall i \in N. \tag{6.4e}$$

Equation (6.4a) minimizes the number of nodes for primary and backup resources of function deployment. Equation (6.4b) ensures that each function in F is allocated into one node in N. Equation (6.4c) ensures that each function protected by a node with at most one strategy, either HB or CB. Equation (6.4d) ensures that the worst-case expected recovery time for each function is smaller than or equal to a given time guarantee, T.

6.1.3 Formulation with uncertain recovery time in resource sharing

In the resource sharing, the recovery time $t_j^0, j \in F$, is also related to the availability of nodes. This work introduces $\tau_{jk}^0, j \in F, k \in \mathbb{N}_j^0$, to replace t_j^0. τ_{jk}^0 represents the recovery time for function j if node k recovers the function when the nodes with higher priority than k are unavailable.

τ_{jk}^0 is related to the availability of nodes in two items: the number of unavailable functions and the number of recoverable functions derived from the remaining capacity. When node i fails, $\Lambda_i = \sum_{j \in F} \chi_{ij}$ functions become unavailable and are required to be recovered. The remaining capacity for node i is $C_i - W_i$.

In the first item, when this work considers the expected recovery time in the first term of (6.3), unavailable function j is recovered by available node k with priority O_{jk} when a node hosting the primary resource of j and the nodes that have higher priority than k fail concurrently. $\sum_{i \in N} \sum_{j' \in F} \chi_{ij}\chi_{ij'}$ primary resources of functions are allocated to the node that hosts the primary resource of function j; $\sum_{i \in N} \sum_{j' \in F} \chi_{ij}\chi_{ij'}$ functions become unavailable concurrently with function j, where $\chi_{ij}\chi_{ij'} = 1, i \in N, j, j' \in F$, represents that the primary resources of function j and j' are allocated to the same node i. Each node k' that has higher priority than node k hosts $\Lambda_{k'}$ functions, which leads to $\sum_{k' \in N_j : O_{jk'} > O_{jk}} \Lambda_{k'}$ unavailable functions. Further, since this work cannot judge the unavailability of the nodes that have lower priority than k, i.e., the nodes that have lower priority than k, this work considers the expected unavailable number of unavailable functions due to the failed nodes, which is equal to $\sum_{k' \in N_j : O_{jk'} < O_{jk}} q_{k'}\Lambda_{k'}$. Let $\mathcal{N}_{jk}$ denote the expected number of unavailable

functions when function j is recovered by available node k. This work has:

$$\mathcal{N}_{jk} = \sum_{i \in N} \sum_{j' \in F} \chi_{ij} \chi_{ij'} + \sum_{k' \in N_j : O_{jk'} > O_{jk}} \Lambda_{k'} + \sum_{k' \in N_j : O_{jk'} < O_{jk}} q_{k'} \Lambda_{k'}, \forall j \in F, k \in N. \tag{6.5}$$

Since each function can only be allocated into one node in N, which is ensured by (6.4b), the three terms are exclusive.

In the second item, this work focuses on the remaining capacity. When node k recovers unavailable function j, $(C_k - W_k)$ remaining capacity of node k can be assigned for recovery. Further, since this work cannot judge the unavailability of the nodes that have lower priority than k, this work considers an expected remaining capacity, $\sum_{k' \in N_j : O_{jk'} < O_{jk}} (1 - q_{k'})(C_{k'} - W_{k'})$. Let $\mathcal{C}_{jk}$ denote the expected remaining capacity when function j is recovered by available node k. This work has:

$$\mathcal{C}_{jk} = (C_k - W_k) + \sum_{k' \in N_j : O_{jk'} < O_{jk}} (1 - q_{k'})(C_{k'} - W_{k'}), \forall j \in F, k \in N. \tag{6.6}$$

For an unavailable function, if it fails and there is not sufficient remaining capacity that can be assigned for recovery of the unavailable function, a waiting time is required to be considered. Let ρ_{jk} denote a binary variable that is set to one if unavailable function j is recovered by node k with priority O_{jk} without any waiting time, and zero otherwise. This work considers whether a waiting time is required by comparing the expected remaining capacity and the expected number of unavailable functions when function j is recovered by node k.

In the mathematical expression, $\rho_{jk} = 1$ when:

$$\mathcal{N}_{jk} \leq \frac{\mathcal{C}_{jk}}{\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'})}, j \in F, k \in N, \tag{6.7}$$

where the denominator of the right hand side, $\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'})$, is the maximum requested load among functions in F for recovery procedure with the updated information.

With linear form of $\mathcal{N}_{jk}$ and $\mathcal{C}_{jk}$, ρ_{jk} can be expressed by:

$$\mathcal{N}_{jk} - \frac{\mathcal{C}_{jk}}{\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'})} \geq -\rho_{jk} A + \epsilon, j \in F, k \in N, \tag{6.8a}$$

$$\mathcal{N}_{jk} - \frac{C_{jk}}{\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'})} \leq (1 - \rho_{jk})A + \epsilon. \tag{6.8b}$$

Let $\mathbb{N}^b_j \subseteq N_j$ represent a set of nodes that protect function j with strategy $b \in B$. The uncertain recovery time is caused by two aspects, which is related to $\rho_{jk}, j \in F, k \in N$. First, this work considers the situation with $\rho_{jk} = 1$, which indicates that function j is recovered by node k without any waiting procedure in resource sharing. This work considers that τ^0_{jk} does not depend on k, and have lower and upper bounds τ^{l}_j and τ^{u}_j, respectively, for each j, which can be collected by the trace logs [1]. This uncertainty is related to the activation time for the function itself. Different from the recovery time t^1_j and t^{m}_j, which depend on recovery mechanisms with the maximum recovery time guarantee given by a service provider, recovery time τ^0_{jk} depends on the size of function and the activation time for it, which has larger estimable uncertainty than t^1_j and t^{m}_j. To cover this aspect, this work considers that each recovery time for function j has its lower and upper bounds τ^{l}_j and τ^{u}_j, which can be expressed by:

$$\rho_{jk}\tau^{\mathrm{l}}_j \leq \tau^0_{jk} \leq \rho_{jk}\tau^{\mathrm{u}}_j, \forall j \in F, k \in \mathbb{N}^0_j. \tag{6.9}$$

Second, this work considers the situation with $\rho_{jk} = 0$. The more concurrent failed nodes are, the more the unavailable functions are. With an insufficient remaining capacity, a longer waiting time is required for a function recovery. To cover the uncertainty of recovery time considering the total remaining capacity, this work introduces a coefficient ω_{jk} that equals $(\max_{j' \in F} (l^{\mathrm{w}}_{j'} - l^{\mathrm{u}}_{j'})\mathcal{N}_{jk} - C_{jk})$ and a given coefficient time for waiting, τ_{w}, where ω_{jk} shows the relationship between the remaining capacity and the number of unavailable functions when function j is recovered by node k. The waiting time τ_{w} can also be collected by the trace logs. The uncertainty set of recovery time with considering resource sharing with $\rho_{jk} = 0$ can be expressed by:

$$\tau^0_{jk} \leq \tau^{\mathrm{u}}_j + (1 - \rho_{jk})\tau_{\mathrm{w}}\omega_{jk}, \forall j \in F, k \in \mathbb{N}^0_j. \tag{6.10}$$

Third, this work considers the average recovery time among nodes that protect function j with the CB strategy for each j, where the upper bound of the average recovery time can also be collected by the trace logs [1]. While

the recovery times of function protected by different nodes may have different variations, it can be upper-bounded by the average downtime or average recovery time [123, 124]. The uncertainty of average recovery time is considered by:

$$\frac{\sum_{k\in\mathbb{N}_j^0}\tau_{jk}^0}{|\mathbb{N}_j^0|} \leq \tau_j^{\mathrm{E}}, \forall j \in F, \tag{6.11}$$

where τ_j^{E} is a given parameter representing the maximum average recovery time.

Let $\boldsymbol{\tau_j} \in \mathbb{R}^{|\mathbb{N}_j^0|}$ be a set of $\tau_{jk}^0, j \in F, k \in \mathbb{N}_j^0$, where $\mathbb{R}^{|\mathbb{N}_j^0|}$ is a set of $|\mathbb{N}_j^0|$-dimensional real-number vectors. Let $U_{jk} = (1-\rho_{jk})\tau_j^{\mathrm{u}} + \rho_{jk}\tau_{\mathrm{w}}\omega_{jk}$ and $L_{jk} = (1-\rho_{jk})\tau_j^{\mathrm{l}}$ represent the upper and lower bounds of τ_{jk}^0, respectively. Combining (6.9), (6.10), and (6.11), the uncertainty set of the recovery time for each function $j \in F$ under resource sharing is defined as:

$$\mathcal{T}_j = \left\{ \boldsymbol{\tau_j} \in \mathbb{R}^{|\mathbb{N}_j^0|} \;\middle|\; \begin{array}{c} L_{jk} \leq \tau_{jk}^0 \leq U_{jk}, \forall k \in \mathbb{N}_j^0 \\ \frac{\sum_{k\in\mathbb{N}_j^0}\tau_{jk}^0}{|\mathbb{N}_j^0|} \leq \tau_j^{\mathrm{E}} \end{array} \right\}, \forall j \in F. \tag{6.12}$$

6.1.4 Dual formulation and MILP formulation

To obtain the optimal function deployment against uncertain recovery time with a non-deceasing workload-dependent failure probability. This work needs to linearize the failure probability and handle the maximization problem under an uncertain recovery time.

First, this work considers how to express the workload-dependent failure probability in the MILP problem. Given the workload-dependent failure probability, which is assumed to be a non-decreasing function, this work can use an $\mathcal{S}$-step function, where $\mathcal{S} \geq 2$, to conservatively approximate a non-decreasing function of the node workload, as shown in Fig. 6.2; $\mathcal{S}$ denotes the number of steps in a step function. q_i can be expressed by the following $\mathcal{S}$-step function; let $s \in \mathbb{S} = [1, \mathcal{S}]$ denote the sth step in $\mathcal{S}$-step workload-dependent failure

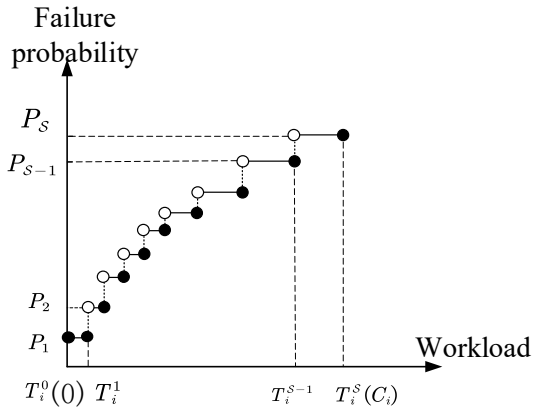


Figure 6.2: Workload-dependent failure probability is expressed by a monotone increasing $\mathcal{S}$-step function.

probability:

$$q_i = \begin{cases} P_1, & T_i^0 \le W_i \le T_i^1 \\ P_2, & T_i^1 < W_i \le T_i^2 \\ \vdots & \vdots \\ P_{\mathcal{S}}, & T_i^{\mathcal{S}-1} < W_i \le T_i^{\mathcal{S}}, \end{cases} \tag{6.13}$$

where $T_i^0 = 0$ and $T_i^{\mathcal{S}} = C_i, i \in N$. When the workload of a node increases from the range of $(T_i^{s-1}, T_i^s]$ to $(T_i^s, T_i^{s+1}]$, the failure probability increases from P_{s-1} to P_s. As the workload increases, even though the node has remaining capacity, the node becomes fragile, and has a higher failure probability to handle the extra workload. By introducing binary variable $z_{is}, i \in N, s \in \mathbb{S}$, which is set to one if $T_i^{s-1} < W_i \le T_i^s$, and zero otherwise, (6.10) is linearized with the same approach in [114] with several auxiliary variables.

Second, to handle the expected recovery time guarantee (6.4d), it is required to guarantee that the recovery time of any function j under the worst case against the uncertain recovery time does not exceed the maximum tolerable recovery time T. This work considers the dual problem for the maximum recovery time under the uncertainty with assuming that x_{ij}^{kb} and ξ_{jk}^0 are fixed; the failure probability q_i is also fixed by following the same approach in [86].

Let $Q_{jk} = \sum_{b \in B} \sum_{i \in N} x_{ij}^{kb} q_i \prod_{O_{jk'} > O_{jk}: k' \in N_j} q_{k'}(1 - q_k)$ represent the failure probability if function j is recovered by node k with priority O_{jk}. Equa-

tion (6.4d) combined with (6.3) for each $j \in F$ can be rewritten by:

$$\max_{\tau_j \in \mathcal{T}_j} \left(\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk} \right) \leq T - \sum_{k \in \mathbb{N}_j^1} \tau_{jk}^1 Q_{jk} - t_j^{\mathrm{m}} \prod_{k \in N_j} \sum_{b \in B} \sum_{i \in N} x_{ij}^{kb} q_i q_k, \forall j \in F. \tag{6.14}$$

With uncertainty set $\mathcal{T}_j$, the left hand side of (6.14) for each $j \in F$ can be formulated by a optimization problem:

$$\max \sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk} \tag{6.15a}$$

$$\text{s.t.} \quad L_{jk} \leq \tau_{jk}^0 \leq U_{jk}, \forall k \in \mathbb{N}_j^0, \tag{6.15b}$$

$$\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 \leq \tau_j^{\mathrm{E}} \sum_{k \in N} \xi_{jk}^0, \tag{6.15c}$$

$$\tau_{jk}^0 \geq 0, \forall k \in \mathbb{N}_j^0. \tag{6.15d}$$

The dual problem of (6.15a)-(6.15d) is given by:

$$\min \sum_{k \in \mathbb{N}_j^0} \left(U_{jk} \pi_{jk}^0 - L_{jk} \lambda_{jk}^0 \right) + \mu_j^0 \tau_j^{\mathrm{E}} \sum_{k \in N} \xi_{jk}^0 \tag{6.16a}$$

$$\text{s.t.} \ \pi_{jk}^0 - \lambda_{jk}^0 + \mu_j^0 \geq Q_{jk}, \forall j \in F, k \in \mathbb{N}_j^0, \tag{6.16b}$$

$$\pi_{jk}^0 \geq 0, \forall j \in F, k \in \mathbb{N}_j^0, \tag{6.16c}$$

$$\lambda_{jk}^0 \geq 0, \forall j \in F, k \in \mathbb{N}_j^0, \tag{6.16d}$$

where π_{jk}^0, λ_{jk}^0, and μ_j^0 are variables introduced in the dual formulation.

Since the duality gap between (6.16a)-(6.16d) and (6.15a)-(6.15d) is zero, by using the duality theorem, this work has:

$$\max_{\tau_j \in \mathcal{T}_j} \sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk} = \min \sum_{k \in \mathbb{N}_j^0} \left(U_{jk} \pi_{jk}^0 - L_{jk} \lambda_{jk}^0 \right) + \mu_j^0 \tau_j^{\mathrm{E}} \sum_{k \in N} \xi_{jk}^0, \forall j \in F. \tag{6.17}$$

Since π_{jk}^0, λ_{jk}^0, and μ_{jk}^0 are not binary variables and (6.16b) considers only the nodes that protects function j with the CB strategy, this work takes π_{jk}^0

as an example and introduce π'^0_{jk} is equal to π^0_{jk}, if $\xi^0_{jk} = 1$. With $\xi^0_{jk} = 0$, $\pi'^0_{jk} = 0$. π'^0_{jk} can be lineaized as:

$$\pi'^0_{jk} \leq \pi^0_{jk} + (1 - \xi^0_{jk})A, \forall j \in F, k \in N, \tag{6.18a}$$

$$\pi'^0_{jk} \geq \pi^0_{jk} - (1 - \xi^0_{jk})A, \forall j \in F, k \in N, \tag{6.18b}$$

$$\pi'^0_{jk} \leq \xi^0_{jk}A, \forall j \in F, k \in N, \tag{6.18c}$$

$$\pi'^0_{jk} \geq -\xi^0_{jk}A, \forall j \in F, k \in N. \tag{6.18d}$$

Similarly, λ^0_{jk}, μ^0_{jk}, and the left hand of (6.14) can be linearized with the same approach in (6.18a)-(6.18d).

Since the minimum value among a set does not exceed a given value if there exists one value in the set that does not exceed the given value, (6.11) can be rewritten by:

$$\sum_{k \in \mathbb{N}^0_j} \left(U_{jk}\pi^0_{jk} - L_{jk}\lambda^0_{jk} \right) + \mu^0_j \tau^{\mathrm{E}}_j \sum_{k \in N} \xi^0_{jk} \leq T - \sum_{k \in \mathbb{N}^1_j} \tau^1_{jk} Q_{jk} - t^{\mathrm{m}}_j \prod_{k \in N_j} \sum_{b \in B} \sum_{i \in N} x^{kb}_{ij} q_i q_k, \forall j \in F. \tag{6.19}$$

Finally, (6.4a)-(6.4e) are transformed into the following optimization problem:

$$\min \sum_{i \in N} \vee_{j \in F} (\chi_{ij} \vee \xi^0_{ji} \vee \xi^1_{ji}) \tag{6.20a}$$

$$\text{s.t.} (6.4\text{b})\text{-}(6.4\text{c}), (6.4\text{e}), (6.5)\text{-}(6.7), (6.14), (6.16\text{b})\text{-}(6.16\text{d}), (6.19). \tag{6.20b}$$

χ_{ij}, ξ^b_{jk}, and Q_{jk} can be linearized with the same approach described in [114] and [33]. Appendix G summarizes a part of the linearization process that is not mentioned in the previous works.

6.2 Heuristic algorithm

This work defines a decision problem related to the considered problem as: given a set of functions F and a set of nodes N, is it possible to find an assignment of function deployment so that the maximum expected recovery

time among functions is no more than c? Similar to [33], when $L_{jk} = U_{jk}$ and $\tau_j^{\mathrm{E}} = U_{jk}$, the decision version of the considered problem is an NP-complete problem by reducing the multiple backup resource allocation problem in [33], which is an NP-complete problem. It indicates that the considered problem becomes difficult to solve in a practical time as the problem size increases.

This section presents an idea of a heuristic algorithm that can obtain an approximate solution of the considered problem with a larger problem size in a practical time with a randomized heuristic algorithm. The procedure of simulated annealing is as follows. This work assumes a minimization problem. Firstly, an initial solution is generated with a greedy approach (Algorithm 6.1), and the objective value is evaluated. Secondly, a neighboring point of the current solution is randomly chosen as a new solution, and the new objective value is evaluated. The new solution is accepted probabilistically, where the probability is determined by the relationship between the current and new objective values.

Algorithm 6.1 Greedy algorithm with protection types and backup strategies

Input: $F, N, C_i, l_j^{\mathrm{w}}, l_j^{\mathrm{u}}, T$

Output: x_{ij}^{kb}, t_j

1: Sort N by C_i decreasingly.

2: Iteratively allocate the primary resource of function j to each node until the workload exceeds its capacity. Record $N_j'' = N \backslash \{\text{the allocated node}\}$.

3: **for** $j = 1 \rightarrow |F|$ **do**

4: **while** $t_j \leq T \times \epsilon_1$ **do**

5: Sort N_j'' according to the number of allocated functions on each element decreasingly.

6: Allocate a backup resource for function j to the first node in N_j'' until the workload exceeds $C_i' = \epsilon_2 \times C_i$, where the function is allocated to the next node in sorted N_j'' if the workload exceeds.

7: Calculate the expected recovery time t_j for each function. If $t_j \leq T \times \epsilon_1, \forall j \in F$, turn to Line 11; otherwise, eliminate the allocation and backtracking to the line 3 with only the primary allocation and turn to Line 8.

8: Allocate a backup resource for function j to the first node in N_j''

until the workload exceeds its capacity, where the function is allocated to the next node in sorted N_j'' if the workload exceeds. Update N_j''.

9: Calculate the expected recovery time for each function with each backup strategy, i.e., HB or CB. Record the smaller one as t_j.

10: **end while**

11: **end for**

12: Record the number of active nodes as n.

13: **return** x_{ij}^{kb} ,t_j, n

Algorithm 6.1 determines an initial allocation on function deployments by allocating functions on each node $i \in N$ iteratively for the primary resource (line 2). The backup resources are also allocated on each node $i \in N$ iteratively with further considering protection types and backup strategies (lines 3-11). Since Algorithm 6.1 is only developed to provide an approximate solution of the considered problem, parameters ϵ_1 and ϵ_2 are designed for providing tolerances with exceeding the expected time guarantee and capacity, respectively. Algorithm 6.1 tries to provide a solution of function deployment with as few nodes as possible with tolerance parameters ϵ_1 and ϵ_2.

Lines 6-7 and 8-9 present the allocation of shared and dedicated protection, respectively. Algorithm 6.1 first tries the allocation with the shared protection to check whether the allocation satisfies the recovery time guarantee with tolerance parameter ϵ_1. In the allocation with the shared protection, ϵ_2 times of the capacity of each node is allowed to be used; the backup resources are allowed to be allocated to the same node with exceeding capacity (lines 6-7). With the same number of active nodes, if the allocation with the shared protection cannot satisfy the recovery time guarantee, Algorithm 6.1 eliminates the allocation and backtracking to Line 3 with only the primary allocation and turns to Line 8. Lines 8-9 then obtain the minimum number of active nodes with the dedicated protection that satisfies ϵ_1 times of the recovery time guarantee. More specially, the backup resources are allocated to the node iteratively until exceeding capacity. After the allocation of all the functions for each round, Algorithm 6.1 calculates the expected recovery time for each function with each backup strategy, i.e., HB or CB, and records the smaller one as t_j. After Algorithm 6.1 obtains a feasible solution that satisfies the re-

laxed recovery time guarantee, line 12 records the number of active nodes as n, which is used in the following randomized heuristic algorithm. Algorithm 6.1 terminates with returning the resource allocation x_{ij}^{kb}, the expected recovery time t_j, and the active number of nodes n.

Algorithm 6.2 Simulated annealing

Input: $F, N, C_i, l_j^{\mathrm{w}}, l_j^{\mathrm{u}}, T_{\min}, T_{\mathrm{init}}, x_{ij}^{kb}$ obtained by Algorithm 6.1
Output: x_{ij}^{kb}, t_j, n

1: **for** $n \rightarrow |N|$ **do**
2: $T \leftarrow T_{\mathrm{init}}$
3: Add a new backup resource of each function in F on the newly active node regardless of the capacity constraint.
4: **while** $T \geq T_{\min}$ **do**
5: Release a randomly chosen backup resource of a randomly chosen function if the function has multiple backup resources. Update N_j''.
6: Allocate a new backup resource of a randomly chosen function to a randomly chosen node in N_j''. Update N_j''.
7: Re-allocate j with random number from a randomly chosen node in $N \backslash N_j''$ to a randomly chosen node in N_j''. Update N_j''.
8: **while** the workload W_i of a node exceeds its capacity C_i **do**
9: Switch the allocation a primary function hosted by the node with one of its backup resources.
10: **end while**
11: Calculate the expected recovery time t_j by Algorithm 6.3 using updated solution $\mathbf{x}'$.
12: **if** $t_j \leq t$ **then**
13: **Return** Δ, x_{ij}^{kb}, t_j
14: **else**
15: $x_{ij}^{kb} \leftarrow x_{ij}^{kb\prime}$ with probability of $\min(1, q)$, where $q = e^{-\frac{t_j' - t_j}{T}}$.
16: $T = T \cdot$ a given decreasing rate
17: **end if**
18: **end while**
19: **end for**

Algorithm 6.2 aims to find a feasible solution that satisfies the recovery

time guarantee with as few active nodes as possible by adjusting the function deployments. This algorithm first gives an initial function deployment in a greedy manner (Algorithm 6.1) and record the number of active nodes as n. Since the backup resource of each function is not allowed to be allocated to the same nodes that host the primary resource of the function, this algorithm only considers nodes in set N''_j, which is N excluding the allocated node, for function j (lines 2, 6, and 8 in Algorithm 6.1). The backup resource of function j is allocated to a node in N''_j that hosts the most functions, i.e., the busiest node, to activate as few nodes as possible that are assigned for function deployment. When the workload of the first node exceeds the capacity, the function is allocated to the next node hosting the most functions with enough remaining capacity to satisfy the capacity constraints.

Since the considered problem aims to minimize the number of active nodes while satisfying the reliability guarantees, Algorithm 6.2 reduces the expected recovery time and unsuccessful recovery probability when a given number of nodes are admitted to be used (lines 2-18), where the expected recovery time is calculated by Algorithm 6.3. Similar to the algorithm developed in [125], with fixing the number of active nodes, in each iteration of decreasing the temperature, the existing function deployment changes randomly to generate a new solution (lines 5-9) and calculates new reliability indicators with Algorithm 6.3. When the recovery time calculated by Algorithm 6.3 satisfies the recovery time guarantee, Algorithm 6.2 is terminated by accepting and returning the corresponding solution (lines 12-13). When any of the reliability indicators does not satisfy the reliability guarantees until the temperature is decreased to the set minimum temperature $T_{\min}$, one more node is permitted to be activated (line 1); a new round of the iteration of decreasing the temperature from T_{init} (repeating lines 4-18) with randomly changing the deployment from the initial allocation obtained in lines 2-6. The algorithm accepts the solution with a worse reliability indicator than the existing solution with a certain probability of $\min(1, e^{-\frac{t'_j - t_j}{T}})$; the higher temperature is, the higher probability to accept a worse solution is (line 15). The iterations terminate when the obtained allocation satisfies the requirement of the recovery time guarantee or the minimum temperature with the maximum admissible number of nodes is reached. The accuracy of solutions by using simulated annealing based on Algorithm 6.1 can

be improved with the cost of longer computation time.

Algorithm 6.3 Calculation of recovery times under uncertainty

Input: $x_{ij}^{km}, T_i^s, P_s, \tau_j^{\mathrm{u}}, \tau_j^{\mathrm{l}}, \tau_j^{\mathrm{E}}$.

Output: t_j

1: Calculate the workload-dependent failure probability of each node based on their workload based on (6.1) and (6.13).
2: Calculate the unavailability Q_{jk} with $\sum_{b\in B}\sum_{i\in N} x_{ij}^{kb} q_i \prod_{O_{jk'}>O_{jk}:k'\in N_j} q_{k'}(1-q_k)$.
3: Calculate the expected remaining capacity C_{jk} and the expected number of unavailable functions $\mathcal{N}_{jk}$ when function j is recovered by node k based on (6.6) and (6.7).
4: Judge the protection type by the relationship between C_{jk} and $\mathcal{N}_{jk}$ and obtain ρ_{jk}.
5: Calculate the upper bound U_{jk} of recovery time t_{jk} with considering the protection types based on (6.10).
6: Find maximum value of $\sum_{k\in\mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$, where τ_{jk}^0 is in the range of uncertainty set $\mathcal{T}_j$ introduced in (6.12).
7: Calculate the expected recovery time based on (6.3)

Algorithm 6.2 focuses on finding a solution that satisfies the recovery time guarantee by random changing the resource allocation while Algorithm 6.3 calculates the expected recovery time with handling the uncertain recovery time. More specifically, Algorithm 6.3 fixes the primary and backup resources allocation obtained in Algorithm 6.2 and calculates the upper and lower bounds of the recovery time in the allocation. It finds the maximum expected recovery time t_j with uncertain recovery time τ_{jk}^0 among the uncertain recovery time set $\mathcal{T}_j$ in (6.12).

Then this algorithm focuses on how to find the maximum value of $\sum_{k\in\mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$, for each $j \in F$. With the fixed allocation, it becomes an LP problem. Each LP problem can be transformed into a minimum cost flow problem, which is inspired from [126]. A directed graph is expressed by $G_j(V_j, E_j)$, where V_j and E_j denote a set of all nodes and a set of all edges, respectively. Instead of solving (6.15a)-(6.15d), this algorithm considers the directed graph in Fig. 6.3.

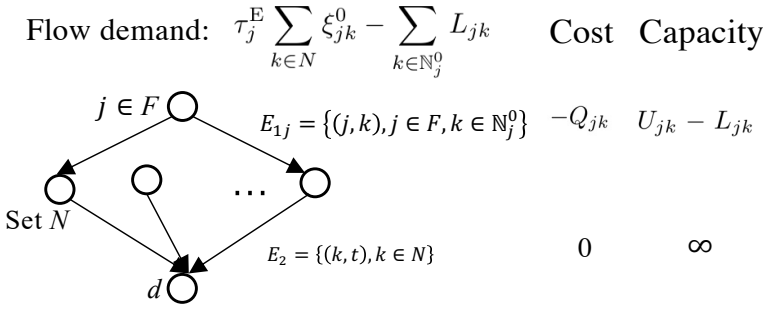


Figure 6.3: Directed graph $G_j(V_j, E_j)$ used for computing maximum value of $\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$.

Directed graph $G_j(V_j, E_j), j \in F$, consists of $V_j = \{j\} \cup N \cup \{t\}$ and $E_{j1} = E_{j1} \cup E_2$, where E_{j1} and E_2 are defined by $E_{j1} \coloneqq \{(j,k) : j \in F, k \in \mathbb{N}_j^0(\xi_{jk}^0 = 1)\}$, and $E_2 \coloneqq \{(k,t) : k \in N\}$, respectively; a source node is j and a destination node is d.

The maximum out flow demand of j is set to $\tau_j^{\mathrm{E}} \sum_{k \in N} \xi_{jk}^0 - \sum_{k \in \mathbb{N}_j^0} L_{jk}$ and capacity of $(j,k) \in E_{j1}$ is set to $U_{jk} - L_{jk}$. The capacity of edge $(k,t) \in E_2$ is set to an infinite value. The cost of each edge $(j,k) \in E_{j1}$ is $-Q_{jk}$. All other edges have 0 cost. The problem is transformed to a minimum cost flow problem in Fig. 6.3. With considering $-Q_{jk}$ as the cost and solving the minimum cost problem, this algorithm can obtain the maximum value of $\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$, which is the absolute value of the minimum cost with the maximum flow demand $(\tau_j^{\mathrm{E}} \sum_{k \in N} \xi_{jk}^0 - \sum_{k \in \mathbb{N}_j^0} L_{jk})$. The problem can be solved by the capacity scaling algorithm [127], which is a capacity scaling successive shortest augmenting path algorithm with computing complexity $O(m^2 \log C)$ [128], where m is the total edge amount and C is the maximum capacity among edges. Please note that, for calculating $\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$, the obtained each flow that outflows from node j to k needs to be added to L_{jk}; $\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$ is calculated with the updated flows.

Thus, $\sum_{k \in \mathbb{N}_j^0} \tau_{jk}^0 Q_{jk}$ for each $j \in F$ in the worst case can be obtained in polynomial time.

6.3 Numerical evaluations

The MILP problems are solved by the IBM(R) ILOG(R) CPLEX(R) Interactive Optimizer with version 12.7.1 [93] implemented by Python 3.7, using Intel Core i7-7700 3.60 GHz 4-core CPU with 32 GB memory. In this section, this work investigates the effects of adopting the recovery time guarantee, uncertainty set, and shared protection in the proposed model; this work investigates the dependency on the bounds of recovery time.

6.3.1 Experiment settings, baselines, and demonstration

In the experiments, this evaluation sets the recovery time of each function randomly in a certain range. Recovery times t_j^{l} and t_j^{u} for CB are randomly distributed over ranges of [0.5, 1] and [1, 5], respectively, for different functions. Maximum average recovery time τ_j^{E} and waiting time τ_{w} are distributed over ranges of [2, 8] and [0.5, 2], respectively. Unavailable times t_j^{l} for HB and t_j^{m} for maintenance are randomly distributed over the range of ranges of [0.01, 0.5] and [20, 200], respectively, where the unit for the times is second. This evaluation sets $P_{\mathrm{H}} = 0.0175$ and $P_{\mathrm{L}} = 0.0025$ and each threshold is equivalent to the half of each capacity of a node to approximate a non-decreasing workload-dependent failure probability. The work in [88] discussed the procedure of the approximation of non-decreasing function, where two examples of step functions fitting the workload-dependent failure probability curve were presented. The requested loads of functions l_j^{w} and l_j^{u} are randomly distributed over the range of [1, 5] and [0.1, 0.5], respectively.

Then, this work introduces five baseline models, as listed in Table 6.1, where the features of the proposed model and baselines are summarized. Baseline 1, which was introduced in [33], addresses the function deployment, where each function can be protected by HB or CB in dedicated protection to minimize the maximum t_j among functions, which is the maximum expected unavailable time (MEUT) among functions described in (6.3). Baseline 1 considers the uncertain recovery time handled by the same approach as the proposed model. It is solved by an MILP problem minimizing (6.3) and constrained to

Table 6.1: Features of proposed model and baselines.

Model	Objective (Minimize)	Recovery time guarantee	Recovery time description	Protection type
Proposed model	Cost	Yes	Uncertain	Dedicated & shared
Baseline 1	MEUT	No	Uncertain	Dedicated
Baseline 2	Cost	Yes	Deterministic	Dedicated
Baseline 3	Cost	Yes	Deterministic	Dedicated & shared
Baseline 4	Cost	Yes	Uncertain	Dedicated
Baseline 5	Cost	Yes	Uncertain†	Dedicated & shared

Uncertain†: In baseline 5, the upper bound of the uncertain recovery time with the shared protection is given in consideration of the maximum arbitrary recoverable functions for nodes. In the proposed model, it is given in consideration of the expected remaining capacity of a node and the expected number of unavailable functions.

(6.20b). Baselines 2 and 3 consider the deterministic recovery time in each model. All the functions are limited to being protected by the dedicated protection in baselines 2, whereas functions in baseline 3 can be protected by both dedicated and shared protection. Baselines 2 and 3 are solved by an MILP problem to minimize the deployment costs while satisfying the recovery time guarantee. Since the uncertainty set (6.12) is not utilized in these baselines, the maximum value of the recovery time is directly given by U_{jk} given in (6.10). Baseline 2 minimizes the deployment cost described in (6.20a) constrained to (6.4b)- (6.4e), (6.5)-(6.7) and (6.14); baseline 3 minimizes the deployment cost described in (6.20a) constrained to (6.4b)- (6.4e) (6.14). Baseline 4, which was introduced in [118], restricts the functions to be protected

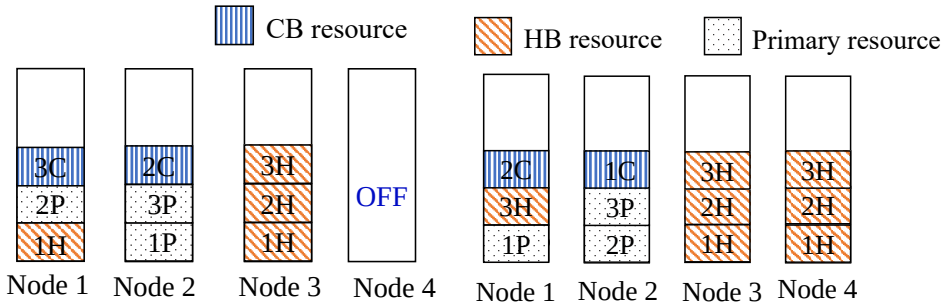
only by dedicated protection, and the uncertainty of the recovery time of functions are considered. The upper bound of its recovery time is given by (6.9). Baseline 4 minimizes the deployment cost described in (6.20a) constrained to (6.4b)- (6.4e) (6.14), (6.14), (6.16b)-(6.16d), and (6.19). Baseline 5 also considers the uncertain recovery time in the dedicated and shared protection, but the baseline considers the approach in [115] to derive the upper bounds of the recovery time with maximum arbitrary recoverable functions for nodes.

This works presents a demonstration to observe the basic characteristics of the proposed model with four nodes and four functions, where the node capacities are 10. This evaluation sets the maximum time guarantee $T = 0.0015$ [s]. As shown in Fig. 6.4, each function in baseline 1 is protected by three nodes with four active nodes and MEUT is 0.001251 [s]. Each function in the proposed model is protected by two nodes with only three active nodes and one node remaining to save cost; MEUT in the proposed model is 0.001321 [s] with satisfying the time guarantee. The proposed model saves 25% cost with 5% increase on MEUT; the increase of MEUT corresponds to a situation that occurs with a probability that all of the primary and three backup nodes fail, which is about 10^{-8}. The computational time to solve the problem is 1.02 [s].

Further, as the number of functions increases, baseline 1 with considering the dedicated protection cannot find a feasible solution with no sufficient remaining capacity for all backup resources. The proposed model allows the CB resource sharing and considers the uncertain recovery time. When this evaluation considers six functions and $T = 0.005$ [s], the proposed model activates all four nodes and MEUT is 0.004379 [s]; baseline 1 cannot obtain a feasible solution.

6.3.2 Effect of recovery time guarantee

This evaluation investigates the effect of recovery time guarantee with focusing on cost-efficiency. The proposed model saves 43% and 26% on the number of active nodes compared with baseline 1 among the tested cases, with setting $T = 0.01$ [s] and 0.005 [s], respectively, as shown in Fig. 6.5. All the five nodes are activated in baseline 1 among the tested cases to distribute the workloads and provide multiple protection against failures of primary and backup nodes.



(a) Propsoed model. (b) Baseline 1.

Figure 6.4: Demonstration of function deployment.

Since baseline 1 only considers minimizing MEUT without saving cost, it tends to provide excessive backup instances for covering improbable scenarios, which leads to excessive redundancy. The proposed model minimizes the number of active nodes with satisfying a time guarantee. Taking the tests with $|F| = 8$ and 10 as examples, with $T = 0.01$ [s], the proposed model only activates two nodes for function deployment. The primary resources are distributed into two nodes to decrease the workload-dependent failure probability. When $T = 0.005$ [s], one CB resource for protection cannot guarantee such an expected recovery time, the proposed model activates the third node so that the functions can be protected by the HB strategy while the workload of primary and HB resources can be distributed into three nodes to decrease the workload-dependent failure probability. Among all the tests, we can observe that more nodes are required to be activated and provide protection for functions as T decreases. The cost increases by 30% when T decreases by 50%. When the time guarantee is relatively loose, the functions can be protected with the CB strategy with shared protection while distributing the primary workload to reduce the failure probability.

Functions in the proposed model can be protected by both dedicated and shared protection. As the number of activated node increases, each function can have a higher probability to be hosted nodes with lower workload and lower τ^0_{jk}, which reduces the expected recovery time for each function. Compared the proposed model with $T = 0.01$ and $T = 0.005$, we observe that the number of activated nodes to satisfy the worst-case expected recovery time guarantee

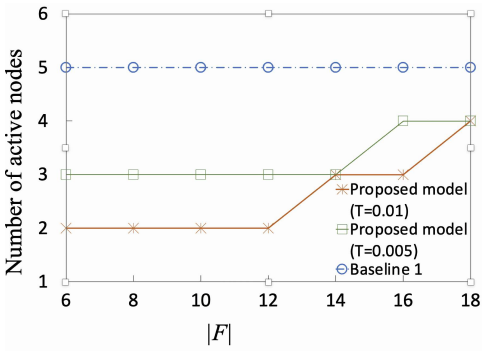


Figure 6.5: Effect of recovery time guarantee.

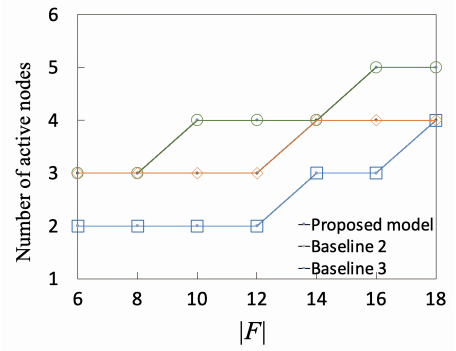


Figure 6.6: Effect of uncertain set of recovery time.

decreases as the recovery time guarantee increases. HB can be adopted when the workload is not too high and functions can be protected with shared CB to reduce cost while satisfying the recovery time guarantee. For the same recovery time guarantee, more nodes are required to be activated for hosting functions to distribute the workload while the functions is required to shared protected with CB to reduce cost when $|F|$ increases and the failure probability of each node increases. Otherwise, the worst-case expected recovery time increases and does not satisfy the recovery time guarantee.

6.3.3 Effect of uncertainty set of recovery time

This evaluation investigates the effect of utilizing the uncertainty set to describe the recovery time. This evaluation compares the proposed model with baselines 2 and 3. The proposed model utilizes the uncertainty set of recovery time with CB. Baselines 2 and 3 can only utilize the maximum empirical value of the recovery time to conservatively provide robustness disregarding that the actual recovery time may be less than the empirical maximum recovery time; they do not consider the empirical average and lower bound of recovery time. U_{jk} in baseline 2 is directly given by the empirical value and U_{jk} in baseline 3 is calculated with the empirical value and the resource allocation based on (6.10).

The proposed model saves 36% and 26% on the number of active nodes compared with baselines 2 and 3 among the tested cases, respectively, as shown

in Fig. 6.6. Since the recovery time τ_{jk}^{0} in baselines 2 and 3 is a deterministic value of U_{jk}, the calculated MEUT in the baselines are always larger than the proposed model when $\sum_{k\in\mathbb{N}_j^0} U_{jk} > \tau_j^{\mathrm{E}} \times |\mathbb{N}_j^0|$. When the number of functions is relatively small, only two nodes are activated in the proposed model. With the same number of activated nodes, baselines 2 and 3 adopting with CB cannot satisfy the recovery time guarantee, since the utilized τ_{jk}^{0} for each k is the maximum value of recovery time; baselines 2 and 3 adopting with HB cannot satisfy the recovery time guarantee since the workload is distributed in a smaller number of nodes so that the workload-dependent failure probability is higher. One more node must be activated to distribute the workload to suppress the failure probability and HB is adopted in the baselines. An increasing tendency of recovery time with the increase of $|F|$ in the proposed model can be observed and more nodes are activated when the worst-case expected recovery time cannot satisfy the recovery time guarantee anymore.

Then this evaluation focuses on the tendency in Fig. 6.6. As $|F|$ increases, the number of functions hosted by the same node increases, while the workload rises, baselines 2 and 3 adopting HB cannot suppress the expected recovery time to satisfy the recovery time guarantee, one more node is activated to distribute the workload. With the same recovery time guarantee, since the proposed model with considering the uncertain set to describe the recovery time, the worst-case expected recovery time with only adopting CB can satisfy the recovery time requirement to achieve the reduction of cost. As $|F|$ further increases, activated nodes in baseline 2 cannot host the functions while satisfying the recovery time requirement, whereas baseline 3 can partially utilize the shared CB to reduce the workload and the deployment cost, this is why baseline 3 has better performance than baseline 2. As the number of functions that are shared and protected by the same node increases, both unsuccessful recovery probability and expected waiting time may increase. The calculated MEUT in baseline 3 cannot satisfy the recovery time guarantee since the utilized τ_{jk}^{0} is the maximum value of recovery time. As $|F|$ increases, the worst-case expected recovery time in the proposed model obtained by utilizing the uncertainty set satisfies the recovery time guarantee while the MEUT calculated in baseline 3 with the upper bound of the recovery time considering the waiting procedure does not satisfy the recovery time guarantee so that one

Table 6.2: Worst-case expected recovery time in proposed model and baselines with different $|F|$.

	$\|F\|$				
	8	10	12	14	16
Proposed model	0.0101	0.0698	0.0125	0.0125	0.0149
Baseline 2	0.0251	-	0.0250	-	0.0250
Baseline 3	0.0251	0.1752	0.0250	0.1750	0.0250
Baseline 4	0.0101	-	0.0125	-	0.0149

−: No feasible solution can be obtained with limited number of activated nodes.

node is required to be activated and distribute the workload and to reduce MEUT.

To compare the worst-case expected recovery time in the proposed model and baselines 2, 3, and 4, this evaluation fixes the number of nodes that can host the resource of functions as the minimum number of active nodes obtained by the proposed model with each parameter setting; this evaluation compares the worst-case expected recovery time obtained by the proposed model and baselines 2, 3, and 4 when only these nodes are activated. In Table 6.2, this evaluation considers that only a limited number of nodes can be used and the worst-case expected recovery time is minimized. To compare the proposed model and baselines 2, 3, and 4 more clearly in terms of the worst-case expected recovery time, this evaluation restricts the proposed model and the baselines to use only CB. From Table 6.2 we can observe the effect of the uncertainty set on the worst-case expected recovery time in the proposed model and compared with baselines 2 and 3, which do not consider the uncertainty set; the worst-case expected recovery time in the proposed model is 52.2% smaller than that of baseline 3.

6.3.4 Dependency on bounds of recovery time

This evaluation investigates the dependency of the settings of bounds describing the recovery time of a function protected by different nodes; τ^0_{jk} is related

to the uncertainty set (6.12). In (6.12), τ_j^{E} represents the average recovery time, L_{jk} and U_{jk} represent the upper and lower bounds of the recovery time, respectively. This evaluation investigates deployment costs obtained by different approaches for different values of coefficient $\eta = \frac{L_{jk}}{U_{jk}}$, $0 \leq \eta \leq 1$, and τ_j^{E}. For each setting, this evaluation conducts 10 trials with different resource allocations and calculate the average deployment cost of all 10 trials with the given settings. This evaluation compares the proposed model with different η and different upper bounds of total recovery time ($\tau_j^{\mathrm{E}}|\mathbb{N}_j^0|$) with baseline 3, which is a model with the same recovery time guarantee and objective function with the same protection types that do not utilize the uncertainty set to efficiently provision functions. Instead, baseline 3 always applies the deterministic recovery time τ_{jk}^0, to provide robustness with ignoring (6.12).

Figure 6.7 shows the deployment costs obtained by the proposed model and baseline 3 for different values of η and upper bound of total recovery time ($\tau_j^{\mathrm{E}}|\mathbb{N}_j^0|$). This evaluation obtains that the deployment cost from the proposed model increases as the value of τ_j^{E} increases. This is because the worst-case expected recovery time increases as τ_j^{E} increases so that more nodes are activated against recovery time guarantee violation. After the value of τ_j^{E} reaches that of $\frac{\sum_{k \in \mathbb{N}_j^0} U_{jk}}{\mathbb{N}_j^0}$, the deployment costs of the proposed model and baseline 3 are comparable. Since baseline 3 is not aware of the uncertainty set, its deployment costs are kept as the same for different values of the upper bound of total recovery time, which are greater than or equivalent to those of the proposed model. We observe that, when $\tau_j^{\mathrm{E}} \leq \frac{\sum_{k \in \mathbb{N}_j^0} U_{jk}}{\mathbb{N}_j^0}$, the deployment cost in the proposed model decreases as the value of η increases. This is because the range of uncertainty decreases as η increases, which decreases the worst-case expected recovery time. By utilizing the uncertainty set, the proposed model with different η saves the deployment cost on average 13%, 18%, and 23% with different η compared to baseline 3.

6.3.5 Effect of shared protection

This evaluation investigates the effect of the shared protection, which reduces deployment costs while increasing waiting time and may further increase the

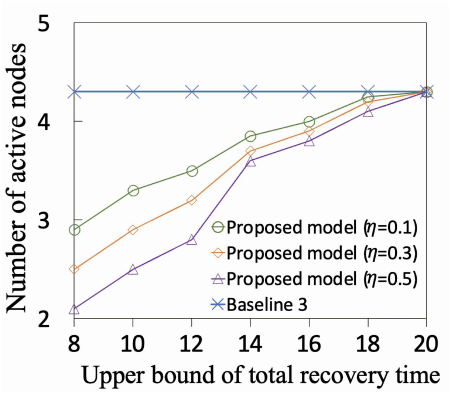


Figure 6.7: Dependency on bounds of recovery time.

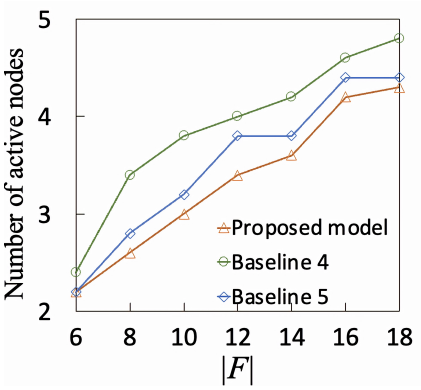


Figure 6.8: Effect of the shared protection.

upper bound of the uncertain recovery time τ_{jk}, as shown in (6.10). This work investigates the cases of dedicated protection and shared protection. This evaluation compares the proposed model with baseline 4, which does not allow backup resource sharing. Figure 6.8 shows the activated number of nodes of the proposed model baseline 4; it is obvious that leveraging the shared protection can increase resource efficiency since multiple functions can be shared protected by a commodity node with CB. By utilizing the shared protection, the proposed model with different $|F|$ saves the deployment cost on average 14% compared to baseline 4 in Fig. 6.8. In the case that only dedicated protection is needed to meet the guarantee, the difference between the proposed model and baseline 4 is relatively small. From Table 6.2 we can observe the effect of the shared backup on the worst-case expected recovery time by comparing the proposed model with baseline 4, which can only adopt the dedicated protection; we can observe that in some cases, the proposed model, which allows shared protection to reduce the resource utilization, can obtain a feasible solution with a limited number of activated nodes; baseline 4 cannot obtain any feasible solution since each backup resource is dedicatedly reserved for each function and the total required workload exceeds the node capacity. Shared protection needs less amount of backup resources than that of functions with dedicated protection but it may not provide complete recovery when multiple functions fail concurrently. More specifically, the utilization of shared protection may lead to the waiting procedure for recovery. With considering the

expected remaining capacity of a node and the expected number of unavailable functions hosted by the node, the upper bound of the τ_{jk}^{0} increases as the number of $\mathcal{N}_{jk}$ increases and C_{jk} decreases, as shown in (6.5), (6.6), and (6.12). When the recovery time guarantee is relatively relaxed, the proposed model shows a larger superiority over baseline 4 compared with a stricter recovery time guarantee.

Then this evaluation compares two approaches to calculate the upper bound of the recovery time of the shared protection. Previous work in [115] introduced the theoretical derivation of the maximum arbitrary recoverable functions for nodes by viewing an instance of assignment between functions $j \in F$ and nodes $k \in N$ in the model as a bipartite graph. Similarly to the work in [115], this work can derive the number of maximum arbitrary recoverable functions of each node. When the number of unavailable functions exceeds the recovery capacity of the host node, there may be a delay in the recovery of functions due to waiting time. Thus, this work can obtain an upper bound of the recovery time τ_{jk} under consideration of waiting time in the shared protection by leveraging the number of maximum arbitrary recoverable functions of each node to replace C_{jk} in the proposed model in (6.6). In baseline 5, this evaluation compares the number of maximum arbitrary recoverable functions and the expected C_{jk}. Similar to (6.7), this evaluation compares $\mathcal{N}_{jk}$ and the number of maximum arbitrary recoverable functions in baseline 5 to obtain ρ_{jk} and ω_{jk}. Since the number of maximum arbitrary recoverable functions is not related to j, this evaluation sets ρ_{jk} and ω_{jk} in baseline 5 are the same for each j protected by the same node k.

Figure 6.8 compares the worst-case expected recovery times obtained by the proposed model and baseline 5 for different numbers of functions with different approaches to obtain the upper bound of the recovery time. By utilizing the shared protection, the proposed model with different $|F|$ saves the deployment cost on average 6% compared to baseline 5. Comparing the worst-case expected recovery times obtained by the proposed model and baseline 5, which adopts different approaches to obtain the uncertain set, we can observe that mode nodes are activated in baseline 5 due to the recovery time guarantee violation. Since the derivation of the number of maximum arbitrary recoverable functions does not consider the current recovery situation (function j is

recovery by which node and the node's priority), it is a coarser granularity and more conservative expression of the recoverable ability of the nodes. Therefore, baseline 5 has a larger upper bound of the uncertain recovery time than the proposed model. Further, the computation time of the proposed model to obtain the optimal solution by solving the MILP problem is about 1.38 times less than that of baseline 5, since the number of maximum arbitrary recoverable functions in baseline 5 requires an addition algorithm to solve the problem in a bipartite graph, which is more complex calculating the expected remaining capacity, C_{jk}, and the expected number of unavailable functions, N_{jk}.

6.3.6 Competitive evaluation on computation time and accuracy

This work shows the comparisons on the number of activated nodes and computation time between the MILP approach and the heuristic algorithm for different numbers of nodes and functions to evaluate the heuristic algorithm for different sizes of problems. This work sets the allowable computation time to 10000 [s] to solve each MILP problem. This work considers a larger size problem with eight nodes with around 20 functions. The developed algorithm is adopted to solve larger-size problems; this evaluation sets $T_{\text{init}} = 100$ and $T_{\text{init}} = 1000$ for two sets of cases; the temperature decreasing rate is set to 0.95 for SA.

Table 6.3 shows the number of activated nodes and computation times for the MILP approach and the heuristic algorithm, where Obj. represents the number of activated nodes and the Comp. time indicates the computation time. We observe that the computation times of the heuristic algorithm with T_{init}=100 and T_{init}=1000 are 74 and 58 times smaller than that of the MILP approach on average in the seven tests, respectively; as the problem size increases, the computation time of the MILP approach increases. The larger the problem size is, the more the computation time of the heuristic algorithm is reduced compared with that of the MILP approach. The case with T_{init}=100 has a solution with higher deployment cost within a shorter computation time than that of T_{init} = 1000. On the other hand, the case with T_{init}=1000 gets a better solution as much as possible in a tolerable time than that of T_{init} = 100.

Table 6.3: Number of activated nodes and computation times of MILP approach and heuristic algorithm with different parameters.

$\|F\|$	MILP		Heuristic			
			T_{init}=100		T_{init}=1000	
	Obj.	Comp. time [s]	Obj.	Comp. time [s]	Obj.	Comp. time [s]
14	3	616.1	3	52.7	3	88.2
16	3	980.4	4	68.8	4	137.2
18	4	1796.3	4	109.7	4	219.2
20	4	3823.5	5	281.3	4	327.3
22	5	57683.5	6	412.5	5	542.3
24	6†	10^5	6	519.7	5	608.7
26	8†	10^5	6	571.6	6	867.1

†: Feasible solution obtained within 10^5 [s]

The computation time of the heuristic algorithm is similar when the same number of nodes is activated. The number of activated nodes derived by the heuristic algorithm is 14% larger than that of the MILP approach on average among the seven tests with T_{init} = 100; it is 28% smaller than that of the MILP approach on average among the seven tests with T_{init} = 1000 since the computation time of MILP approach is limited in a tolerable computing time, 10^5 [s]. The difference between the number of activated nodes derived by the heuristic algorithm and that of the MILP approach does not exceed 1 in the seven tests. The heuristic algorithm activates a fixed number of nodes and randomly adjusts the resource allocation to satisfy the reliability indicators. Similar to the MILP approach, the related calculation of (6.15a)-(6.15d) and (6.16a)-(6.16d) introduces additional variables and increases the computational time complexity to find the worst-case expected recovery time when the recovery time t_j^0 is in the uncertainty set (6.12). In the heuristic algorithm, the worst-case expected recovery time needs to be computed in each initial solution given by Algorithm 6.1 and in each iteration in Algorithm 6.2

by executing Algorithm 6.3, which requires building a graph in Fig. 6.3 and finding the maximum flow of the graph and brings a longer computation time compared with the algorithm that does not consider calculating the worst-case expected recovery time in the uncertain recovery time set.

6.4 Summary

This chapter proposed a robust function deployment model against uncertain recovery time with satisfying an expected recovery time guarantee in a cost-efficient manner. Each node fails with a workload-dependent failure probability; preventive deployed backup resources can recover the unavailable function hosted by a failed node within a period of time related to the backup strategy. The expected recovery time of a function is related to the backup strategy (HB or CB strategy), protection types (dedicated or shared protection), the workload of the node hosting it, and the number of unavailable functions and remaining capacity of available nodes. This chapter introduced an uncertainty set that considers the upper bound of the average recovery time among nodes and the upper and lower bounds of each recovery time. The robust optimization technique was applied to handle the worst-case expected recovery time among the uncertain recovery time satisfying a recovery time guarantee; the model was formulated as an MILP problem. A greedy-based simulated annealing algorithm was developed to address the considered problem in practical scenarios. In the algorithm, this chapter transformed the linear-programming problem to obtain the worst-case expected recovery time among uncertain times into a graph problem. The algorithm decreases the number of active nodes while decreasing the worst-case expected recovery time until the recovery time satisfies the recovery time guarantee. The numerical results showed the superiority of the proposed models by taking into account the recovery time guarantee, uncertainty set, and shared protection, and this chapter investigated the dependency on the uncertain recovery time boundaries.

Chapter 7

Resource allocation strategies for accelerating recovery under reliability guarantees with workload-dependent failure probability

This chapter proposes a primary and backup resource allocation model under reliability guarantees to minimize the deployment cost [125].

The remainder of the chapter is organized as follows. Section 7.1 presents the motivation of the proposed model. Section 7.2 describes the proposed model. Section 7.3 introduces a heuristic algorithm. Section 7.4 presents numerical results that show the performance of the proposed model in different cases. Section 4.5 summarizes this chapter.

7.1 Motivation for considering workload-related recovery time in two aspects

The recovery time relates to different backup modes [60]; the modes have different workloads to provide different pre-configuration states for recovery [61]. If

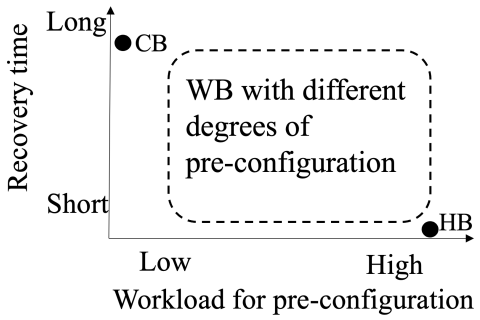


Figure 7.1: Comparison among different backup modes.

the backup resources are only deployed without being activated as an instance, the recovery time of an unavailable function relates to the activation and instantiation time; the backup mode is called cold backup (CB). If a backup instance is activated and synchronized with the primary function so that the backup instance can take over the task running on the function immediately, the recovery time is only affected by the monitoring mechanism of the availability of physical nodes [62]; the backup mode is called hot backup (HB). In addition to the two typical backup modes, warm backup (WB) is also commonly used for cost-efficient prompt failure recovery [63]. In the warm backup, a backup instance of the function is already resident in memory; it is partially or fully initialized for standby; the backup instance synchronizes the state of services from the processing primary function periodically. When the primary function fails, the backup instance can take over the tasks running on the primary function faster than CB based on the pre-configuration. Compared with the HB mode, where the backup instance is fully pre-configured, backup resources with different degrees of the pre-configuration in WB recover the unavailable primary function in a longer or equal time. In other words, the more sufficient pre-configuration (including the instantiation, initialization, and synchronization) a backup instance provides, the faster the recovery procedure can be, as shown in Fig. 7.1. This work considers that there exist multiple states of pre-configuration for different types of service, i.e., stateful and stateless services [64] with different degrees of instantiation, initialization, and synchronization.

The recovery of an unavailable function requires workloads, which include

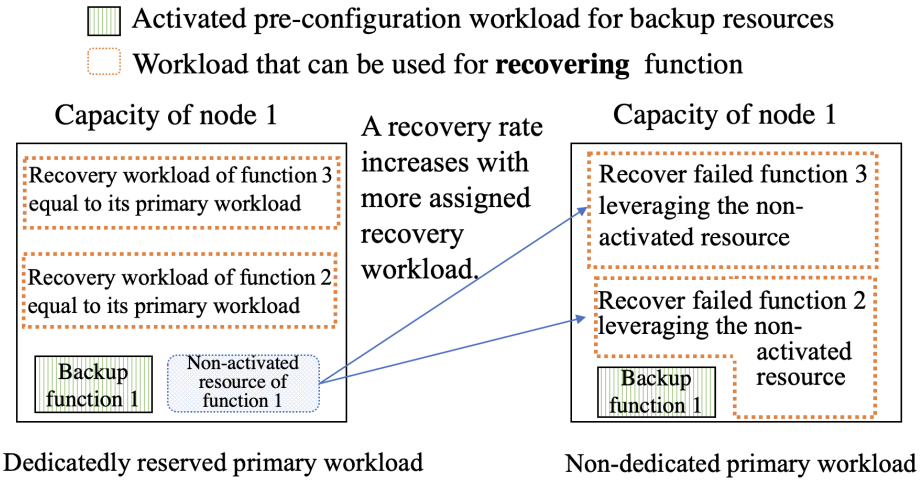


Figure 7.2: Left: primary workload is dedicatedly assigned for each function; right: when function 1 is not being recovered by node 1, the non-activated workload of function 1 can be utilized by functions 2 and 3 to speed up the recovery, where node 1 hosts each backup resource of functions 1, 2, and 3.

the workloads for instantiation, initialization, and synchronization. After recovery, the workload increases from backup workload for pre-configuration to primary workload for steady running. For higher reliability, the workload for a primary resource is commonly dedicatedly reserved for each function with providing redundant backup instances for covering improbable situation of unavailability of nodes, which leads to redundancy. However, if a function is not being recovered by a node, the non-activated resource, which is equal to the difference between the primary and backup workloads, is not utilized. Service providers can utilize the non-activated workload, which is similar to resource scale-up in cloud computing [129, 130]; they should consider a suitable toleration on the function reliability degradation. Figure 7.2 shows an example in which node 1 protects functions 1, 2, and 3. When function 1 is not being recovered by node 1, the non-activated workload of function 1 can be utilized by functions 2 and 3 to speed up their recovery, where a function can be recovered faster if more computing resources are permitted to be used. On the other aspect, the workload of a node during recovering unavailable functions may exceed its capacity and lead to unsuccessful recovery. Thus, suitable toleration on

the function reliability degradation should be considered. The work in [131] considered reducing operating cost by scaling computational resources with satisfying the consumer demand. This work considers the balance between the deployment cost and the recovery time that is affected by the allocated resources. The proposed model reduces the cost while keeping a certain level of reliability, rather than offering unnecessary redundancy; it considers the active number of nodes as the deployment cost, and the workload-dependent failure probability as the proportionate cost [83].

7.2 Optimization model

7.2.1 Model description

Let N and F denote a set of nodes and a set of functions, respectively. Consider N_j, which is a subset of N with $|N_j| \geq 1$, as nodes hosting the backup resources of function $j \in F$. This work considers that each node can host both primary and backup resources. Let w_j^{P} represent the given workload of each running primary function $j \in F$. Considering the backup resources for function recovery, the required pre-configuration state for each function is not the same; this work considers that $M_j, j \in F$, represents a set of backup modes for function j with different degrees of instantiation, initialization, and synchronization. For instance, periodic synchronization can speed up the recovery for the functions of a stateful service, e.g., database; it cannot improve the recovery time for the functions of a stateless service, e.g., web service [64]. Each backup mode $m \in M_j$ corresponds to a processing workload of the backup resource for function $j \in F$, which is represented by w_{jm}^{B}. Let w_{jk}^{R} represents the workload in node $k \in N$ that is permitted to be used by function $j \in F$ for recovery. C_i represents the maximum resource capacity of node $i \in N$ for processing the resources hosted by the node.

x_{ij}^{km}, $i \in N, j \in F, k \in N \backslash \{i\}, m \in M_j$, represents a binary variable; x_{ij}^{km} is 1 if the primary resource of function j is hosted by node i and the function is protected by node k with backup mode m, and otherwise 0. Let χ_{ij} denote a binary variable that equals $\vee_{m \in M_j} \vee_{k \in N \backslash \{i\}} x_{ij}^{km}$; it is 1 if primary resource of function j is hosted by node i, and otherwise 0. ξ_{jk} denotes a binary variable

that equals $\vee_{m \in M_j} \vee_{i \in N \setminus \{k\}} x_{ij}^{km}$; it is 1 if the backup resource of function j is hosted by node k, and otherwise 0.

$L_i^{\mathrm{W}}, i \in N$, denotes the workload of node i, which includes the primary workload of running functions and the pre-configuration backup workload for functions hosted by node i. L_i^W can be given by:

$$L_i^{\mathrm{W}} = \sum_{j \in F} \{w_j^{\mathrm{P}} \chi_{ij} + \sum_{m \in M_j} w_{jm}^{\mathrm{B}} \zeta_{ji}^m\}, \forall i \in N, \tag{7.1}$$

where $\sum_{j \in F} w_j^{\mathrm{P}} \chi_{ij}$ is the workload for the primary resource hosted by node i; $\sum_{j \in F} \sum_{m \in M_j} w_{jm}^{\mathrm{B}} \zeta_{ji}^m$ is the total workloads for the pre-configuration of the backup functions hosted by node i.

By analyzing the empirical relationship in tracelog, the workload-dependent failure probability is given; an $\mathcal{S}$-step function can be adopted to conservatively approximate the failure probability depending on the node workload, as shown in Fig. 7.3. Let $\mathcal{S}$ denote the number of steps; $s \in \mathbb{S} = [1, \mathcal{S}]$ represents the sth step in the failure probability, and this work has $T_i^0 = 0$ and $T_i^{\mathcal{S}} = C_i, i \in N$. q_i can be described by:

$$q_i = \begin{cases} P_1, & T_i^0 \leq L_i^W \leq T_i^1 \\ P_2, & T_i^1 < L_i^W \leq T_i^2 \\ \vdots & \vdots \\ P_{\mathcal{S}}, & T_i^{\mathcal{S}-1} < L_i^W i \leq T_i^{\mathcal{S}}, \end{cases} \tag{7.2}$$

When the workload of a node grows from $(T_i^{s-1}, T_i^s]$ to $(T_i^s, T_i^{s+1}]$,, the failure probability grows from P_{s-1} to P_s. Even if there is remaining capacity for a node, as the workload grows, the node becomes more vulnerable and has a higher failure risk to host some additional workloads.

The backup modes of each function affect the workload-dependent failure probability of the node hosting it and the recovery time if it becomes unavailable under node failures. Let $t_{jm}^{\mathrm{B}}, j \in F, m \in M_j$, be a given recovery time for backup mode m, each of which corresponds to backup workload w_{jm}^{B} for pre-configuration. The recovery time of a function is also related to the recovery ability and the assigned recovery workload of each node that protects the function; the recovery times of a function protected by different nodes assigned with different recovery workloads can be collected and analyzed. Let

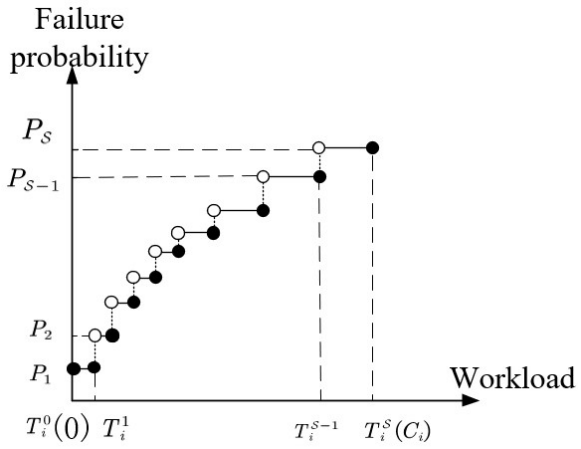
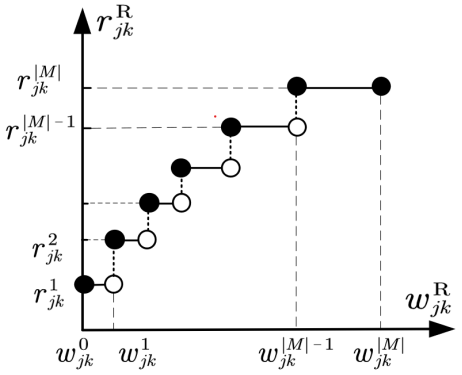


Figure 7.3: Workload-dependent failure probability can be described by an $\mathcal{S}$-step function.

$r_{jk}^{\mathrm{R}}, j \in F, k \in N_j$, represent the rate of node k to recover function j, each of which is related to the assigned recovery workload w_{jk}^{R}. Each recovery rate has been normalized, where normalization refers to a scale of the recovery time on each node whose assigned recovery workload is equal to the primary workload, as the example shown in the left side of Fig. 7.2. The relationship between assigned recovery workload and recovery rate can be approximated by a step function shown Fig. 7.4(a), where Fig. 7.4(b) shows a numerical example of the relationship. The recovery time of function j protected with mode m by node k is $\frac{t_{jm}^{\mathrm{B}}}{r_{jk}^{\mathrm{R}}}$.

As this work considers that a larger assigned recovery workload can increase the recovery rate, this work sets the node assigned with a larger recovery workload to the higher priority among multiple backup resources. O_{jk} is an integer that reflects the priority of a backup instance in $k \in N_j$ for recovering an unavailable function j, which is affected by the assigned recovery and backup workloads. Let $O_{jk} = M_1 w_{jk}^{\mathrm{R}} + M_2 w_{jm}^{\mathrm{B}} \zeta_{jk}^{m}$, where M_1 and M_2 are given multipliers to make $M_1 w_{jk}^{\mathrm{R}}$ and $M_2 w_{jm}^{\mathrm{B}} \zeta_{jk}^{m}$ as integers. This work sets $M_1 \gg M_2$ so that the assigned backup workload affects the priority when the assigned recovery workload is the same for some nodes that host the backup resources of a function. The larger O_{jk} is, the higher priority node k is responsible for recovering function j with; $O_{jk} = 0, k \in N \backslash N_j$.



(a) Relationship between assigned recovery workload and recovery rate is approximated by M-step function.

Range of w^{R}_{jk}	$1/r^{\mathrm{R}}_{jk}$
$[w^{\mathrm{p}}_j, 1.1 \times w^{\mathrm{p}}_j]$	1
$(1.1 \times w^{\mathrm{p}}_j, 1.2 \times w^{\mathrm{p}}_j]$	0.9
$(1.2 \times w^{\mathrm{p}}_j, 1.4 \times w^{\mathrm{p}}_j]$	0.8
$(1.4 \times w^{\mathrm{p}}_j, 2 \times w^{\mathrm{p}}_j]$	0.7

(b) Example, where assigned reocvery workload is comapred with primary workload.

Figure 7.4: Relationship between assigned recovery workload and recovery rate.

7.2.2 Reliability guarantees

This work considers two reliability guarantees for different recovery scenarios. The first one is a recovery time guarantee with considering a situation that a function can be successfully recovered by an available node with its backup resource. The recovery time for each function is restricted not to exceed the recovery time guarantee. The second one is an unsuccessful recovery probability guarantee, where the unsuccessful recovery probability is restricted not to be greater than a given survivability parameter.

Assuming that each node independently fails with a corresponding probability. The recovery time guarantee considers the expected recovery time of unavailable function j if the function can be successfully recovered by an available node with backup resource; the expected recovery time can be expressed by:

$$\tau_j = \sum_{m \in M_j} \sum_{i \in N} \sum_{k \in N_j} \frac{t^{\mathrm{B}}_{jm}}{r^{\mathrm{R}}_{jk}} x^{km}_{ij} q_i \prod_{O_{jk'} > O_{jk}} q_{k'}(1 - q_k), \forall j \in F, \tag{7.3}$$

where multiple concurrent nodes failures are considered.

Let $\mathcal{P}_\Gamma$ be a collection of of all feasible failure configurations, each with at most Γ failed nodes in N. Let $\mathcal{P}^\sigma_\Gamma, \sigma \in \mathbb{C} := [1, |\mathcal{P}_\Gamma|]$, represent the σth

element in $\mathcal{P}_\Gamma$; it is a set of failed nodes, which is called a failure configuration.

Each P_Γ^σ occurs with a specific probability w_σ with considering a workload-dependent failure probability of each node, where w_σ is a weight of the unavailability for the failure configuration. w_σ can be described by:

$$w_\sigma = \prod_{i \in \mathcal{P}_\Gamma^\sigma} q_i \prod_{i \in N \backslash \mathcal{P}_\Gamma^\sigma} (1 - q_i), \forall \sigma \in \mathbb{C}, \tag{7.4}$$

The unsuccessful recovery probability corresponds to two cases. In one case, the nodes hosting all of the primary and backup resources of a function fail; in the other case, the total workload of any node during recovery (with utilizing the extra-assigned recovery workload) exceeds the capacity under a failure configuration.

Let $h_j^\sigma, j \in F, \sigma \in \mathbb{C}$, be a binary; it is 1 if all the nodes that host the resources of function j fail concurrently (in other words, all the nodes are in $\mathcal{P}_\Gamma^\sigma$), and otherwise 0. More precisely, In a mathematical formula, h_j^σ is 1 if

$$\sum_{i \in N} (\chi_{ij} + \xi_{ji}) = \sum_{i \in \mathcal{P}_\Gamma^\sigma} (\chi_{ij} + \xi_{ji}), j \in F, \sigma \in \mathbb{C}. \tag{7.5}$$

Binary variable $\vee_{j \in F} h_j^\sigma$ is 1 if at least one function under failure configuration $\mathcal{P}_\Gamma^\sigma$ cannot be recovered by any available backup resource of it, and otherwise 0.

When the primary workload for a function is not dedicatedly reserved, unsuccessful recovery may occur under some failure configurations, each of which occurs with a probability. In other words, since the non-activated workload can be utilized by other functions protected by the node for higher-speed recovery, all the unavailable functions may not be recovered simultaneously with the assigned recovery workload.

In other words, α_{jk}^σ is 1 if all k' that satisfy $O_{jk'} > O_{jk}$ is in P_Γ^σ. Let δ_{jk}^σ represent the probability of $\alpha_{jk}^\sigma = 1$, which can be expressed by:

$$\delta_{jk}^\sigma = \alpha_{jk}^\sigma \sum_{i \in N} \chi_{ij} q_i \prod_{k' \in P_\Gamma^\sigma : O_{jk'} > O_{jk}} q_{k'} (1 - q_k), \forall j \in F, k \in N \backslash P_\Gamma^\sigma, \sigma \in \mathbb{C}. \tag{7.6}$$

This work introduces $N_{jk\sigma}^*$ which denotes a set containing the unavailable nodes hosting the backup and primary resources of function j when it is recovered by node k under failure configuration P_Γ^σ, i.e., $\alpha_{jk}^\sigma = 1$. More specifically,

$N^*_{jk\sigma}$ includes node i that hosts the primary resource of function j with $\chi_{ij} = 1$; it also includes the unavailable nodes k' that have higher recovery priority $O_{jk'}$ than O_{jk}. Let $e^j_{kk'} = 1$ if $O_{jk'} > O_{jk}$; 0 otherwise. In a mathematical formulation, $N^\star_{jk\sigma}$ can be expressed by: $N^*_{jk\sigma} = \{k' \in N_j \backslash \{k\} | e^j_{kk'} = 1 \text{ and } \alpha^\sigma_{jk} = 1\} \cup \{i \in N \backslash \{k\} | \chi_{ij} = 1\}, j \in F, k \in N \backslash P^\sigma_\Gamma, \sigma \in \mathbb{C}$.

Multiple functions can be recovered a node simultaneously. Let F_c denote a set containing all combinations of functions. Let $\mathcal{F}^\omega_\mathrm{c}, \omega \in \mathbb{F} \coloneqq [1, |F_\mathrm{c}|]$, denote the ωth element in F_c. Node k recovers the functions in $\mathcal{F}^\omega_\mathrm{c}$ under $\mathcal{P}^\sigma_\Gamma$ with a certain probability, $\zeta^\sigma_{\omega k}$, which is expressed by:

$$\zeta^\sigma_{\omega k} = \prod_{k' \in \bigcup\limits_{j \in \mathcal{F}^\omega_\mathrm{c}} N^*_{jk\sigma}} q_{k'}, \forall \omega \in \mathbb{F}, k \in N \backslash P^\sigma_\Gamma, \sigma \in \mathbb{C}, \tag{7.7}$$

where $\bigcup\limits_{j \in \mathcal{F}^\omega_\mathrm{c}} N^*_{jk\sigma}$ represents a node set that contains all possible unavailable nodes among functions $j \in \mathcal{F}^\omega_\mathrm{c}$; $\cup$ expresses the "or" operation among each set $N^*_{jk\sigma}$.

Let $L^\mathrm{R}_{i\omega\sigma}, i \in N \backslash P^\sigma_\Gamma, \omega \in \mathbb{F}, \sigma \in \mathbb{C}$, represent the workload of node i during recovering unavailable functions $j \in \mathcal{F}^\omega_\mathrm{c}$ under failure configuration $\mathcal{P}^\sigma_\Gamma$, which can be described by:

$$\begin{aligned} L^\mathrm{R}_{i\omega\sigma} = &\sum_{j \in F} w^\mathrm{P}_j \chi_{ij} + \sum_{j \in \mathcal{F}^\omega_\mathrm{c}} \Big\{ \alpha^\sigma_{ji} w^\mathrm{R}_{ji} \xi_{ji} + \\ &\sum_{m \in M_j} (1 - \alpha^\sigma_{ji}) w^\mathrm{B}_{jm} \zeta^m_{ji} \Big\}, \forall i \in N, \omega \in \mathbb{F}, \sigma \in \mathbb{C}, \end{aligned} \tag{7.8}$$

where the first term expresses the total primary workload of functions hosted by node i, the former part of second term expresses the total recovery workload of functions which are recovered by node i, and the latter of second term expresses the total backup workload of functions hosted by node i.

Let $g^\sigma_{i\omega}$ represent a binary variable, which is set to 1 if $L^\mathrm{R}_{i\omega\sigma}$ exceeds capacity C_i when functions in $\mathcal{F}^\omega_\mathrm{c}$ is being recovered by node i under failure configuration $\mathcal{P}^\sigma_\Gamma$, and 0 otherwise. In a mathematical formulation, it can be expressed by:

$$\begin{aligned} &\textbf{If} \quad L^\mathrm{R}_{i\omega\sigma} \le C_i \quad \textbf{then} \\ &\qquad g^\sigma_{i\omega} = 0 \\ &\textbf{Else} \\ &\qquad g^\sigma_{i\omega} = 1. \end{aligned} \tag{7.9}$$

The total probability for unsuccessful recovery among all failure configurations is described by:

$$\mathcal{U} = \sum_{\sigma \in \mathbb{C}} \left(\sum_{\omega \in \mathbb{F}} \sum_{i \in N \setminus P_\Gamma^\sigma} \zeta_{\omega i}^\sigma g_{i\omega}^\sigma + w_\sigma \left(\vee_{j \in F} h_j^\sigma \right) \right). \tag{7.10}$$

This work formulates the function deployment problem as the following optimization problem:

$$\min \sum_{i \in N} \Delta_i \tag{7.11a}$$

$$\text{s.t.} \Delta_i = \vee_{j \in F} (\chi_{ij} \vee \xi_{ji}) \tag{7.11b}$$

$$\sum_{i \in N} \chi_{ij} = 1, \forall j \in F, \tag{7.11c}$$

$$\sum_{m \in M_j} \sum_{i \in N \setminus \{k\}} x_{ij}^{km} \leq 1, \forall j \in F, k \in N, \tag{7.11d}$$

$$t_j \leq T, \forall j \in F, \tag{7.11e}$$

$$\mathcal{U} \leq \epsilon, \tag{7.11f}$$

$$L_i^{\mathrm{W}} \leq C_i, \forall i \in N, \tag{7.11g}$$

$$w_{ji}^{\mathrm{R}} \leq C_i, \forall i \in N, j \in F, \tag{7.11h}$$

$$(7.1) - (7.10), \tag{7.11i}$$

$$x_{ij}^{km}, \chi_{ij}, \xi_{ji}, \zeta_{ij}^m \in \{0,1\}, \forall i \in N, k \in N \setminus \{i\}, j \in F, m \in M_j, \tag{7.11j}$$

$$h_j^\sigma \in \{0,1\}, \forall k \in N, j \in F, \sigma \in \mathbb{C}, \tag{7.11k}$$

$$\alpha_{jk}^\sigma \in \{0,1\}, \forall k \in N \setminus P_\Gamma^\sigma, j \in F, \sigma \in \mathbb{C}, \tag{7.11l}$$

$$e_{kk'}^j \in \{0,1\}, \forall j \in F, k \in N_j, k' \in N_j \setminus \{k\}, \tag{7.11m}$$

$$g_{i\omega}^\sigma \in \{0,1\}, \forall i \in N \setminus P_\Gamma^\sigma, \omega \in \mathbb{F}, \sigma \in \mathbb{C}. \tag{7.11n}$$

Equation (7.11a) minimizes the active number of nodes hosting primary and backup resources with different modes of function deployment. Equation (7.11c) ensures that the primary resource of each function in F is hosted by only one node in N. Equation (7.11d) ensures that each function can be protected by a node with at most one backup mode. Equation (7.11e) imposes that the expected recovery time of each function does not exceed a given time guarantee, T. Equation (7.11f) imposes that the total unsuccessful recovery probability

in all failure configurations does not exceed a given survivability parameter, ϵ. Equations (7.11g) and (7.11h) impose that the workload on each node does not exceed its maximum computing capacity C_i.

Appendix H and the work in [114] describe a part of linearization process of the model; the problem can be formulated as an MILP problem.

7.3 Heuristic algorithm

The computational time for solving the MILP problem is time-consuming, to solve a larger-size problem in a practical time, a heuristic algorithm for obtaining an approximate solution of the considered problem is presented in this section.

Algorithm 7.1 Simulated annealing

Input: $F, N, T_i^n, P_n, C_i, w_j^{\mathrm{P}}, w_{jm}^{\mathrm{B}}, T_{\min}, T_{\mathrm{init}},$

Output: $x_{ij}^{km}, w_{jk}^{\mathrm{R}}$

1: Sort N by C_i decreasingly.
2: Iteratively allocate the primary resource of function j to each node until the workload exceeds its capacity. Record $N_j'' = N \backslash \{\text{the allocated node}\}$.
3: **for** $j = 1 \rightarrow |F|$ **do**
4: Sort N_j'' based on the number of hosting functions on each node decreasingly.
5: Allocate a backup resource for function j with a random backup mode and random assigned recovery workload to the first node in N_j'' until the workload exceeds its capacity, where the function is allocated to the next node in sorted N_j'' if the workload exceeds. Update N_j''.
6: Record the number of activated nodes as n.
7: **end for**
8: **for** $n \rightarrow |N|$ **do**
9: $T \leftarrow T_{\mathrm{init}}$
10: Add a new backup resource of each function in F on the newly activated node regardless of the capacity constraint.
11: **while** $T \geq T_{\min}$ **do**
12: Randomly re-allocate function j from a node in $N \backslash N_j''$ to an activated node in N_j''. Update N_j''.

13: Randomly adjust the backup mode of a randomly chosen function.
14: Sort nodes with the number of functions hosted by it increasingly. Randomly release a backup instance of a function hosted by the first node in the sorted set if the function has multiple backup resources; increase the assigned recovery workload of the remaining backup resource of the function. Update N_j''.
15: Randomly assign a new backup instance of a function to a node in N_j''. The function is preferentially assigned to an activated node; a new node is activated with a probability ρ. The new backup resource is assigned with a random backup mode and random recovery workload. Update N_j''.
16: **while** L_i^{w} exceeds the capacity of node i **do**
17: Randomly swap the primary and backup resource allocation of a function hosted by node i.
18: **end while**
19: Calculate the expected recovery time τ_j and unavailable probability $\mathcal{U}$ using updated solution $\mathbf{x}'$.
20: **if** $\tau_j \leq t$ and $\mathcal{U} \leq \epsilon$ **then**
21: **Return** Δ, $\mathbf{x}$,τ_j, and $\mathcal{U}$
22: **else**
23: $\mathbf{x} \leftarrow \mathbf{x}'$ with probability of $\min(1, q)$, where $q = e^{-\frac{\tau_j' - \tau_j}{T}}$.
24: $T = T \cdot$ a given decreasing rate
25: **end if**
26: **end while**
27: **end for**

Algorithm 7.2 Calculation of reliability indicators

Input: $x_{ij}^{km}, w_{jk}^{\mathrm{R}}, N_j''$
Output: $\tau_j, \mathcal{U}$

1: Calculate the failure probability of each node based on their workload.
2: Calculate the recovery rate r_{jk}^{R} of each node for each function based on their assigned recovery workload w_{jk}^{R}; Calculate priority O_{jk}.
3: Calculate the expected recovery time τ_j based on (7.3) with considering the assigned backup mode.
4: Set $\alpha_{jk}^{\sigma} = 1$ if all k' that satisfy $e_{kk'}^{j} = 1$ is in P_{Γ}^{σ}.

5: If $\prod_{j \in \mathcal{F}_c^\omega} \alpha_{jk}^\sigma = 1$, calculate the workload of node i after failure configuration σ with (7.8) when recovering functions in $\mathcal{F}_c^\omega$. Determine $g_{i\omega}^\sigma$.
6: Calculate δ_{jk}^σ based on (7.6).
7: Calculate the unsuccessful recovery probability $\mathcal{U}$ based on (7.10).

This work uses simulated annealing (SA) [94] to minimize the number of nodes activated for function deployment while satisfying the reliability guarantees. This work first gives an initial function deployment in a greedy manner (lines 1-6) and record the number of activated nodes as n. This work considers nodes in set N_j'', which is N excluding the allocated node, for function j (lines 2-6). Function j is deployed to a node in N_j'' that hosts the most functions to activate as few as possible nodes are assigned for function deployment. When the workload of the first node exceeds the capacity, the function is allocated to the next node that has enough capacity to satisfy the capacity constraints.

Since the considered problem aims to minimize the number of activated nodes while satisfying the reliability guarantees, Algorithm 7.1 minimizes the expected recovery time and unsuccessful recovery probability when a given number of nodes are admitted to be used (lines 8-26), where the reliability indicators are calculated by Algorithm 7.2. Fixing the number of activated nodes, the resource allocation, including the function deployment with backup modes and assigned recovery workload, changes randomly to generate a new solution (lines 9-17) and calculates new reliability indicators with Algorithm 7.2 in each iteration of the temperature decreasing. When both calculated reliability indicators satisfy the reliability guarantees, Algorithm 7.1 is terminated with accepting and returning the corresponding solution (lines 18-21). When any of the reliability indicators does not satisfy the reliability guarantees until the temperature decrease to the set minimum temperature $T_{\min}$, one more node is permitted to be activated (line 8); a new round of the iteration of decreasing the temperature from T_{init} (repeating lines 9-25) with randomly changing the deployment from the initial allocation obtained in lines 2-6. SA accepts a worse solution with larger reliability indicators than the existing solution with probability of $\min\{1, e^{-\frac{\tau_j' - \tau_j}{T}}\}$ (line 22).

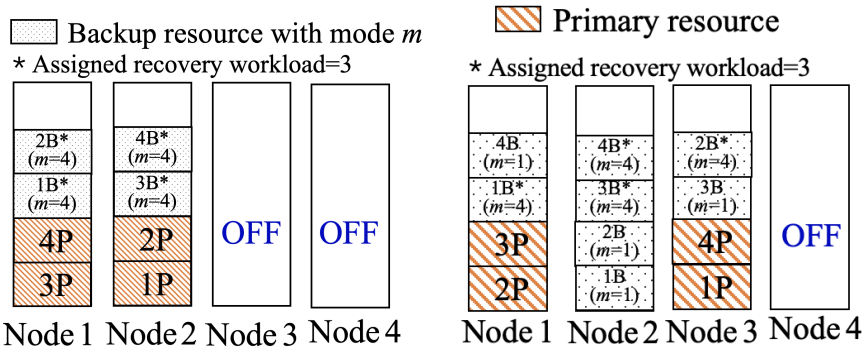
7.4 Numerical evaluations

This work first introduces demonstrations to show the basic characteristics of the proposed model. Further, this work describes the experiment settings and the baseline models; this work compares the proposed model with three baselines.

7.4.1 Demonstration

This work introduces two demonstrations with four nodes and four functions, where the node capacities are 10 and the primary workloads of functions are set to 2. This work considers a two-step function with setting $P_1 = 0.0025$ and $P_2 = 0.01$ and set each threshold to 50% of each node's capacity. Four backup modes with $m = 1, 2, 3$, and 4 are considered for each function; the backup workload is set to 0.5, 1, 1.5, and 2 for each mode, respectively; the corresponding recovery time is set to 2, 1.5, 1, and 0.5, respectively. The recovery workload, w^{R}_{jk}, can be assigned for faster recovery to a range over [2,10], where 2 is the primary workload of function and 10 is the node capacity. For a model linearization purpose, this range is divided into four sub-ranges, which are [2,3), [3,4), [4,5), and [5,10] similar to the horizon axis of Fig. 7.4(a); each sub-range has its reciprocal of the recovery rate, $1/r^{\mathrm{R}}_{jk}$, that is set to 1, 0.9, 0.8, and 0.7, respectively, similar to the vertical axis of Fig. 7.4(a). The maximum time guarantee is set to $T = 0.005$ [s] for demonstrations. This work sets $\epsilon = 2 \times 10^{-4}$ and $\epsilon = 10^{-6}$ for demonstrations 1 and 2, respectively, as reliability guarantees.

Figure 7.5 shows the primary ("P") and backup ("B") resource allocation with backup mode m of function deployment. As shown in demonstration 1 of Fig. 7.5(a), each function is protected by two nodes to fit the reliability guarantee $\epsilon = 10^{-6}$, where totally three nodes are activated. Except for the situation that nodes 1, 2, and 3 fail simultaneously, the workload during recovery of any failure configuration, $L^{\mathrm{R}}_{i\omega\sigma}$, is smaller than each capacity so that unsuccessful recovery does not occur. For each function, the recovery workload of a node with a higher priority to recovery the function is assigned to 3; its corresponding reciprocal of the recovery rate is 0.9. The maximum recovery time among the four functions is 0.004851 [s] to fit the maximum recovery time



(a) Demostration 1: $T = 0.005$, $\epsilon = 2 \times 10^{-4}$.

(b) Demostration 2: $T = 0.005$, $\epsilon = 10^{-6}$.

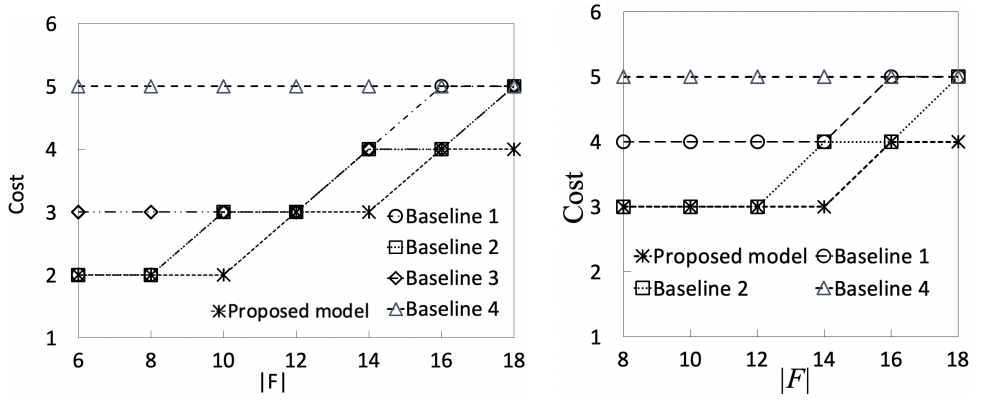
Figure 7.5: Demonstration of function deployment (recovery workload of each node protects each function without star mark is 2).

guarantee.

When the reliability guarantee is relaxed with maintaining the same maximum recovery time guarantee, each function is protected by one node, as shown in demonstration 2 of Fig. 7.5(b). The recovery workload, w^{R}_{jk}, of each function by its backup node is assigned to 3 and the corresponding reciprocal of the recovery rate, $1/r^{\mathrm{R}}_{jk}$, is 0.9. The maximum recovery time among the four functions is 0.004455 [s]. The probability that the nodes hosting the functions fail simultaneously is 10^{-4}. Similar to demonstration 1, the extra-assigned recovery workload speeds up the recovery while avoiding the unsuccessful recovery probability under some failure configurations to fit the reliability guarantee. The computation time for this size problem in the demonstration is at most 10 [s].

7.4.2 Baselines and experiment settings

Baseline 1 considers a situation that each function has only one backup mode and its recovery time corresponding to the selected backup mode is given and fixed. Baseline 2 considers a fixed assigned recovery workload which is equal to the primary workload for each function. Baseline 3 does not consider the workload-dependent failure probability; the failure probability is set to a constant value as $\max_{s \in \mathcal{S}} P_s$ since the failure probabilities with different



(a) Experiment 1: $T = 0.008$ and $\epsilon = 10^{-4}$. (b) Experiment 2: $T = 0.005$ and $\epsilon = 10^{-6}$.

Figure 7.6: Comparison with baselines.

workloads cannot be judged. The objective value of baselines 1, 2, and 3 is to minimize the number of activated nodes with satisfying the reliability guarantees. Baseline 4 considers the same model setting with minimizing a weighted value of maximum recovery time and reliability among functions without considering the deployment cost.

This work considers a smaller-size problem with five nodes where the node capacities are randomly distributed over [15, 25]. The ratios of $\frac{T_i}{C_i}$ for each node i are randomly distributed over [0.4, 0.8]. The primary workload of functions is randomly distributed over the range of [1, 4]. Each function can have 2, 3, or 4 backup modes with different pre-configuration states, i.e., $|M_j| = 2, 3$, or 4. Each backup mode has different backup workloads for pre-configuration. The backup workload of function j for backup mode m is randomly distributed over the range of $(\frac{(m-1)w_j^{\mathrm{P}}}{|M_j|}, \frac{mw_j^{\mathrm{P}}}{|M_j|}]$, where $m \in [1, |M_j|]$. The recovery workload and recovery rate settings are the same as the demonstrations in Section 7.4.1. This work sets $P_1 = 0.0025$ and $P_2 = 0.0175$ as the same settings with Chapter 4.

7.4.3 Comparison with baselines on cost-efficiency

This work compares the proposed model with the baselines on cost by conducting two experiments with setting $T = 0.008$, $\epsilon = 10^{-4}$ and $T = 0.005$, $\epsilon = 10^{-6}$ for experiments 1 and 2, respectively. In experiment 1, the reliability

guarantees are relatively loose; each function can be protected by at least one node while satisfying the guarantees. As shown in Fig. 7.6(a), the number of activated nodes in baselines 1, 2, and 3 is on average 14%, 11%, and 21%, respectively, larger than that of the proposed model, over the different values of $|F|$. The number of activated nodes is 43% reduced by considering the proposed model compared with baseline 4 in the examined case. All the five nodes are activated in baseline 4 to distribute the workloads and provide protection against failures as much as possible. In experiment 2, the reliability guarantees restrict that each function must be protected by more than one node to satisfy that the total unsuccessful recovery probability does not exceed $\epsilon = 10^{-6}$. As shown in Fig. 7.6(b), the number of activated nodes in baselines 1 and 2 is on average 23% and 8%, respectively, larger than that of the proposed model; any feasible solution cannot be found in baseline 3.

Then this work discusses the reasons for the differences between each baseline and the proposed model.

Baseline 1 considers that each function is protected with one fixed backup mode with a fixed workload; it lacks flexibility for adapting the resource allocation and backup modes to reduce the maximum recovery time. In comparison between the proposed model and baseline 1, when the total workload is so small that the reliable time in the proposed model can be easily suppressed to be smaller than T with the admitted activated nodes, baseline 1 shows similar performance with the proposed model. As $|F|$ increases, each function is protected by more than one node with the same backup mode in baseline 1; the baseline cannot suppress node workloads with a similar approach demonstration 1 in Section V.A. It leads to the increase of maximum recovery time and further leads to the increase of the number of activated nodes. When the reliability is relatively strict in experiment 2, baseline 1 with considering only one backup mode shows worse performance than a loose reliability guarantee in experiment 1.

Baseline 2 cannot find any feasible solution with activating nodes with the same number of the proposed model without an extra-assigned recovery workload in experiment 1; the baseline needs to activate one more node than the proposed model to distribute the workload and reduce the workload-dependent failure probability to satisfy the recovery time guarantee with some different

values of $|F|$. On the other aspect, the extra-assigned recovery workload may lead to the unsuccessful recovery under some failure configurations. Thus, when the reliability guarantee is relatively loose in experiment 1, which has higher tolerance on the unsuccessful recovery than experiment 2, more recovery workload can be assigned at the cost of complying with unsuccessful recovery probability.

In baseline 3, each function needs to be protected with a backup mode with the shortest recovery time and highest backup workload to satisfy the recovery time guarantee in baseline 3, since it considers a fixed failure probability. Further, the recovery workload must be extra-assigned to speed up the recovery, which may lead to the unsuccessful recovery under some failure configurations. In experiment 1, one more node is required to be activated to reduce the unsuccessful recovery probability compared with the proposed model with some different values of $|F|$. Baseline 3 cannot find any feasible solution that satisfies the reliability guarantees in experiment 2.

7.4.4 Competitive performance of heuristic algorithm for larger-size problems

Table 7.1 compares the performance of the developed heuristic algorithm with the MILP approach, where Obj. represents the number of activated nodes and the time indicates the computation time. This work considers a larger size problem with eight nodes with $T = 0.005$ and $\epsilon = 10^{-6}$. This work can observe that there is a gap between the computation times of the MILP approach and the heuristic algorithm; the computation time gap increases as the problem size increases. Considering 10^5 [s] as the allowable maximum computation time in the tests, the deployment cost obtained with the algorithm is 1.9% larger than that of the MILP approach. The heuristic algorithm considers to activate a fixed number of nodes and randomly adjust the resource allocation to satisfy the reliability indicators. The case with T_{init}=100 and ρ=0.1 is to give some tolerance on a relatively high cost to get a solution in a shorter computation time. The case with T_{init}=1000 and ρ=0.05 is to get a better solution as much as possible in a tolerable time. Without the initial allocation with the greedy algorithm described in lines 1-6 in Algorithm 7.1, the heuristic

Table 7.1: Number of activated nodes and computation times of MILP approach and heuristics algorithm with different parameters.

$\|F\|$	MILP		Heuristic			
			T_{init}=100, ρ=0.1		T_{init}=1000, ρ=0.05	
	Obj.	Comp. time [s]	Obj.	Comp. time [s]	Obj.	Comp. time [s]
14	3	345.6	4	58.2	3	89.4
16	3	852.4	4	79.5	4	143.2
18	3	1169.3	4	93.7	4	169.2
20	4	3932.5	5	183.3	4	174.3
22	5	69545.5	5	212.5	5	402.3
24	7†	10^5	6	391.7	5	398.7
26	8†	10^5	6	417.6	6	435.1

†: Feasible solution obtained within 10^5 [s].

algorithm cannot find any solution with a random given initial allocation to reduce the number of activated nodes from eight (the maximum number of nodes that can be activated) with satisfying the constraints.

7.5 Summary

This chapter proposed a primary and backup resource allocation model under reliability guarantees to minimize the number of activated nodes. The resource allocation was considered with two kinds of workload distribution, the backup workload and the recovery workload. Different backup modes with different backup workloads corresponding to varying degrees of pre-configuration and recovery times. The extra-assigned recovery workload can be adopted to speed up the recovery while improving the resource efficiency; it may lead to unsuccessful recovery in a specific failure configuration. The proposed model minimizes the number of activated nodes as deployment cost while restricting

the recovery time and the total unsuccessful recovery probability not to exceed each guarantee. The proposed model was compared with two baselines, each of which does not consider flexible backup modes and extra-assigned recovery workload, respectively, in different reliability guarantees. The numerical results revealed that the proposed model saves the deployment cost on average 14% and 11% in the experiments compared to the two baselines, respectively. This chapter analyzed the reasons for the superiority of the proposed model compared with baselines.

Chapter 8

Implementations

This chapter provides two demonstrations on how to implement the resource allocation models in real network system. VNFs can be deployed by NFVO platform, e.g., Kubernetes, which is an open-source system to deploy and manage functions automatically [62, 132]. This work considers customized resources to realize reliable and prompt function deployment.

Section 8.1 designs and implements a two-layer controller structure in Kubernetes to achieve the function deployment in a limited computation time with considering resource migration for allocation optimality. Section 8.2 focuses on the backup resources design with implementing a custom resource and the corresponding controller in Kubernetes to manage the primary and different types of backup resources of network functions. Section 8.3 summarizes this chapter.

8.1 Implementation of real-time function deployment with resource migration in Kubernetes

This section designs and implements controllers to deploy network functions in a real-time and reliable manner. This section introduces two new resource types called *migratable Pod set* (MPS) and *global optimizer* (GO), each of which is *custom resource* in Kubernetes [38]. To achieve the function deployment in a

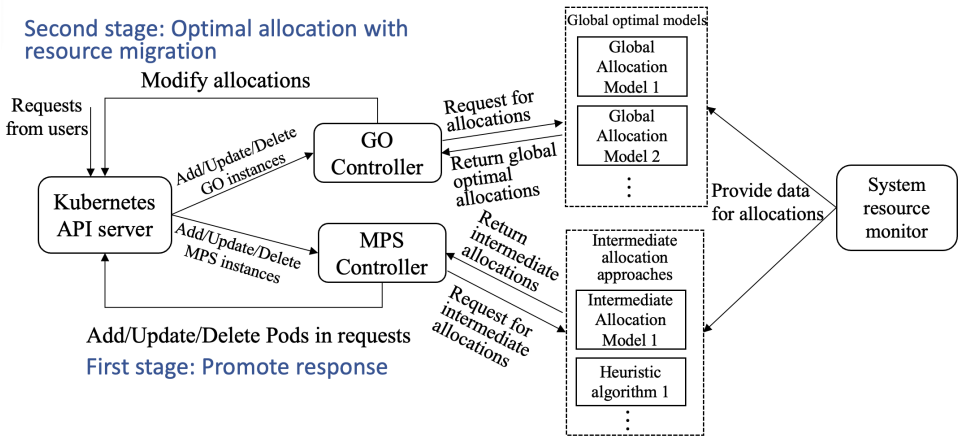


Figure 8.1: Overall structure of the system in 8.1.

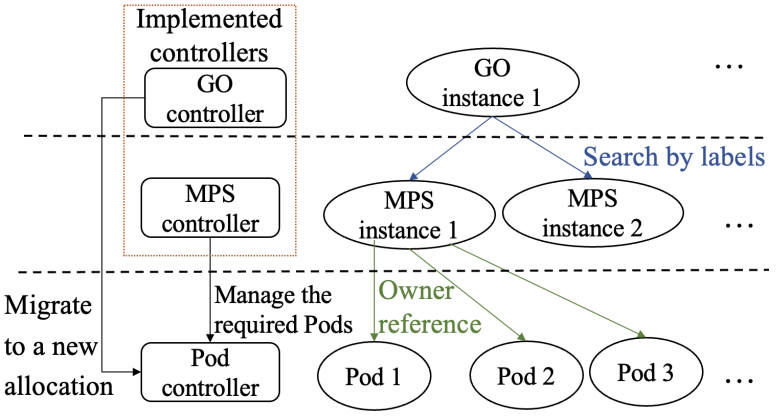


Figure 8.2: Controller structure of the system in 8.1.

limited computation time, this section introduces two controllers in Kubernetes for a two-stage function deployment. The MPS controller manages the Pods for an intermediate allocation with a model or a heuristic algorithm to respond to the requests promptly. The GO controller manages the MPS instances by optimizing Pod allocations with considering resource migration of network functions to maintain the current state of the Pods to keep the current state consistent with the desired state.

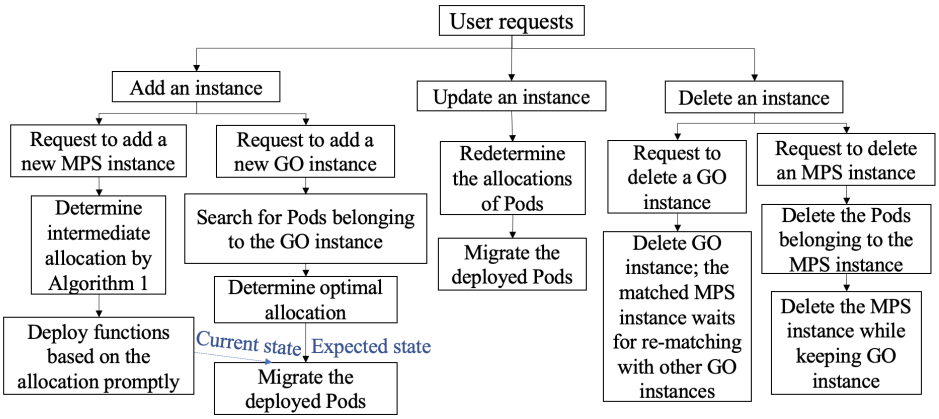


Figure 8.3: Workflow of controllers of the system in 8.1.

8.1.1 Design and Implementation

Overall structure

Figure 8.1 overviews the structure of the reported controllers with considering arrival and releasing requests, the MPS and GO controllers receive the notification from Kubernetes application programming interface (API) server and handle the events caused by MPS and GO instance operations, respectively, corresponding to the received requests. Each controller requests the solution of function deployment corresponding to each model. With the allocation results, each controller adjusts the current state of each Pod to its expected state through the control loop [133]. More specifically, the controller is responsible for the first stage allocation with an intermediate model or heuristic algorithm for a prompt response to user requests. The GO controller is responsible for the second stage allocation with optimization models with different objectives. The GO controller modifies Pod allocations with resource migration. Each controller requests the allocation of different models while the resource monitor provides the system data for model calculation.

For an arrival user request, which includes the requested functions and the intermediate and global optimal requirements, the implemented system transforms the user request into two aspects. First, an MPS instance is created and labeled. Second, a GO instance is created if the requested model has not been requested before. If the requested GO instance exists, the newly created

MPS instance is automatically detected and matched with the GO instance, while the optimal allocations are calculated with a corresponding model.

Controllers design

In a dynamic scenario with arrival and releasing requests, a one-controller structure is not sufficient to handle multiple requests, each of which may require different intermediate allocation approaches to achieve different requests. In addition, requests arrive continuously in sequence instead of concurrently so that each request needs to be managed independently, where a request is the smallest operable unit in the implemented system by a service provider. The requested GO and MPS instances are designed to be flexibly combined and replaced at will. Based on this principle, this work designs a two-layer controller structure in the implemented system, which is designed for managing MPS instances in the lower layer and managing GO instances in the upper layer, as shown in Fig. 8.2. If a new request of adding functions arrives, an MPS instance is created to manage the corresponding Pods by setting the ownership between the MPS instance and the corresponding Pods to deploy containerized VNFs. The allocations of these Pods are determined by an intermediate allocation approach appointed by the MPS instance. A GO instance represents a global optimizer for the Pods owned by the appointed MPS instances, which are identified by labels defined in the GO instance. When a GO instance detects a new MPS instance filtered by the label, optimal allocations of Pods are calculated and the Pods are migrated to the new allocations. If an MPS instance is deleted, its owned Pods are correspondingly deleted with the MPS instance. If a GO instance is deleted, the MPS instances are not affected. This enables flexible matches of global optimizers and the MPS instances.

Workflow

When user requests of creating/updating/deleting an instance for a containerized function arrive, the request is automatically translated to the operation for MPS and GO instances. If a new function is required to be deployed according to the arrival request, the MPS controller promptly determines the intermediate allocation with a model or an algorithm to respond to the re-

quest. Simultaneously, the GO controller determines the optimal allocation to obtain the expected state of the function deployment, which commonly requires a longer time than the determination of MPS controller. The GO controller manages the allocation of Pods to let the current state of them, i.e., intermediate allocation, keep pace with the expected state configured by optimal allocation. The management of resource migration is achieved by the control loop [133]; it firstly instantiates and initializes the Pods at the optimal allocation configured by its corresponding GO instance and secondly terminates the running Pods owned by the MPS instance to keep continuous service for the request. With updating an instance according to a request, the corresponding GO instance determines the optimal allocation and migrates the deployed Pods while keeping the service continuity. When a request is released, the corresponding MPS instance is deleted while its owned Pods are deleted automatically while keeping the GO instance. The workflow is shown in Fig. 8.3.

8.1.2 Demonstration

This work implements the controllers by Operator SDK v1.4.2 [134], Golang 1.16 [135], and Kubernetes v1.20.4 [108] running on a 5-node cluster including one master and four worker nodes. This work deploys the designed controller as a *deployment* in Kubernetes with relative resources, e.g., namespace, CRD, and permissions for modifying the instances and Pods.

This work implements the controllers with the following algorithm and optimization model. This work considers a workload-aware greedy algorithm to balance the utilization ratio among nodes to determine the intermediate allocation for the MPS controller.

Algorithm 8.1 Workload-aware greedy algorithm (WAGA)

Input: Requested functions and their requested load, cluster information
Output: Intermediate allocation

1: **for** each function waited to be allocated **do**
2: Sort nodes by their workload increasingly as a set N.
3: Allocate the function to the first node in N and update its workload.
4: **end for**

```
NAME                    READY   STATUS              IP        NODE
r1-request1-testfunc1   0/1     ContainerCreating   <none>    worker1
r1-request1-testfunc2   0/1     ContainerCreating   <none>    worker2
r1-request1-testfunc3   0/1     ContainerCreating   <none>    worker3
r1-request1-testfunc4   0/1     ContainerCreating   <none>    worker4
r1-request1-testfunc5   0/1     ContainerCreating   <none>    worker1
```

(a) Pods are created and located as intermediate deplyment.

```
NAME                    READY   STATUS    IP             NODE
r1-request1-testfunc1   1/1     Running   10.244.1.61    worker1
r1-request1-testfunc2   1/1     Running   10.244.1.62    worker1
r1-request1-testfunc3   1/1     Running   10.244.3.34    worker3
r1-request1-testfunc4   1/1     Running   10.244.3.35    worker3
r1-request1-testfunc5   1/1     Running   10.244.1.60    worker1
```

(b) Pods are migrated and located as global deplyment.

Figure 8.4: Locations of Pods created by first user.

The GO controller determines the optimal allocation with a mathematical model. The work in [8] introduced a migration policy triggered by an excess CPU utilization compared with a certain threshold of computing resources. The work in [33] introduced workload-dependent failure probability, which can be modelized as a two-step function. This work considers that each node has a certain threshold for resource utilization and a corresponding workload-dependent failure probability; the threshold is different based on each characteristic. The objective of the optimization model is to minimize the maximum failure probability among nodes as the primary objective, the number of active nodes as the secondary one, and the migration cost as the third one.

This work validates the implementation by using the following scenario. Two users request five and three functions, respectively, in sequence. All requests choose the introduced intermediate algorithm and global optimization model. After each request from each user arrives, two YAML configuration files are translated for creating an MPS instance and a GO instance, respectively. For the first request, five Pods are created and deployed as determined by the introduced intermediate algorithm, as shown in Fig. 8.4(a). The intermediate deployment time is 1.1 [s], including the calculation and creation time. Simultaneously, the GO controller determines a global optimal allocation based on the introduced model. After the calculation, the Pods are migrated to optimal allocation, as shown in Fig. 8.4(b). The time for the optimal deployment with

```
NAME                    READY   STATUS              IP             NODE
r1-request1-testfunc1   1/1     Running             10.244.1.61    worker1
r1-request1-testfunc2   1/1     Running             10.244.1.62    worker1
r1-request1-testfunc3   1/1     Running             10.244.3.34    worker3
r1-request1-testfunc4   1/1     Running             10.244.3.35    worker3
r1-request1-testfunc5   1/1     Running             10.244.1.60    worker1
r1-request2-testfunc1   0/1     ContainerCreating   <none>         worker2
r1-request2-testfunc2   0/1     ContainerCreating   <none>         worker4
r1-request2-testfunc3   0/1     ContainerCreating   <none>         worker2
```

(a) Pods are created and located as intermediate deplyment.

```
NAME                    READY   STATUS    IP             NODE
r1-request1-testfunc1   1/1     Running   10.244.1.61    worker1
r1-request1-testfunc2   1/1     Running   10.244.2.22    worker2
r1-request1-testfunc3   1/1     Running   10.244.3.34    worker3
r1-request1-testfunc4   1/1     Running   10.244.3.35    worker3
r1-request1-testfunc5   1/1     Running   10.244.1.60    worker1
r1-request2-testfunc1   1/1     Running   10.244.2.21    worker2
r1-request2-testfunc2   1/1     Running   10.244.1.63    worker1
r1-request2-testfunc3   1/1     Running   10.244.1.64    worker1
```

(b) Pods are migrated and located as global deplyment.

Figure 8.5: Pod deployment created by first and second users.

calculation and migration is 2.9 [s]. Fig. 8.5 shows the deployments and the migration of the functions requested by the second user. The total time for optimal deployment is 3.8 [s]. Fig. 8.6 outlines the entire process, where the green rectangles represent the newly allocated functions with an intermediate deployment to respond to the request and the orange rectangles represent the resource migration for global deployment. Secondly, this work compares the implemented system with the scheduler introduced in [35], which does not consider the two-layer controller structure. The implemented system saves approximately three times computation time compared with [35] to achieve prompt request responses.

8.2 Implementation of backup resource management controller for reliable function allocation in Kubernetes

This section designs and implements a controller to manage the primary and backup resources of network functions. This section introduces a new resource

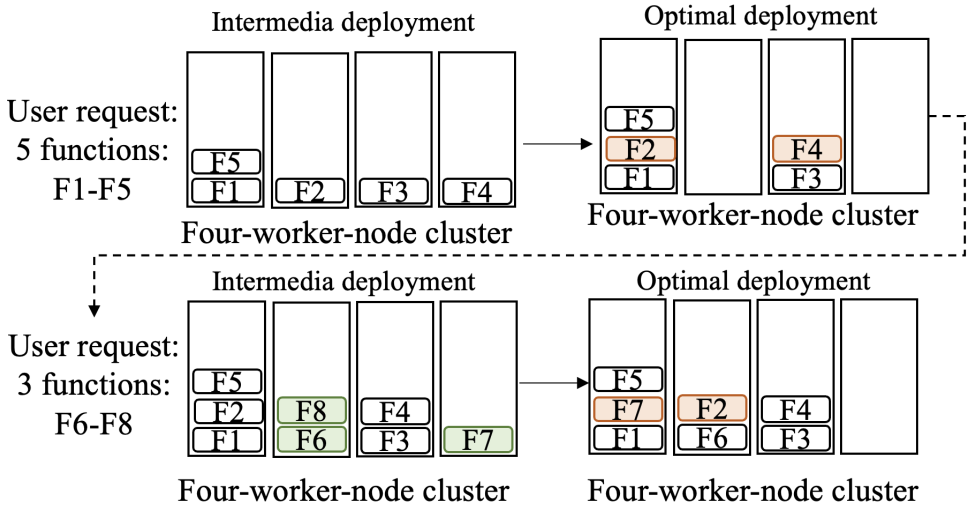


Figure 8.6: Entire process in demonstration.

type called *backup Pod set (BPS)*, which is a *custom resource* in Kubernetes [38]. *BPS* includes a certain number of different types of Pods which are the primary, HB, and CB Pods. The transitions of different types of Pods can be customized by cooperating with the allocation-model-based scheduler introduced in [35] or randomly. Demonstrations validate the effectiveness of the controller.

8.2.1 Design and Implementation

Overall structure

Figure 8.7 overviews the structure of the reported controller and the cooperation between the controller and Kubernetes application programming interface (API) server. The controller handles events triggered by BPS instance operations; BPS is defined by custom resource definitions (CRD) [38].

When a BPS instance is requested to be created, updated, or deleted, *Informer*, which is a bridge between the API server and the controller, receives the notification from the API server and pushes the events caused by BPS instance operations to a First-In, First-Out (FIFO) queue called *WorkQueue*. *Control loop* is the core of the controller, which handles the events in *WorkQueue* by maintaining the current state of a BPS instance until

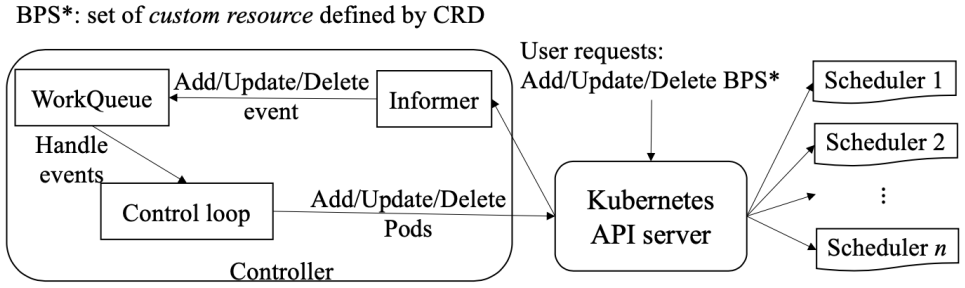


Figure 8.7: Overall structure.

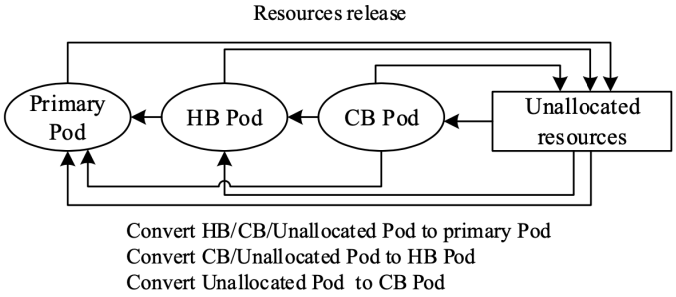


Figure 8.8: Pod state transition diagram.

it is consistent with the desired state. *Control loop* maintains the number of Pods for each type by resource releasing and converting, until the number of Pods for each type is consistent with the desired number requested by the user requirement, as shown in Fig. 8.8. The scheduler decides the allocations of the newly created Pods with the default scheduler or allocation-model-based scheduler in [35].

8.2.2 Backup Pod set (BPS)

A BPS instance is a *custom resource* defined by CRD. CRD includes the required information and specifications for creating a BPS instance with three parts: *Metadata, Spec,* and *Status*. *Metadata* provides descriptive information of a BPS instance including name and creation time. *Spec* provides the specifications and desired state of a BPS instance including the desired numbers of the different types of Pods, the strategies for state transitions, and the name of the specified scheduler. *Status* contains the current state of a BPS instance, including the current numbers and names of the Pods. *Status* is updated pe-

riodically in the control loop.

Primary, HB, and CB Pods are distinguished by the labels, each of which is attached to the Pod when a service is created. Each Pod with a *primary* label is used to balance the traffic for services. Each Pod with a *hot backup* label is activated without being exposed to the service; the Pod takes over the tasks of primary Pod once a failure is detected. When a Pod with a *cold backup* label is requested to be created, this work adds an *init container* [110] in the CB Pod. *Init containers* run before the other containers in the same Pod. Each *init container* must be completed successfully before the activation of other containers in the same Pod. The *init containers* in this demonstration is implemented as a transmission control protocol (TCP) server, which is used for waiting for the activation message from the control loop.

8.2.3 Control loop

The control loop handles the add, delete, and update events of a BPS instance, as shown in Algorithm 8.2.

When a BPS operation is requested, the desired numbers of Pods for different types are changed. The control loop focuses on the comparison between the desired number and the current number of each type of Pods. The control loop automatically maintains the current BPS state until it reaches the desired state by following the transition of different types of Pods with resource releasing and converting shown in Fig. 8.8. Note that, for a different conversion unavailable time, the resource conversion has a priority policy. Compared with the Pods with longer unavailable time, the Pods with shorter time are prioritized to be converted. For example, the conversion of the HB Pod to the primary Pod has higher priority than the conversion of the CB Pod to the primary Pod. A candidate of the same type of Pods can be chosen based on a given policy; it is chosen randomly in this algorithm.

When the current number of the primary/CB/HB Pods is larger than the desired number, each unnecessary primary/CB/HB Pod with the number of the difference between the desired number and the current number of primary/CB/HB Pods is selected by the given strategy and deleted; the occupied resources of the unnecessary Pods are released.

Algorithm 8.2 Control loop

```
1: Update Status defined in CRD
2: if the instance is newly created then
3:     Create desired numbers of primary/HB/CB Pods
4: else
5:     if Current primary Pods < desired primary Pods then
6:         Convert an HB Pod to the primary Pod until there is no more HB Pods or the primary Pods reach the desired number.
7:         Convert a CB Pod to the primary Pod until there is no more CB Pods or the primary Pods reach the desired number.
8:         Create a primary Pod until the desired number of primary Pods.
9:     else if Current primary Pods > desired primary Pods then
10:        Delete a primary Pod until the desired number of primary Pods.
11:    end if
12:    if Current primary Pods = desired primary Pods then
13:        if Current HB Pods < desired HB Pods then
14:            Convert a CB Pod to the HB Pod until there is no more CB Pods or the HB Pods reach the desired number.
15:            Create HB Pod until the desired number of HB Pods.
16:        else if Current HB Pods > desired HB Pods then
17:            Delete HB Pod until the desired number of HB Pods.
18:        end if
19:        if Current HB Pods = desired HB Pods then
20:            if Current CB Pods < desired CB Pods then
21:                Create a CB Pod until the desired number of CB Pods.
22:            else if Current CB Pods > desired CB Pods then
23:                Delete a CB Pod until the desired number of CB Pods.
24:            end if
25:        end if
26:    end if
27: end if
28: Requeue in WorkQueue
```

When the current number of primary Pods is smaller than the desired number, firstly, the current HB Pod is converted to the primary Pod and exposed to the service. When all the HB Pods are converted and the number of current primary Pods is still less than the desired number, secondly, the current CB Pod is converted to the primary Pod with being activated and exposed to the service. When all the CB Pods are converted to the primary Pods and the number of current primary Pods is still less than the desired number, the remaining primary Pods are created with using unallocated resources.

When the current number of the HB Pods is smaller than the desired number, firstly, the current CB Pod is converted to the HB Pod with being activated by the *init container*. When all the CB Pods are converted and the number of current HB Pods is still less than the desired number, the remaining HB Pods are created with using unallocated resources.

When the current number of the CB Pods is smaller than the desired number, the insufficient CB Pod is created following a given strategy. The resource of the CB Pod is reserved without activated; they wait for being activated by the *init container*.

8.2.4 Demonstrations

This work implements the controller by Operator SDK v1.4.2 [134], Golang 1.16 [135], and Kubernetes v1.20.4 [108] running on an Intel Core i7- 10510U 1.80 GHz 2-core CPU, 4 GB memory. This work deploys the designed controller as a *deployment* in Kubernetes with relative resources, e.g., namespace, CRD, and permissions for modifying the BPS instances and Pods. This work creates a BPS instance with two primary Pods, two HB Pods, and two CB Pods by a YAML file shown in Fig. 8.9. Figure 8.10 shows the created Pods in the BPS instance. The time for the controller to handle the BPS creation request is 0.211 [s]. The total creation time for the instance is 5.344 [s].

When the traffic increases, the request of updating the number of Pods for each type in a BPS instance from two to three for load balancing arrives. The controller compares the current and desired states of the different types of Pods; one Pod for each type is added. Figure 8.11(a) shows the results. The Pod index refers to the name of Pod in Fig. 8.10. One HB Pod, Pod 3, is

```
apiVersion: backup.example.com/v1alpha1
kind: BackupPodSet
metadata:
  name: bps-sample
spec:
  replicas: 2
  hotBackups: 2
  coldBackups: 2
  strategy:
    init: "random"
    unallocatedToColdbackup: "random"
    unallocatedToHotbackup: "random"
    unallocatedToReplicas: "random"
    coldbackupToHotbackup: "random"
    coldbackupToReplicas: "random"
    hotbackupToReplicas: "random"
  schedulerName: "default-scheduler"
  wakeupTimeout: 5
  template:
    metadata:
      name: nginx
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
```

Figure 8.9: Configuration file of a BPS instance.

```
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl apply -f deploy_sample.yaml
backuppodset.backup.example.com/bps-sample created
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl get pods -L type
NAME                      READY   STATUS     RESTARTS   AGE   TYPE
bps-sample-3f583358bca8   1/1     Running    0          28s   hotbackup
bps-sample-4cd4ff0807ed   1/1     Running    0          28s   hotbackup
bps-sample-58fe7068249d   0/1     Init:0/1   0          28s   coldbackup
bps-sample-59c4f1208433   0/1     Init:0/1   0          28s   coldbackup
bps-sample-9be87e6d4e00   1/1     Running    0          28s   primary
bps-sample-e6e44b332f58   1/1     Running    0          28s   primary
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl get bps
NAME         NAMESPACE   REPLICAS   HOTBACKUPS   COLDBACKUPS   AGE
bps-sample   default     2/2        2/2          2/2           50s
```

Figure 8.10: List of resources created by the BPS instance.

converted to a primary Pod. Two CB Pods, Pods 5 and 6 are activated and converted to HB Pods. Three Pods, Pods 7, 8,and 9 are newly created as the CB Pods. The time for the controller to handle the BPS updating request is 0.113 [s]. The total transition time from the last state to the current one of the BPS instance is 3.681 [s].

When the traffic decreases, the request of updating the number of Pods for each type to one for higher utilization arrives. Figure 8.11(a) shows the results after resource releasing. The controller compares the current and desired states of the different types of Pods and decreases the numbers of Pods for all types to one randomly with releasing the corresponding resources. The time for the controller to handle the BPS updating request is 0.226 [s]. transition time from the last state to the current one of the BPS instance is 2.137 [s].

In the case that a primary Pod failure is detected, the HB and CB Pods are activated to take over the task of the failed primary Pod. This demonstration deletes a primary Pod, Pod 3, to demonstrate the case of primary Pod failure. Figure 8.11(b) shows the maintenance results in confronting with a failure. The controller maintains the number of Pods for different types until the current BPS instance state becomes the desired state. The HB Pod, Pod 6, is converted to a primary Pod and is exposed to the service. The CB Pod, Pod 9, is successfully activated and converted to an HB Pod. A CB Pod, Pod 10, is newly created. The controller is normally running as shown in Fig. 8.12. The time for the controller to handle a primary Pod failure is 0.079 [s]. The total transition time is 1.933 [s].

8.3 Summary

This chapter firstly designed and implemented a two-layer controller structure in Kubernetes for automatic function deployment and management in a real-time and optimal manner. The controllers achieve the function deployment in a limited computation time with considering resource migration for allocation optimality. The demonstration validates that the controller automatically manages the resources promptly and correctly. The time for the controller to respond a request is within one second. The total transition time for the optimal deployment of the MPS instance is within four seconds. The

Initial		Increase the number of Pods for each type		Decrease the number of Pods for each type	
Pod	Type	Pod	Type	Pod	Type
1	Primary	1	Primary		
2	Primary	2	Primary		
3	HB	3	Primary	3	Primary
4	HB	4	HB		
5	CB	5	HB		
6	CB	6	HB	6	HB
		7	CB		
		8	CB		
		9	CB	9	CB
Time for controller: 0.211 [s]. Total creation time: 5.344 [s].		Time for controller: 0.113 [s]. Total transition time: 3.681 [s].		Time for controller: 0.226 [s]. Total transition time: 2.137 [s].	

(a) List of resources after updating the BPS instance.

Before failure		After failure	
Pod	Type	Pod	Type
3 ✗	Primary		
6	HB	6	Primary
9	CB	9	HB
		10	CB
Time for controller: 0.079 [s]. Total transition time: 1.933 [s].			

✗: failure

(b) List of resources after failure of primary Pod.

Figure 8.11: State transition of Pods triggered by requests.

```
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl delete pod bps-sample-4cd4ff0807ed
pod "bps-sample-4cd4ff0807ed" deleted
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl get pods -L type
NAME                          READY   STATUS     RESTARTS   AGE   TYPE
bps-sample-386d11b75f44   1/1     Running    0          11m   hotbackup
bps-sample-58fe7068249d   1/1     Running    0          13m   primary
bps-sample-92cea736875d   0/1     Init:0/1   0          34s   coldbackup
ubuntu@ubuntu:~/BackupResourcesController_k8s$ kubectl get bps
NAME         NAMESPACE   REPLICAS   HOTBACKUPS   COLDBACKUPS   AGE
bps-sample   default     1/1        1/1          1/1           13m
```

Figure 8.12: List of resources after deleting a primary Pod in BPS instance.

implemented system responds to the request approximately three times faster than the previous work in the tested case.

This chapter secondly designed and implemented a custom resource and the corresponding controller in Kubernetes to manage the primary and backup resources of network functions. The controller manages the state of each BPS instance to keep the current state consistent with the desired state. Demonstration validated that the controller automatically manages the resources correctly and rapidly. The time for the controller to handle a BPS operation is within one second. The total transition time from the last state to the current one of the BPS instance is within four seconds.

Chapter 9

Conclusions

The revolution brought about by NFV has made the deployment of functions more efficient, fast, and convenient. However, the occurrence of software and hardware failures poses significant challenges to the availability and continuity of services. In the face of failures, the backup and recovery mechanisms play a crucial role in ensuring the resilience of the network and the uninterrupted delivery of services. This thesis explores different protection approaches, i.e., dedicated protection and shared protection, as well as various backup strategies, including cold backup and hot backup, and their impact on recovery time; this thesis investigates the impact of node workload on failure probability. This thesis studies six specific problems about the resource allocations with workload-dependent failure probability with relative implementations in containerized networks.

Firstly, this thesis proposed a primary and backup resource allocation model with considering a workload-dependent failure probability aiming to minimize the maximum expected unavailable time. The workload-dependent failure probability and the consideration of different backup strategies cause different recovery time and lead to a nonlinear programming problem to calculate unavailable time for each VM. With step functions to approximate the given non-decreasing workload-dependent failure portability, this work formulated the problem as an MILP problem to obtain the primary and backup resource allocation of each VM in PMs, where the expected unavailable time is suppressed. This work proved that MEUT of the proposed model is equal

to the smaller value between the two MEUTs obtained by applying only HB and CB strategies with the same total requested load by comprehensively discussing different values of MEUT corresponding to different parameters. The heuristic algorithm inspired by the water-filling algorithm was developed based on the proved theorem. The numerical results showed that the proposed model suppresses MEUT compared with the conventional model which does not consider the workload-dependent failure probability. The developed heuristic algorithm is approximately 10^5 times faster than the MILP approach with 10^{-2} performance penalty on MEUT. This work discussed and implemented the approximation of step function for a non-decreasing function with a goal; this work investigated the performance of approximation for different problem sizes.

Secondly, this thesis proposed a multiple backup resource allocation model with the workload-dependent failure probability to minimize MEUT under a priority policy. This thesis analyzed the superiority of the protection priority policy to express the expected unavailable time for each function protected by multiple backup resources in the proposed model. This thesis derived the theorems that clarify the influence of policies on MEUT. This thesis formulated the optimization problem as an MILP problem. A lower bound of the optimal objective value in the proposed model was derived. This thesis proved that the decision version of the multiple resource allocation problem in the proposed model is NP-complete. A heuristic algorithm inspired by the water-filling algorithm was developed with providing an upper bound of the expected unavailable time obtained by the algorithm. The numerical results showed that the proposed model reduces MEUT compared with the single backup model in which each function is protected by only one server without protection priority of servers and the conventional model without the workload-dependent failure probability. The priority policy adopted in the proposed model specifying that the server which adopts the HB strategy has higher priority than that with the CB strategy for multiple backup resources suppresses MEUT compared with other priority policies. The developed heuristic algorithm is approximately 10^6 times faster than the MILP approach with 10^{-4} performance penalty on MEUT.

Thirdly, this thesis proposed a primary and backup resource allocation

model with preventive recovery priority setting against multiple failures to minimize W-UP for both dedicated and shared protection. Each node fails with different workload-dependent failure probabilities; each failure pattern has its corresponding weight. This thesis introduced a recovery strategy which is determined at the operation start time and can be applied for each failure pattern. Once failures are detected, the recoveries are operated with the workload variation according to the priority setting. Besides, this thesis introduced an approach to obtain unsuccessful recovery probability without priority setting. The numerical results observed that the proposed model reduces W-UP compared with baselines. The proposed model, which jointly considers the unsuccessful recovery and load balancing against failures, outperforms the baseline models which consider each type of unavailability separately. The developed heuristic algorithm is approximately 729 times faster than the MILP approach with 1.6% performance penalty on W-UP.

Fourthly, this thesis proposed a robust function deployment model against uncertain recovery time with satisfying an expected recovery time guarantee in a cost-efficient manner. Each node fails with a workload-dependent failure probability; preventive deployed backup resources can recover the unavailable function hosted by a failed node within a period of time related to the backup strategy. The expected recovery time of a function is related to the backup strategy (HB or CB strategy), protection types (dedicated or shared protection), the workload of the node hosting it, and the number of unavailable functions and remaining capacity of available nodes. this thesis introduced an uncertainty set that considers the upper bound of the average recovery time among nodes and the upper and lower bounds of each recovery time. The robust optimization technique was applied to handle the worst-case expected recovery time among the uncertain recovery time satisfying a recovery time guarantee; the model was formulated as an MILP problem. A greedy-based simulated annealing algorithm was developed to address the considered problem in practical scenarios. In the algorithm, this thesis transformed the linear programming problem to obtain the worst-case expected recovery time among uncertain times into a graph problem. The algorithm decreases the number of active nodes while decreasing the worst-case expected recovery time until the recovery time satisfies the recovery time guarantee. The numerical results

showed the superiority of the proposed models by taking into account the recovery time guarantee, uncertainty set, and shared protection, and this thesis investigated the dependency on the uncertain recovery time boundaries.

Fifthly, this thesis proposed a primary and backup resource allocation model under reliability guarantees to minimize the number of activated nodes. The resource allocation was considered with two kinds of workload distribution, the backup workload and the recovery workload. Different backup modes with different backup workloads corresponding to varying degrees of pre-configuration and recovery times. The extra-assigned recovery workload can be adopted to speed up the recovery while improving the resource efficiency; it may lead to unsuccessful recovery in a specific failure configuration. The proposed model minimizes the number of activated nodes as deployment cost while restricting the recovery time and the total unsuccessful recovery probability not to exceed each guarantee. The proposed model was compared with two baselines, each of which does not consider flexible backup modes and extra-assigned recovery workload, respectively, in different reliability guarantees. The numerical results revealed that the proposed model saves the deployment cost on average 14% and 11% in the experiments compared to the two baselines, respectively. This thesis analyzed the reasons for the superiority of the proposed model compared with baselines.

Sixthly, this thesis designed and implemented two custom resources and the corresponding controllers in Kubernetes to manage the function deployment in a real-time and optimal manner and the primary and backup resources of network functions, respectively. In the first demonstration, the controllers achieve the function deployment in a limited computation time with considering resource migration for allocation optimality. The demonstration validates that the controller automatically manages the resources promptly and correctly. The time for the controller to respond to a request is within one second. The total transition time for the optimal deployment of the MPS instance is within four seconds. The implemented system responds to the request approximately three times faster than the previous work in the tested case. In the second demonstration, the controller manages the state of each BPS instance to keep the current state consistent with the desired state. Demonstration validated that the controller automatically manages the resources correctly and rapidly.

The time for the controller to handle a BPS operation is within one second. The total transition time from the last state to the current one of the BPS instance is within four seconds.

The thesis introduced the five proposed models with workload-dependent unavailability and two implementations that address different distinct application scenarios in the realm of enhanced network virtualization reliability and fault tolerance, taking into account their respective attributes. Each model is accompanied by optimal modeling, algorithm, and theoretical analyses, offering diverse approaches to cater to specific needs. This comprehensive work equips network operators and service providers with the flexibility to choose the most suitable model and approach based on their specific requirements, thereby enabling them to establish a network virtualization environment that is adaptable, cost-efficient, and high available.

For future studies in resource management, two challenges with potential research directions stand out. The first challenge concerns the dynamic requests for resource allocation. Addressing this requires online allocation mechanisms capable of adapting to ever-changing demands. Online resource allocation considering dynamic requests can be achieved by different approaches. By leveraging machine learning techniques such as graph neural networks and autoregressive models, precise prediction of resource demands and occurrence of failures can be achieved. In-depth research into online scheduling algorithms holds the potential to enhance network performance, including congestion reduction and improved data transmission rates, particularly vital when expanding NFV services across geographically distributed data centers. Reinforcement learning plays a significant role in dynamic resource allocation algorithms, enabling real-time adaptive decision-making for optimizing resource allocation strategies. Its capacity for learning from real-time feedback facilitates rapid adaptation to changes and realization of real-time performance optimization. In complex environments, reinforcement learning navigates multifactorial influences by learning the relationships between states, actions, and rewards, striking an optimal balance. Its goal of resource allocation optimization aligns with the requirements of dynamic resource allocation, effectively distributing resources efficiently under varying demands and constraints to enhance overall system performance.

The second challenge focuses on migration issues in real-world production environments, such as user mobility in mobile edge computing networks and the complex dependency structures and heterogeneous resource demands these environments present. Key migration issues include ensuring seamless data transfer across different platforms, minimizing downtime, and maintaining consistency, especially for stateful applications. Seamless data transfer is crucial, as it requires maintaining data integrity across various architectures. Consistency is also vital to prevent data loss or functionality problems. Addressing these challenges necessitates strategies that balance technical requirements with operational continuity. This involves scheduling the migration of NFV services across multiple nodes or executors to achieve optimal service performance, meeting both dependency and user-perceived delay requirements. It requires identifying suitable VNF scheduling and migration strategies. Research should focus on optimizing handover delay and developing efficient algorithms for seamless management in NFV environments. This can be achieved by optimizing resource allocation and traffic management, ensuring that services have the necessary resources for faster data transmission and reduced job completion times.

Appendix A

Linearization of S-step function in Chapter 3

By introducing binary variable $z_{is}, i \in K, s \in \mathcal{S}$, where z_{is} is set to one if $T_i^{s-1} < W_i \leq T_i^s$, and zero otherwise, (3.2) can be expressed by:

$$\mathcal{P}_i^{\mathrm{S}} = \sum_{s \in \mathcal{S}} P_s z_{is}, \forall i \in K, \tag{A.1a}$$

$$W_i \leq \sum_{s \in \mathcal{S}} T_i^s z_{is}, \forall i \in K, \tag{A.1b}$$

$$W_i > \sum_{s \in \mathcal{S}} T_i^{s-1} z_{is}, \forall i \in K, \tag{A.1c}$$

$$\sum_{s \in \mathcal{S}} z_{is} = 1, \forall i \in K, \tag{A.1d}$$

$$z_{is} \in \{0, 1\}, \forall i \in K, s \in \mathcal{S}. \tag{A.1e}$$

Equation (A.1a) expresses the failure probability of a PM. With binary variable $z_{is}, i \in K, s \in \mathcal{S}$, when the workload of PM i is in the range of $(T_i^{s-1}, T_i^s]$, (A.1b) and (A.1c) restrict $z_{is} = 1$. Equation (A.1d) imposes that each PM corresponds to the failure probability in only one step among multiple steps.

Appendix B

Analysis of possible values of MEUT in Chapter 3

B.1 Possible values of MEUT

From (3.3d), we observe that multiple values of MEUT in the proposed model exist, each of which corresponds to a situation. Since the work in Chapter 3 considers that each VM is allocated into only one PM and protected by another PM with selecting one of the protection strategies, the possible values of (3.3d) can be simplified as:

$$r'_j = t_1 q_i (1 - q_k) + t_3 q_i q_k, \tag{B.1a}$$

$$r''_j = t_0 q_i (1 - q_k) + t_3 q_i q_k, \tag{B.1b}$$

where q_i and q_k denote the failure probabilities of PMs, which are either P_{L} or P_{H}. Therefore, MEUT has eight possible values:

$$r'^1_j = t_1 P_{\mathrm{L}} (1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}, \tag{B.2a}$$

$$r'^2_j = t_1 P_{\mathrm{L}} (1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.2b}$$

$$r'^3_j = t_1 P_{\mathrm{H}} (1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.2c}$$

$$r'^4_j = t_1 P_{\mathrm{H}} (1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}, \tag{B.2d}$$

$$r''^1_j = t_0 P_{\mathrm{L}} (1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}, \tag{B.3a}$$

$$r_j''^2 = t_0 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.3b}$$

$$r_j''^3 = t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.3c}$$

$$r_j''^4 = t_0 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}, \tag{B.3d}$$

where the four values in (B.2a)-(B.2d) are derived from (B.1a) by considering (q_i, q_k) as $(P_{\mathrm{L}}, P_{\mathrm{L}})$, $(P_{\mathrm{L}}, P_{\mathrm{H}})$, $(P_{\mathrm{H}}, P_{\mathrm{L}})$, and $(P_{\mathrm{H}}, P_{\mathrm{H}})$, respectively; the four values in (B.3a)-(B.3d) are derived from (B.1b) with the same consideration with (B.2a)-(B.2d), respectively.

This work investigates all the possible situations associated with MEUT with optimal consideration to minimize MEUT. With the assumption of $t_1 < t_0$, the proposed model chooses the HB strategy to reduce MEUT with the same allocation. The CB strategy does not affect the failure probability of backup resource in a PM compared with the HB strategy. If the failure probability of a PM whose backup resource provides protection with the CB strategy is P_{H}, the failure probability of that by adopting the HB strategy is also P_{H}. Therefore, $t_0 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$, and $t_0 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$ do not exist when MEUT is minimized. Both primary resource and backup resource that protects the primary one with the HB strategy work actively; the proposed model can swap the primary resource and backup resource without changing the failure probabilities of PMs. With the assumption of $P_{\mathrm{L}} < P_{\mathrm{H}}$, $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$ does not exist when MEUT is minimized since $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}} < t_1 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}$. There are five feasible values with optimal consideration left:

$$\mathcal{T}_1 = t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}, \tag{B.4a}$$

$$\mathcal{T}_2 = t_1 P_{\mathrm{L}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.4b}$$

$$\mathcal{T}_3 = t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}, \tag{B.4c}$$

$$\mathcal{T}_4 = t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{H}}, \tag{B.4d}$$

$$\mathcal{T}_5 = t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}. \tag{B.4e}$$

B.2 Boundary values

This work compares the values of $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, $\mathcal{T}_4$, and $\mathcal{T}_5$ to obtain the rank of the values of MEUT.

$$\mathcal{T}_2 - \mathcal{T}_1 = (t_3 - t_1) P_{\mathrm{L}} P_{\mathrm{H}} - (t_3 - t_1) P_{\mathrm{L}} P_{\mathrm{L}} \tag{B.5a}$$

$$\mathcal{T}_3 - \mathcal{T}_2 = t_0(P_\mathrm{L} - P_\mathrm{L}^2) + t_3 P_\mathrm{L}^2 - t_1(P_\mathrm{L} - P_\mathrm{L}P_\mathrm{H}) - t_3 P_\mathrm{L}P_\mathrm{H} \tag{B.5b}$$

$$\mathcal{T}_4 - \mathcal{T}_3 = t_0(P_\mathrm{H} - P_\mathrm{L})(1 - P_\mathrm{L}) + t_3(P_\mathrm{L}P_\mathrm{H} - P_\mathrm{L}P_\mathrm{L}) \tag{B.5c}$$

$$\mathcal{T}_5 - \mathcal{T}_3 = t_1(P_\mathrm{H} - P_\mathrm{H}^2) + t_3 P_\mathrm{H}^2 - t_0(P_\mathrm{L} - P_\mathrm{L}^2) - t_3 P_\mathrm{L}^2 \tag{B.5d}$$

$$\mathcal{T}_5 - \mathcal{T}_4 = t_1(P_\mathrm{H} - P_\mathrm{H}^2) + t_3 P_\mathrm{H}^2 - t_0(P_\mathrm{H} - P_\mathrm{L}P_\mathrm{H}) - t_3 P_\mathrm{L}P_\mathrm{H}. \tag{B.5e}$$

When $t_3 > t_1$, $t_3 > t_0$, and $P_\mathrm{L} < P_\mathrm{H}$, $\mathcal{T}_2 > \mathcal{T}_1$ and $\mathcal{T}_4 > \mathcal{T}_3$ can be easily derived from (B.5a) and (B.5c).

Assuming that (B.5b) is equal to 0, a boundary value of t_1 is obtained as:

$$v_2 = \frac{1 - P_\mathrm{L}}{1 - P_\mathrm{H}} t_0 + \frac{P_\mathrm{L} - P_\mathrm{H}}{1 - P_\mathrm{H}} t_3. \tag{B.6}$$

If $t_1 \leq v_2$, $\mathcal{T}_3 \leq \mathcal{T}_2$, and otherwise, $\mathcal{T}_2 > \mathcal{T}_3$.

Assuming that (B.5d) is equal to 0, a boundary value of t_1 is obtained as:

$$v_1 = \frac{P_\mathrm{L}(1 - P_\mathrm{L})}{P_\mathrm{H}(1 - P_\mathrm{H})} t_0 + \frac{P_\mathrm{L}^2 - P_\mathrm{H}^2}{P_\mathrm{H}(1 - P_\mathrm{H})} t_3. \tag{B.7}$$

If $t_1 \leq v_1$, $\mathcal{T}_5 \leq \mathcal{T}_3$, and otherwise, $\mathcal{T}_5 > \mathcal{T}_3$.

Assuming that (B.5e) is equal to 0, the same boundary value of t_1 as: $v_2 = \frac{1-P_\mathrm{L}}{1-P_\mathrm{H}} t_0 + \frac{P_\mathrm{L}-P_\mathrm{H}}{1-P_\mathrm{H}} t_3$ is obtained; if $t_1 \leq v_2$, $\mathcal{T}_5 \leq \mathcal{T}_4$, and otherwise, $\mathcal{T}_5 > \mathcal{T}_4$.

$$\begin{aligned} v_2 - v_1 &= \frac{(1 - P_\mathrm{L})(P_\mathrm{H} - P_\mathrm{L})}{P_\mathrm{H}(1 - P_\mathrm{H})} t_0 + \frac{P_\mathrm{L}P_\mathrm{H} - P_\mathrm{H}^2 - P_\mathrm{L}^2 + P_\mathrm{H}^2}{P_\mathrm{H}(1 - P_\mathrm{H})} t_3 \\ &= \frac{(P_\mathrm{H} - P_\mathrm{L})t_0}{P_\mathrm{H}(1 - P_\mathrm{H})} + \frac{(P_\mathrm{L}^2 - P_\mathrm{L}P_\mathrm{H})(t_0 - t_3)}{P_\mathrm{H}(1 - P_\mathrm{H})} t_3 \\ &= \frac{(P_\mathrm{H} - P_\mathrm{L})t_0}{P_\mathrm{H}(1 - P_\mathrm{H})} + \frac{(P_\mathrm{H} - P_\mathrm{L})P_\mathrm{L}(t_3 - t_0)}{P_\mathrm{H}(1 - P_\mathrm{H})} t_3 > 0. \end{aligned} \tag{B.8}$$

Therefore, $v_2 > v_1$.

B.3 Rank of the values of MEUT

In the condition of $t_1 \leq v_1$, the rank of MEUT among the five values is $t_1 P_\mathrm{L}(1 - P_\mathrm{L}) + t_3 P_\mathrm{L}P_\mathrm{L} < t_1 P_\mathrm{L}(1 - P_\mathrm{H}) + t_3 P_\mathrm{L}P_\mathrm{H} < t_1 P_\mathrm{H}(1 - P_\mathrm{H}) + t_3 P_\mathrm{H}P_\mathrm{H} \leq t_0 P_\mathrm{L}(1 - P_\mathrm{L}) + t_3 P_\mathrm{L}P_\mathrm{L} < t_0 P_\mathrm{H}(1 - P_\mathrm{L}) + t_3 P_\mathrm{H}P_\mathrm{L}$. $t_1 P_\mathrm{L}(1 - P_\mathrm{L}) + t_3 P_\mathrm{L}P_\mathrm{L}$ corresponds to the situation that the total requested load is so low that all VMs are protected with the HB strategy with failure probability P_L. Since each value of MEUT with

the CB strategy is not less than $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$, the proposed model can switch the CB strategy to the HB strategy to reduce MEUT. It indicates that each value of MEUT with the CB strategy does not exist when MEUT is minimized. MEUT of the proposed model is $t_1P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}, t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}}$, and $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ as the total requested load increases.

In the condition of $v_1 < t_1 \leq v_2$, the rank of MEUT among the five values is $t_1P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}} < t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}} \leq t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}} < t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}} \leq t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}}$. $t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}}$ does not exist when MEUT is minimized. The CB strategy can be switched to the HB strategy, which has smaller MEUT of $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ to reduce MEUT. Similar to the condition of $t_1 \leq v_1$, with the same total requested load, $t_1P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}$ and $t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}}$ can replace $t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}$ by switching the CB strategy to the HB strategy to reduce MEUT. As the increase of total requested load, $t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}$ can replace $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ until the total requested load is larger than the sum of thresholds of all PMs. $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ can replace $t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}}$ when the failure probability of the primary PM which is protected with the CB strategy is P_{H}. The protection strategies can be changed according to the total requested load so that the larger value of MEUT does not exist when MEUT is minimized. The proposed model adopts both HB and CB strategies to balance the recovery time and workload. MEUT is $t_1P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}, t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}}$, $t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}$, and $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ as the total requested load increases.

In the condition of $v_2 < t_1$, the rank of MEUT among the five values is $t_1P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}} < t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}} < t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}} < t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}} < t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$. $t_1P_{\mathrm{L}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{L}}P_{\mathrm{H}}$ does not exist when MEUT is minimized. The HB strategy can be switched to the CB strategy, which has smaller MEUT of $t_0P_{\mathrm{L}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{L}}P_{\mathrm{L}}$ to reduce MEUT. Similar to the condition of $v_1 < t_1 \leq v_2$, $t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}}$ can replace $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ by switching the HB strategy to the CB strategy to reduce MEUT until the total requested load becomes so large that the failure probabilities of both primary and backup PMs with the CB strategy is P_{H}. MEUT is $t_1P_{\mathrm{H}}(1-P_{\mathrm{H}})+t_3P_{\mathrm{H}}P_{\mathrm{H}}$ when the total requested load is larger than that of the case that MEUT is $t_0P_{\mathrm{H}}(1-P_{\mathrm{L}})+t_3P_{\mathrm{H}}P_{\mathrm{L}}$. The proposed model

in Chapter 3 adopts both HB and CB strategies to balance the recovery time and the workload. MEUT is $t_1 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}, t_0 P_{\mathrm{L}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{L}} P_{\mathrm{L}}$, $t_0 P_{\mathrm{H}}(1 - P_{\mathrm{L}}) + t_3 P_{\mathrm{H}} P_{\mathrm{L}}$, and $t_1 P_{\mathrm{H}}(1 - P_{\mathrm{H}}) + t_3 P_{\mathrm{H}} P_{\mathrm{H}}$ as the total requested load increases.

Appendix C

Linearization of proposed model in Chapter 4

$\phi^b_{ijj'} = e^i_{jj'} x^b_{ij} y_{j'}$, $\pi^b_{ijj'} = e^i_{jj'} x^b_{ij}$, $i \in F, j \in N, j' \in S_j, b \in B$, and $\zeta^b_{ij} = x^b_{ij} y_j, i \in F, j \in N, b \in B$, can be linearized as:

$$\phi^b_{ijj'} \leq e^i_{jj'}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1a}$$

$$\phi^b_{ijj'} \leq x^b_{ij'}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1b}$$

$$\phi^b_{ijj'} \leq y_{j'}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1c}$$

$$\phi^b_{ijj'} \geq e^i_{jj'} + x^b_{ij'} + y_{j'} - 2, \forall i \in F, j \in N, \tag{C.1d}$$
$$j' \in S_j, b \in B,$$

$$\pi^b_{ijj'} \leq e^i_{jj'}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1e}$$

$$\pi^b_{ijj'} \leq x^b_{ij'}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1f}$$

$$\pi^b_{ijj'} \geq e^i_{jj'} + x^b_{ij'} - 1, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1g}$$

$$\zeta^b_{ij} \leq x^b_{ij}, \forall i \in F, j \in N, b \in B, \tag{C.1h}$$

$$\zeta^b_{ij} \leq y_j, \forall i \in F, j \in N, b \in B, \tag{C.1i}$$

$$\zeta^b_{ij} \geq x^b_{ij} + y_j - 1, \forall i \in F, j \in N, b \in B, \tag{C.1j}$$

$$\phi^b_{ijj'}, \pi^b_{ijj'} \in \{0, 1\}, \forall i \in F, j \in N, j' \in S_j, b \in B, \tag{C.1k}$$

$$\zeta^b_{ij} \in \{0, 1\}, \forall i \in F, j \in N, b \in B. \tag{C.1l}$$

By introducing binary variables $\lambda^{ijb}_{mm'} = \omega^{\mathrm{L}}_{mij} \omega^{\mathrm{H}}_{m'ij} x^b_{ij}$, $\theta^{ijb}_{mm'} = \omega^{\mathrm{L}}_{mij} \omega^{\mathrm{H}}_{m'ij} x^b_{ij} y_j$, where $m, m' \in [0, |S|], i \in F, j \in N, b \in B$, $\eta_{mm'i} = z^{\mathrm{L}}_{mi} z^{\mathrm{H}}_{m'i}$, where $m, m' \in$

$[0, |S|], i \in F$, (4.4e) can be expressed by a linear form as:

$$u_i = \sum_{b\in B}\sum_{j\in N}\{t_b p_i \sum_{m\in M}\sum_{m'\in M}((P_{\mathrm{L}})^m (P_{\mathrm{H}})^{m'} \tag{C.2a}$$
$$((1-P_{\mathrm{H}})\lambda^{ijb}_{mm'} + (P_{\mathrm{H}} - P_{\mathrm{L}})\theta^{ijb}_{mm'}))\}+$$
$$t_3 p_i \sum_{m\in M}\sum_{m'\in M}\{(P_{\mathrm{L}})^m (P_{\mathrm{H}})^{m'}\eta_{mm'i}\}, \forall i \in F,$$
$$\lambda^{ijb}_{mm'} \le \omega^{\mathrm{L}}_{mij}, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2b}$$
$$\lambda^{ijb}_{mm'} \le \omega^{\mathrm{H}}_{m'ij}, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2c}$$
$$\lambda^{ijb}_{mm'} \le x^b_{ij}, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2d}$$
$$\lambda^{ijb}_{mm'} \ge \omega^{\mathrm{L}}_{mij} + \omega^{\mathrm{H}}_{m'ij} + x^b_{ij} - 2, \forall m, m' \in M, \tag{C.2e}$$
$$i \in F, j \in N, b \in B,$$
$$\theta^{ijb}_{mm'} \le \lambda^{ijb}_{mm'}, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2f}$$
$$\theta^{ijb}_{mm'} \le y_j, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2g}$$
$$\theta^{ijb}_{mm'} \ge \lambda^{ijb}_{mm'} + y_j - 1, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2h}$$
$$\eta_{mm'i} \le z^{\mathrm{L}}_{mi}, \forall m, m' \in M, i \in F, \tag{C.2i}$$
$$\eta_{mm'i} \le z^{\mathrm{H}}_{m'i}, \forall m, m' \in M, i \in F, \tag{C.2j}$$
$$\eta_{mm'i} \ge z^{\mathrm{L}}_{mi} + z^{\mathrm{H}}_{m'i} - 1, \forall m, m' \in M, i \in F, \tag{C.2k}$$
$$\lambda^{ijb}_{mm'}, \theta^{ijb}_{mm'} \in \{0,1\}, \forall m, m' \in M, i \in F, j \in N, b \in B, \tag{C.2l}$$
$$\eta_{mm'i} \in \{0,1\}, \forall m, m' \in M, i \in F. \tag{C.2m}$$

Appendix D

Performance bounds in Chapter 4

Let $n_i = \sum_{b \in B} \sum_{j \in N} x_{ij}^b, \forall i \in F$, denote the number of servers which provide protection for function $i \in F$. This work reorders the set of F by sorting i by w_i decreasingly, i.e., $w_i \geq w_{i-1}, \forall i \in F \backslash \{1\}$. Let G denote a set of ordered function sets $G = \{\emptyset, \{1\}, \{1,2\}, \{1,2,3\}, \cdots, F\}$. Let F' denote each element in G, which contains $|F'|$ functions with largest requested loads. This work introduces a binary parameter $\alpha_{j|F'|}, j \in N, F' \in G$. $\alpha_{j|F'|}$ is set to 1 if $C_j \geq \sum_{i \in F \backslash F'} w_i$, and 0 otherwise. This work introduces an integer parameter $\beta_j, j \in N$, which equals $\max_{F' \in G} \left\{(|F| - |F'|)\alpha_{j|F'|}\right\}$.

Lemma D.1 $\sum_{i \in F} n_i \leq \sum_{j \in N} \beta_j$.

Proof : Since this work reorders the set of F by sorting i by w_i decreasingly, if a server has enough capacity to protect function $i \in F$, it must be able to protect function $i' \in F, i' \geq i$. On the contrary, if a server protects function $i' \in F$, it may not be able to protect function $i \in F, i < i'$. If $C_j \geq \sum_{i \in F \backslash F'} w_i$, functions in set $F \backslash F'$ can be allocated to server $j \in N$. The smaller $|F'|$ is, the more functions can be allocated to server $j \in N$. $\beta_j = \max_{F' \in G} \left\{(|F| - |F'|)\alpha_{j|F'|}\right\}$ is the maximum number of functions which can be allocated to server $j \in N$ according to the capacity of server j and the total requested load. $\sum_{j \in N} \beta_j$ is the maximum number of servers which provide protection for all functions in F; it is the upper bound of $\sum_{i \in F} n_i$. □

Theorem D.1 *When the failure probability of each function $i \in F$ is the same and only the HB strategy is considered, the function which corresponds to MEUT is protected by at most* $\min\{|N|, \frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}\}$ *servers.*

Proof : Firstly, without loss of generality, this work assumes two different functions i_1 and i_2; function $i_1 \in F$ is protected by n_1 servers; function $i_2 \in F$ is protected by n_2 servers. The failure probability of each function $i \in F$ is denoted by p. Since only the HB strategy is considered, the upper bound of the expected unavailable time of i_1 is $p(t_1+(t_3-t_1)P_{\mathrm{H}}^{n_1})$; the lower bound of the expected unavailable time of i_2 is $p(t_1+(t_3-t_1)P_{\mathrm{L}}^{n_2})$. By subtracting the lower bound of u_{i_2} from the upper bound of u_{i_1}, this work obtains that, if $u_{i_1} \geq u_{i_2}$, then $P_{\mathrm{H}}^{n_1} \geq P_{\mathrm{L}}^{n_2}$. It indicates that, if $u_{i_1} \geq u_{i_2}$, then $\frac{n_1}{n_2} \leq \frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}$.

Let i' denote the function which corresponds to MEUT, $n_{i'}$ denote the number of servers which protect i', and $u_{i'}$ denote MEUT. Since $u_{i'} \geq u_i, \forall i \in F$, this work has $n_{i'} \leq n_i\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}, \forall i \in F$. It indicates that $|F|n_{i'} \leq \sum_{i\in F} n_i\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}$. With Lemma D.1, this work has $n_{i'} \leq \frac{\sum_{i\in F} n_i}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}} \leq \frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}$. Each function can be protected by at most $|N|$ servers, while the function which corresponds to MEUT can be protected by at most $\frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}$ servers. Therefore, the function which corresponds to MEUT is protected by at most $\min\{|N|, \frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}\}$ servers with the HB strategy. □

Theorem D.2 *When the failure probability of each function $i \in F$ is the same and only the HB strategy is considered, MEUT in an optimal solution is at least* $t_1P_{\mathrm{L}}^{\mathcal{M}}(1-P_{\mathrm{H}}) + t_3P_{\mathrm{L}}^{\mathcal{M}+1}$, *where* $\mathcal{M} = \min\{|N|, \frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}\}$.

Proof : Let u^* denote MEUT obtained by the algorithm and i' denote the function which corresponds to MEUT. Let $p_{i'}$ denote the failure probability of function with MEUT. Theorem D.1 indicates that the maximum size of $S_i, i \in F$, is $\mathcal{M} = \min\{|N|, \frac{\sum_{j\in N}\beta_j}{|F|}\frac{\log P_{\mathrm{L}}}{\log P_{\mathrm{H}}}\}$. Therefore, this work obtains that $\prod_{O_{ij'}>O_{ij}:j'\in S_i} q_{j'}(1-q_j) \geq P_{\mathrm{L}}^{\mathcal{M}-1}(1-q_j), j \in N_i$; $\prod_{j\in N_i} q_j \geq P_{\mathrm{L}}^{\mathcal{M}}$. By using (4.4e), this work has:

$$\begin{aligned} u^* &\geq \sum_{b\in B}\sum_{j\in N_{i'}} t_b x_{i'j}^b p_{i'} P_{\mathrm{L}}^{\mathcal{M}-1}(1-q_j) + t_3 p_{i'} P_{\mathrm{L}}^{\mathcal{M}}, \\ &\geq t_1 P_{\mathrm{L}}^{\mathcal{M}}(1-P_{\mathrm{H}}) + t_3 P_{\mathrm{L}}^{\mathcal{M}+1}. \end{aligned} \tag{D.1}$$

Therefore, a lower bound of the optimum objective value is obtained as $u^* \geq t_1 P_{\mathrm{L}}^{\mathcal{M}}(1-P_{\mathrm{H}}) + t_3 P_{\mathrm{L}}^{\mathcal{M}+1}$. □

Lemma D.2 *The lower bound of the minimum size of* $S_i, i \in F$, *is* $\frac{\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i'\in F} w_i'}\}|N|}{|F|}$.

Proof : The maximum requested load among functions in F, is expressed by $\max_{i\in F} r_i$. The number of functions that can be protected by a server is at least $\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i\in F} w_i}\}$. If $\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i\in F} w_i}\}\frac{|N|}{|F|} \geq |F|$, each function can be protected by at least $|N|$ servers and the minimum size of $S_i, i \in F$, is $|N|$. If $\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i\in F} w_i}\}\frac{|N|}{|F|} < |F|$, this work can find a set of functions that can be protected by each server without any duplicated element. $\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i\in F} w_i}\}|N|$ is the minimum total number of functions that can be protected by all servers. Therefore, the lower bound of the minimum size of $S_i, i \in F$, is $\frac{\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i'\in F} w_i'}\}|N|}{|F|}$. □

Theorem D.3 *The maximum expected unavailable time obtained by Algorithm 4.1 is at most* $t_1 \frac{P_H(1-P_L)(1-P_H^{\mathcal{N}})}{1-P_H} + t_3 P_H^{\mathcal{N}'+1}$, *where* $\mathcal{N} = \min\{|N|, \lfloor \frac{\sum_{j\in N} C_j}{\sum_{i'\in F} w_i'} \rfloor\}$ *and* $\mathcal{N}' = \frac{\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i'\in F} w_i'}\}|N|}{|F|}$.

Proof : Let u^* denote MEUT obtained by the algorithm and i' denote the function which corresponds to MEUT. Let $p_{i'}$ denote the failure probability of function with MEUT. The maximum size of $S_i, i \in F$, is $\mathcal{N} = \min\{|N|, \lfloor \frac{\sum_{j\in N} C_j}{\sum_{i'\in F} w_i'} \rfloor\}$ and the lower bound of the minimum size of $S_i, i \in F$, is $\mathcal{N}' = \frac{\min\{|F|, \frac{\min_{j\in N} C_j}{\max_{i'\in F} w_i'}\}|N|}{|F|}$, where $\mathcal{N}' \leq \mathcal{N}$.

Therefore, this work obtains that $\prod_{j\in N_{i'}} q_j \leq P_{\mathrm{H}}^{\mathcal{N}'}$. Since $\sum_{j\in N_{i'}} \prod_{O_{i'j'}>O_{i'j}:j'\in N_{i'}} q_{j'}(1-q_j)$, can be approximated as a geometric sequence with initial value $(1-P_{\mathrm{L}})$ and common ratio P_{H}, this work obtains that $\sum_{j\in N_{i'}} \prod_{O_{i'j'}>O_{i'j}:j'\in N_{i'}} q_{j'}(1-q_j) \leq \frac{(1-P_{\mathrm{L}})(1-P_{\mathrm{H}}^{\mathcal{N}})}{1-P_{\mathrm{H}}}$. By using (4.4e), this work has:

$$\begin{aligned} u^* &\leq t_1 p_{i'} \frac{(1-P_{\mathrm{L}})(1-P_{\mathrm{H}}^{\mathcal{N}})}{1-P_{\mathrm{H}}} + t_3 p_{i'} P_{\mathrm{H}}^{\mathcal{N}'}, \\ &\leq t_1 \frac{P_{\mathrm{H}}(1-P_{\mathrm{L}})(1-P_{\mathrm{H}}^{\mathcal{N}})}{1-P_{\mathrm{H}}} + t_3 P_{\mathrm{H}}^{\mathcal{N}'+1}. \end{aligned} \tag{D.2}$$

Therefore, an upper bound of the objective value obtained by the algorithm is obtained as $u^* \leq t_1 \frac{P_{\mathrm{H}}(1-P_{\mathrm{L}})(1-P_{\mathrm{H}}^N)}{1-P_{\mathrm{H}}} + t_3 P_{\mathrm{H}}^{N'+1}$. □

Appendix E

NP-Completeness of the problem in Chapter 4

This work defines the decision version of the multiple backup resource allocation (MBRA) problem as: given a set of functions F and a set of servers S, is it possible to find a assignment of servers to functions so that MEUTs among all functions is no more than c?

Theorem E.1 *The decision version of the multiple backup resource allocation problem is NP-complete.*

Proof : Firstly, the MBRA problem is NP, as this work can verify whether a certificate of any instance of the MBRA decision problem, which is an assignment of functions to backup servers, makes MEUTs among all functions no more than c by calculating (4.4e) and (4.4f) in a polynomial time of $O(|F||N|)$.

Then, this work shows a reduction from the partition problem, which is a known NP-complete problem [136], to prove the NP-completeness of MBRA. The partition problem 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?

First, this work constructs an instance of MBRA from any instance of the partition problem. An instance of the partition problem consists of a set of positive integers $G = \{I_g : g \in [1, |G|]\}$. An instance of the MBRA decision problem is constructed with the following steps, which performs in

a polynomial time of $O(|G|)$. An instance of MBRA is constructed with the following steps:

1) Set $|F| = 2$ and both failure probabilities of two functions are set to 1;

2) Set $w_i = 1, \forall i \in F$ and $C_j = 1, \forall j \in N$. It indicates that each server can protect at most one function;

3) Set $|N| = |G|$. For each positive integer $g \in G$, there is a corresponding server $j \in N$ with $T_j = 0, \forall j \in N$, which means threshold of each server is set to 0. The failure possibility of each server is set to $q_j = e^{-I_g}$;

4) Set $t_0 = t_3$;

5) Set $c = (t_3 - t_1)e^{\frac{-\sum_{I_g \in G} I_g}{2}} + t_1 = (t_3 - t_1)\sqrt{\prod_{j \in N} q_j} + t_1$.

This construction imposes that each server can protect at most one function with the HB strategy in MBRA. According to the proof of Theorem 4.1, the expected unavailable time of each function $i \in F$, u_i, can be simplified as (4.5). u_i is expressed by $t_1 + (t_3 - t_1) \prod_{j \in N_i} q_j$.

Consider that a partition problem instance is a Yes instance, which indicates that there exists two subsets of G_1 and G_2 with $\sum_{I_g \in G_1} I_g = \sum_{I_g \in G_2} I_g$. By using the above presented steps to define the corresponding MBRA instance from any partition problem instance, each function is assigned to multiple servers. The production of failure probabilities for each subset of servers, is the same and equal to $e^{\frac{-\sum_{I_g \in G} I_g}{2}}$. The expected unavailable time of each function protected by servers which correspond to each of G_1 and G_2 is equal to $(t_3 - t_1)e^{\frac{-\sum_{I_g \in G} I_g}{2}} + t_1$. As a result, it is possible to find an assignment of the two functions to servers so that MEUTs of the two functions is no more than c.

Conversely, consider that an MBRA instance is a Yes instance, which indicates that it is possible to find an assignment of the two functions to servers making both expected unavailable time of the two functions are no more than c. Let S_1 and S_2 be the sets of servers each of which protects each function in the assignment. G_1 and G_2 are two subsets of G, which correspond to S_1 and S_2, respectively. Let $G' = G_1 \cup G_2$ and $G' \subseteq G$. Since each server can protect at most one function, $S_1 \cap S_2 = \emptyset$, which indicates that $G_1 \cap G_2 = \emptyset$ and $G' = G$. The expected unavailable time of two functions protected by servers with the HB strategy which correspond to

G_1 and G_2 is equal to $t_1 + (t_3 - t_1)e^{-\sum_{I_g \in G_1} I_g}$ and $t_1 + (t_3 - t_1)e^{-\sum_{I_g \in G_2} I_g}$, respectively. Since both expected unavailable time of the two functions are no more than $t_1 + (t_3 - t_1)e^{\frac{-\sum_{I_g \in G} I_g}{2}}$, this work can obtain that the production of failure probabilities for each subset of servers which protect each function is $e^{-\sum_{I_g \in G_1} I_g} \leq e^{\frac{-\sum_{I_g \in G} I_g}{2}}$ and $e^{-\sum_{I_g \in G_2} I_g} \leq e^{\frac{-\sum_{I_g \in G} I_g}{2}}$, respectively, which indicates that $\sum_{I_g \in G_1} I_g \geq \frac{\sum_{I_g \in G} I_g}{2}$ and $\sum_{I_g \in G_2} I_g \geq \frac{\sum_{I_g \in G} I_g}{2}$. Since $G' = G_1 \cup G_2 = G$, $\sum_{I_g \in G_1} I_g + \sum_{I_g \in G_2} I_g = \sum_{I_g \in G} I_g$. Therefore, $\sum_{I_g \in G_1} I_g = \sum_{I_g \in G_2} I_g = \frac{\sum_{I_g \in G} I_g}{2}$. As a result, this work is able to partition G into two subsets G_1 and G_2 such that the two sums of numbers in the two subsets are equal. If an MBRA instance is a Yes instance, then the corresponding partition problem instance is a Yes instance.

This work confirmed that if a partition problem instance is a Yes instance, then the corresponding MBRA instance is a Yes instance, and vice versa. This work can transform any instance of the partition problem into an MBRA instance in a polynomial time. This confirms that the partition problem, which is NP-complete, is polynomial time reducible to MBRA. Since MBRA belongs to NP, MBRA is NP-complete. □

Appendix F

Parts of linearzion processing for proposed model in Chapter 5

For the sake of brevity and readability, let $p \in \mathbb{P} \coloneqq [1, |N|]$; $a \in \mathbb{A} \coloneqq [1, \binom{|F|}{\Gamma}]$. Equation (1), $\chi_{ij}, i \in N, j \in F$, $\xi_{jk}, j \in F, k \in N$, and $\theta_{ij}^{kp}, i \in N, j \in F, k \in N \backslash \{i\}, p \in \mathbb{P}$, are linearized to (8a)-(8f), (9a)-(9f) and (10a)-(10g), respectively. Equations (3) and (5f) are linearized to (11a)-(14f) with some auxiliary variables as follows:

$$p_j^* \leq \sum_{i \in N \backslash \{k\}} p x_{ij}^{kp} + (1 - \sigma_{jk}^p) B, \forall j \in F, p \in \mathbb{P}, k \in N, . \tag{F.1a}$$

$$p_j^* \geq \sum_{i \in N \backslash \{k\}} p x_{ij}^{kp} - (1 - \sigma_{jk}^p) B, \forall j \in F, p \in \mathbb{P}, k \in N, \tag{F.1b}$$

$$\sum_{i \in N \backslash \{k\}} p x_{ij}^{kp} \geq (\sigma_{jk}^p - 1) B + \sum_{i \in N \backslash \{k'\}} p' x_{ij}^{k'p'}, \forall j \in F,$$

$$(p, k) \in (\mathbb{P}, N), (p', k') \in (\mathbb{P}, N) \backslash \{(p, k)\}, \tag{F.1c}$$

$$\sum_{p \in \mathbb{P}} \sum_{k \in N} \sigma_{jk}^p = 1, \forall j \in F, \tag{F.1d}$$

$$p_j^* \geq \sum_{i \in N \backslash \{k\}} p x_{ij}^{kp}, \forall j \in F, p \in \mathbb{P}, k \in N, \tag{F.1e}$$

$$\sigma_{jk}^p \in \{0, 1\}, \forall j \in F, p \in \mathbb{P}, k \in N.. \tag{F.1f}$$

$$\chi_{ij} \geq \frac{1}{|N|(|N|-1)} \sum_{p\in\mathbb{P}} \sum_{k\in N\setminus\{i\}} x_{ij}^{kp}, \forall i \in N, j \in F, . \tag{F.2a}$$

$$\chi_{ij} \leq \sum_{p\in\mathbb{P}} \sum_{k\in N\setminus\{i\}} x_{ij}^{kp}, \forall i \in N, j \in F, \tag{F.2b}$$

$$\xi_{jk} \geq \frac{1}{|N|(|N|-1)} \sum_{p\in\mathbb{P}} \sum_{i\in N\setminus\{k\}} x_{ij}^{kp}, \forall j \in F, k \in N, \tag{F.2c}$$

$$\xi_{jk} \leq \sum_{p\in\mathbb{P}} \sum_{i\in N\setminus\{k\}} x_{ij}^{kp}, \forall j \in F, k \in N, \tag{F.2d}$$

$$\chi_{ij} \in \{0,1\}, \forall i \in N, j \in F, \tag{F.2e}$$

$$\xi_{jk} \in \{0,1\} \forall j \in F, k \in N. \tag{F.2f}$$

$$\theta_{ij}^{kp} \leq x_{ij}^{kp}, \forall i \in N, j \in F, k \in N\setminus\{i\}, p \in \mathbb{P}, \tag{F.3a}$$

$$\theta_{ij}^{kp} \leq \delta_j^p, \forall i \in N, j \in F, k \in N\setminus\{i\}, p \in \mathbb{P}, \tag{F.3b}$$

$$\theta_{ij}^{kp} \geq x_{ij}^{kp} + \delta_j^p - 1, \forall i \in N, j \in F, k \in N\setminus\{i\}, p \in \mathbb{P}, \tag{F.3c}$$

$$\theta_{ij}^{kp} \in \{0,1\}, \forall i \in N, k \in N\setminus\{i\}, j \in F, p \in \mathbb{P}, \tag{F.3d}$$

$$1 - \delta_j^p \leq p_j^* - p, \forall j \in F, p \in \mathbb{P}, \tag{F.3e}$$

$$1 - \delta_j^p \geq (p_j^* - p)\epsilon, \forall j \in F, p \in \mathbb{P}, \tag{F.3f}$$

$$\delta_j^p \in \{0,1\}, \forall j \in F, p \in \mathbb{P}. \tag{F.3g}$$

where $0 < \epsilon < \frac{1}{|N|}$.

Let y_{ms}^{σ}, y'^{σ}_{ms}, $m \in [0,|N|]$, $s \in \mathbb{S}$, $\sigma \in \mathbb{C}$, be binary variables to linearize (5.4). y_{ms}^{σ} is set to one if $\sum_{i\in\mathcal{P}_\Gamma^\sigma} z_{is}^0$ is equal to m among nodes in $\mathcal{P}_\Gamma^\sigma$, and zero otherwise. y'^{σ}_{ms} is set to one if $\sum_{i\in N\setminus\mathcal{P}_\Gamma^\sigma} z_{is}^0$ is equal to m for nodes in $N\setminus\mathcal{P}_\Gamma^\sigma$, and zero otherwise. This work can express y_{ms}^{σ} and y'^{σ}_{ms} in the form of the following constraints:

$$\sum_{i\in\mathcal{P}_\Gamma^\sigma} z_{is}^0 = \sum_{m\in[0,|N|]} m y_{ms}^{\sigma}, \forall \sigma \in \mathbb{C}, s \in \mathbb{S}, \tag{F.4a}$$

$$\sum_{i\in N\setminus\mathcal{P}_\Gamma^\sigma} z_{is}^0 = \sum_{m\in[0,|N|]} m y'^{\sigma}_{ms}, \forall \sigma \in \mathbb{C}, s \in \mathbb{S}, \tag{F.4b}$$

$$\sum_{m\in[0,|N|]} y^{\sigma}_{ms} = 1, s \in \mathbb{S}, \sigma \in \mathbb{C}, \tag{F.4c}$$

$$\sum_{m\in[0,|N|]} y'^{\sigma}_{ms} = 1, s \in \mathbb{S}, \sigma \in \mathbb{C}, \tag{F.4d}$$

$$y^{\sigma}_{ms}, y'^{\sigma}_{ms} \in \{0,1\}, \forall m \in [0,|N|], \sigma \in \mathbb{C}, s \in \mathbb{S}. \tag{F.4e}$$

Q_σ can be expressed as a linear form as:

$$Q_\sigma \leq q_{i\sigma} + (1-\omega_{i\sigma})B, \forall i \in N\backslash\mathcal{P}^{\sigma}_{\Gamma}, \sigma \in \mathbb{C}, \tag{F.5a}$$

$$Q_\sigma \geq q_{i\sigma} - (1-\omega_{i\sigma})B, \forall i \in N\backslash\mathcal{P}^{\sigma}_{\Gamma}, \sigma \in \mathbb{C}, \tag{F.5b}$$

$$q_{i\sigma} \geq (\omega_{i\sigma}-1)B + q_{i'\sigma}, \forall i, i' \in N\backslash\mathcal{P}^{\sigma}_{\Gamma}, \sigma \in \mathbb{C}, i' \neq i, \tag{F.5c}$$

$$\sum\nolimits_{i\in N\backslash\mathcal{P}^{\sigma}_{\Gamma}} \omega_{i\sigma} = 1, \forall \sigma \in \mathbb{C}, \tag{F.5d}$$

$$Q_\sigma \geq q_{i\sigma}, \forall i \in N\backslash\mathcal{P}^{\sigma}_{\Gamma}, \sigma \in \mathbb{C}, \tag{F.5e}$$

$$\omega_{i\sigma} \in \{0,1\}, \forall i \in N\backslash\mathcal{P}^{\sigma}_{\Gamma}, \sigma \in \mathbb{C}. \tag{F.5f}$$

$e^{\sigma}_j \in \{0,1\}, g^{\sigma}_i \in \{0,1\}$, and $r_\sigma \in \{0,1\}$ can be expressed as a linear form as:

$$\sum_{i\in N}(\chi_{ij}+\xi_{ji}) - \sum_{i\in\mathcal{P}^{\sigma}_{\Gamma}}(\chi_{ij}+\xi_{ji}) \geq -e^{\sigma}_j A + \epsilon, \forall j \in F, \sigma \in \mathbb{C}, \tag{F.6a}$$

$$\sum_{i\in N}(\chi_{ij}+\xi_{ji}) - \sum_{i\in\mathcal{P}^{\sigma}_{\Gamma}}(\chi_{ij}+\xi_{ji}) \leq (1-e^{\sigma}_j)A, \forall j \in F, \sigma \in \mathbb{C}, \tag{F.6b}$$

$$L^{\mathrm{W}}_{i\sigma} - C_i \leq g^{\sigma}_i A, \forall i \in N, \sigma \in \mathbb{C}, \tag{F.6c}$$

$$L^{\mathrm{W}}_{i\sigma} - C_i \geq (g^{\sigma}_i - 1)A, \forall i \in N, \sigma \in \mathbb{C}, \tag{F.6d}$$

$$r_\sigma \geq \frac{1}{|F|+|N|}\left(\sum_{j\in F} e^{\sigma}_j + \sum_{i\in N} g^{\sigma}_i\right), \forall \sigma \in \mathbb{C}, \tag{F.6e}$$

$$r_\sigma \leq \sum_{j\in F} e^{\sigma}_j + \sum_{i\in N} g^{\sigma}_i, \forall \sigma \in \mathbb{C}, \tag{F.6f}$$

where A is a sufficiently large number that satisfies $A \geq 2|F|$. R_σ can be expressed as a linear form as:

$$R_\sigma \leq w_\sigma + (1-r_\sigma)A, \forall \sigma \in \mathbb{C}, \tag{F.7a}$$

$$R_\sigma \geq w_\sigma - (1-r_\sigma)A, \forall \sigma \in \mathbb{C}, \tag{F.7b}$$

$$R_\sigma \leq r_\sigma A, \forall \sigma \in \mathbb{C}, \tag{F.7c}$$

$$R_\sigma \geq -r_\sigma A, \forall \sigma \in \mathbb{C}, \tag{F.7d}$$

$$r_\sigma \in \{0,1\}, \forall \sigma \in \mathbb{C}. \tag{F.7e}$$

Appendix G

Parts of linearzion processing for proposed model in Chapter 6

To linearize ρ_{jk} and ω_{jk}, the linear form of $\mathcal{N}_{jk}$ and C_{jk} are required. First, this work introduces binary variable $e^j_{kk'}$, $j \in F, k \in N, k' \in N \setminus \{k\}$: it is set to one if $O_{jk'} > O_{jk}$, and zero otherwise. $e^j_{kk'}$ can be expressed as a linear form as:

$$O_{jk'} - O_{jk} \leq e^j_{kk'} A, \forall j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.1a}$$

$$O_{jk'} - O_{jk} \geq (e^j_{kk'} - 1)A, \forall j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.1b}$$

$$e^j_{kk'} + e^j_{k'k} = 1, \forall j \in F, k \in N, k' \in N \setminus \{k\}. \tag{G.1c}$$

With $e^j_{kk'}$, this work introduces binary variable $\lambda_{jj'kk'}$ that satisfies $\sum_{O_{jk'} > O_{jk}} \Lambda_{k'} = \sum_{j' \in F} \sum_{k' \in N} \lambda_{jj'kk'}$, which can expressed in a linear form as:

$$\lambda_{jj'kk'} \leq \chi_{k'j'}, j, j' \in F, k \in N, , k' \in N\{k\} \tag{G.2a}$$

$$\lambda_{jj'kk'} \leq e^j_{kk'}, j, j' \in F, k \in N, k' \in N\{k\} \tag{G.2b}$$

$$\lambda_{jj'kk'} \geq \chi_{k'j'} + e^j_{kk'} - 1, j, j' \in F, k \in N, k' \in N\{k\}. \tag{G.2c}$$

Similarly, this work introduces binary variable $f^j_{kk'}$, $j \in F, k \in N, k' \in N \setminus \{k\}$: it is set to one if $O_{jk'} < O_{jk}$ and $\sum_{p \in B} \xi^p_{jk'} = \sum_{p \in B} \xi^p_{jk} = 1$, and zero otherwise. $f^j_{kk'}$ can be expressed as a linear form as:

$$f^j_{kk'} = f1^j_{kk'} - f2^j_{kk'}, j \in F, k \in N, k' \in N \setminus \{k\} \tag{G.3a}$$

$$f1^j_{kk'} \leq \sum_{p \in B} \xi^p_{jk'}, j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.3b}$$

$$f1^j_{kk'} \leq \sum_{p\in B} \xi^p_{jk}, j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.3c}$$

$$f1^j_{kk'} \leq \sum_{p\in B} \xi^p_{jk} + \sum_{p\in B} \xi^p_{jk'} - 1, j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.3d}$$

$$f2^j_{kk'} \leq f1^j_{kk'}, j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.3e}$$

$$f2^j_{kk'} \leq e^j_{kk'}, j \in F, k \in N, k' \in N \setminus \{k\}, \tag{G.3f}$$

$$f2^j_{kk'} \geq f1^j_{kk'} + e^j_{kk'} - 1, j \in F, k \in N, k' \in N \setminus \{k\}. \tag{G.3g}$$

$\sum_{k'\in N_j: O_{jk'}<O_{jk}} q_{k'}\Lambda_{k'}$ and $\sum_{O_{jk'}<O_{jk}}(1-q_{k'})(C_{k'}-W_{k'})$ can be linearized with the same approach with (G.2a)-(G.2c).

Let $\sum_{i\in N}\Delta_i = \vee_{j\in F}(\chi_{ij} \vee \xi^0_{ji} \vee \xi^1_{ji})$ represent the objective function of the optimization problem. It can be linearized as:

$$\delta_{ij} \geq \frac{1}{3}(\chi_{ij} + \xi^0_{ji} + \xi^1_{ji}), \forall i \in N, j \in F, \tag{G.4a}$$

$$\delta_{ij} \leq (\chi_{ij} + \xi^0_{ji} + \xi^1_{ji}), \forall i \in N, j \in F, \tag{G.4b}$$

$$\Delta_i \geq \frac{1}{|F|}(\sum_{j\in F} \delta_{ij}), \forall i \in N, \tag{G.4c}$$

$$\Delta_i \leq (\sum_{j\in F} \delta_{ij}), \forall i \in N. \tag{G.4d}$$

Appendix H

Parts of linearzion processing for proposed model in Chapter 7

Let $d_{k'i}^{jk\sigma} = |k'e_{kk'}^{j} - i|, j \in F, k \in N \setminus \mathcal{P}_{\Gamma}^{\sigma}, k' \in N \setminus \{k\}, i \in \mathcal{P}_{\Gamma}^{\sigma}, \sigma \in \mathbb{C}$, whose range is in $[0, |N|]$. $d_{k'i}^{jk\sigma}$ can be expressed in a linear form as:

$$d_{k'i}^{jk\sigma} \leq k'e_{kk'}^{j} - i + (1 - \delta_{k'i}^{jk\sigma})A, \tag{H.1a}$$
$$d_{k'i}^{jk\sigma} \geq k'e_{kk'}^{j} - i - (1 - \delta_{k'i}^{jk\sigma})A, \tag{H.1b}$$
$$d_{k'i}^{jk\sigma} \leq i - k'e_{kk'}^{j} + \delta_{k'i}^{jk\sigma}, \tag{H.1c}$$
$$d_{k'i}^{jk\sigma} \geq i - k'e_{kk'}^{j} - \delta_{k'i}^{jk\sigma}, \tag{H.1d}$$
$$k'e_{kk'}^{j} - i \geq (\delta_{k'i}^{jk\sigma} - 1)A, \tag{H.1e}$$
$$i - k'e_{kk'}^{j} \geq -\delta_{k'i}^{jk\sigma}A, \tag{H.1f}$$
$$\delta_{k'i}^{jk\sigma} \in \{0, 1\}. \tag{H.1g}$$

Let $c_{k'i}^{jk\sigma}, j \in F, k \in N \setminus \mathcal{P}_{\Gamma}^{\sigma}, k' \in N \setminus \{k\}, i \in \mathcal{P}_{\Gamma}^{\sigma}, \sigma \in \mathbb{C}$, denote a binary variable that is set to zero if $k'e_{kk'}^{j} = i$, i.e., $d_{k'i}^{jk\sigma} = 0$, and otherwise one. $c_{k'i}^{jk\sigma}$ can be expressed in a linear form as:

$$c_{k'i}^{jk\sigma} \geq \frac{d_{k'i}^{jk\sigma}}{|N|}, \tag{H.2a}$$
$$c_{k'i}^{jk\sigma} \leq d_{k'i}^{jk\sigma}. \tag{H.2b}$$

For each k' satisfying $e^j_{kk'} = 1$, $\phi^{jk\sigma}_{k'} = \prod_{i \in \mathcal{P}^\sigma_\Gamma} c^{jk\sigma}_{k'i} = 0$ represents that k' is in P^σ_Γ. $\sum_{k' \in N} \phi^{jk\sigma}_{k'} = 0$ represents that each k' satisfying $e^j_{kk'} = 1$ is in P^σ_Γ. α^σ_{jk} can be expressed in a linear form as:

$$\alpha^\sigma_{jk} = 1 - \sum_{k' \in N \setminus \{k\}} \phi^{jk\sigma}_{k'}, j \in F, k \in N, \sigma \in \mathbb{C}, \tag{H.3a}$$

$$\phi^{jk\sigma}_{k'} \leq c^{jk\sigma}_{k'i}, \forall i \in \mathcal{P}^\sigma_\Gamma, j \in F, k \in N \setminus \mathcal{P}^\sigma_\Gamma, k' \in N \setminus \{k\}, \sigma \in \mathbb{C}, \tag{H.3b}$$

$$\phi^{jk\sigma}_{k'} \geq \sum_{i \in \mathcal{P}^\sigma_\Gamma} c^{jk\sigma}_{k'i} - |\mathcal{P}^\sigma_\Gamma| + 1, \forall j \in F, k \in N \setminus \mathcal{P}^\sigma_\Gamma, k' \in N \setminus \{k\}, \sigma \in \mathbb{C}, \tag{H.3c}$$

$$\phi^{jk\sigma}_{k'}, c^{jk\sigma}_{k'i} \in \{0, 1\}, \forall i \in \mathcal{P}^\sigma_\Gamma, j \in F, k \in N \setminus \mathcal{P}^\sigma_\Gamma, k' \in N \setminus \{k\}, \sigma \in \mathbb{C}. \tag{H.3d}$$

Bibliography

[1] M. Alam, K. A. Shakil, and S. Sethi, "Analysis and clustering of workload in google cluster trace based on resource usage," in *2016 Int. Conf. on Comput. Sci. and Eng.* IEEE, 2016, pp. 740–747.

[2] J. J. Meza, "Large scale studies of memory, storage, and network failures in a modern data center," Ph.D. dissertation, Carnegie Mellon University, 2018.

[3] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technol. Center for Cloud Comput., Tech. Rep*, vol. 84, 2012.

[4] S. Shen, V. van Beek, and A. Iosup, "Statistical characterization of business-critical workloads hosted in cloud datacenters," in *15th IEEE/ACM Int. Symp. on Cluster, Cloud and Grid Comput.* IEEE, 2015, pp. 465–474.

[5] C. Delimitrou and C. Kozyrakis, "Cross-examination of datacenter workload modeling techniques," in *31st Int. Conf. on Distrib. Comput. Syst. Workshops*. IEEE, 2011, pp. 72–79.

[6] J. Grohmann, P. K. Nicholson, J. O. Iglesias, S. Kounev, and D. Lugones, "Monitorless: Predicting performance degradation in cloud applications with machine learning," in *20th Int. Middleware Conf.*, 2019, pp. 149–162.

[7] M. Rohr, "Workload-sensitive timing behavior analysis for fault localization in software systems," Dissertation (PhD thesis), Faculty of Engineering, Kiel University, Kiel, Germany, Jan. 2015.

[8] A. Roy, R. Ganesan, D. Dash, and S. Sarkar, "Reducing service failures by failure and workload aware load balancing in SaaS clouds," in *43rd Annu. IEEE/IFIP Conf. on Dependable Syst. and Netw. Workshop*. IEEE, 2013, pp. 1–6.

[9] R. K. Sahoo, M. S. Squillante, A. Sivasubramaniam, and Y. Zhang, "Failure data analysis of a large-scale heterogeneous server environment," in *Int. Conf. on Dependable Syst. and Netw.* IEEE, 2004, pp. 772–781.

[10] R. K. Iyer and D. J. Rossetti, "A measurement-based model for workload dependence of CPU errors," *IEEE Trans. Comput.*, vol. 100, no. 6, pp. 511–519, 1986.

[11] B. Schroeder, E. Pinheiro, and W.-D. Weber, "DRAM errors in the wild: a large-scale field study," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 37, no. 1, pp. 193–204, 2009.

[12] K. Tovletoglou, L. Mukhanov, D. S. Nikolopoulos, and G. Karakonstantis, "HaRMony: Heterogeneous-reliability memory and QoS-aware energy management on virtualized servers," in *25th Int. Conf. on Architectural Support for Program. Languag. and Operating Syst.*, 2020, pp. 575–590.

[13] H. Huang and S. Guo, "Proactive failure recovery for NFV in distributed edge computing," *IEEE Commun. Mag.*, vol. 57, no. 5, pp. 131–137, 2019.

[14] 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, 2016.

[15] 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 J. of the Commun. Soc.*, vol. 1, pp. 700–712, 2020.

[16] J. Zhang, Z. Wang, C. Peng, L. Zhang, T. Huang, and Y. Liu, "RABA: Resource-aware backup allocation for a chain of virtual network functions," in *IEEE Conf. Comput. Commun.* IEEE, 2019, pp. 1918–1926.

[17] F. He and E. Oki, "Unavailability-aware shared virtual backup allocation for middleboxes: A queueing approach," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 2, pp. 2388–2404, 2020.

[18] S. Kirkpatrick, C. D. Gelatt Jr, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[19] F. He, T. Sato, B. C. Chatterjee, T. Kurimoto, U. Shigeo, and E. Oki, "Robust optimization model for primary and backup resource allocation in cloud providers," *IEEE Trans. on Cloud Comput.*, 2021.

[20] I. A. Moschakis and H. D. Karatza, "Multi-criteria scheduling of bag-of-tasks applications on heterogeneous interlinked clouds with simulated annealing," *J. Syst. Soft.*, vol. 101, pp. 1–14, 2015.

[21] N. Su, A. Shi, C. Chen, E. Chen, and Y. Wang, "Research on virtual machine placement in the cloud based on improved simulated annealing algorithm," in *2016 World Automat. Congr.* IEEE, 2016, pp. 1–7.

[22] S. K. Addya, A. K. Turuk, B. Sahoo, M. Sarkar, and S. K. Biswash, "Simulated annealing based vm placement strategy to maximize the profit for cloud service providers," *Eng. Sci. Technol. an Int. J.*, vol. 20, no. 4, pp. 1249–1259, 2017.

[23] R. Kang, F. He, and E. Oki, "Robust virtual network function allocation in service function chains with uncertain availability schedule," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 2987–3005, 2021.

[24] R. Chandran, S. Rakesh Kumar, and N. Gayathri, "Genetic algorithm-based tabu search for optimal energy-aware allocation of data center resources," *Soft. Comput.*, vol. 24, pp. 16 705–16 718, 2020.

[25] N. Téllez, M. Jimeno, A. Salazar, and E. Nino-Ruiz, "A tabu search method for load balancing in fog computing," *Int. J. Artif. Intell.*, vol. 16, no. 2, pp. 1–30, 2018.

[26] M. Mitchell, *An introduction to genetic algorithms*. MIT press, 1998.

[27] R. Fujita, F. He, T. Sato, and E. Oki, "Optimization of backup resource assignment for middleboxes," in *IEEE 20th Int. Conf. on High Perform. Switch. and Rout.* IEEE, 2019, pp. 1–6.

[28] 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.

[29] ——, "Designing optimal middlebox recovery schemes with performance guarantees," *IEEE J. Sel. Areas Commun.*, vol. 36, no. 10, pp. 2373–2383, 2018.

[30] F. He, T. Sato, and E. Oki, "Optimization model for backup resource allocation in middleboxes with importance," *IEEE/ACM Trans. Netw.*, vol. 27, no. 4, pp. 1742–1755, 2019.

[31] L. Qu, C. Assi, K. Shaban, and M. J. Khabbaz, "A reliability-aware network service chain provisioning with delay guarantees in NFV-enabled enterprise datacenter networks," *IEEE Trans. Netw. Serv. Manag.*, vol. 14, no. 3, pp. 554–568, 2017.

[32] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "Reliability aware service placement using a viterbi-based algorithm," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 1, pp. 622–636, 2019.

[33] M. Zhu, F. He, and E. Oki, "Optimization model for multiple backup resource allocation with workload-dependent failure probability," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 3733–3752, 2021.

[34] 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 Trans. Netw. Serv. Manag.*, Jul. 2020.

[35] R. Kang, M. Zhu, F. He, T. Sato, and E. Oki, "Design of scheduler plugins for reliable function allocation in Kubernetes," in *Conf. Design of Reliab. Comm. Netw.* IEEE, 2021.

[36] R. Kang, F. He, and E. Oki, "Resilient virtual network function allocation with diversity and fault tolerance considering dynamic requests," in *2022 IEEE/IFIP Netw. Oper. Manage. Symp.*, Apr. 2022, pp. 1–9.

[37] M. Zhu, F. He, and E. Oki, "Optimal primary and backup resource allocation with workload-dependent failure probability," in *2020 Int. Conf. on Inf. and Commun. Technol. Convergence.* IEEE, 2020, pp. 1–6.

[38] The Kubernetes Authors, "Custom resources," https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources.

[39] Z. Lv and L. Qiao, "Optimization of collaborative resource allocation for mobile edge computing," *Comput. Commun.*, vol. 161, pp. 19–27, 2020.

[40] F. He and E. Oki, "Backup allocation model with probabilistic protection for virtual networks against multiple facility node failures," *IEEE Trans. Netw. Serv. Manag.*, vol. 18, no. 3, pp. 2943–2959, 2021.

[41] S. Wang, M. Chen, X. Liu, C. Yin, S. Cui, and H. V. Poor, "A machine learning approach for task and resource allocation in mobile-edge computing-based networks," *IEEE Internet Things J.*, vol. 8, no. 3, pp. 1358–1372, 2020.

[42] C. Singhal and S. De, *Resource allocation in next-generation broadband wireless access networks.* IGI Global, 2017.

[43] P. Vamvakas, E. E. Tsiropoulou, and S. Papavassiliou, "Dynamic spectrum management in 5g wireless networks: A real-life modeling approach," in *IEEE Conf. Comput. Commun.*, 2019, pp. 2134–2142.

[44] 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 *2018 IEEE Int. Conf. on Commun.* IEEE, 2018, pp. 1–6.

[45] A. Alleg, T. Ahmed, M. Mosbah, R. Riggio, and R. Boutaba, "Delay-aware vnf placement and chaining based on a flexible resource allocation approach," in *13th Int. Conf. on Netw. and Serv. Manage.* IEEE, 2017, pp. 1–7.

[46] M. Liu, C. Li, and T. Li, "Understanding the impact of vcpu scheduling on dvfs-based power management in virtualized cloud environment," in *IEEE 22nd Int. Symp. on Model., Anal. & Simul. of Comput. and Telecommun. Syst.* IEEE, 2014, pp. 295–304.

[47] C. Natalino, F. Coelho, G. Lacerda, A. Braga, L. Wosinska, and P. Monti, "A proactive restoration strategy for optical cloud networks based on failure predictions," in *20th Int. Conf. on Transparent Opt. Netw.* IEEE, 2018, pp. 1–5.

[48] I. S. Moreno, P. Garraghan, P. Townend, and J. Xu, "Analysis, modeling and simulation of workload patterns in a large-scale utility cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 208–221, 2014.

[49] P. Garraghan, I. S. Moreno, P. Townend, and J. Xu, "An analysis of failure-related energy waste in a large-scale cloud environment," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 2, pp. 166–180, 2014.

[50] Y. Chen, C. Chang, Wan, and P. Wang, "Dynamic replication scheduling for cloud datacenters based on workload statistics," in *2019 IEEE Int. Conf. on Dependable, Autonomic and Secure Comp.* IEEE, 2019, pp. 1093–1096.

[51] M. T. Beck, J. F. Botero, and K. Samelin, "Resilient allocation of service function chains," in *2016 IEEE Conf. on Netw. Function Virtualization and Softw. Defined Netw.* IEEE, 2016, pp. 128–133.

[52] B. Han, V. Gopalakrishnan, G. Kathirvel, and A. Shaikh, "On the resiliency of virtual network functions," *IEEE Commun. Mag.*, vol. 55, no. 7, pp. 152–157, 2017.

[53] S. Herker, X. An, W. Kiess, S. Beker, and A. Kirstaedter, "Data-center architecture impacts on virtualized network functions service chain embedding with high availability requirements," in *2015 IEEE Global Commun. Conf. Workshops*. IEEE, 2015, pp. 1–7.

[54] J. Fan, Z. Ye, C. Guan, X. Gao, K. Ren, and C. Qiao, "Grep: Guaranteeing reliability with enhanced protection in NFV," in *ACM SIGCOMM Workshop on Hot Topics in Middleboxes and Netw. Function Virtualization*, 2015, pp. 13–18.

[55] K. Sharma and K. R. Singh, "Online data back-up and disaster recovery techniques in cloud computing: A review," *Int. Journal of Eng. and Innovative Technol.*, vol. 2, no. 5, pp. 249–254, 2012.

[56] F. He, T. Sato, and E. Oki, "Backup resource allocation model for virtual networks with probabilistic protection against multiple facility node failures," in *15th Int. Conf. on the Des. of Reliab. Commun. Netw.* IEEE, 2019, pp. 37–42.

[57] R. Kang, F. He, and E. Oki, "Fault-tolerant resource allocation model for service function chains with joint diversity and redundancy," *Comput. Netw.*, vol. 217, p. 109287, 2022.

[58] W. Ding, H. Yu, and S. Luo, "Enhancing the reliability of services in NFV with the cost-efficient redundancy scheme," in *2017 IEEE Int. Conf. on Commun.* IEEE, 2017, pp. 1–6.

[59] M. Karimzadeh-Farshbafan, V. Shah-Mansouri, and D. Niyato, "A dynamic reliability-aware service placement for network function virtualization (NFV)," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 2, pp. 318–333, 2020.

[60] L. Sun, J. An, Y. Yang, and M. Zeng, "Recovery strategies for service composition in dynamic network," in *2011 Int. Conf. on Cloud and Serv. Comput.* IEEE, 2011, pp. 60–64.

[61] S.-Y. Song and F. J. Lin, "Dynamic fault management in service function chaining," in *IEEE 44th Annu. Comput., Softw., and Appl. Conf.*, 2020, pp. 1477–1482.

[62] M. Zhu, R. Kang, F. He, and E. Oki, "Implementation of backup resource management controller for reliable function allocation in kubernetes," in *IEEE 7th Int. Conf. on Netw. Softwarization*, Jun. 2021, pp. 360–362.

[63] S. V. Amari and G. Dill, "Redundancy optimization problem with warm-standby redundancy," in *Annu. Reliability and Maintainability Symp.* IEEE, 2010, pp. 1–6.

[64] Red Hat, "Stateful vs stateless," https://www.redhat.com/en/topics/cloud-native-apps/stateful-vs-stateless, accessed Nov 9, 2021.

[65] Y. Zhang, F. He, and E. Oki, "Availability-aware service chain provisioning with sub-chain-enabled coordinated protection," in *IEEE/IFIP Int. Symp. on Integr. Netw.* IEEE, 2021, pp. 1–6.

[66] P. M. Mohan and M. Gurusamy, "Resilient vnf placement for service chain embedding in diversified 5g network slices," in *2019 IEEE Global Commun. Conf.* IEEE, 2019, pp. 1–6.

[67] R. Kang, F. He, and E. Oki, "Resilient resource allocation model in service function chains with diversity and redundancy," in *2021 IEEE Int. Conf. on Commun.*, Jun. 2021, pp. 1–6.

[68] A. Zhou, S. Wang, C.-H. Hsu, M. H. Kim, and K.-s. Wong, "Virtual machine placement with (m, n)-fault tolerance in cloud data center," *Cluster Comput.*, vol. 22, no. 5, pp. 11 619–11 631, 2019.

[69] R. Fujita, F. He, T. Sato, and E. Oki, "Shared backup resource assignment for middleboxes," *Opt. Switch. and Netw.*, vol. 37, p. 100569, 2020.

[70] T. Hu, Z. Guo, J. Zhang, and J. Lan, "Adaptive slave controller assignment for fault-tolerant control plane in software-defined networking," in *2018 IEEE Int. Conf. on Commun.* IEEE, 2018, pp. 1–6.

[71] F. He and E. Oki, "Load balancing model against multiple controller failures in software defined networks," in *2020 IEEE Int. Conf. on Commun.* IEEE, 2020, pp. 1–6.

[72] H. Moens and F. De Turck, "VNF-P: A model for efficient placement of virtualized network functions," in *10th Int. Conf. on Netw. and Serv. Manage.* IEEE, 2014, pp. 418–423.

[73] N. T. Hieu, M. Di Francesco, and A. Y. Jääski, "A virtual machine placement algorithm for balanced resource utilization in cloud data centers," in *IEEE 7th Int. Conf. on Cloud Comput.* IEEE, 2014, pp. 474–481.

[74] I. M. Kamrul and E. Oki, "PSO: preventive start-time optimization of OSPF link weights to counter network failure," *IEEE Commun. Lett.*, vol. 14, no. 6, pp. 581–583, 2010.

[75] X. Su, B. Liu, X. Zhu, J. Zeng, and C. Xiao, "Low-complexity iterative approximated water-filling based power allocation in an ultra-dense network," *Entropy*, vol. 18, no. 5, p. 158, 2016.

[76] C. Xing, Y. Jing, S. Wang, S. Ma, and H. V. Poor, "New viewpoint and algorithms for water-filling solutions in wireless communications," *IEEE Trans. Signal Process.*, vol. 68, pp. 1618–1634, 2020.

[77] A. Zamani, B. Bakhshi, and S. Sharifian, "An efficient load balancing approach for service function chain mapping," *Comput. Electr. Eng.*, vol. 90, p. 106890, 2021.

[78] P. He, M. Li, L. Zhao, B. Venkatesh, and H. Li, "Water-filling exact solutions for load balancing of smart power grid systems," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 1397–1407, 2016.

[79] J. Cheng, S.-H. Tseng, and A. Tang, "Worst-case latency performance of load balancing through distributed waterfilling algorithm," in *53rd Annu. Conf. on Inf. Sci. and Syst.* IEEE, 2019, pp. 1–6.

[80] D. Zhang, Z. Zheng, X. Lin, X. Chen, and C. Wu, "Dynamic backup sharing scheme of service function chains in NFV," *China Commun.*, vol. 19, no. 5, pp. 178–190, 2022.

[81] B. Yi, X. Wang, M. Huang, S. K. Das, and K. Li, "Fairness-aware vnf sharing and rate coordination for high efficient service scheduling," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 4597–4611, 2022.

[82] C. Pham, N. H. Tran, S. Ren, W. Saad, and C. S. Hong, "Traffic-aware and energy-efficient vnf placement for service chaining: Joint sampling and matching approach," *IEEE Trans. Serv. Comput.*, vol. 13, no. 1, pp. 172–185, 2017.

[83] F. Malandrino, C. F. Chiasserini, G. Einziger, and G. Scalosub, "Reducing service deployment cost through vnf sharing," *IEEE/ACM Trans. Netw.*, vol. 27, no. 6, pp. 2363–2376, 2019.

[84] R. Kang, M. Zhu, F. He, and E. Oki, "Implementation of virtual network function allocation with diversity and redundancy in kubernetes," in *2021 IFIP Network. Conf.*, 2021, pp. 1–2.

[85] J. Chu and C.-T. Lea, "Optimal link weights for ip-based networks supporting hose-model vpns," *IEEE/ACM Trans. Netw.*, vol. 17, no. 3, pp. 778–788, 2008.

[86] M. Ito, F. He, and E. Oki, "Robust optimization for probabilistic protection with primary and backup allocations under uncertain demands," in *17th Int. Conf. on the Des. of Reliab. Commun. Netw.* IEEE, 2021, pp. 1–6.

[87] M. A. Khoshkholghi, A. Abdullah, R. Latip, S. Subramaniam, and M. Othman, "Disaster recovery in cloud computing: A survey," *Stud. Comput. Intell.*, vol. 7, no. 4, p. 39, 2014.

[88] M. Zhu, F. He, and E. Oki, "Optimization model for primary and backup resource allocation with workload-dependent failure probability," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 1, pp. 452–471, 2022.

[89] VMware, "Vmware vcenter site recovery manager performance and best practices for performance," https://www.vmware.com/pdf/Perf_SiteRecoveryManager10_Best-Practices.pdf, 2009, accessed Mar. 12, 2023.

[90] A. Lenk and S. Tai, "Cloud standby: disaster recovery of distributed systems in the cloud," in *Eur. Conf. on Service-Oriented and Cloud Comput.* Springer, 2014, pp. 32–46.

[91] R. G. Gallager *et al.*, "Energy limited channels: Coding, multiaccess, and spread spectrum," *Tech. Rep. LIDS-P-1714*, Nov. 1987.

[92] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: Fair allocation of multiple resource types." in *8th USENIX Symp. on Networked Syst. Des. and Implementation*, vol. 11, no. 2011, 2011, pp. 24–24.

[93] IBM Knowledge Center, "Introducing IBM ILOG CPLEX Optimization Studio v12.9.0.0," https://www.ibm.com/support/knowledgecenter/SSSA5P_12.9.0/ilog.odms.studio.help/Optimization_Studio/topics/COS_home.html, accessed June 24, 2019.

[94] T. Segaran, *Programming collective intelligence: building smart web 2.0 applications.* " O'Reilly Media, Inc.", 2007.

[95] M. Zhu, F. He, and E. Oki, "Optimal multiple backup resource allocation with workload-dependent failure probability," in *2020 IEEE Global Commun. Conf.* IEEE, 2020, pp. 1–6.

[96] 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 *IEEE 7th Int. Conf. on Cloud Netw.* IEEE, 2018, pp. 1–3.

[97] Z. Wang, J. Zhang, T. Huang, and Y. Liu, "A clustering-based approach for virtual network function mapping and assigning," in *IEEE/ACM 25th Int. Symp. on Qual. of Serv.* IEEE, 2017, pp. 1–5.

[98] B. Shen, R. Sundaram, A. Russell, S. Aiyar, K. Gupta, A. Nagpal, A. Ramesh, and H. Shukla, "High availability for vm placement and a stochastic model for multiple knapsack," in *26th Int. Conf. on Comput. Commun. and Netw.* IEEE, 2017, pp. 1–9.

[99] R. Birke, I. Giurgiu, L. Y. Chen, D. Wiesmann, and T. Engbersen, "Failure analysis of virtual and physical machines: patterns, causes and characteristics," in *44th Annu. IEEE/IFIP Int. Conf. on Dependable Syst. and Netw.* IEEE, 2014, pp. 1–12.

[100] J. Wilkes, "Yet more Google compute cluster trace data," Google research blog, Apr. 2020, posted at https://ai.googleblog.com/2020/04/yet-more-google-compute-cluster-trace.html.

[101] A. Hmaity, M. Savi, F. Musumeci, M. Tornatore, and A. Pattavina, "Protection strategies for virtual network functions placement and service chains provisioning," *Networks*, vol. 70, no. 4, pp. 373–387, 2017.

[102] M. Casazza, M. Bouet, and S. Secci, "Availability-driven NFV orchestration," *Comput. Netw.*, vol. 155, pp. 47–61, 2019.

[103] F. He, T. Sato, and E. Oki, "Survivable virtual network embedding model with shared protection over elastic optical network," in *IEEE 8th Int. Conf. on Cloud Netw.* IEEE, 2019, pp. 1–3.

[104] T. Kimura, K. Ishibashi, T. Mori, H. Sawada, T. Toyono, K. Nishimatsu, A. Watanabe, A. Shimoda, and K. Shiomoto, "Spatio-temporal factorization of log data for understanding network events," in *IEEE Conf. Comput. Commun.* IEEE, 2014, pp. 610–618.

[105] S. Ferdousi, F. Dikbiyik, M. F. Habib, M. Tornatore, and B. Mukherjee, "Disaster-aware datacenter placement and dynamic content management in cloud networks," *J. Opt. Commun. Netw.*, vol. 7, no. 7, pp. 681–694, 2015.

[106] R. Kang, F. He, T. Sato, and E. Oki, "Virtual network function allocation to maximize continuous available time of service function chains," in *IEEE 8th Int. Conf. on Cloud Netw.*, Nov. 2019, pp. 1–6.

[107] A. Gember-Jacobson, R. Viswanathan, C. Prakash, R. Grandl, J. Khalid, S. Das, and A. Akella, "Opennf: Enabling innovation in network function control," in *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4. ACM, 2014, pp. 163–174.

[108] The Kubernetes Authors, "Kubernetes," https://kubernetes.io/, accessed Dec. 16, 2020.

[109] Prometheus Authors, "Prometheus - monitoring system & time series database," https://prometheus.io, accessed Jan. 9, 2021.

[110] The Kubernetes Authors, "Init containers," https://kubernetes.io/docs/concepts/workloads/pods/init-containers, accessed Mar. 18, 2021.

[111] Y. Wang, L. Zhang, P. Yu, K. Chen, X. Qiu, L. Meng, M. Kadoch, and M. Cheriet, "Reliability-oriented and resource-efficient service function chain construction and backup," *IEEE Trans. Netw. Serv. Manag.*, 2020.

[112] Network Functions Virtualisation ETSI Industry Specification Group (ISG), "Network functions virtualisation (NFV); pre-deployment testing; report on validation of NFV environments and services," Network Functions Virtualisation ETSI Industry Specification Group (ISG), Standard, 2016. [Online]. Available: https://www.etsi.org/deliver/etsi_gs/NFV-TST/001_099/001/01.01.01_60/gs_NFV-TST001v010101p.pdf

[113] M. Ito, F. He, and E. Oki, "Robust optimization for probabilistic protection with multiple types of resources in cloud," in *IEEE 9th Int. Conf. on Cloud Netw.* IEEE, 2020, pp. 1–6.

[114] M. Zhu, F. He, and E. Oki, "Load balancing model under multiple failures with workload-dependent failure probability," in *17th Int. Conf. on the Des. of Reliab. Commun. Netw.*, Apr. 2021, pp. 1–6.

[115] ——, "Resource allocation model against multiple failures with workload-dependent failure probability," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1098–1116, 2022.

[116] J. Kleinberg and E. Tardos, *Algorithm design*. Pearson Education India, 2006.

[117] Z. Fan, H. Shen, Y. Wu, and Y. Li, "Simulated-annealing load balancing for resource allocation in cloud environments," in *2013 Int. Conf. on Parallel and Distrib. Comput., Appl. and Technol.* IEEE, 2013, pp. 1–6.

[118] M. Zhu, F. He, and E. Oki, "Robust function deployment against uncertain recovery time with Workload-Dependent failure probability," in *IEEE 19th Annu. Consum. Commun. & Netw. Conf.*, Jan. 2022.

[119] M. Zhu and E. Oki, "Robust function deployment against uncertain recovery time in different protection types with workload-dependent failure probability," *Comput. Netw.*, vol. 231, p. 109826, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128623002712

[120] R. Kang, M. Zhu, and E. Oki, "Implementation of service function chain deployment with allocation models in kubernetes," in *IEEE Conf. Comput. Commun. Workshops*, 2022, pp. 1–2.

[121] The OVN community, "OVN, Open Virtual Network :: OVN project documentation website," https://www.ovn.org/en/, accessed Mar. 12, 2023.

[122] The CNI Authors, "The Container Network Interface," https://www.cni.dev, accessed Mar. 12, 2023.

[123] H. Naser and H. T. Mouftah, "Availability analysis and simulation of mesh restoration networks," in *9th Int. Symp. on Comput. and Commun.*, vol. 2. IEEE, 2004, pp. 779–785.

[124] S. Garg, J. Singh, and D. Singh, "Availability and maintenance scheduling of a repairable block-board manufacturing system," *Int. J. Reliab. Saf.*, vol. 4, no. 1, pp. 104–118, 2010.

[125] M. Zhu and E. Oki, "Resource allocation in multiple backup modes under reliability guarantee with workload-dependent failure probability," in *IEEE 11th Int. Conf. on Cloud Netw.*, 2022, pp. 40–48.

[126] A. Juttner, I. Szabó, and A. Szentesi, "On bandwidth efficiency of the hose resource management model in virtual private networks," in *22nd Annu. Joint Conf. of the IEEE Comput. and Commun. Societies*, vol. 1. IEEE, 2003, pp. 386–395.

[127] C. Çalışkan, "On a capacity scaling algorithm for the constrained maximum flow problem," *Networks*, vol. 53, no. 3, pp. 229–230, 2009.

[128] NetworkX Developers, "Networkx," https://networkx.org/documentation/stable/reference/algorithms/generated/networkx.algorithms.flow.capacity_scaling.html/, accessed Nov. 26, 2022.

[129] K. Hwang, Y. Shi, and X. Bai, "Scale-out vs. scale-up techniques for cloud performance and productivity," in *IEEE 6th Int. Conf. on Cloud Comput. Technol. and Sci.*, 2014, pp. 763–768.

[130] E. Radhika and G. Sudha Sadasivam, "A review on prediction based autoscaling techniques for heterogeneous applications in cloud environment," *Mater. Today: Proc.*, vol. 45, pp. 2793–2800, 2021, int. Conf. on Advances in Materials Research - 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214785320394657

[131] A. Mazidi, M. Golsorkhtabaramiri, and M. Yadollahzadeh Tabari, "An autonomic risk-and penalty-aware resource allocation with probabilistic resource scaling mechanism for multilayer cloud resource provisioning," *Int. J. Commun. Syst.*, vol. 33, no. 7, p. e4334, 2020.

[132] M. Zhu, R. Kang, and E. Oki, "Implementation of real-time function deployment with resource migration in kubernetes," in *IEEE/IFIP Netw. Oper. and Manage. Symp.*, 2022, pp. 1–3.

[133] The Kubernetes Authors, "Controllers," https://kubernetes.io/docs/concepts/architecture/controller/, accessed Jan. 18, 2022.

[134] Red Hat, Inc., "Operator SDK," https://sdk.operatorframework.io.

[135] Google, "The Go programming language," https://golang.org.

[136] R. M. Karp, "Reducibility among combinatorial problems," in *Complexity of computer computations*. Springer, 1972, pp. 85–103.

Publication List

Journal Papers

1. M. Zhu and E. Oki, "Robust Function Deployment against Uncertain Recovery Time in Different Protection Types with Workload-Dependent Failure Probability," Comput. Netw., vol. 231, no. 109826, pp. 1-15, Jul. 2023.

2. M. Zhu, F. He, and E. Oki, "Resource Allocation Model Against Multiple Failures with Workload-Dependent Failure Probability," IEEE Trans. Netw. and Serv. Manag., vol. 19, no. 2, pp. 1098-1116, Jun. 2022.

3. M. Zhu, F. He, and E. Oki, "Optimization Model for Primary and Backup Resource Allocation with Workload-Dependent Failure Probability," IEEE Trans. Netw. and Serv. Manag., vol. 19, no. 1, pp. 452-471, Mar. 2022.

4. M. Zhu, F. He, and E. Oki, "Optimization Model for Multiple Backup Resource Allocation with Workload-Dependent Failure Probability," IEEE Trans. Netw. and Serv. Manag., vol. 18, no. 3, pp. 3733-3752, Sep. 2021.

International Conference Papers

1. M. Zhu and E. Oki, "Resource Allocation in Multiple Backup Modes under Reliability Guarantee with Workload-Dependent Failure Probability," IEEE Global Internet (GI) Symp. 2022, Nov. 2022.

2. R. Kang, M. Zhu, and E. Oki, "Implementation of Service Function Chain Deployment with Allocation Models in Kubernetes," IEEE Conf. Comput. Commun. Workshops, May 2022.

3. M. Zhu, R. Kang, and E. Oki, "Implementation of Real-time Function Deployment with Resource Migration in Kubernetes," IEEE/IFIP Netw. Oper. and Manage. Symp., Apr. 2022.

4. M. Zhu, F. He, and E. Oki, "Robust Function Deployment against Uncertain Recovery Time with Workload-Dependent Failure Probability," IEEE Consum. Commun. and Netw. Conf., Jan. 2022.

5. R. Kang, M. Zhu, F. He, and E. Oki, "Implementation of Virtual Network Function Allocation with Diversity and Redundancy in Kubernetes," IFIP Network., Jun. 2021.

6. M. Zhu, R. Kang, F. He, and E. Oki, "Implementation of Backup Resource Management Controller for Reliable Function Allocation in Kubernetes," 7th IEEE Int. Conf. on Netw. Softwarization, Jun. 2021.

7. M. Zhu, F. He, and E. Oki, "Load Balancing Model under Multiple Failures with Workload-Dependent Failure Probability," 17th Int. Conf. on the Des. of Reliab. Commun. Netw., Apr. 2021.

8. R. Kang, M. Zhu, F. He, T. Sato, and E. Oki, "Design of Scheduler Plugins for Reliable Function Allocation in Kubernetes," 17th Int. Conf. on the Des. of Reliab. Commun. Netw., Apr. 2021.

9. M. Zhu, F. He, and E. Oki, "Multiple Backup Resource Allocation with Workload-Dependent Failure Probability," IEEE Global Commun. Conf., Dec. 2020.

10. M. Zhu, F. He, and E. Oki, "Optimal Primary and Backup Resource Allocation with Workload-Dependent Failure Probability," The 11th Int. Conf. on Inf. and Commun. Technol. on Convergence, Oct. 2020.

Awards

1. Japan Society for the Promotion of Science Research Fellowships for Young Scientists (DC1) from 2022 to 2025.
2. Travel grant at ACM SIGCOMM in 2022.
3. Graduate representative of Graduate School of Informatics, Kyoto University, 2021.
4. Aeon Scholarship from 2019-2021.