

Article

# Optimization-Based Resource Management Algorithms with Considerations of Client Satisfaction and High Availability in Elastic 5G Network Slices

Chiu-Han Hsiao <sup>1,\*†‡</sup> , Frank Yeong-Sung Lin <sup>2‡</sup>, Evana Szu-Han Fang <sup>2</sup> , Yu-Fang Chen <sup>2</sup>, Yean-Fu Wen <sup>3</sup>, Yennun Huang <sup>1</sup>, Yang-Che Su <sup>1</sup>, Ya-Syuan Wu <sup>2</sup> and Hsin-Yi Kuo <sup>1</sup>

<sup>1</sup> Research Center for Information Technology Innovation, Academia Sinica, Taipei 115, Taiwan; yennunhuang@citi.sinica.edu.tw (Y.H.); R06725052@ntu.edu.tw (Y.-C.S.); R05725060@ntu.edu.tw (H.-Y.K.)

<sup>2</sup> Department of Information Management, National Taiwan University, Taipei 10617, Taiwan; flin@ntu.edu.tw (F.Y.-S.L.); D06725003@ntu.edu.tw (E.S.-H.F.); D09725003@ntu.edu.tw (Y.-F.C.); R06725016@ntu.edu.tw (Y.-S.W.)

<sup>3</sup> Graduate Institute of Information Management, National Taipei University, New Taipei City 23799, Taiwan; yeanfu@mail.ntpu.edu.tw

\* Correspondence: chiuhanhsiao@citi.sinica.edu.tw

† Current address: 128 Academia Road, Section 2, Nankang, Taipei 115, Taiwan.

‡ These authors contributed equally to this work.



**Citation:** Hsiao, C.-H.; Lin, F.Y.-S.; Fang, E.S.-Z.; Chen, Y.-F.; Wen, Y.-F.; Huang, Y.; Su, Y.-C.; Wu, Y.-S.; Kuo, H.-Y. Optimization-Based Resource Management Algorithms with Considerations of Client Satisfaction and High Availability in Elastic 5G Network Slices. *Sensors* **2021**, *21*, 1882. <https://doi.org/10.3390/s21051882>

Academic Editor:  
Peter Han Joo Chong

Received: 2 February 2021  
Accepted: 1 March 2021  
Published: 8 March 2021

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.



**Copyright:** © 2021 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

**Abstract:** A combined edge and core cloud computing environment is a novel solution in 5G network slices. The clients' high availability requirement is a challenge because it limits the possible admission control in front of the edge cloud. This work proposes an orchestrator with a mathematical programming model in a global viewpoint to solve resource management problems and satisfying the clients' high availability requirements. The proposed Lagrangian relaxation-based approach is adopted to solve the problems at a near-optimal level for increasing the system revenue. A promising and straightforward resource management approach and several experimental cases are used to evaluate the efficiency and effectiveness. Preliminary results are presented as performance evaluations to verify the proposed approach's suitability for edge and core cloud computing environments. The proposed orchestrator significantly enables the network slicing services and efficiently enhances the clients' satisfaction of high availability.

**Keywords:** network slicing; resource allocation; load balancing; admission control; high availability; Lagrangian relaxation (LR)

## 1. Introduction

Novel applications of 5G services are cataloged with three kinds of quality of service (QoS) features: massive machine-type communication (mMTC), enhanced mobile broadband (eMBB), and ultra-reliable low-latency communication (uRLLC) in 5G cellular networks [1]. Two system architectures have been proposed for ensuring the scalability and flexibility of resource allocation and scheduling for the diverse QoS requirements. The first highly acceptable approach is the cloud radio access network (C-RAN). The baseband processing and networking functions are virtually centralized in a resource pool for resource scalability and flexibility [2]. The other one is mobile edge computing (MEC). It supports interactive and real-time applications associated with nearby cloud hosts to obtain the required latency for addressing urgent and distributed requirements [3]. To implement C-RAN and MEC, software-defined networking (SDN) and network function virtualization (NFV) have been integrated to develop network slicing technologies in 5G [4]. Network slices are end-to-end (E2E) mutually separate sets of programmable infrastructure resources with independent control. A slice is a logical network adapted for particular applications regarding the diverse QoS requirements [1].

However, SDN and NFV have paved the way for employing the slicing concept. The network slice-as-a-service (NSaaS) strategy has several problems and challenges [4,5]. Furthermore, the failure of virtualized network functions (VNFs) may impact the QoS for service provisioning of the control plane (e.g., MME) and the data plane (e.g., S-GW or PDN-GWs), respectively [4]. The related factors would be addressed and collected to design an orchestrator with the resource management efficiently and effectively for the VNFs, such as resource allocation, load balancing, and high availability (HA) for building up a high performance and robust slicing network.

Furthermore, HA is considered a high stringency requirement for providing a sustainable business with emergency response applications to support a reliable performance with zero downtime and costs. Some technologies can be adopted by the redundant array of independent discs, the replication of nodes, or master-slave database redundancy to offer data protection in an HA-equipped system [6–8]. From the network operator perspective, 5G network slices' resources are limited for simultaneous profit maximization. Load balancing is typically adopted by an orchestrator to achieve resource planning and provisioning and enhance performance [4,9].

In this paper, the efficient elastic mechanisms of resource management in virtual machines (VMs) with VNFs are proposed to integrate admission control, load balancing, resource allocation, and HA arrangement in 5G network slices. The system architecture is shown in Figure 1. An integer programming problem is formulated to maximize system utilization, subject to the quality of services, resource requirements, and HA constraints. The resource allocation problem is combined with knapsack and bin-packing problems. Generally, the VNFs or called tasks with various amounts of resource requirements emulated as VMs should be packed into a finite number of servers. A few servers in active status and then stepping forward servers by increasing demands are possible to reduce the operating costs in a bin-packing problem. A set of tasks for applications with diverse weights and values is selected for inclusion to remain within the resource limit and reach the maximum benefit, a well-known knapsack problem. Furthermore, the knapsack problem is the aforementioned combinatorial nondeterministic polynomial time (NP) hard problem. One possible solution is adopted by the Lagrangian relaxation (LR) method efficiently and effectively. Near-optimal solutions provide practical methods to overcome the bottleneck within limited computing resources of slices. The virtualization, parallel computing, and mathematical programming techniques are also developed in this paper to make near-optimal decisions from an NSaaS service provider's perspective. The proposed resource management algorithms ensure that computing resources are equitably distributed in the slices of cloudlets and core clouds. The computation in the edge and core clouds is performed to achieve maximum resource utilization with HA and minimal cost. Moreover, the admission control scheme with the additional HA requirements is devised to admit the maximum number of jobs. Each job requires various types of resources.

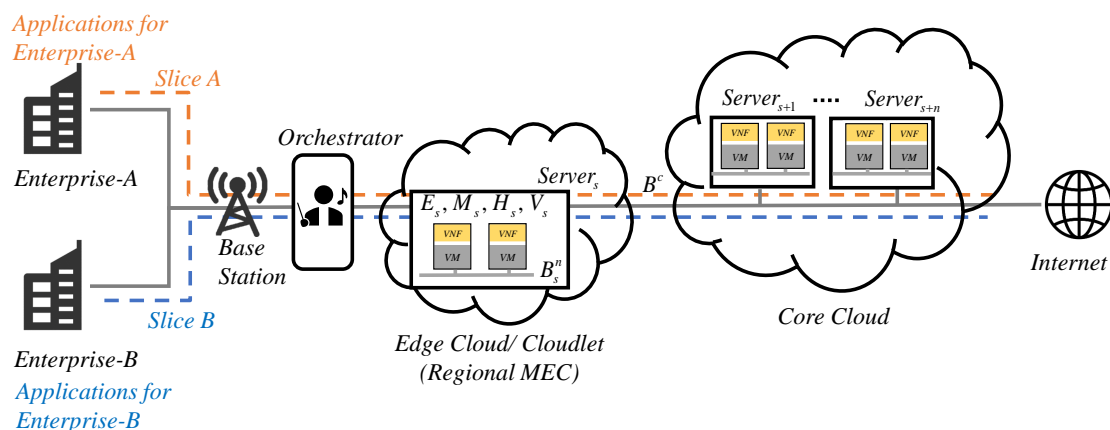


Figure 1. Network topology in slicing networks.

The remainder of this paper is organized as follows: Section 2 reviews related works. Section 3 introduces the mathematical model and problem description. The LR-based solution approach is developed, as shown in Section 4. Section 5 presents computational experiments. Conclusions are drawn in Section 6.

## 2. Literature Review

A cloud computing environment in 5G network slices supports share-based services in a pay-as-you-go approach. It provides operators with a flexible architecture from the perspective of a network operator. Resource management with complex on-demand traffic in the share-based cloud computing environment is a considerable challenge. Inappropriately designed resource management solutions might lead to the network's increasing costs. This is related to unsatisfactory quality of services (QoS) and system reliability regarding the penalty of user experience and call-blocking probability due to the virtualized network function failure. Tables 1–3 compare the resource management problems investigated with those investigated in related studies in terms of (i) resource allocation, (ii) load balancing, and (iii) admission control.

### 2.1. Resource Allocation

For the maximization of the benefits of the cloud system and service quality enhancement, Liu et al. [10] developed a joint multi-resource allocation model based on a semi-Markov decision process (SMDP). They solved the optimization problem using linear programming to attain wireless resource allocation and near-optimal decisions in cloud computing. However, cloud service providers fulfill individual requirements to achieve a win-win situation for both parties in terms of computational efficiency, budget balance, and truthfulness. Jin et al. [11] designed an incentive-compatible auction mechanism to appropriately allocate matching among cloud resources and user demands. In this auction model, the buyers are mobile devices, and the sellers are cloudlets. The nearest control center can adopt the role of the auctioneer to reduce transmission costs and latency. Furthermore, by assuming that all cloudlets can provide the same resources with distinct rewards, Liu and Fan [12] presented a two-stage optimization strategy to achieve optimal cloudlet selection in a multi-cloudlet environment and optimal resource allocation in response to the request in the cloudlet. Studies on resource allocation in a cloud computing environment are shown in Table 1. Resource allocation and scheduling algorithms have been proposed in numerous research areas, such as transportation management, industrial management, operational research, computer science, and particularly in real-time operating systems [13,14]. For instance, the earliest deadline first (EDF) is a dynamic scheduling algorithm used in real-time operating systems to allocate computing resources in central processing units (CPUs) using a priority queue. The queue searches for the task with the closest deadline; if a job cannot be completed within its time frame, the operating system must release the job. The main idea is to maximize resource use for several competing interests while balancing the load. A weighted priority for each data flow can be assigned inversely proportional to the respective flow's anticipated resource consumption [15,16]. Ding et al. [17] proposed a Linux scheduling policy with a priority queue rather than a first-in-first-out (FIFO) queue to improve kernel-based virtual machine performance. Zhao et al. [18] proposed online VM placement algorithms to cost efficiently allocate resources to VMs to increase revenues in a managed server farm. First-fit (FF), FF migration (FFM), least reliable first (LRF), and decreased density greedy (DDG) algorithms are packing strategies relevant to task optimization for achieving desirable performance.

Based on the analysis of the previous studies, this paper regards offloading tasks as virtual machines with distinct rewards with various levels of demands in the cloudlets and core clouds. The Lagrangian relaxation-based approach is proposed to maximize system revenue, subject to constraints such as computing capacity, assignments, and quality of service requirements. The approach is adopted by considering finding the optimal solutions other than SMDP within limited states or constraints. The objective is to maximize the total

value of tasks using proposed heuristics in polynomial time. It also appropriately rearranges resources using the next-fit algorithm to allocate tasks and satisfy application requirements.

**Table 1.** Resource allocation comparisons with existing methods. SMDP, semi-Markov decision process; EDF, earliest deadline first; FF, First-fit; FFM, FF migration; LRF, least reliable first; DDG, decreased density greedy.

Classification	Objective	Strategy	Related Studies	Proposed Methods
Resource allocation	Maximizing revenue and satisfying QoS	SMDP; auction; EDF; FIFO; FF; FFM; LRF; DDG	[10–18]	LR and next-fit are adopted

#### Comparison with related studies

This paper regards offloading tasks as virtual machines with distinct rewards with various levels of demands in the cloudlet and core cloud environment. The Lagrangian relaxation-based approach is proposed to maximize system revenue, subject to constraints such as computing capacity, assignments, and quality of service requirements. The objective is to maximize the total value of tasks by using proposed heuristics to appropriately rearrange resources using the next-fit algorithm to allocate tasks and satisfy application requirements.

## 2.2. Load Balancing

Load balancing is a crucial technology that optimizes mobile application performance. Several studies have proposed cloudlet technology solutions [19] to achieve load balancing for mobile devices (Table 2). Jia et al. [19] devised a load-balancing algorithm to balance the workload among multiple cloudlets in wireless metropolitan area networks. A redirection scheme shortened the average offload task response time and improved user experience. Yao et al. [20] studied load balancing for cloudlet nodes. The task allocation problem was formulated as an integer linear problem. A two-step centralized appointment-driven strategy was used to obtain a solution with minimal average response time and a balanced load. From a network operator perspective, time-varying and periodically changing demands are challenges in providing on-demand services. Furthermore, cloud service providers face load-balancing problems in cloud computing environments. The system availability of on-demand access should be considered using an adjustable resource assignment mechanism to satisfy QoS. If resource demands are stochastic, their accurate forecast is severe. Historical data are measured for estimating usage data to determine the load distribution and evaluate the proposed algorithm [21].

**Table 2.** Load balancing comparisons with existing methods.

Classification	Objective	Strategy	Related Studies	Proposed Methods
Load balance	Maximum system availability	Bin packing; 0/1 knapsack	[19–21]	Based on the system utilization in global view

#### Comparison with related studies

In general, as many tasks as possible are admitted to maximize total revenue; however, the supply and demand for resources are unbalanced during peak traffic hours. This study determines which tasks are selected and dispatched to achieve maximum objective values subject to task assignment and limited capacity. The problems are classified as bin-packing and 0/1 knapsack problems. Herein, the assignment decisions are based on resource utilization functions, including a central processing unit, random access memory, hard drive, and bandwidth for each server type. The assignment strategies are designed to fit the QoS requirements, such as those of computing and transmission in variant traffic loads in a global view adopted by the heuristics of bin-packing strategies.

In general, as many tasks as possible are admitted to maximize total revenue; however, the supply and demand for resources are unbalanced during peak traffic hours. This study

determines the orchestrator's decisions for which tasks are selected and dispatched to achieve the maximum objective values subject to task assignment and limited capacity. The problems are classified as the bin-packing and 0/1 knapsack combined problems. Herein, the orchestrator's decisions are based on resource utilization functions, including a central processing unit, random access memory, hard drive, and bandwidth for each server type. The orchestrator's objective is also designed for that assignment strategy fitting the QoS requirements, such as those of computing and transmission in variant traffic loads in a global view adopted by the heuristics of bin-packing strategies.

### 2.3. Admission Control

Preventing server overload and ensuring application performance are the goals of admission control [22]. This mechanism decides whether to admit a particular service request to the server. It is implemented for meeting QoS requirements and achieving service providers' expected revenue [23]. Hoang et al. [24] developed a semi-Markov decision process (SMDP)-based optimization model for mobile cloud computing that considers constraints such as resources, bandwidth, and QoS to perform admission control tasks. This framework ensures QoS performance and maximizes the reward. Xia et al. [25] aimed to optimize cloud system throughput. An effective admission algorithm based on the proposed resource cost paradigm to model various resource consumptions was devised according to the online request following the current workload. Table 3 presents the comparison of the admission control methods.

**Table 3.** Admission control comparisons with existing methods.

Classification	Objective	Strategy	Related Studies	Proposed Methods
Admission control	Maximum QoS and throughput	SMDP; priority; no-priority; 0/1 knapsack	[22–25]	Non-priority and Lagrangian multipliers are adopted
<b>Comparison with related studies</b>				
The semi-Markov decision process (SMDP) approach relies on partial data, such as resources, bandwidth, and quality of service (QoS), to decide whether to accept or reject a task. SMDP computing time is inefficient for jobs with time constraints, such as delay and delay tolerance sensitive applications. The priority of user information is unknown for decision making under a computing time constraint. In this paper, call admission control mechanisms within the conditions with non-priority and QoS constraints are jointly considered. The Lagrangian relaxation-based approach is proposed to maximize system revenue combined with the proposed resource allocation methods to admit tasks and satisfy application requirements appropriately.				

The SMDP-based approach relies on partial data. The computing time is also inefficient for jobs with time constraints, such as delay and delay tolerance sensitive applications. The priority of user information is unknown for decision making under a computing time constraint. In this paper, call admission control mechanisms within the conditions with non-priority and QoS constraints are jointly considered. The Lagrangian relaxation-based approach is proposed to maximize system revenue combined with the proposed resource allocation methods to appropriately admit tasks and satisfy application requirements.

### 2.4. Research Scope

This research focuses on resource management in various scenarios in 5G network slices. An optimization-based approach (LR) is used to solve the mathematical programming problem to maximize system revenue. In our proposed cloud computing framework (cloudlets and core clouds) in 5G network slices, two primary research questions are considered. What is the most efficient algorithm for resource allocation within admission control, resource scheduling, and load-balancing policies in a limited-resource cloud computing environment? Is the HA of VMs considerably influenced in the system?

Additionally, the problem is addressed under rapidly increasing data traffic conditions and a limited resource pool to obtain near-optimal policies using a combination of LR approaches. The solutions satisfy the QoS-related constraints in transmission and stand on computation perspectives to fulfill the throughput, delay, and delay jitter requirements. They are compared with other resource management schemes for efficiency and effectiveness.

### 3. Mathematical Formulation

A mathematical model is proposed to manage an optimization programming problem. It focuses on developing a well-designed algorithm in a parallel computing environment for admission control, resource scheduling, and an HA arrangement to maximize cloud service provider profits in 5G network slices. Based on the combined cloudlet and core cloud network architecture (Figure 1), VMs with VNFs are requested by various applications and require specific resources. They should be allocated to the server  $k$  with computation and transmission capacities concerning CPU, RAM, storage, and internal or shared bandwidth, expressed as  $P_k, M_k, H_k, B_k^n$ , and  $B^c$ , respectively, in cloudlets and core clouds. The proposed model in the orchestrator supports network resource management in front of the cloudlet and core cloud environments when assuming that resource management strategies are followed when facing a batch of requests. Tables 4 and 5 present lists of the given parameters and decision variables.

**Table 4.** Given parameters.

Notation	Description
$S$	Index set of physical servers in the cloud computing system (cloudlets and core clouds), where $S = \{1, 2, \dots, s, \dots,  S \}$ .
$I$	Index set of applications on the cloud computing system, where $I = \{1, 2, \dots, i, \dots,  I \}$ .
$W$	Index set of standard VMs, $W = \bigcup_{i=1}^I W_i$ and $W_i = \{1, 2, \dots, j, \dots,  W_i \}$ , where $W_i$ is also an index set of standard VMs required by application $i$ , where $i \in I$ .
$T$	Index set of VMs with high availability (HA), $T = \bigcup_{i=1}^I T_i$ and $T_i = \{1, 2, \dots, \ell, \dots,  T_i \}$ , where $T_i$ is also an index set of VMs with HA required by application $i$ , where $i \in I$ .
$N$	Set of total VMs, $N = \bigcup_{i=1}^I N_i$ and $N_i = W_i \cup T_i$ , where $N_i$ is the total number of standard VMs and VMs with HA required by application $i$ , where $i \in I$ .
$r$	Dual-rate; represents the ratio between the number of standard VMs and the VMs with HA, formulated as $T_i = \lceil  W_i  \times r \rceil$ , where $i \in I$ .
$B_s^n$	Internal transmission bandwidth of server $s$ , where $s \in S$ .
$B^c$	Shared transmission bandwidth within the cloud computing system (cloudlets and core clouds).
$E_s$	Processing capability of each central processing unit (CPU) core in server $s$ , where $s \in S$ .
$M_s$	Total random access memory (RAM) capacity in server $s$ , where $s \in S$ .
$H_s$	Total storage capacity in server $s$ , where $s \in S$ .
$V_s$	Maximum number of VMs allowable on server $s$ , where $s \in S$ .
$\beta_s$	Cost rate for opening server $s$ , where $s \in S$ .
$R_i$	Reward of admitting application $i$ (application $i$ can be admitted only if the demands on all types of resources are fully satisfied), where $i \in I$ .
$P_i$	Penalty of rejecting application $i$ (application $i$ is rejected only if all of the types of resource requirements are not fully satisfied), where $i \in I$ .
$D_{ij}$	Total CPU processing capability on VM $j$ required by application $i$ , where $i \in I, j \in W_i$ .
$G_{ij}$	Total RAM capability required by application $i$ on VM $j$ , where $i \in I, j \in W_i$ .
$Q_{ij}$	Total storage capability required by application $i$ on VM $j$ , where $i \in I, j \in W_i$ .
$X_{ij\ell}$	Total transmission channel capacity required by application $i$ between VMs $j$ and $\ell$ , where $i \in I, j \in W_i, \ell \in T_i$ .

**Table 5.** Decision variables.

Notation	Description
$y_i$	Binary variable, 1 if the application $i$ is completely allocated to and served in the computing system, and 0 otherwise, where $i \in I$ .
$a_{ijs}$	Binary variable used for task admission control and assignment, 1 if the standard VM $j$ of application $i$ is admitted in the cloud computing networks and allocated to server $s$ , and 0 otherwise, where $i \in I, j \in W_i, s \in S$ .
$b_{i\ell s}$	Binary variable used for task admission control and assignment, 1 if the VM with HA $\ell$ of application $i$ is admitted in the cloud computing networks and allocated to server $s$ , and 0 otherwise, where $i \in I, \ell \in T_i, s \in S$ .
$f_{ij\ell s}$	Binary variable used for exclusive setting, 1 if VMs $j$ and $\ell$ of application $i$ are allocated to server $s$ , and 0 otherwise, where $i \in I, j \in W_i, \ell \in T_i, s \in S$ .
$z_s$	Binary variable for server power-on or power-off status, 1 if server $s$ is turned on, and 0 otherwise, where $s \in S$ .

The system's profit is the combination of the rewards of admitting applications, the penalties of rejecting the other applications, and the cost for turning on the servers.

The objective function  $Z_{IP}$  is shown in Equation (1), and its goal is to maximize system profits among all VMs requested by the applications.

$$Z_{IP} = \sum_{i \in I} R_i y_i - \sum_{i \in I} P_i (1 - y_i) - \sum_{s \in S} \beta_s z_s \quad (1)$$

To acquire the highest revenue, the optimization problem is shown as:

Objective function:

$$\max Z_{IP} = \min -Z_{IP}$$

subject to:

$$C1: (|W_i| + |T_i|)y_i \leq \sum_{j \in W_i} \sum_{s \in S} a_{ijs} + \sum_{\ell \in T_i} \sum_{s \in S} b_{i\ell s}, \forall i \in I,$$

$$C2: \sum_{s \in S} a_{ijs} \leq 1, \forall i \in I, \forall j \in W_i,$$

$$C3: \sum_{s \in S} b_{i\ell s} \leq 1, \forall i \in I, \forall \ell \in T_i,$$

$$C4: \sum_{i \in I} \sum_{j \in W_i} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} b_{i\ell s} \leq V_s, \forall s \in S,$$

$$C5: \sum_{i \in I} \sum_{j \in W_i} D_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} D_{i\ell} b_{i\ell s} \leq E_s, \forall s \in S,$$

$$C6: \sum_{i \in I} \sum_{j \in W_i} G_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} G_{i\ell} b_{i\ell s} \leq M_s, \forall s \in S,$$

$$C7: \sum_{i \in I} \sum_{j \in W_i} Q_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} Q_{i\ell} b_{i\ell s} \leq H_s, \forall s \in S,$$

$$C8: \sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} X_{ij\ell} f_{ij\ell s} \leq B_s^n, \forall s \in S,$$

$$C9: \sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} X_{ij\ell} (1 - f_{ij\ell s}) \leq B^c, \forall s \in S,$$

$$C10: a_{ijs} \leq z_s, \forall i \in I, \forall j \in W_i, \forall s \in S,$$

$$C11: b_{i\ell s} \leq z_s, \forall i \in I, \forall \ell \in T_i, \forall s \in S,$$

$$C12: a_{ijs} + b_{i\ell s} \leq f_{ij\ell s} + 1, \forall i \in I, \forall j \in W_i, \forall \ell \in T_i, \forall s \in S.$$

where C1–C3 belong to the admission control and assignment constraints, C4–C9 belong to the capacity constraints, and C10–C12 belong to the HA constraints.

For admission control and assignment constraints,  $(|W_i| + |T_i|)$  is the total number of standard or additional HA VMs required by an application, in which the VMs are in

disjoint sets, where  $W_i \cap T_i = \emptyset$  in constraint C1.  $\sum_{s \in S} a_{ijs}$  is the number of standard VMs that must be admitted and allocated into servers.  $\sum_{s \in S} b_{i\ell s}$  is the number of admitted and allocated HA VMs. The total number of allocated HA and standard VMs should be greater than or equal to the requirement, as shown in constraint C1. That is, while application  $i$  is completely served, the summation of  $a_{ijs}$  and  $b_{i\ell s}$  should be greater than and equal to the demand on the right-hand side of constraint C1.

$\sum_{s \in S} a_{ijs}$  and  $\sum_{s \in S} b_{i\ell s}$  are shown in the constraints C2 and C3, which means when the value of  $\sum_{s \in S} a_{ijs}$  or  $\sum_{s \in S} b_{i\ell s}$  is less than or equal to one, the VMs are inseparably assigned to servers. In other words, a virtual machine is not partially allocated to servers.

For capacity constraints, the resources offered by a server are defined as a set of four factors: the maximum number of VMs in the server  $s$  ( $V_s$ ), the processing capacity of each CPU core ( $E_s$ ), the RAM capacity ( $M_s$ ), and the storage capacity ( $H_s$ ). The total resources required by VMs for each server cannot exceed its available resources as formulated in the constraints C4–C7.

We also set the internal bandwidth rate ( $B_s^n$ ) in the server  $s$  for internal transmission between VMs. For example, two applications,  $i$  and  $i + 1$ , and their index sets of VMs are  $W_i = \{1_i, 2_i\}$ ,  $T_i = \{3_i\}$ ,  $W_{i+1} = \{1_{i+1}, 2_{i+1}\}$ , and  $T_{i+1} = \{3_{i+1}\}$ , as shown in Figure 2. The link between VM  $3_i$  and VM  $2_{i+1}$  represents the internal bandwidth required by VM  $3_i$  to connect to VM  $2_{i+1}$  in server  $s$ . The constraint C8 indicates that the total bandwidth required by all the VMs should not exceed the internal bandwidth  $B_s^n$  in server  $s$ . The external bandwidth rate ( $B^c$ ) is set for the transmission between servers in the cloud computing environment (cloudlets and core clouds), as illustrated in Figure 2. The link between server  $s - 1$  and server  $s$  represents the bandwidth requested by VM  $1_i$  to connect to VM  $3_i$ . The constraint C9 indicates that the total bandwidth required by all VMs accepted by servers in a cloud should not exceed the external bandwidth.

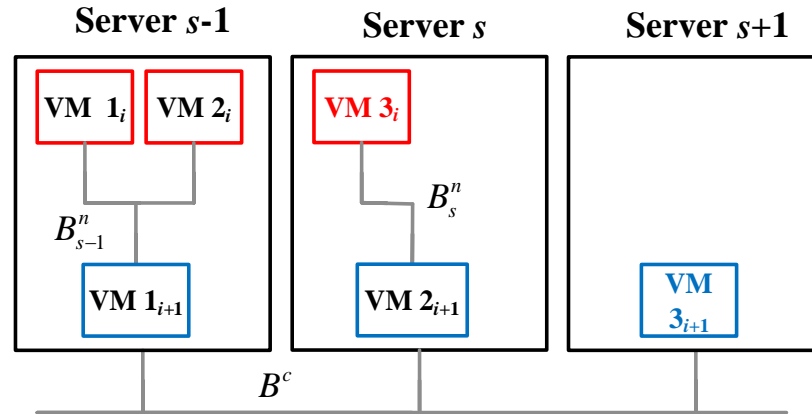


Figure 2. Representation of internal and external transmissions between VMs in servers.

In this work, we assume two kinds of VMs, standard VMs and VMs with HA, of the same application, cannot be assigned to the same server, as shown in Figure 2. In the beginning, all servers contain no VMs; then, for example: application  $i$  comes in and needs two standard VMs and one VM with HA, as mentioned in the previous paragraph. First, we put two standard VMs, VM  $1_i$  and VM  $2_i$ , in server  $s - 1$ , while the two kinds of VMs cannot be assigned to the same server, then VM  $3_i$  has to be assigned to server  $s$ . To continue, the next application  $i + 1$  comes in, and we put its standard VMs  $1_{i+1}$  and  $2_{i+1}$  in server  $s - 1$  and server  $s$ , respectively, while server  $s - 1$  and server  $s$  have no residual capacity to handle one more VM. Meanwhile, the VM with HA, VM  $3_{i+1}$ , cannot be put in server  $s - 1$  or server  $s$  due to our exclusive assumption. Thus, VM  $3_{i+1}$  is assigned to server  $s + 1$ .

For HA configuration constraints, the decision variable  $f_{ij\ell s}$  for application  $i$  is set for the VM assignments. The index sets  $j$  and  $\ell$  for different kinds of VMs must be collocated to server  $s$  and separated into different servers, where  $j \in W_i$  and  $\ell \in T_i$ . The relationships



are expressed as the constraint C12. The constraints C10 and C11 indicate the server power status. If any VM is admitted and assigned to server  $s$ , server  $s$  must be powered on, and  $z_s$  should be set to one. The constraint C12 assures that when VMs  $j$  and  $\ell$  are assigned to the same server  $s$ , which means that  $a_{ijs}$  and  $b_{i\ell s}$  are both set to one, the exclusive setting  $f_{ij\ell s}$  must also be one. In other words, the standard VM and the HA VM requested from the same application cannot be allocated to the same server  $s$ .

#### 4. Lagrangian Relaxation-Based Solution Processes

The Lagrangian relaxation method is proposed for solving large-scale mathematical programming problems, including optimization problems with linear, integer, and nonlinear programming problems in many practical applications [26]. The key idea is to relax complicated constraints into a primal optimization problem and extend feasible solution regions to simplify the primal problem. Based on the relaxation, the primal problem is transformed into an LR problem associated with Lagrangian multipliers [27–29]. Figure 3 illustrates the six procedural steps in the LR method.

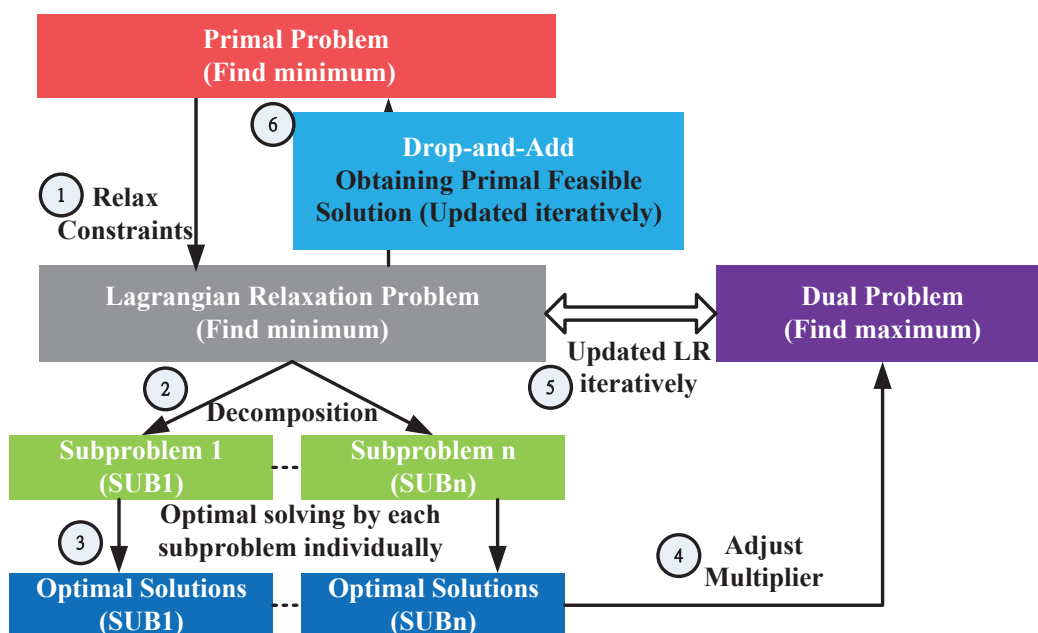


Figure 3. Lagrangian relaxation-based solution process flow.

##### 4.1. Procedures of Step 1: Relaxation

To separate the feasible region of the primal problem into several subproblems with an independent set of decision variables, the primal problem is transformed into an LR problem. The relaxation method associated with Lagrangian multipliers is applied to the constraints C1 and C4–C13 in Step 1, as presented in Figure 3. Then, the constraints C1 and C4–C12 with the multipliers are added to the primal problem Equation (1), as shown in Equation (2) and denoted as  $Z_{LR}$ .

$$\begin{aligned}
Z_{LR} = & - \sum_{i \in I} R_i y_i + \sum_{i \in I} P_i (1 - y_i) + \sum_{s \in S} \beta_s z_s \\
& + \sum_{i \in I} \mu_i^1 [(|W_i| + |T_i|) y_i - \sum_{j \in W_i} \sum_{s \in S} a_{ijs} - \sum_{\ell \in T_i} \sum_{s \in S} b_{i\ell s}] \\
& + \sum_{s \in S} \mu_s^2 [\sum_{i \in I} \sum_{j \in W_i} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} b_{i\ell s} - V_s] \\
& + \sum_{s \in S} \mu_s^3 [\sum_{i \in I} \sum_{j \in W_i} D_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} D_{i\ell} b_{i\ell s} - E_s] \\
& + \sum_{s \in S} \mu_s^4 [\sum_{i \in I} \sum_{j \in W_i} G_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} G_{i\ell} b_{i\ell s} - M_s] \\
& + \sum_{s \in S} \mu_s^5 [\sum_{i \in I} \sum_{j \in W_i} Q_{ij} a_{ijs} + \sum_{i \in I} \sum_{\ell \in T_i} Q_{i\ell} b_{i\ell s} - H_s] \\
& + \sum_{s \in S} \mu_s^6 [\sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} X_{ij\ell} f_{ij\ell s} - B_s^n] \\
& + \sum_{s \in S} \mu_s^7 [\sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} X_{ij\ell} (1 - f_{ij\ell s}) - B^c] \\
& + \sum_{i \in I} \sum_{j \in W_i} \sum_{s \in S} \mu_{ijs}^8 [a_{ijs} - z_s] \\
& + \sum_{i \in I} \sum_{\ell \in T_i} \sum_{s \in S} \mu_{i\ell s}^9 [b_{i\ell s} - z_s] \\
& + \sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} \sum_{s \in S} \mu_{ij\ell s}^{10} [a_{ijs} + b_{i\ell s} - f_{ij\ell s} - 1]
\end{aligned} \tag{2}$$

Then, the optimization problem can be reformulated as:

$$\begin{aligned}
& \text{Objective function:} \\
& \min Z_{LR} \\
& \text{subject to: C2, C3,}
\end{aligned}$$

where  $y_i \in \{0, 1\}$ ,  $a_{ijs} \in \{0, 1\}$ ,  $b_{i\ell s} \in \{0, 1\}$ ,  $f_{ij\ell s} \in \{0, 1\}$ , and  $z_s \in \{0, 1\}$ .

#### 4.2. Procedures of Steps 2 and 3: Decomposition and Solving Subproblems

The LR problem can be decomposed into several independent subproblems, with their related decision variables. The divide-and-conquer approach is used to solve the subproblems correspondingly.

##### 4.2.1. Subproblem 1 (Related to $y_i$ )

By extracting items with decision variable  $y_i$ , the optimization problem of Subproblem 1 can be developed as:

$$\begin{aligned}
& \text{Objective function:} \\
& \min \sum_{i \in I} (-R_i - P_i + \mu_i^1 |W_i| + \mu_i^1 |T_i|) y_i \\
& \text{subject to: } y_i \in \{0, 1\}.
\end{aligned} \tag{3}$$

Equation (3) can be divided into  $|I|$  independent subproblems. For each application  $i$ , where  $i \in I$ , the decision variable  $y_i$  is set to one when the coefficient  $(-R_i - P_i + \mu_i^1 |W_i| + \mu_i^1 |T_i|)$  is less than zero. Otherwise,  $y_i$  is set to zero. The run time is  $O(|I|)$ , and the pseudocode is illustrated in Algorithm 1.

**Algorithm 1** Subproblem 1.

---

**Input:** Given parameters  $\mathbf{R}, \mathbf{P}, \mathbf{W}, \mathbf{T}$  and Lagrangian multipliers  $\mu^1$ .  
**Output:** Decision variable  $\mathbf{y}$ .  
**Initialize:**  $y_i \leftarrow 0, \forall i \in I$   
**for**  $i = 0$  to  $(|I| - 1)$  **do**  
     $c \leftarrow -R_i - P_i + \mu_i^1 |W_i| + \mu_i^1 |T_i|$   
    **if**  $c < 0$  **then**  
         $y_i \leftarrow 1$   
    **end if**  
**end for**

---

4.2.2. Subproblem 2 (Related to  $a_{ijs}$ )

By extracting items with decision variable  $a_{ijs}$ , the optimization problem of Subproblem 2 can be developed as:

Objective function:

$$\min \sum_{i \in I} \sum_{j \in W_i} \sum_{s \in S} (-\mu_i^1 + \mu_s^2 + \mu_s^3 D_{ij} + \mu_s^4 G_{ij} + \mu_s^5 Q_{ij} + \mu_{ijs}^8 + \sum_{\ell \in T_i} \mu_{ij\ell s}^{10}) a_{ijs} \quad (4)$$

subject to C2:  $\sum_{s \in S} a_{ijs} \leq 1, \forall i \in I, \forall j \in W_i,$

$$a_{ijs} \in \{0, 1\}.$$

Equation (4) can be divided into  $|I||W_i||S|$  cases. The decision variable  $a_{ijs}$  is set to one, and the minimum coefficient  $(-\mu_i^1 + \mu_s^2 + \mu_s^3 D_{ij} + \mu_s^4 G_{ij} + \mu_s^5 Q_{ij} + \mu_{ijs}^8 + \sum_{\ell \in T_i} \mu_{ij\ell s}^{10})$  is less than zero and corresponds to alliance subindices  $i, j$ , and  $s$ . To satisfy the constraint C2, the decision variable should be set to one only if the minimum coefficient with subindex  $s$  is a required sorting processes. Otherwise,  $a_{ijs}$  is set to zero. The run time is  $O(|I||W_i||S||T_i|)$ . The pseudocode is illustrated in Algorithm 2.

**Algorithm 2** Subproblem 2.

---

**Input:** Given parameters  $\mathbf{D}, \mathbf{G}, \mathbf{Q}$  and Lagrangian multipliers  $\mu^1, \mu^2, \mu^3, \mu^4, \mu^5, \mu^8, \mu^{10}$ .  
**Output:** Decision variable  $\mathbf{a}$ .  
**Initialize:**  $a_{ijs} \leftarrow 0, \forall i \in I, \forall j \in W_i, \forall s \in S$   
**for**  $i = 0$  to  $(|I| - 1)$  **do**  
    **for**  $j = 0$  to  $(|W_i| - 1)$  **do**  
        **for**  $s = 0$  to  $(|S| - 1)$  **do**  
             $c_s \leftarrow -\mu_i^1 + \mu_s^2 + \mu_s^3 D_{ij} + \mu_s^4 G_{ij} + \mu_s^5 Q_{ij} + \mu_{ijs}^8 + \sum_{\ell \in T_i} \mu_{ij\ell s}^{10}$   
        **end for**  
        Find the index  $m$  that  $c_m$  has the minimum value in  $\mathbf{c}$ .  
        **if**  $c_m < 0$  **then**  
             $a_{ijm} \leftarrow 1$   
        **end if**  
    **end for**  
**end for**

---

#### 4.2.3. Subproblem 3 (Related to $b_{i\ell s}$ )

By extracting items with decision variable  $b_{i\ell s}$ , the optimization problem of Subproblem 3 can be developed as:

Objective function:

$$\min \sum_{i \in I} \sum_{\ell \in T_i} \sum_{s \in S} (-\mu_i^1 + \mu_s^2 + \mu_s^3 D_{i\ell} + \mu_s^4 G_{i\ell} + \mu_s^5 Q_{i\ell} + \mu_{i\ell s}^9 + \sum_{j \in W_i} \mu_{ij\ell s}^{10}) b_{i\ell s} \quad (5)$$

$$\text{subject to C3: } \sum_{s \in S} b_{i\ell s} \leq 1, \forall i \in I, \forall \ell \in T_i,$$

$$b_{i\ell s} \in \{0, 1\}.$$

The solution process of Equation (5) is similar to that of Equation (4) and can be also divided into  $|I||T_i||S|$  subproblems. The decision variable  $b_{i\ell s}$  is set to one when the minimum coefficient  $(-\mu_i^1 + \mu_s^2 + \mu_s^3 D_{i\ell} + \mu_s^4 G_{i\ell} + \mu_s^5 Q_{i\ell} + \mu_{i\ell s}^9 + \sum_{j \in W_i} \mu_{ij\ell s}^{10})$  is less than zero and corresponds to alliance subindices  $i, \ell$ , and  $s$ . Otherwise,  $b_{i\ell s}$  is set to zero. The run time of this subproblem is  $O(|I||T_i||S||W_i|)$ . The pseudocode is illustrated in Algorithm 3.

---

#### Algorithm 3 Subproblem 3.

---

**Input:** Given parameters  $\mathbf{D}, \mathbf{G}, \mathbf{Q}$  and Lagrangian multipliers  $\mu^1, \mu^2, \mu^3, \mu^4, \mu^5, \mu^9, \mu^{10}$ .

**Output:** Decision variable  $\mathbf{b}$ .

**Initialize:**  $b_{i\ell s} \leftarrow 0, \forall i \in I, \forall \ell \in T_i, \forall s \in S$

**for**  $i = 0$  to  $(|I| - 1)$  **do**

**for**  $\ell = 0$  to  $(|T_i| - 1)$  **do**

**for**  $s = 0$  to  $(|S| - 1)$  **do**

$$c_s \leftarrow -\mu_i^1 + \mu_s^2 + \mu_s^3 D_{i\ell} + \mu_s^4 G_{i\ell} + \mu_s^5 Q_{i\ell} + \mu_{i\ell s}^9 + \sum_{j \in W_i} \mu_{ij\ell s}^{10}$$

**end for**

    Find the index  $m$  that  $c_m$  has the minimum value in  $\mathbf{c}$ .

**if**  $c_m < 0$  **then**

$$b_{i\ell m} \leftarrow 1$$

**end if**

**end for**

**end for**

---

#### 4.2.4. Subproblem 4 (Related to $f_{ij\ell s}$ )

By extracting items with decision variable  $f_{ij\ell s}$ , the optimization problem of Subproblem 4 can be developed as:

Objective function:

$$\min \sum_{i \in I} \sum_{j \in W_i} \sum_{\ell \in T_i} \sum_{s \in S} (\mu_s^6 X_{ij\ell} - \mu_s^7 X_{ij\ell} - \mu_{ij\ell s}^{10}) f_{ij\ell s} \quad (6)$$

$$\text{subject to: } f_{ij\ell s} \in \{0, 1\}.$$

Equation (6) can be divided into  $|I||W_i||S||T_i|$  cases. For the alliance subindices  $i, j, \ell$ , and  $s$ , the decision variable  $f_{ij\ell s}$  is set to one when the coefficient  $(\mu_s^6 X_{ij\ell} - \mu_s^7 X_{ij\ell} - \mu_{ij\ell s}^{10})$  is less than zero. Otherwise,  $f_{ij\ell s}$  is set to zero. The run time is  $O(|I||W_i||S||T_i|)$ . The pseudocode is illustrated in Algorithm 4.

**Algorithm 4** Subproblem 4.

**Input:** Given parameters  $\mathbf{X}$  and Lagrangian multipliers  $\mu^6, \mu^7, \mu^{10}$ .

**Output:** Decision variable  $\mathbf{f}$ .

**Initialize:**  $f_{ij\ell s} \leftarrow 0, \forall i \in I, \forall j \in W_i, \forall \ell \in T_i, \forall s \in S$

```

for  $i = 0$  to  $(|I| - 1)$  do
  for  $j = 0$  to  $(|W_i| - 1)$  do
    for  $\ell = 0$  to  $(|T_i| - 1)$  do
      for  $s = 0$  to  $(|S| - 1)$  do
         $c \leftarrow \mu_s^6 X_{ij\ell} - \mu_s^7 X_{ij\ell} - \mu_{ij\ell s}^{10}$ 
        if  $c < 0$  then
           $f_{ij\ell s} \leftarrow 1$ 
        end if
      end for
    end for
  end for
end for

```

4.2.5. Subproblem 5 (Related to  $z_s$ )

By extracting items with decision variable  $z_s$ , the optimization problem of Subproblem 5 can be developed as:

Objective function:

$$\min \sum_{s \in S} [\beta_s - \sum_{i \in I} (\sum_{j \in W_i} \mu_{ijs}^8 + \sum_{\ell \in T_i} \mu_{i\ell s}^9)] z_s \quad (7)$$

subject to:  $z_s \in \{0, 1\}$ .

Equation (7) can be divided into  $|S||I||W_i|$  or  $|S||I||T_i|$  cases. In each case with subindex  $s$ , the decision variable  $z_s$  is set to one when the coefficient  $[\beta_s - \sum_{i \in I} (\sum_{j \in W_i} \mu_{ijs}^8 + \sum_{\ell \in T_i} \mu_{i\ell s}^9)]$ , which corresponds to alliance subindex  $s$ , is less than zero. Otherwise,  $z_s$  is set to zero. The run time is  $O(|S||I||W_i|)$  or  $O(|S||I||T_i|)$ , and the pseudocode is illustrated in Algorithm 5.

**Algorithm 5** Subproblem 5.

**Input:** Given parameters  $\beta$  and Lagrangian multipliers  $\mu^8, \mu^9$ .

**Output:** Decision variable  $\mathbf{z}$ .

**Initialize:**  $z_s \leftarrow 0, \forall s \in S$

```

for  $s = 0$  to  $(|S| - 1)$  do
   $c \leftarrow 0$ 
  for  $i = 0$  to  $(|I| - 1)$  do
    for  $j = 0$  to  $(|W_i| - 1)$  do
       $c \leftarrow c + \mu_{ijs}^8$ 
    end for
    for  $\ell = 0$  to  $(|T_i| - 1)$  do
       $c \leftarrow c + \mu_{i\ell s}^9$ 
    end for
  end for
  if  $\beta_s - c < 0$  then
     $z_s \leftarrow 1$ 
  end if
end for

```

## 4.3. Procedure of Step 4: Dual Problem and the Subgradient Method

According to the weak Lagrangian duality theorem [30], the objective values of the LR problem  $Z_{LR}$  are the lower bounds (LBs) of the primal problem  $Z_{IP}$  with multiples

$\mu_i^1, \mu_s^2, \mu_s^3, \mu_s^4, \mu_s^5, \mu_s^6, \mu_s^7, \mu_{ijs}^8, \mu_{i\ell s}^9, \mu_{ij\ell s}^{10} \geq 0, \forall i \in I, \forall j \in W_i, \forall \ell \in T_i, \forall s \in S$ . The formulation of the dual problem (D) is constructed to calculate the tightest LB ( $\max Z_D$ ), where  $\max Z_D = \min Z_{LR}$ . Then, the dual problem can be formulated as:

Objective function:

$$\max Z_D$$

$$\text{subject to: } \mu^1 \geq 0, \mu^2 \geq 0, \mu^3 \geq 0, \mu^4 \geq 0, \mu^5 \geq 0,$$

$$\mu^6 \geq 0, \mu^7 \geq 0, \mu^8 \geq 0, \mu^9 \geq 0, \mu^{10} \geq 0.$$

The subgradient method is commonly used for solving the dual problem by iteratively updating the Lagrangian multipliers [26,31,32].

First, let vector  $\mathbf{S}$  be a subgradient of  $Z_D$ . In the  $q$ th iteration of the subgradient optimization procedure, the multiplier vector  $\pi^q = (\mu^{1,q}, \mu^{2,q}, \mu^{3,q}, \mu^{4,q}, \mu^{5,q}, \mu^{6,q}, \mu^{7,q}, \mu^{8,q}, \mu^{9,q}, \mu^{10,q})$  is updated by  $\pi^{q+1} = \pi^q + t^q \mathbf{S}^q$ . The step size  $t^q$  is determined by equation  $t^q = \lambda \frac{(Z_{IP}^h - Z_D(\pi^q))}{\|\mathbf{S}^q\|^2}$ . The denominator  $\mathbf{S}^q$  is the sum of relaxed constraints concerning to the decision variable values based on the  $q$ th iteration.  $Z_{IP}^h$  is the primal objective value in the  $h$ th iteration.  $Z_D(\pi^q)$  is the objective value of the dual problem in the  $q$ th iteration.  $\lambda$  is a constant, where  $0 \leq \lambda \leq 2$ . Accordingly, the optimal objective value of the dual problem is obtained iteratively.

#### 4.4. Procedure of Step 5: Obtaining the Primal Feasible Solutions

Applying the LR and the subgradient methods to solve the LR and dual problems determines a theoretical LB from the primal feasible solution. Crucial information regarding the primal feasible solution can be identified [28]. The feasible region of a mathematical programming problem defined by the solutions must be satisfied by all constraints. A set of primal feasible solutions to  $Z_{IP}$  is a subset of the infeasible region solutions to  $Z_{LR}$  or  $Z_D$ . Several alternative methods can be used sophisticatedly to obtain the primal feasible solutions from the observations of the infeasible region. For example, Figure 4 presents an experimental case. The green curve represents the process of obtaining the primal feasible solutions iteratively. The objective is to identify the minimum value of the primal problem ( $Z_{IP}^h$ ). Then, the LBs are determined using the subgradient method to iteratively obtain the tightest LB ( $\max Z_D$ ), represented by the purple line. The proposed resource management approach for obtaining primal feasible solutions is called the drop-and-add algorithm. It is a heuristic-based allocation mechanism for optimizing the objective function. It searches for a solution that satisfies not only all the user demands, but also the constraints. The initial solution is adopted using next-fit (NF) or first-fit (FF) as baselines for evaluating solution quality [33]. The proposed algorithm, FF, and NF are simultaneously implemented for result comparison.

#### 4.5. Procedure of Step 6: Drop-and-Add Algorithm

Lagrangian multipliers determined from the dual problem have significant values for evaluating the sensitivity of objective value improvement [26,28,30,32]. Through the subproblems, the integrated weighting factor of applications is represented in (3) as  $(-R_i - P_i + \mu_i^1 |W_i| + \mu_i^1 |T_i|)$ ; therefore, the sum of the values can be used as an index to interpret the application's significance of  $i$ . The corresponding multipliers determine the ordering set of admitting, assigning, and scheduling decision variables. The other decision variables  $a_{ijs}$  or  $b_{i\ell s}$  and the assignment of indices  $j, \ell$ , and  $s$  can be regarded as two bin-packing problems. The VM  $j$  can be packed into the server  $s$ . The VM  $\ell$  can be packed into servers except server  $s$  to comply with the HA conditions for  $b_{i\ell s}$ . For performance evaluation, NF is adopted for sequentially performing the algorithm of assignment for  $a_{ijs}$  and then  $b_{i\ell s}$ . The flowchart of the drop-and-add algorithm is shown in Figure 5.

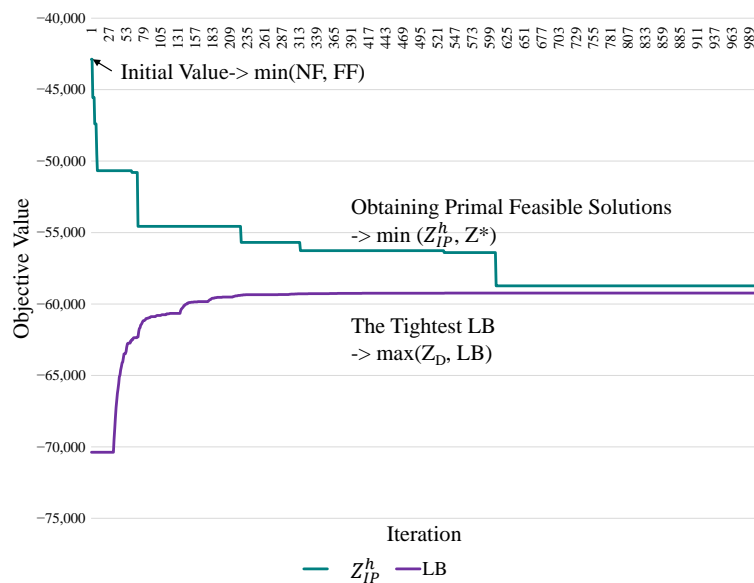


Figure 4. Obtaining primal feasible solutions and the tightest lower bound (LB). NF, next-fit.

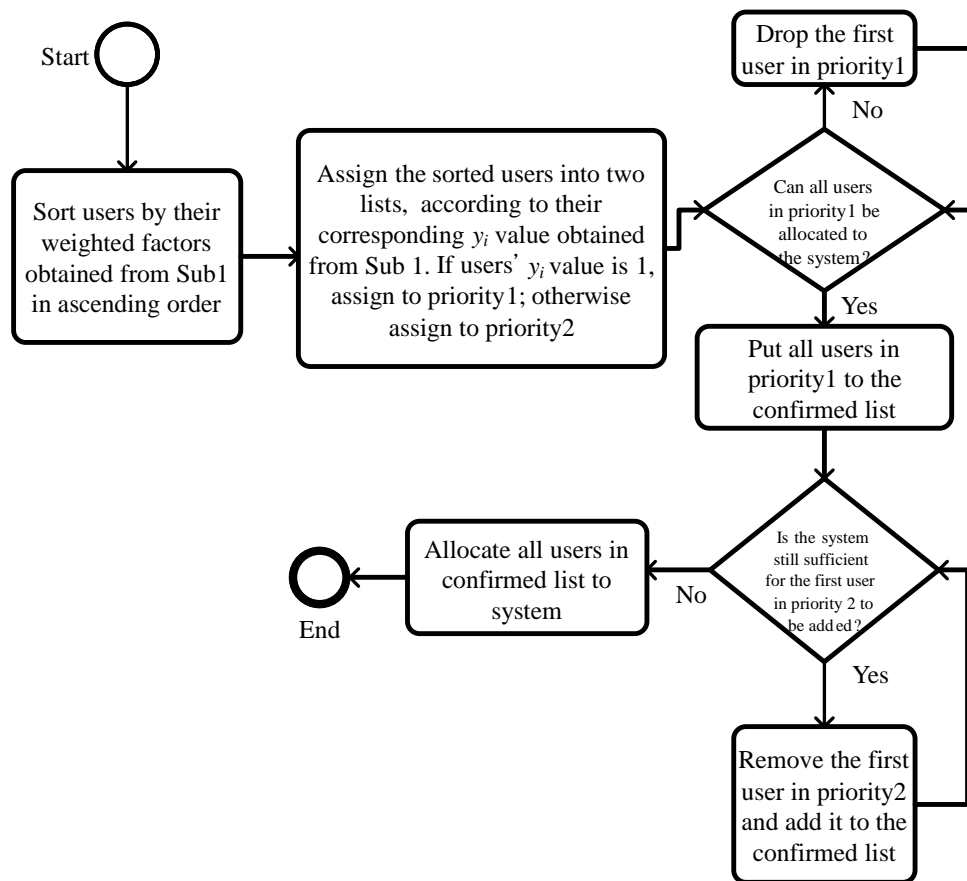


Figure 5. Flowchart of the drop-and-add algorithm.

### 5. Computational Experiments

A cloud service provider’s resource parameters and the applications’ demand attributes were simulated in a cloud computing experimental environment and are presented in Table 6.  $U(x, y)$  means a number uniform distributed between the parameters of  $x$  and  $y$ .

**Table 6.** Given parameters for the experiments.

Given Parameter	Value
Number of servers, $ S $	56–84
Number of applications, $ I $	24–52
Dual-rate, $r$	0.5
Number of standard VMs for application $i$ , $ W_i $	$ W_i  \sim U(1, 10), \forall i \in I$
Number of VMs with HA for application $i$ , $ T_i $	$T_i = \lceil  W_i  \times r \rceil, \forall i \in I$
Total number of VMs for application $i$ , $ N_i $	$ N_i  =  W_i  +  T_i , \forall i \in I$
Host internal bandwidth capacity, $B_s^n$ (Mbps)	120–225
Shared transmission bandwidth within the cloud computing system, $B^c$ (Mbps)	1000
Host CPU processing capacity, $E_s$ (GHz)	480–900
Host memory capacity, $M_s$ (GB)	120–225
Host storage capacity, $H_s$ (GB)	1200–2250
The maximum number of VMs allowable on server $s$ , $V_s$	8–15
Cost rate for opening server $s$ , $\beta_s$	$\beta_s = (E_s + M_s + H_s + V_s)/40,000$
Reward rate of each application, $R_i$	$R_i \sim \sum_j \binom{D_{ij}}{50}, \forall i \in I$
Penalty rate of each application, $P_i$	$P_i \sim \sum_j \binom{D_{ij}}{100}, \forall i \in I$
CPU requests of a task, $D_{ij}$ (GHz)	$D_{ij} \sim U(1, 120), \forall i \in I, \forall j \in W_i$
Memory requests of a task, $G_{ij}$ (GB)	$G_{ij} \sim U(1, 30), \forall i \in I, \forall j \in W_i$
Storage requests of a task, $Q_{ij}$ (TB)	$Q_{ij} \sim U(1, 300), \forall i \in I, \forall j \in W_i$
Total transmission channel capacity, $X_{ij\ell}$ (Mbps)	$X_{ij\ell} \sim U(0, 30), \forall i \in I, \forall j \in W_i, \forall \ell \in T_i$
Bandwidth requests of a task, $C_{ijs}$ (Mbps)	$C_{ijs} \sim U(1, 2000), \forall i \in I, \forall j \in W_i, s \forall s \in S$

The algorithms were constructed and implemented to analyze solution quality and improvement ratios (IRs) in several simulation cases. Solution quality is defined as the objective value gap (denoted as GAP) between the proposed algorithm and the LR problem, which is expressed as  $GAP = \frac{|V_{\text{Drop\&Add}} - V_{\text{LR}}|}{\max(V_{\text{Drop\&Add}}, V_{\text{LR}})} \times 100\%$ , where  $V_{\text{Drop\&Add}}$  is the objective value of applying the drop-and-add algorithm and  $V_{\text{LR}}$  is the objective value of the LR problem.  $IR^{\text{NF}}$  is expressed as  $IR^{\text{NF}} = \frac{V_{\text{Drop\&Add}} - V_{\text{NextFit}}}{\max(V_{\text{NextFit}}, V_{\text{Drop\&Add}})} \times 100\%$ .  $IR^{\text{FF}}$  is expressed as  $IR^{\text{FF}} = \frac{V_{\text{Drop\&Add}} - V_{\text{FirstFit}}}{\max(V_{\text{FirstFit}}, V_{\text{Drop\&Add}})} \times 100\%$ , where  $V_{\text{NextFit}}$  and  $V_{\text{FirstFit}}$  are the objective values of employing the NF algorithm or FF algorithm, respectively. The experiments were developed in Python and implemented in a VM on a workstation with a quad-core CPU, 8 GB RAM, and Ubuntu 14.04. The traffic loads of tasks and arrival time intervals were randomly generated. The experimental environment was initialized for a data center including edge and core clouds. VMs were requested by applications to represent computing requirements. Based on resource admission control strategies and scheduling models, the VMs were packed into the corresponding servers in the cloud computing environments.

### 5.1. Performance Evaluation Case: Traffic Load

This experiment was designed to analyze VMs requested by applications in a time slot, which can be interpreted as a snapshot of the system loading. First, the trend of the objective function with the number of applications as the control variable was examined. The result in Figure 6 indicates that the objective value of (IP) increases with the number of applications. The drop-and-add and LB values were almost the same in some cases (20~80), indicating that the optimal solution was determined when the GAP was less than 0.80% (the minimized one was 0.60%). Table 7 compares drop-and-add with NF or FF (higher values are preferable) in the primal problem. The maximum IR was 89.48%, with



180 applications in both FF and NF. The penalty of unsatisfied applications significantly increased the objective value for indicating when arriving applications exceeded system capacity with the NF or FF algorithm in the cases of the number of applications being over 100. Otherwise, the drop-and-add algorithm can select valuable applications to pack into servers, which results in a higher objective value than NF or FF in the cases of the number of applications being over 100.

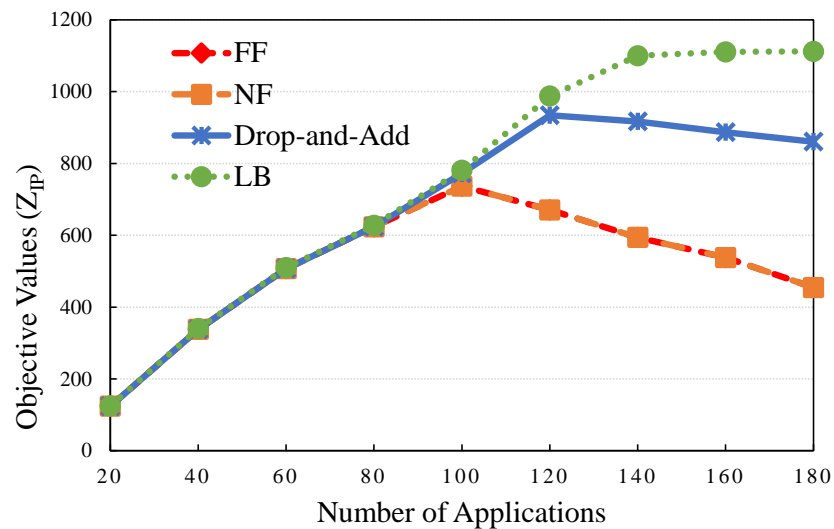


Figure 6. Objective value by the number of applications.

Table 7. Comparison of solution qualities under different number of applications.

Number of Applications	Drop-And-Add	LB	Gap (%)	NF	FF	IR <sup>NF</sup> (%)	IR <sup>FF</sup> (%)
20	123.743	124.760	0.80	123.687	123.687	0.05	0.05
40	337.910	340.080	0.64	337.910	337.910	0.00	0.00
60	506.729	509.780	0.60	506.729	506.729	0.00	0.00
80	623.692	627.760	0.65	623.523	623.523	0.02	0.02
100	772.445	780.680	1.06	736.595	736.595	4.87	4.87
120	934.295	987.845	5.73	670.535	670.535	39.33	39.33
140	916.985	1099.91	19.94	593.945	593.945	54.39	54.39
160	886.925	1110.70	25.23	537.905	537.905	64.89	64.89
180	860.705	1112.32	29.23	454.235	454.235	89.48	89.48

In some cases (20~80), the lower bound (LB) value is extremely close to, but not equal to that of the drop-and-add algorithm. In other cases, the drop-and-add algorithm has a significant improvement rate compared with NF or FF.

### 5.2. Performance Evaluation Case: Number of Servers

The cost function is emulated as the capital expenditure (CAPEX) of the cloudlets and core clouds related to network infrastructure deployment of appropriate levels of servers, where the budget is a significant constraint. The cost difference between high- and low-end servers is also typically significant. Service providers generally conduct a comprehensive system analysis to determine the methods of how to manage, control, and operate a system appropriately. The plans, such as turning servers on or off, provide sufficient QoS to applications efficiently and effectively. Furthermore, the servers purchased were deployed at four levels with different capacities in this experiment. Nonhomogeneous servers were deployed under the same limited budget. The rapid increases in data traffic are represented as traffic loads to determine which method delivered superior QoS for applications at affordable costs and preserved revenue. The following are the experimental scenarios tested. Figure 7 and Table 8 present the results of the proposed methods (drop-and-add,

LB, NF, and FF). The drop-and-add algorithm attained the most practical objective value (higher is preferable) compared with NF and FF in the case of 40.

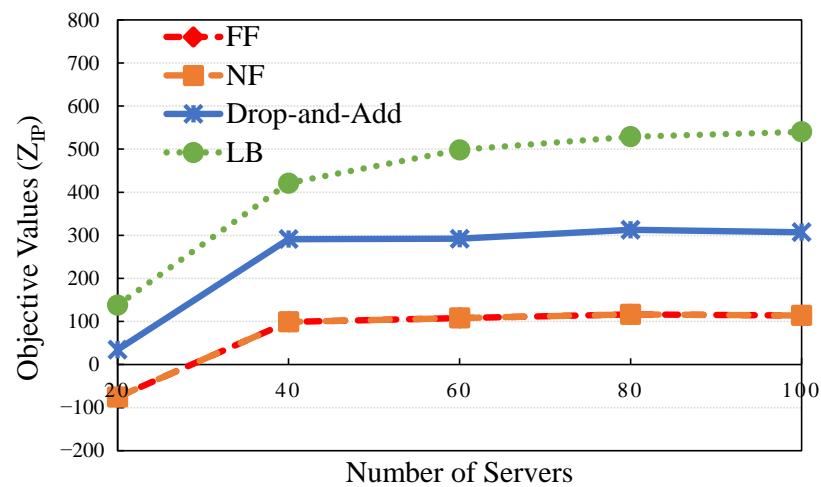


Figure 7. Objective value by the number of servers.

Table 8. Comparison of solution qualities under different number of servers.

Number of Servers	Drop-And-Add	LB	Gap (%)	NF	FF	IR <sup>NF</sup> (%)	IR <sup>FF</sup> (%)
20	34.439	137.926	300.5	-75.601	-75.601	145.55	145.55
40	291.198	421.117	44.62	99.138	99.138	193.73	193.73
60	291.880	498.201	70.69	108.01	108.01	170.23	170.23
80	312.825	528.898	69.07	116.41	116.41	168.73	168.73
100	307.056	540.088	75.89	113.894	113.894	169.6	169.6

Regarding the GAP, some values in the other situations were calculated to determine the difference in rate value between drop-and-add and LB. The GAP values indicate that the minimum value (44.62%) was determined in one of the cases with numerous servers (40). A data center has sufficient resource space in servers with the drop-and-add, NF, and FF algorithms. The maximum improvement ratio is represented by the ratio of improvement of feasible solutions and was 193.73% in numerous servers (i.e., 40). The result reveals that the resource allocation algorithm has a significant impact on system performance with a limited resource constraint.

### 5.3. Performance Evaluation Case: Effect of HA

As far as crucial business requirements are concerned, cloud service providers should offer high quality, excellent availability, good performance, and reliable services. In this case, the applications are divided into levels by using VM replication for HA requests. The VMs requesting to be in the  $T_i$  set with HA by the application  $i$  asking for VM placement must be mutually and exclusively allocated into different physical servers. The level of exclusivity is called the dual-rate. In the subsequent experiment, the dual-rate was the parameter configured for HA VMs. Moreover, a dual-rate of 0.5 indicates that the HA VMs,  $|T_i|$ , requested by half of  $|W_i|$  for application  $i$  are allocated to different servers. The dual-rate equals 0.3, which indicates 30% of standard VMs required for application  $i$  with HA capability. In Figure 8, it is evident that the dual-rate significantly affected the application satisfaction rate. Thus, the drop-and-add algorithm offers more benefits (higher values are preferable) than NF or FF in the dual-rate cases. The improvement ratios increased significantly when the dual-rate was higher than 0.3, and the maximum IR was 608.39%. The drop-and-add algorithm achieved flexibility and efficiency and could sufficiently obtain

the maximum objective value to deal with HA requests. As observed in Table 9, FF and NF performed poorly when the dual-rate was beyond 0.3. The tasks were not assigned to appropriate servers with HA. This resulted in turning on more servers, which corresponded to cost generation.

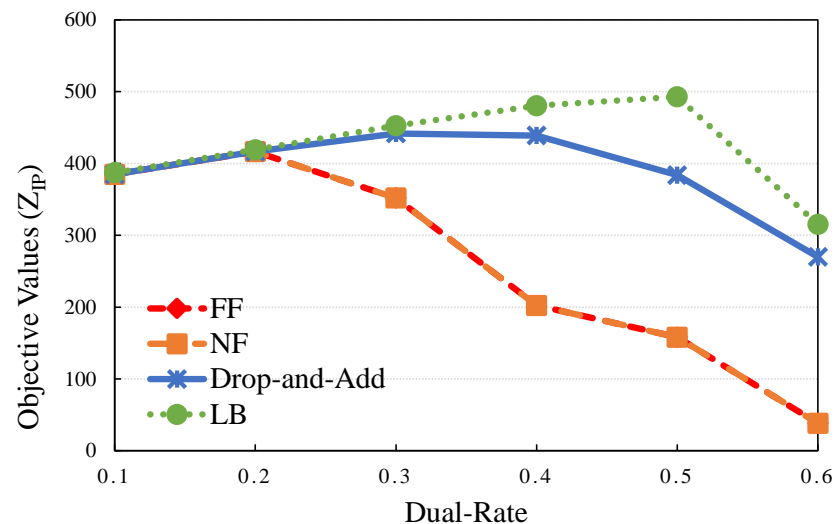


Figure 8. Objective value when scaling the dual-rate.

Table 9. Comparison of solution qualities under different the dual-rate.

Dual-Rate	Drop-and-Add	LB	Gap (%)	NF	FF	IR <sup>NF</sup> (%)	IR <sup>FF</sup> (%)
0.1	384.827	387.200	0.61	384.827	384.827	0.00	0.00
0.2	416.531	419.040	0.60	416.531	416.531	0.00	0.00
0.3	441.730	452.618	2.46	351.926	351.926	25.52	25.52
0.4	438.900	480.493	9.48	202.224	202.224	117.04	117.04
0.5	383.708	492.960	28.47	158.026	158.026	142.81	142.81
0.6	269.938	315.366	16.83	38.106	38.106	608.39	608.39

#### 5.4. Performance Evaluation Case: Time complexity comparison

Table 10 shows the time complexity of the LR-based solution for resource management in a sliced network. We added an existing scheme, brute force, for comparison in Table 10. The time complexity of this scheme was higher than the proposed LR-based algorithm. Furthermore, the corresponding explanations were included to support our statements. The time complexity of the proposed LR-based solutions was  $O(N|I||W_i||T_i||S|)$ . Table 10 shows the time complexity of the LR-based solution. The Lagrange dual solution was determined by Sub-problems (4)–(6), which were solved using the minimum algorithm with  $|S|$  servers. Each sub-problem required  $O(|I||W_i||T_i|)$  time minimum coefficient among servers. Since the sub-problems were solved individually by divide-and-conquer algorithms, the time complexity for each of the sub-problems was constant. Thus, the worst case of time complexity among these subproblems was considered significant in each iteration. The Lagrange dual solutions for the sub-problems could be obtained after a maximum number of iterations  $N$ , and the time complexity was  $O(N|I||W_i||T_i||S|)$ . The number of iterations pre-defined to converge was about 600 based on the experiment results shown in Figure 4. Fortunately, all given parameters and multipliers for the solution did not need the same initial values. The convergence was achieved in a small number of iterations with the previous execution results. Furthermore, the proposed algorithms can be set as the output results at any complete iteration. Thus, the time can be controlled in practice.

**Table 10.** Time complexity comparison between the proposed LR-based approach and other schemes.

Algorithm	Time Complexity	Annotation
Equation (3)	$O( I )$	$ I $ subproblems determine the value of the decision variable for each application $i$ .
Equation (4)	$O( I  W_i  S  T_i )$	$ I  W_i $ subproblems with summation of $\ell \in T_i$ , where $i \in I$ , determine the binary decision variable $a_{ijs}$ that the coefficient is minimized among $ S $ .
Equation (5)	$O( I  T_i  S  W_i )$	$ I  T_i $ subproblems with summation of $j \in W_i$ , where $i \in I$ determine the binary decision variable $b_{i\ell s}$ that the coefficient is minimized among $ S $ .
Equation (6)	$O( I  W_i  S  T_i )$	$ I  W_i  S  T_i $ subproblems determine decision variable $f_{ij\ell s}$ .
Equation (7)	$O( S  I  W_i )$ or $O( S  I  T_i )$	$ S $ subproblems with summation $\mu_{ijs}^8$ or $\mu_{i\ell s}^9$ determine decision variable $z_s$ .
Dual problem (D)	$O(N I  W_i  T_i  S )$	$N$ times of the maximum complexity by the sub-problems, (4)–(6).
Drop-and-add algorithm	$O(N I  W_i  T_i  S )$	The algorithm adjusts the values of decision variables $y_i$ , $a_{ijs}$ , $b_{i\ell s}$ , $f_{ij\ell s}$ , and $z_s$ based on the dual problems. The complexity is the worst case determined by the decision variables of the subproblems.
FF	$O( I  W_i  T_i  S )$	The resource allocation.
NF	$O( I  W_i  T_i  S )$	The resource allocation.
Brute force	$O(2^{ I ^4 W_i ^2 T_i ^2 S ^4})$	The total combination of decision variables.

### 5.5. Discussion

To test the proposed algorithm, we designed three experiments involving changes in application demands, the cost of servers, and dual-rates for HA with application requests, as shown in Table 11. The research problem was formulated as a mathematical programming problem to determine both admission control and resource scheduling problems from a service provider perspective. Overall, the drop-and-add algorithm is the most effective for obtaining the optimal solution in the fewest iterations. The optimization-based energy-efficient admission control and resource allocation algorithms have significant benefits in cloud computing systems. The significance of applications was inspired by the coefficient of (3), which sorts applications by a composition of application weights, client satisfaction, and high availability. Therefore, using the LR-based solution approach with multipliers to obtain the primal feasible solutions was suitable for allocating VMs to servers efficiently and effectively. The mathematical formulation was decomposed into five subproblems. It was solved optimally using parallel computation techniques to reduce computation time substantially. Furthermore, the experimental results reveal and confirm that the drop-and-add algorithm was active in a few minutes by the most significant in the sliced network stages shown in Tables 7–11. The following suggested problems and limitations of this paper can be further studied and solved: QoS requirements can be classified into more categories. Resource sharing between multiple service providers is a new research problem that necessitates considering related issues, such as the sharing economy, from multiple network service provider perspectives. Hence, operator pricing policies could be a worthwhile topic.

**Table 11.** Execution time in the experiments.

Traffic load (number of servers: 80, dual-rate: 0.5)			
<b>Number of applications</b>	<b>60</b>	<b>120</b>	<b>180</b>
min (FF, NF)	0.0418 s.	0.0826 s.	0.1067 s.
Proposed method	373 s.	744 s.	1117 s.
Number of servers (number of users: 80, dual-rate: 0.5)			
<b>Number of servers</b>	<b>40</b>	<b>80</b>	<b>100</b>
min (FF, NF)	0.213 s.	0.159 s.	0.269 s.
Proposed method	236 s.	478 s.	604 s.
Effect of HA (number of applications: 80, number of servers: 80)			
<b>Dual-rate</b>	<b>0.2</b>	<b>0.4</b>	<b>0.6</b>
min (FF, NF)	0.059 s.	0.246 s.	0.430 s.
Proposed method	284 s.	437 s.	583 s.

## 6. Summary and Conclusions

A mathematical programming model is used to develop resource management by simulating the cloud service provider role in this paper's cloud computing systems in 5G network slices. The mathematical model is solved using an LR-based approach. The proposed algorithm increases the cloud computing network infrastructure's flexibility, including cloudlets, and core clouds, to maximize rewards by admitting as many applications as possible. The gaps between upper bounds and lower bounds in the computational experiments demonstrate the drop-and-add heuristic optimal solution qualities. The main contribution is demonstrating that the orchestrator designed the resource management algorithm significantly determined using Lagrangian multipliers to indicate task significance. A promising and straightforward resource allocation approach is proposed to combine client satisfaction and high availability for network planning in a sliced network. The developed optimization-based efficient admission control and resource allocation algorithms are confirmed through various experimental cases. The proposed method has excellent effectiveness and efficiency compared with the LB, FF, and NF solutions based on the experimental results. The resource management mechanisms enables the slicing network as services to efficiently and maximize system revenue in 5G networks.

**Author Contributions:** Conceptualization, C.-H.H. and F.Y.-S.L.; Formal analysis, C.-H.H. and F.Y.-S.L.; Methodology, C.-H.H., F.Y.-S.L., Y.-C.S., Y.-S.W. and H.-Y.K.; Supervision, F.Y.-S.L., Y.-F.W. and Y.H.; Visualization, C.-H.H. and Y.-F.C.; Writing review and editing, C.-H.H., E.S.-H.F., Y.-F.C. and Y.-F.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was supported in part by Ministry of Science and Technology (MOST), Taiwan, under Grant Number MOST 109-2221-E-002-144 and MOST 110-2222-E-001-002.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Ordonez-Lucena, J.; Ameigeiras, P.; Lopez, D.; Ramos-Munoz, J.J.; Lorca, J.; Folgueira, J. Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges. *IEEE Commun. Mag.* **2017**, *55*, 80–87. [[CrossRef](#)]
2. Checko, A.; Christiansen, H.L.; Yan, Y.; Scolari, L.; Kardaras, G.; Berger, M.S.; Dittmann, L. Cloud RAN for Mobile Networks—A Technology Overview. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 405–426. [[CrossRef](#)]
3. Chen, M.; Hao, Y. Task Offloading for Mobile Edge Computing in Software Defined Ultra-Dense Network. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 587–597. [[CrossRef](#)]

4. Taleb, T.; Ksentini, A.; Sericola, B. On Service Resilience in Cloud-Native 5G Mobile Systems. *IEEE J. Sel. Areas Commun.* **2016**, *34*, 483–496. [[CrossRef](#)]
5. Li, X.; Samaka, M.; Chan, H.A.; Bhamare, D.; Gupta, L.; Guo, C.; Jain, R. Network Slicing for 5G: Challenges and Opportunities. *IEEE Internet Comput.* **2017**, *21*, 20–27. [[CrossRef](#)]
6. Al-Fuqaha, A.; Guizani, M.; Mohammadi, M.; Aledhari, M.; Ayyash, M. Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 2347–2376. [[CrossRef](#)]
7. AWS Marketplace. High Availability. Available online: <https://aws.amazon.com/marketplace/solutions/infrastructure-software/high-availability> (accessed on 2 February 2021).
8. Mao, Y.; You, C.; Zhang, J.; Huang, K.; Letaief, K.B. A Survey on Mobile Edge Computing: The Communication Perspective. *IEEE Commun. Surv. Tutor.* **2017**, *19*, 2322–2358. [[CrossRef](#)]
9. Sotiriadis, S.; Bessis, N.; Amza, C.; Buyya, R. Elastic Load Balancing for Dynamic Virtual Machine Reconfiguration Based on Vertical and Horizontal Scaling. *IEEE Trans. Serv. Comput.* **2019**, *12*, 319–334. [[CrossRef](#)]
10. Liu, Y.; Lee, M.J.; Zheng, Y. Adaptive Multi-Resource Allocation for Cloudlet-Based Mobile Cloud Computing System. *IEEE Trans. Mob. Comput.* **2016**, *15*, 2398–2410. [[CrossRef](#)]
11. Jin, A.; Song, W.; Zhuang, W. Auction-Based Resource Allocation for Sharing Cloudlets in Mobile Cloud Computing. *IEEE Trans. Emerg. Top. Comput.* **2018**, *6*, 45–57. [[CrossRef](#)]
12. Liu, L.; Fan, Q. Resource Allocation Optimization Based on Mixed Integer Linear Programming in the Multi-Cloudlet Environment. *IEEE Access* **2018**, *6*, 24533–24542. [[CrossRef](#)]
13. Ahmad, A.; Arshad, R.; Mahmud, S.A.; Khan, G.M.; Al-Raweshidy, H.S. Earliest-Deadline-Based Scheduling to Reduce Urban Traffic Congestion. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 1510–1526. [[CrossRef](#)]
14. Meneguetta, R.I.; Boukerche, A.; Pimenta, A.H.M. AVARAC: An Availability-Based Resource Allocation Scheme for Vehicular Cloud. *IEEE Trans. Intell. Transp. Syst.* **2019**, *20*, 3688–3699. [[CrossRef](#)]
15. Garey, M.R.; Johnson, D.S. *Computers and Intractability: A Guide to the Theory of NP-Completeness*; W. H. Freeman & Co.: San Francisco, CA, USA, 1990.
16. Hou, I.; Gupta, P. Proportionally Fair Distributed Resource Allocation in Multiband Wireless Systems. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1819–1830. [[CrossRef](#)]
17. Ding, T.; Hao, Q.; Zhang, B.; Zhang, T.; Huai, L. Scheduling Policy Optimization in Kernel-Based Virtual Machine. In Proceedings of the International Conference on Computational Intelligence and Software Engineering (CiSE), Wuhan, China, 10–12 December 2010; pp. 1–4.
18. Zhao, L.; Lu, L.; Jin, Z.; Yu, C. Online Virtual Machine Placement for Increasing Cloud Provider’s Revenue. *IEEE Trans. Serv. Comput.* **2017**, *10*, 273–285. [[CrossRef](#)]
19. Jia, M.; Liang, W.; Xu, Z.; Huang, M. Cloudlet Load Balancing in Wireless Metropolitan Area Networks. In Proceedings of the IEEE International Conference on Computer Communications (IEEE INFOCOM), San Francisco, CA, USA, 10–14 April 2016; pp. 1–9.
20. Yao, D.; Gui, L.; Hou, F.; Sun, F.; Mo, D.; Shan, H. Load Balancing Oriented Computation Offloading in Mobile Cloudlet. In Proceedings of the IEEE Vehicular Technology Conference (VTC-Fall), Toronto, ON, Canada, 24–27 September 2017; pp. 1–6.
21. Nguyen, D.D.; Nguyen, H.X.; White, L.B. Reinforcement Learning With Network-Assisted Feedback for Heterogeneous RAT Selection. *IEEE Trans. Wirel. Commun.* **2017**, *16*, 6062–6076. [[CrossRef](#)]
22. Yuan, H.; Bi, J.; Tan, W.; Li, B.H. CAWSAC: Cost-Aware Workload Scheduling and Admission Control for Distributed Cloud Data Centers. *IEEE Trans. Autom. Sci. Eng.* **2016**, *13*, 976–985. [[CrossRef](#)]
23. Bashar, A. BN-Based Approach for Predictive Admission Control of Cloud Services. In Proceedings of the IEEE International Advance Computing Conference (IACC), Hyderabad, India, 5–7 January 2017; pp. 59–64.
24. Hoang, D.T.; Niyato, D.; Wang, P. Optimal Admission Control Policy for Mobile Cloud Computing Hotspot with Cloudlet. In Proceedings of the 2012 IEEE Wireless Communications and Networking Conference (WCNC), Paris, France, 1–4 April 2012; pp. 3145–3149.
25. Xia, Q.; Liang, W.; Xu, W. Throughput Maximization for Online Request Admissions in Mobile Cloudlets. In Proceedings of the IEEE Conference on Local Computer Networks (LCN), Sydney, NSW, Australia, 21–24 October 2013; pp. 589–596.
26. Geoffrion, A. Lagrangian Relaxation and its Uses in Integer Programming. *Math. Program.* **1974**, *2*, 82–114.
27. Bertsekas, D.P. Multiplier Methods: A Survey. *Automatica* **1976**, *12*, 133–145. [[CrossRef](#)]
28. Bertsekas, D.P. *Constrained Optimization and Lagrange Multiplier Methods*; Academic Press: Cambridge, MA, USA, 1982.
29. Hestenes, M.R. Multiplier and Gradient Methods. *J. Optim. Theory Appl.* **1969**, *4*, 303–320. [[CrossRef](#)]
30. Fisher, M.L. The Lagrangian Relaxation Method for Solving Integer Programming Problems. *Manag. Sci.* **2004**, *50*, 1861–1871. [[CrossRef](#)]
31. Rockafellar, R.T. A Dual Approach to Solving Nonlinear Programming Problems by Unconstrained Optimization. *Math. Program.* **1973**, *5*, 354–373. [[CrossRef](#)]
32. Fisher, M.L. An Applications Oriented Guide to Lagrangian Relaxation. *Interfaces* **1985**, *15*, 10–21. [[CrossRef](#)]
33. Xu, X.; Zhang, J.; Ji, Y.; Li, H.; Gu, R.; Yu, H.; Zhang, J. BBU Aggregation for Maximizing the Resource Utilization in Optical-Enabled Cloud Radio Access Networks. In Proceedings of the International Conference on Optical Communications and Networks (ICOON), Hangzhou, China, 24–27 September 2016; pp. 1–3.