Hindawi
Wireless Communications and Mobile Computing
Volume 2018, Article ID 4593208, 11 pages
https://doi.org/10.1155/2018/4593208

WILEY | Hindawi

*Research Article*

# AIMING: Resource Allocation with Latency Awareness for Federated-Cloud Applications

**Jie Wei, Ao Zhou, Jie Yuan, and Fangchun Yang**

*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Haidian, Beijing 100876, China*

Correspondence should be addressed to Jie Wei; wjwb@bupt.edu.cn

Received 27 October 2017; Revised 24 January 2018; Accepted 14 February 2018; Published 10 April 2018

Academic Editor: Anna Kobusinska



Federated-cloud has been widely deployed due to the growing popularity of real-time applications, and hence allocating resources among clouds becomes nontrivial to meet the stringent service requirements. The challenges lie in achieving minimized latency constrained by virtual machines rental overhead and resource requirement. This becomes further complicated by the issues of datacenter selection. To this end, we propose AIMING, a novel resource allocation approach which aims to minimize the latency constrained by monetary overhead in the context of federated-cloud. Specifically, the network resources are deployed and selected according to $k$-means clustering. Meanwhile, the total latency among datacenters is optimized based on binary quadratic programming. The evaluation is conducted with real data traces. The results show that AIMING can reduce total datacenter latency effectively compared with other approaches.

## 1. Introduction

Real-time applications continue to grow rapidly, with evermore functionality and evermore users around the globe. Because of this growth, major real-time application providers now use tens of geographically dispersed datacenters to support service. For early real-time application providers, they have an urgent need to migrate their service from their own servers to distributed datacenters to ensure short latency and compete with new application providers. Hence, the main unmet challenge of leveraging these datacenters is effectively allocating resource and optimizing latency for applications. Usually the resource comes in the form of virtual machines (VM). The application providers deploy different types of VM configurations which consist of specific amount of CPU, memory, and disk resource.

It is challenging to determine the location and the number of VMs for application providers, without any knowledge of the latency among datacenters (since the application providers are in the SaaS layer). However, the cloud providers have their own fiber links to interconnect all major regions, and the latency among datacenters is stable and measurable. By this feature, it is feasible to determine the number of VMs and optimize delivery latency for application providers.

In this paper, a resource allocation approach AIMING ($k$-meAns & bInary quadratic programMING) is proposed for VM placement to optimize latency among datacenters for real-time application providers in federated-cloud. (The federated-cloud is the deployment and management of multiple external and internal cloud computing services to meet business needs.) The AIMING optimizes total latency from the perspective of application providers and provides a VMs rental strategy for real-time application providers, especially for the early providers who need to migrate their services to federated-cloud. We get the measured latency among datacenters and then select datacenters according to the latency. There are several constraints to be addressed.

(1) Choosing the well-connected datacenters with low latency is essential, since there are dozens of datacenters around the world and the paths' latency among these datacenters is different. Latency significantly affects the QoE of real-time application users. Thus, ensuring low latency is the initial aim of the application providers.

(2) Compared to standalone datacenter, different datacenters have different VM rental prices in federated-cloud.

An application provider may choose a cheaper datacenter even if with higher latency. The application providers have their own budget for renting VMs. Hence, latency optimization should be constrained by the limitation of monetary overhead.

(3) In addition to the VMs rental overhead, the network overhead also should be considered. Real-time applications such as VoIP may generate accumulative huge data along with the increasing duration time since the data interaction exists among VMs located in different datacenters.

Many latency optimization approaches have been proposed for latency-sensitive applications in federated-cloud scenario. Hao et al. [1] have proposed an online algorithm to guarantee the service latency. Guo et al. [2] and Grozev and Buyya [3] optimize latency for virtual desktops and web applications, respectively, by calculating. Aral and Ovatman [4] have optimized the latency for interdatacenters and intradatacenters by heuristic algorithm. Alicherry and Lakshman [5] have proposed an algorithm to minimize network latency and bandwidth utilization based on mapping VM clusters onto the federated-cloud. Compared with existing resource allocation approaches, the main contributions of our work are listed as follows:

(1) We propose the AIMING to allocate resource (VMs and network) from the perspective of application providers with different types of VMs taken into consideration.

(2) The distribution of federated-cloud is not uniform. We use $k$-means to classify the datacenters into several regions according to real-world latency data traces, which is better than functional calculation method. The classification is more accurate and effective than graph theory since we use within-cluster sum of squares (WCSS) to evaluate.

(3) The monetary overhead and latency are a couple of tradeoffs. The application providers may limit budget for each region of datacenters. Small budget may lead to higher latency. We optimize the latency under the constraint of budget.

This paper is aimed at advancing the current state-of-the-art researches in latency optimization of real-time applications. The proposed AIMING is two-fold: First, $k$-means is used to classify the datacenters according to latency. Then, the binary quadratic programming is utilized to place VMs for application providers in federated-cloud.

The rest of this paper is organized as follows. Section 2 introduces the related work. In Section 3, we use $k$-means to classify the datacenters. Then, we use binary quadratic programming to place VMs in Section 4. Section 5 presents the implementation, experiment, and analysis of the proposed approach, and we conclude the paper in Section 6.

## 2. Related Work

A number of schemes have been proposed for latency and monetary overhead optimization in cloud. In this section, we briefly review previous works related to this work from three aspects, namely, standalone cloud resource allocation, federated-cloud resource allocation, and mobile cloud resource allocation.

*2.1. Standalone Cloud Resource Allocation.* Many works allocate VMs in standalone cloud to reduce latency and monetary overhead. Meng et al. [6] have proposed a two-tier approximation algorithm to efficiently place VMs. The algorithm consists of two components: SlotClustering and VMMinKcut. SlotClustering is to partition $n$ slots into $k$ clusters by using the cost between slots as the partition criterion. VMMinKcut is to partition $n$ VMs into $k$ VM clusters with minimum intercluster traffic. Fang et al. [7] have designed VMPlanner to optimize both VM placement and traffic flow routing so as to turn off unneeded network elements for power saving. An offline VM placement through emulated VM migration is proposed by Li et al. [8]. As long as the suitable physical machine has enough capacity, the migration algorithm can place the VM to its best physical machine directly. Furthermore, they study a hybrid scheme by employing a batch to accept upcoming VMs for online scenarios. Ant colony is deployed by Gao et al. for resource allocation in cloud. Both of them optimize the power consumption of CPU utilization.

In addition, Gao et al. [9, 10] also take the potential cost of wasted resources into consideration. A cloud resource allocation based on imperfect information Stackelberg game (IISG) and hidden Markov model (HMM) has been proposed by Wei et al. [11]. They firstly use the HMM to predict the service provider's current bid based on demand. Then they establish the IISG to choose the optimal bidding strategy and achieve maximum profits. Li et al. [12] focus on cost optimization of network traffic and physical machines utilization. They solve the VM placement problem from two aspects. The first aspect is putting emphasis on the cost of network traffic with fixed physical machines cost for two cases (the number of requested VMs is same or different). The second aspect is placing VMs in general case and taking both network cost and physical machine utilization cost into consideration. Bandwidth allocation is considered by Tian et al. and Yu et al. A distributed host-based bandwidth allocation design with underlay congestion control layer and private congestion control layer has been presented by Tian et al. [13]. To address the issue that existing works do not consider the impact of primary embedding on backup resource consumption, Yu et al. [14] have proposed a novel algorithm to compute the most efficient resource embedding given a tenant request.

*2.2. Federated-Cloud Resource Allocation.* Some notable schemes are proposed for federated-cloud resource allocation. Jiao et al. and Wu and Madhyastha optimize the network traffic by resource allocation in federated-cloud. Multiobjective optimization for placing users' data over multiple clouds for socially aware services has been studied by Jiao et al. [15]. They build a framework that can accommodate different objectives. An optimization approach based on graph cuts is leveraged for decomposing their original problem. To minimize user perceived latency, Wu and Madhyastha [16] have conducted a deep study of latency benefits when deploying web-services across AWS, Microsoft Azure, and Google Compute Engine. Their measurement shows that it is necessary to replicate data to ensure low latency for users. The capacity of cloud infrastructure has been studied by Narayanan et al. and Tordsson et al. Towards a leaner

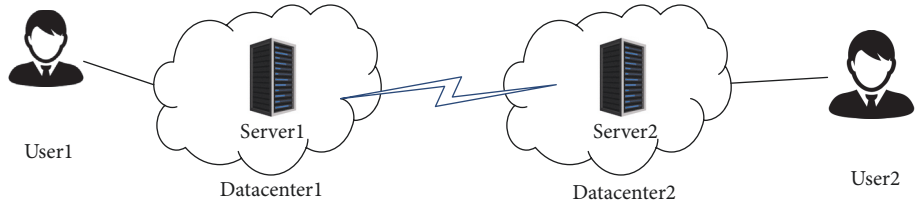


Figure 1: Distributed clouds communication network model for real-time application users.

geodistributed cloud infrastructure, Narayanan et al. [17] discuss several factors that influence the capacity provisioning and illustrate the research challenges for software design.

A novel cloud brokering approach has been proposed by Tordsson et al. [18] to optimize the placement of virtual infrastructure across multiple clouds. In a high throughput computing cluster case, they verify the feasibility of the approach. The communication overhead is considered by Lee et al. and Yi et al. Lee et al. [19] have proposed a distributed resource allocation approach for resource competition in federated-cloud. In their approach, each job consists of tasks and the communication behavior among tasks can be profiled. According to the communication behavior, their approach minimizes the communication overhead and tries to allocate grouped tasks to get balance in the case of resource competition. From the perspective of customers, Yi et al. [20] focus on network-aware resource allocation and develop a mixed binary quadratic programming optimal model to minimize the rental cost for each customer. Except for energy, the bandwidth is also considered by Xu and Li [21].

To solve the large-scale optimization, a distributed algorithm based on alternating direct method of multipliers has been proposed. To achieve energy efficiency and satisfy deadline constraints, Ding et al. [22] focus on dynamic scheduling of VMs. Mihailescu et al. and Wang et al. present pricing schemes for users according to their resource need. Mihailescu and Teo [23] have discussed a strategic-proof dynamic pricing scheme suitable for allocating resources in federated-cloud. A new cloud brokerage service is proposed by Wang et al. [24] based on leveraging both pricing benefits and multiplexing gains. The proposed service reserves a large pool of instances and serves users with price discounts. Hung et al. [25] coordinate job scheduling across datacenters with low overhead, while achieving near-optimal performance. A data hosting scheme has been proposed by Zhang et al. [26] to select several clouds to store data with monetary minimizing. Ardagna et al. [27] use game-theory to management runtime of resources from multiple IaaS providers to multiple SaaS with a revenue and penalty cost model. Jiao et al. [28] allocate and reconfigure resource in the multitier resource pool. And Palmieri et al. [29] optimize the overall communication and runtime resource utilization of cloud infrastructures by reoptimizing the communication paths between VMs and big data sources.

*2.3. Mobile Cloud Resource Allocation.* The resource allocation problem is also a big challenge for mobile cloud environment. Gai et al. [30] have proposed a dynamic energy-aware cloudlet-based mobile cloud computing model concentrating on energy consumption during the wireless communications. In their work, they have solved energy problem with dynamic network and provided guidelines and theoretical supports. To efficiently handle the peak load and satisfy the requirements of remote program execution, Tong et al. [31] have deployed cloud servers at the network edge and designed the edge cloud as a tree hierarchy of geodistributed servers. Furthermore, a workload placement algorithm is proposed to decide which edge cloud servers mobile programs should be placed on. There are also many works for mobile service optimization [32–34].

Different from the previous research, AIMING allocates resources by latency optimization with monetary overhead constraint. The resources contain CPU and memory. We combine the $k$-means and binary quadratic programming to solve the optimization problem.

## 3. Preliminary

To facilitate presenting the proposed approach, we first define some terms and notations that will be used for the paper in Notations. Then, we illustrate latency model and monetary overhead model for AIMING.

*3.1. Latency Model.* In this section, we introduce our latency model in detail. We first discuss the communication model of real-time application among two users in federated-cloud through datacenters. Based on this communication model, we then model the latency of data transfer among VMs.

We assume that a real-time application is distributed on federated-cloud and provides service for users through the datacenter network. As shown in Figure 1, user1 and user2 are users of real-time application and server1 and server2 are distributed in datacenter1 and datacenter2, respectively. Server1 in datacenter1 is the nearest available application server to user1, and it is the same as server2 and datacenter2. The message source user1 connects with destination user2 through datacenter1 and datacenter2.

In the federated-cloud scenario, each VM may contact with each other and transfer data. We first show the latency model between two VMs for data transmission. Let $\mathrm{VM}^{ij}{}_p$ denote $p$th VM of $i$th instance type in $j$th datacenter. And it is the same as $\mathrm{VM}_q{}^{lk}$. $l(\mathrm{VM}_p{}^{ij}, \mathrm{VM}_q{}^{lk})$ is the latency between $\mathrm{VM}_p{}^{ij}$ and $\mathrm{VM}_q{}^{lk}$. $v(\mathrm{VM}_p{}^{ij}, \mathrm{VM}_q{}^{lk})$ is the transmission data volume between $\mathrm{VM}_p{}^{ij}$ and $\mathrm{VM}_q{}^{lk}$. We get the latency by ping. $t_{jk}(\mathrm{ping})$ is the time of ping, and $v(\mathrm{packet})$ is the packet

size of ping. Through $t_{jk}$(ping) and $v$(packet), we can model the latency between $\mathrm{VM}_p^{ij}$ and $\mathrm{VM}_q^{lk}$ as follows:

$$l\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) = v\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot \frac{t_{jk}(\mathrm{ping})}{v(\mathrm{packet})} \cdot x_p^{ij} \cdot x_q^{lk}, \tag{1}$$

where $x_p^{ij} \in \{0, 1\}$ and $x_p^{ij} = 1$ indicates that $\mathrm{VM}_p^{ij}$ is selected to rent for service; otherwise $x_p^{ij} = 0$. And it is the same as $x_q^{lk}$. The latency between VMs approximately equals the latency between datacenters where VMs are located, since the datacenter is extremely large. The formula of total latency can be shown as follows:

$$l_{\mathrm{total}} = \sum_{j,k=1}^{n_{\mathrm{dc}}} \sum_{i,l=1}^{n_{\mathrm{type}}} \sum_{p,q=1}^{n(\mathrm{VM}^{ij})} l\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right). \tag{2}$$

$n_{\mathrm{dc}}$ is the number of datacenters where VMs are placed. $n_{\mathrm{type}}$ is the number of the types of VM instances. $n(\mathrm{VM}^{ij})$ is the VM number of $i$th instance types in $j$th datacenter. We consider the data communication among each VM in formula (2). After finishing the latency model, we show the monetary overhead model in the following Section 3.2.

*3.2. Monetary Overhead Model.* We depict the monetary overhead model in this section. First, we discuss the data communication mode among VMs and pricing mechanism of federated-cloud. Figure 2 shows the data communication model between two VMs. VM 1 and 2 have different rental prices, and the data communication also needs expense. Thus, the total monetary overhead consists of two parts: VM rental monetary overhead and data transmission monetary overhead.

In addition, for federated-cloud scenario, different cloud providers have different prices. Even if the datacenters belong to the same cloud provider, they may have different VM rental prices and data transmission prices. The pricing mechanism of federated-cloud is shown in Figure 3. There are three cloud providers A, B, and C, and each provider has two datacenters. Users can get the prices of datacenters from the pricing module. They can choose the datacenters to rent VMs according to the pricing module.

According to the above discussion, the total monetary overhead consists of two parts: VM rental monetary overhead and data transmission monetary overhead. Different datacenters have different VM rental prices and data transmission prices. We model the VM rental monetary overhead and data transmission monetary overhead separately as follows:

$$m_{\mathrm{rental}} = \sum_{j=1}^{n_{\mathrm{dc}}} \sum_{i=1}^{n_{\mathrm{type}}} \sum_{p=1}^{n(\mathrm{VM}^{ij})} \left[p\left(\mathrm{VM}_p^{ij}\right) \cdot t\left(\mathrm{VM}_p^{ij}\right) \cdot x_p^{ij}\right], \tag{3}$$

$$m_{\mathrm{transmission}} = \sum_{j,k=1}^{n_{\mathrm{dc}}} \sum_{i,l=1}^{n_{\mathrm{type}}} \sum_{p,q=1}^{n(\mathrm{VM}^{ij})} \left[p'\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot v\left(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk}\right) \cdot x_p^{ij} \cdot x_q^{lk}\right], \tag{4}$$

$$m_{\mathrm{total}} = m_{\mathrm{rental}} + m_{\mathrm{transmission}}. \tag{5}$$

$m_{\mathrm{rental}}$ of formula (3) represents the rental monetary overhead and $m_{\mathrm{transmission}}$ of formula (4) represents the data transmission monetary overhead. $m_{\mathrm{total}}$ of formula (5) represents the total monetary overhead. $p(\mathrm{VM}_p^{ij})$ denotes the rental price of $\mathrm{VM}_p^{ij}$. $t(\mathrm{VM}_p^{ij})$ denotes the rental duration time of $\mathrm{VM}_p^{ij}$. $p'(\mathrm{VM}_p^{ij}, \mathrm{VM}_q^{lk})$ denotes the data transmission price between $\mathrm{VM}_p^{ij}$ and $\mathrm{VM}_q^{lk}$.

## 4. Proposed Approach

In this section, we illustrate our resource allocation approach AIMING. In real-world, most real-time communications happen in a region, such as a country. Thus, the communications in a region are far more frequent than the communications among regions. Based on this assumption, we optimize the total latency according to regions. We first divide the datacenters into $K$ regions; then resource is allocated for each region to minimize the total latency.

*4.1. Datacenter Preprocess.* In this section, we preprocess $N$ datacenters by dividing the datacenters into $K$ regions. The distribution of datacenters is heterogeneous. Thus, for convenience, we divide the datacenters according to the end-to-end latency $L = t(\mathrm{ping})/v(\mathrm{packet})$. We use $k$-means clustering to divide the datacenters into $K$ regions. Given a set of datacenter latency observations $(\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_N)$, where each observation is a $N$-dimensional real vector, $k$-means clustering aims to partition the $N$ observations into $K$ ($\leqslant N$) sets $\mathbf{S} = \{S_1, S_2, \ldots, S_K\}$ so as to minimize the within-cluster sum of squares (WCSS). Hence, the objective is to find

$$\arg\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{L \in \mathbf{S}_i} \left\| L - \mu_i \right\|^2, \tag{6}$$

where $\mu_i$ is the mean of points in $S_i$.

Figure 4 is an example of datacenters clustering when the number of datacenters $N$ equals 5. We divide the datacenters into $K$ regions ($K$ is equal to 2). The latency between two

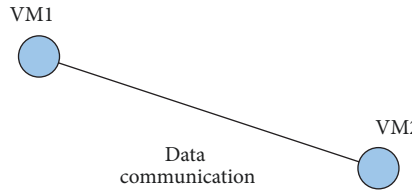


Figure 2: Data communication mode between two VMs.

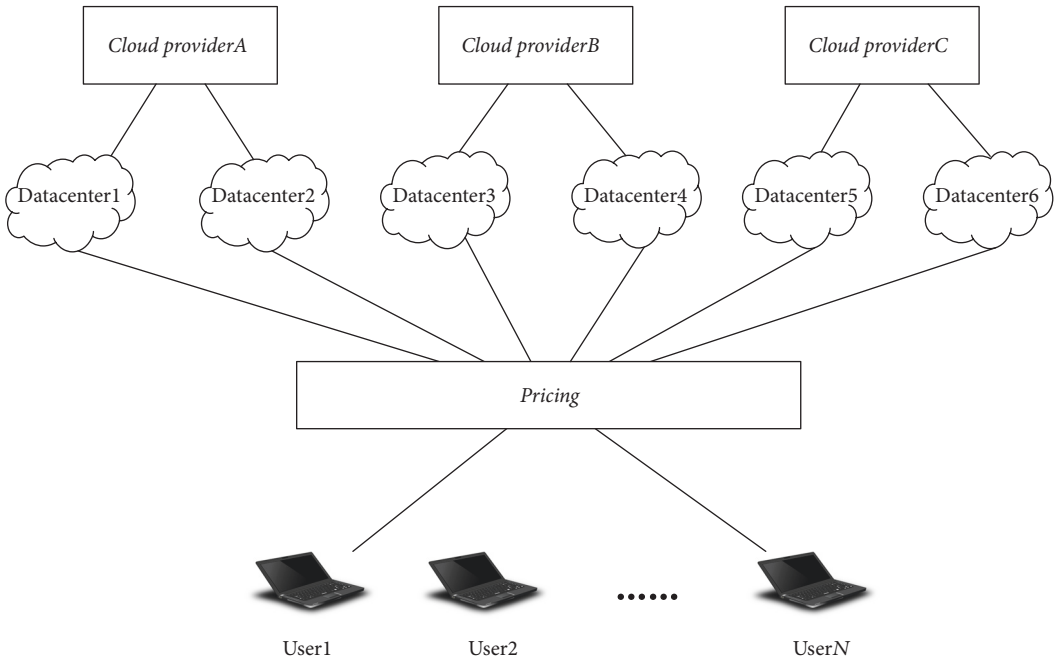


Figure 3: Pricing mechanism of federated-cloud.

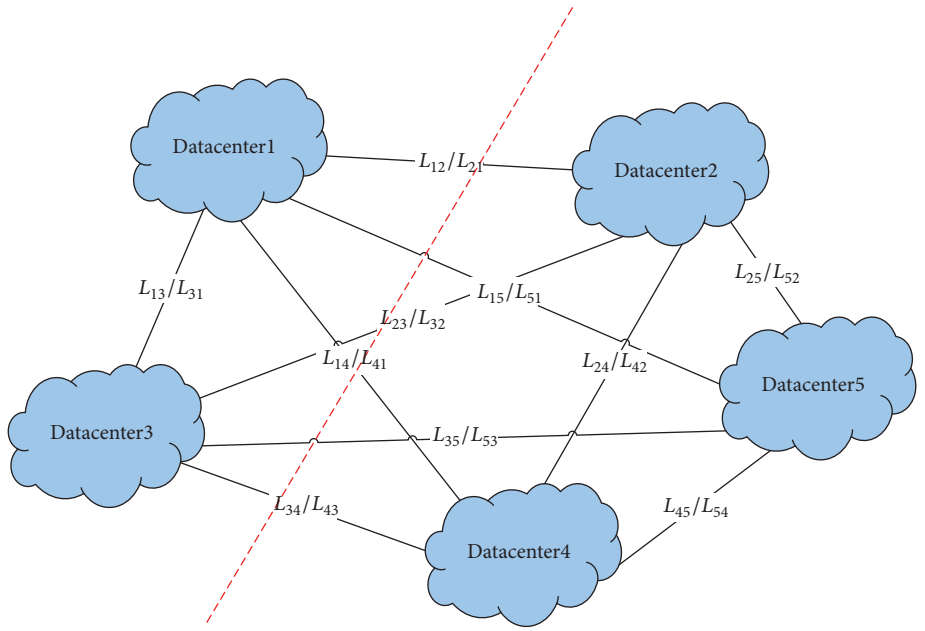


Figure 4: An example of datacenters clustering when $N$ is equal to 5 and $K$ is equal to 2.

Table 1: End-to-end latency of datacenters.

| Latency | Datacenter1 | Datacenter2 | Datacenter3 | Datacenter4 | Datacenter5 |
|---|---|---|---|---|---|
| Datacenter1 | $L_{11}$ | $L_{12}$ | $L_{13}$ | $L_{14}$ | $L_{15}$ |
| Datacenter2 | $L_{12}$ | $L_{22}$ | $L_{23}$ | $L_{24}$ | $L_{25}$ |
| Datacenter3 | $L_{13}$ | $L_{23}$ | $L_{33}$ | $L_{34}$ | $L_{35}$ |
| Datacenter4 | $L_{14}$ | $L_{24}$ | $L_{34}$ | $L_{44}$ | $L_{45}$ |
| Datacenter5 | $L_{15}$ | $L_{25}$ | $L_{35}$ | $L_{45}$ | $L_{55}$ |

datacenters is nondirectional, which means that $L_{ij} = L_{ji}$, $1 \leqslant i$, and $j \leqslant 5$. We get the sets $\mathbf{S} = \{S_1, S_2\}$ which are the two parts shown in Figure 4. The latency set of datacenters is shown in Table 1. Each observation is a 5-dimensional real vector, $L_i = \{L_{i1}, L_{i2}, L_{i3}, L_{i4}, L_{i5}\}$, $1 \leqslant i \leqslant 5$.

*4.2. Latency Optimization of Datacenter Region.* In this section, we minimize the total latency of each region by placing VMs. The region total latency optimization problem is modeled as binary quadratic programming. For region $S_r$, $N_r$ represents the total need number of datacenters of region $S_r$. The number of selected datacenters of region $S_r$ is represented as $N_r' \leqslant N_r$. The purpose is to select $N_r'$ datacenters from $N_r$ and place VMs to minimize the total latency of the region. We assume that the capacity need of each region for customer request is known to us. The capacity performance of a given instance type $i$ is denoted as $C_i$. Hence, the objective of region $S_r$ can be shown as follows:

$$\min \quad l_{\text{total}}^{r} = \min \sum_{j,k=1}^{N_r} \sum_{i,l=1}^{n_{\text{type}}} \sum_{p,q=1}^{n(\text{VM}^{ij})} v\left(\text{VM}_p^{ij}, \text{VM}_q^{lk}\right) \cdot \frac{t_{jk}\left(\text{ping}\right)}{v\left(\text{packet}\right)} \cdot x_p^{ij} \cdot x_q^{lk}. \tag{7}$$

Users can specify the following types of deployment constraints as follows:

$$\sum_{j=1}^{N_r} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \geq N^{ir}{}_{\text{vm}} \quad i = 1, 2, \ldots, n_{\text{type}}, \tag{8}$$

$$\sum_{i=1}^{n_{\text{type}}} C_i \cdot \left( \sum_{j=1}^{N_r} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \right) \geq C^{r}{}_{\text{need}}, \tag{9}$$

$$N^{\min j}{}_{\text{vm}} \leq \sum_{i=1}^{n_{\text{type}}} \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \leq N^{\max j}{}_{\text{vm}} \quad j = 1, 2, \ldots, N_r, \tag{10}$$

$$C^{\min j}{}_{\text{vm}} \leq \sum_{i=1}^{n_{\text{type}}} C_i \cdot \left( \sum_{p=1}^{n(\text{VM}^{ij})} x_p^{ij} \right) \leq C^{\max j}{}_{\text{vm}} \quad j = 1, 2, \ldots, N_r, \tag{11}$$

$$m_{\text{total}}^{r} = m_{\text{rental}}^{r} + m_{\text{transmission}}^{r} \leq \text{Budget}_r. \tag{12}$$

(1) *VM hardware configuration constraints are expressed by restricting the number of each instance type and total infrastructure capacity need*. Formula (8) is the number of VMs of each instance type constraint. $N^{ir}{}_{\text{vm}}$ is the minimum VMs number of instance type $i$ of region $S_r$. Formula (9) is the total infrastructure capacity constraint of region $S_r$. $C^{r}{}_{\text{need}}$ is the minimum infrastructure capacity need of region $S_r$.

(2) *Load balancing constrains are expressed as VM number and infrastructure capacity need of each datacenter*. Formula (10) is the VM number constraint of each datacenter. Users can determine this type of constraint of datacenter $j$ by $N^{\min j}{}_{\text{vm}}$ and $N^{\max j}{}_{\text{vm}}$, the minimum and maximum VMs number of datacenter $j$, $1 \leqslant j \leqslant N'{}_r$. Formula (11) is the infrastructure capacity need constraint of each datacenter. Users can determine this type of constraint of datacenter $j$ by $C^{\min j}{}_{\text{vm}}$ and $C^{\max j}{}_{\text{vm}}$, the minimum and maximum capacity need of datacenter $j$, $1 \leqslant j \leqslant N'{}_r$.

(3) *Monetary overhead constraint is expressed as budget of VM rental overhead and data transmission overhead*. $\text{Budget}_r$ of formula (12) is the monetary constraint of region $S_r$.

Moreover, the number of selected datacenters for region $S_r$ is $N_r'$. And we should note that each VM has to be exactly one instance type and placed in exactly one cloud.

**Lemma 1.** *The region total latency optimization problem is NP-hard.*

*Proof.* We prove the NP-hardness by reduction from 0-1 Knapsack problem. Given a knapsack instance with $n$ items, each with a weight $w_i$ and a value $v_i$, along with a maximum weight capacity $W$, we create the region total latency optimization problem as follows. The number of datacenters $N_r$ and $N_r'$ are equal to 1. The CPU of each type of VM instance is equal to 0, which means no infrastructure capacity herein. And the VM rental price is particular. There are $n$ links among VMs corresponding to the items. Assuming that the latency of each link between VMs is $L_i$, then $-L_i$ is corresponding to value $v_i$ and data transmission price is corresponding to weight $w_i$. The budget constraint is the only limitation of the datacenter, which is corresponding to maximum weight capacity $W$. Finally, 0-1 knapsack problem can be reduced to the region total latency optimization problem. Thus, the region total latency optimization problem is NP-hard. □

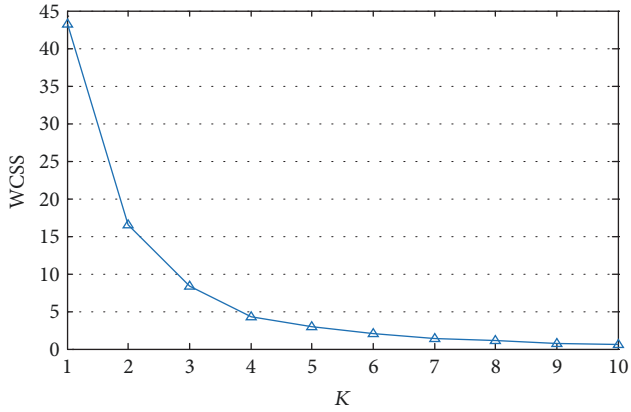


Figure 5: The WCSS of datacenter preprocess.

Table 2: VM performance parameters.

| Cloud provider | Instance type | CPU | Memory (GiB) |
|---|---|---|---|
| Azure | D1 v2 | 1 | 3.5 |
| | D2 v2 | 2 | 7 |
| | D3 v2 | 4 | 14 |
| | D4 v2 | 8 | 28 |
| Google | n1-standard-1 | 1 | 3.75 |
| | n1-standard-2 | 2 | 7.5 |
| | n1-standard-4 | 4 | 15 |
| | n1-standard-8 | 8 | 30 |
| Amazon | t2.small | 1 | 2 |
| | t2.large | 2 | 8 |
| | t2.xlarge | 4 | 16 |
| | t2.2xlarge | 8 | 32 |

There are multiple model languages and solvers which can be used to solve the optimization problem. We choose the AMPL as modelling language. AMPL has good support for sets and its syntax is close to mathematical notation. Since AMPL can also be used for various types of optimization problems for quadratic programming formulations described above, we use the CPLEX solver.

## 5. Performance Evaluation

The AIMING is based on federated-cloud environment that is hard to be implemented in practice due to its large network. We evaluate AIMING by simulation and compare it with several approaches. The evaluation shows total latency and rental overhead under various numbers of VMs and datacenters. Parameters analysis includes budget and deviation of VM number. Extensive evaluation results show that AIMING is efficient in total latency reduction.

*5.1. Experimental Setup.* (1) Using python with PyCharm and based on real-world latency data trace, the data preprocess runs in a variety of realistic settings. We set up a federated-cloud environment with 22 datacenters and 3 cloud providers including Amazon Web Service, Microsoft Azure, and Google Cloud Platform. The 9 datacenters of Microsoft Azure are South USA, Central USA, Sao Paulo, Dublin, Amsterdam, East Asia, Japan West, Singapore, and East Australia. The 4 datacenters of Google are Taiwan, Europe, central USA, and East USA. The other 9 datacenters of Amazon are Oregon, Virginia, California, Sao Paulo, Dublin, Frankfurt, Tokyo, Singapore, and Sydney. We use the interdatacenters latency dataset measured by Mansouri and Buyya [35] from Cloud Laboratory of The University of Melbourne. They have measured the latency between 22 datacenters for 8 hours. The packet size is between 64 bytes and 1 KB and the interval time between each packet is 4 seconds. The latency dataset is measurable and reliable because the cloud providers have their own fiber links interconnecting all major regions. Each network topology of datacenter is fat-tree structure.

Figure 5 shows the WCSS with variety $K$. The number of datacenters $N$ is equal to 22. In Figure 5, the WCSS is below 5 after $K$ equals 4. Thus, we choose 4 as the value of $K$ and get the clustering result as follows.

Cluster 1 contains 5 datacenters: Azure Dublin, Azure Amsterdam, Google Europe, Amazon Dublin, and Amazon Frankfurt.

Cluster 2 contains 8 datacenters: Azure East Asia, Azure Japan West, Azure Singapore, Azure Australia East, Google Taiwan, Amazon Tokyo, Amazon Singapore, and Amazon Sydney.

Cluster 3 contains 7 datacenters: Azure South USA, Azure Central USA, Google Central USA, Google East USA, Amazon Oregon, Amazon Virginia, and Amazon California.

Cluster 4 contains 2 datacenters: Azure Sao Paulo and Amazon Sao Paulo.

Given that each region has the similar trend, we conduct the latency optimization algorithm on region 1 with 5 datacenters in the following part.

(2) In this section, we use AMPL with CPLEX to implement the latency optimization algorithm. Four VM instance types are considered in the evaluation. For the rental VM configuration, the performance parameters are listed in Table 2. As for the rental price and data transmission price, we use the price from their websites [36–38]. We conduct the latency optimization algorithm on 5 datacenters of cluster 1. For convenience, we define a variable, the number of VMs for each instance type, to replace the total VMs number. Only 4 VM instance types are considered here; then the total number of VMs is 4 multiples.

In order to evaluate the performance of AIMING, we compare it with two approaches in terms of total latency and monetary overhead:

(i) Random: this approach is a random method with the constraints of VM number and capacity need.

(ii) Greedy: this approach selects $N_r'$ datacenters with cheapest VM rental price to allocate VMs [39].

(iii) AIMING: our approach is based on binary quadratic programming.

*5.2. Total Latency.* In this section, we show the optimal result of total latency with various numbers of VMs and datacenters.

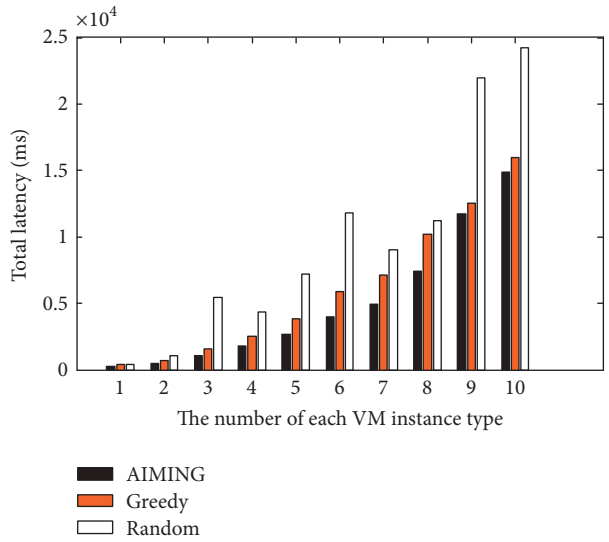


Figure 6: The total latency with various numbers of each VM instance type, which ranges from 1 to 10.

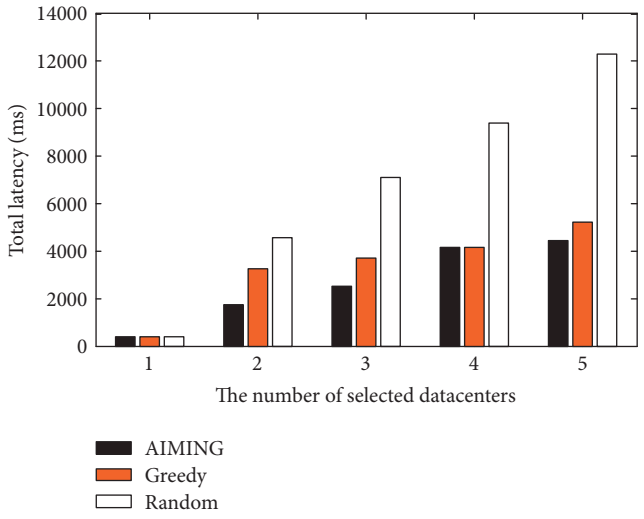


Figure 7: The total latency with various numbers of selected datacenters, which ranges from 1 to 5.

As shown in Figure 6, the total latency of AIMING is better than other two approaches. The total latency of AIMING is about 14906 ms when the number of each VM instance type is 10; however, other approaches (e.g., Random, at about 24198 ms, and Greedy, at about 15977.4 ms) are higher. We set the number of datacenters as 3. The total latency increases along with the number of each VM instance type, which ranges from 1 to 10. However, the difference value between AIMING and Greedy also increases along with the number of VMs. This is because when the number of each VM instance type increases, the data transmission need increases vastly, which leads to higher difference value of total latency.

Figure 7 shows the relationship between total latency and number of selected datacenters. The number of each VM instance type is equal to 5, which means that there are 20 VMs

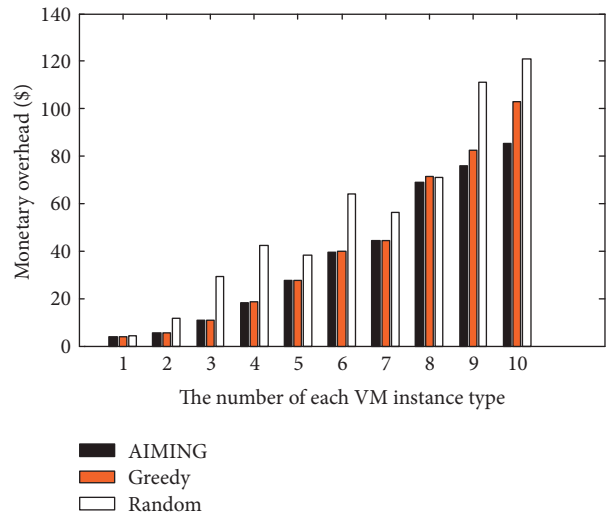


Figure 8: The monetary overhead with various numbers of each VM instance type, which ranges from 1 to 10.

to allocate. When the number of selected datacenters is equal to 1, only the datacenter can be chosen. Then, AIMING, Greedy, and Random have the same total latency which is 400 ms. The Greedy ignores the transmission latency because it often chooses the cheapest datacenters. The AIMING can decrease a maximum 46.1% latency time of Greedy approach when the number of selected datacenters is 2.

### 5.3. Parameter Analysis

*(1) Monetary Overhead and Number of Each VM Instance Type.* Figure 8 describes the relationship between monetary overhead and the number of each VM instance type. We set the number of selected datacenters as 3 which is the same as Figure 6. The Greedy and AIMING nearly have the same rental overhead until the number of each VM instance type equals 6. The reason is as follows: when the number of each VM instance type is not large enough, the VM rental overhead plays a more important role in monetary overhead than data transmission overhead. After the number of each VM instance type equals 6, the data transmission overhead occupies a large proportion and AIMING is obviously better than Greedy and Random. When the number of each VM instance type is 10, the monetary overheads of AIMING, Greedy, and Random are $85.275, $103.1247, and $120.8789.

*(2) Monetary Overhead and Number of Selected Datacenters.* The relationship between monetary overhead and selected datacenters number is shown in Figure 9. The number of VM for each instance type is equal to 5, which is the same with Figure 7. When the number of selected datacenters is 3, the AIMING starts to show its advantage in monetary reduction more than Greedy. When the number of selected datacenters is equal to 3, it will lead to frequent data interaction among VMs and data transmission overhead becomes the major expense. The monetary overhead of the three approaches increases along with the number of selected datacenters.

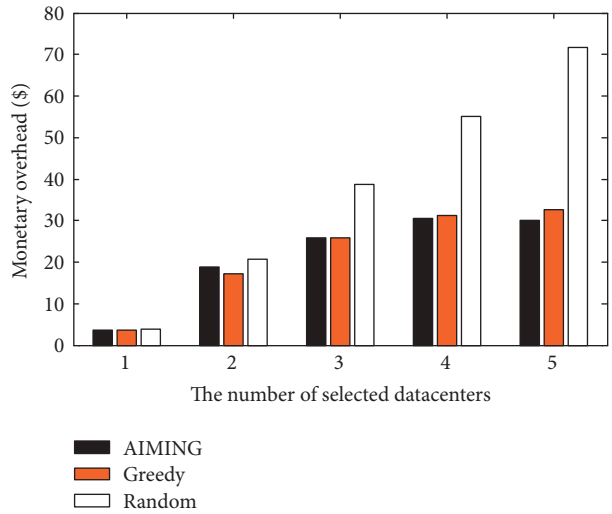


Figure 9: The monetary overhead with various numbers of selected datacenters, which ranges from 1 to 5.

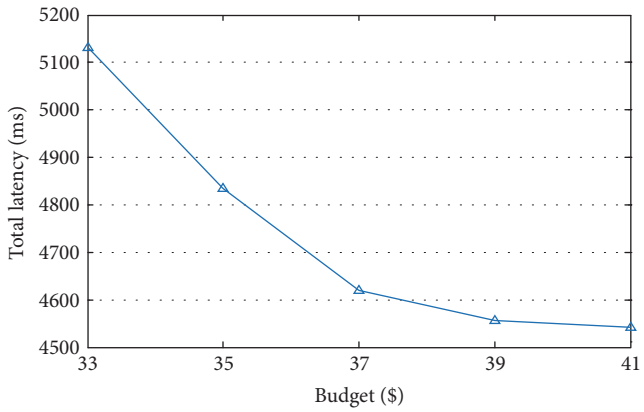


Figure 10: The total latency with budget ranges from $33 to $41.

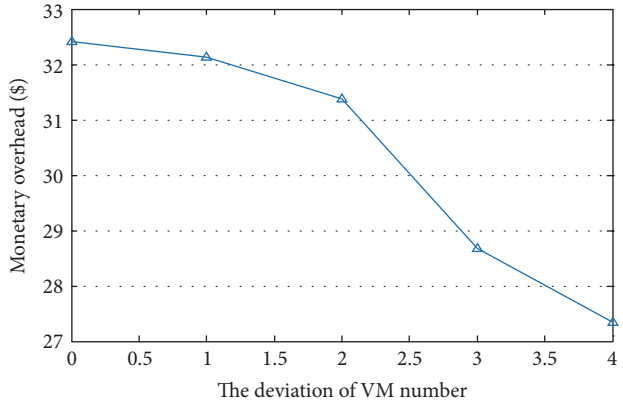


Figure 11: The monetary overhead with various deviations of VM number ranges from 0 to 4.

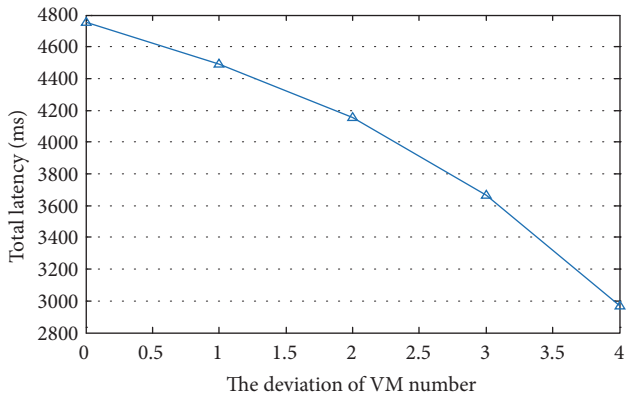


Figure 12: The total latency with various deviations of VM number ranges from 0 to 4.

When the number of selected datacenters is equal to 5, the monetary overheads of AIMING, Greedy, and Random are $30.1725, $32.538, and $71.6723.

*(3) Total Latency and Budget.* The relationship of total latency and budget is shown in Figure 10. The total latency decreases when budget increases. The number of selected datacenters is set to be 5. The number of each VM instance type is 5, which means that there are 20 VMs to be allocated. We control the budget ranging from $33 to $41, and the difference of latency decreases from 5130.8 ms to 4542.9 ms. The decline is caused by relaxing budget constraint. With the increase of budget, the VM allocation choice scope becomes larger. The better allocation solution will appear and lead to lower total latency.

*(4) Monetary Overhead and Deviation of VM Number.* Using deviation of VM number, we discuss the VM number constraint of each datacenter. In Figures 11 and 12, the number of each VM instance type is equal to 5 and the number of selected datacenters is 5, which that means there are 20 VMs to be allocated in 5 datacenters. To explain the deviation of VM number, we take an example: if the deviation of VM number is 0, the constraint of VM number of each datacenter is no more than 5 and no less than 5, which means that the constraint of VM number of each datacenter is 5. If the deviation of VM number is equal to 1, the constraint of VM number of each data center is no more than 6 and no less than 4. In Figure 11, the monetary overhead decreases from $32.4114 to $27.3404 when the deviation of VM number increases from 0 to 4. It is because the increasing of deviation will enlarge the solution scope. Hence, the curve shows an impressive decline after the deviation is equal to 2.

*(5) Total Latency and Deviation of VM Number.* In Figure 12, the total latency decreases along with increase of VM number deviation. The reason is as follows: when the deviation of VM number increases, the VM allocation scope becomes larger. Thus, better allocation solution will decrease the total latency, and the curve declines after deviation equals 2. The total latency decreases from 4755 ms to 2967.8 ms when the deviation of VM number increases from 0 to 4.

## 6. Conclusion

In this paper, we propose an optimization approach named AIMING to optimize the total latency in federated-cloud scenario. The approach includes two steps: datacenter preprocess and latency optimization. The first step is based on $k$-means clustering, and quadratic programming is used for the latter step. With real latency among datacenters, VM rental price, and data transmission price as input, we use python and AMPL to conduct the evaluation. The evaluation shows that AIMING can reduce the total latency and monetary overhead efficiently. Besides, we also do a lot of parameter analyses to show more details and impact factors.

## Notations

| | |
|---|---|
| ${VM_p}^{ij}$: | $p$th VM of $i$th type in $j$th datacenter |
| $n(VM^{ij})$: | The VM number of $i$th instance type in $j$th datacenter |
| $p({VM_p}^{ij})$: | The rental price of $p$th VM of $i$th instance type in $j$th datacenter |
| $p'({VM_p}^{ij}, {VM_q}^{lk})$: | The data transmission price between ${VM_p}^{ij}$ and ${VM_q}^{lk}$ |
| $l({VM_p}^{ij}, {VM_q}^{lk})$: | The latency between ${VM_p}^{ij}$ and ${VM_q}^{lk}$ |
| $t({VM_p}^{ij})$: | The rental duration time of ${VM_p}^{ij}$ |
| $v({VM_p}^{ij}, {VM_q}^{lk})$: | The transmission data volume between ${VM_p}^{ij}$ and ${VM_q}^{lk}$ |
| $v$(packet): | The packet size of ping |
| $t_{jk}$(ping): | The latency of ping from $j$th datacenter to $k$th datacenter |
| $n_{\text{dc}}$: | The number of datacenters which are taken into consideration |
| $n_{\text{type}}$: | The number of the VM instance type |
| $m_{\text{total}}$: | Total monetary overhead |
| $m_{\text{rental}}$: | VM rental monetary overhead |
| $m_{\text{transmission}}$: | Data transmission monetary overhead |
| $l_{\text{total}}$: | The total latency of datacenters |
| ${x_p}^{ij}$: | If ${VM_p}^{ij}$ is selected to be rented, ${x_p}^{ij} = 1$; otherwise, ${x_p}^{ij} = 0$ |
| $N_r$: | Total number of datacenters of region $S_r$ |
| $S_r$: | $r$th datacenter region of clustering |
| ${N'}_r$: | The number of selected datacenters of $S_r$ |
| ${N^{ir}}_{\text{vm}}$: | Minimum number of instance types $i$ of region $S_r$ |
| ${C^r}_{\text{need}}$: | Minimum infrastructure capacity need of region $S_r$ |
| ${N^{\min j}}_{\text{vm}}$: | The minimum VMs number of datacenter $j$, $1 \leqslant j \leqslant {N'}_r$ |
| ${N^{\max j}}_{\text{vm}}$: | The maximum VMs number of datacenter $j$, $1 \leqslant j \leqslant {N'}_r$ |
| ${C^{\min j}}_{\text{vm}}$: | The minimum infrastructure capacity need of datacenter $j$, $1 \leq j \leq {N'}_r$ |
| ${C^{\max j}}_{\text{vm}}$: | The maximum infrastructure capacity need of datacenter $j$, $1 \leq j \leq {N'}_r$ |
| $\text{Budget}_r$: | The monetary constraint of region $S_r$. |

## Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

## Acknowledgments

The work presented in this study is supported by NSFC (61602054) and Beijing Natural Science Foundation (4174100).

## References

[1] F. Hao, M. Kodialam, T. V. Lakshman, and S. Mukherjee, "Online Allocation of Virtual Machines in a Distributed Cloud," *IEEE/ACM Transactions on Networking*, vol. 25, no. 1, pp. 238–249, 2017.

[2] T. Guo, P. Shenoy, K. K. Ramakrishnan, and V. Gopalakrishnan, "Latency-aware virtual desktops optimization in distributed clouds," *Multimedia Systems*, pp. 1–22, 2017.

[3] N. Grozev and R. Buyya, "Regulations and latency-aware load distribution of web applications in Multi-Clouds," *The Journal of Supercomputing*, vol. 72, no. 8, pp. 3261–3280, 2016.

[4] A. Aral and T. Ovatman, "Network-aware embedding of virtual machine clusters onto federated cloud infrastructure," *The Journal of Systems and Software*, vol. 120, pp. 89–104, 2016.

[5] M. Alicherry and T. V. Lakshman, "Network aware resource allocation in distributed clouds," in *Proceedings of the IEEE Conference on Computer Communications (INFOCOM '12)*, pp. 963–971, Orlando, Fla, USA, March 2012.

[6] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of the IEEE Conference on Computer Communications (IEEE INFOCOM '10)*, pp. 1–9, San Diego, Calif, USA, March 2010.

[7] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong, "VMPlanner: optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers," *Computer Networks*, vol. 57, no. 1, pp. 179–196, 2013.

[8] K. Li, H. Zheng, and J. Wu, "Migration-based virtual machine placement in cloud systems," in *Proceedings of the 2013 IEEE 2nd International Conference on Cloud Networking, CloudNet 2013*, pp. 83–90, USA, November 2013.

[9] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[10] C. Gao, H. Wang, L. Zhai, Y. Gao, and S. Yi, "An energy-aware ant colony algorithm for network-aware virtual machine placement in cloud computing," in *Proceedings of the 22nd IEEE International Conference on Parallel and Distributed Systems, ICPADS 2016*, pp. 669–676, chn, December 2016.

[11] W. Wei, X. Fan, H. Song, X. Fan, and J. Yang, "Imperfect information dynamic stackelberg game based resource allocation using hidden markov for cloud computing," *IEEE Transactions on Services Computing*, 2016.

[12] X. Li, J. Wu, S. Tang, and S. Lu, "Let's stay together: Towards traffic aware virtual machine placement in data centers," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 1842–1850, can, May 2014.

[13] C. Tian, A. Munir, A. X. Liu et al., "Multi-tenant multi-objective bandwidth allocation in datacenters using stacked congestion control," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, Ga, USA, May 2017.

[14] R. Yu, G. Xue, X. Zhang, and D. Li, "Survivable and bandwidth-guaranteed embedding of virtual clusters in cloud data centers," in *Proceedings of the IEEE INFOCOM 2017 - IEEE Conference on Computer Communications*, pp. 1–9, Atlanta, Ga, USA, May 2017.

[15] L. Jiao, J. Lit, W. Du, and X. Fu, "Multi-objective data placement for multi-cloud socially aware services," in *Proceedings of the 33rd IEEE Conference on Computer Communications, IEEE INFOCOM 2014*, pp. 28–36, Toronto, Canada, May 2014.

[16] Z. Wu and H. V. Madhyastha, "Understanding the latency benefits of multi-cloud webservice deployments," *Computer Communication Review*, vol. 43, no. 1, pp. 13–20, 2013.

[17] I. Narayanan, A. Kansal, A. Sivasubramaniam et al., "Towards a Leaner Geo-distributed Cloud Infrastructure," in *Proceedings of the USENIX Workshop on Hot Topic in Cloud Computing (HotCloud*, pp. 1–7, 2014.

[18] J. Tordsson, R. S. Montero, R. Moreno-Vozmediano, and I. M. Llorente, "Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers," *Future Generation Computer Systems*, vol. 28, no. 2, pp. 358–367, 2012.

[19] Y.-H. Lee, K.-C. Huang, M.-R. Shieh, and K.-C. Lai, "Distributed resource allocation in federated clouds," *The Journal of Supercomputing*, vol. 73, no. 7, pp. 3196–3211, 2017.

[20] P. Yi, H. Ding, and B. Ramamurthy, "Budget-Optimized Network-Aware Joint Resource Allocation in Grids/Clouds Over Optical Networks," *Journal of Lightwave Technology*, vol. 34, no. 16, pp. 3890–3900, 2016.

[21] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *Proceedings of the 32nd IEEE Conference on Computer Communications, IEEE INFOCOM 2013*, pp. 854–862, Italy, April 2013.

[22] Y. Ding, X. Qin, L. Liu, and T. Wang, "Energy efficient scheduling of virtual machines in cloud with deadline constraint," *Future Generation Computer Systems*, vol. 50, pp. 62–74, 2015.

[23] M. Mihailescu and Y. M. Teo, "Dynamic resource pricing on federated clouds," in *Proceedings of the 10th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2010*, pp. 513–517, Australia, May 2010.

[24] W. Wang, D. Niu, B. Li, and B. Liang, "Dynamic cloud resource reservation via cloud brokerage," in *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013*, pp. 400–409, USA, July 2013.

[25] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in *Proceedings of the 6th ACM Symposium on Cloud Computing, ACM SoCC 2015*, pp. 111–124, USA, August 2015.

[26] Q. Zhang, S. Li, Z. Li, Y. Xing, Z. Yang, and Y. Dai, "CHARM: A Cost-Efficient Multi-Cloud Data Hosting Scheme with High Availability," *IEEE Transactions on Cloud Computing*, vol. 3, no. 3, pp. 372–386, 2015.

[27] D. Ardagna, M. Ciavotta, and M. Passacantando, "Generalized Nash Equilibria for the Service Provisioning Problem in Multi-Cloud Systems," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 381–395, 2017.

[28] L. Jiao, A. M. Tulino, J. Llorca, Y. Jin, and A. Sala, "Smoothed Online Resource Allocation in Multi-Tier Distributed Cloud Networks," *IEEE/ACM Transactions on Networking*, vol. 25, no. 4, pp. 2556–2570, 2017.

[29] F. Palmieri, U. Fiore, S. Ricciardi, and A. Castiglione, "GRASP-based resource re-optimization for effective big data access in federated clouds," *Future Generation Computer Systems*, vol. 54, pp. 168–179, 2016.

[30] K. Gai, M. Qiu, H. Zhao, L. Tao, and Z. Zong, "Dynamic energy-aware cloudlet-based mobile cloud computing model for green computing," *Journal of Network and Computer Applications*, vol. 59, pp. 46–54, 2016.

[31] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, IEEE INFOCOM 2016*, USA, April 2016.

[32] S. Wang, W. Su, X. Zhu, and H. Zhang, "A Hadoop-based approach for efficient web service management," *International Journal of Web and Grid Services*, vol. 9, no. 1, pp. 18–34, 2013.

[33] S. Wang, Y. Zhao, L. Huang, J. Xu, and C. Hsu, "QoS prediction for service recommendations in mobile edge computing," *Journal of Parallel and Distributed Computing*, 2017.

[34] Y. Guo, S. Wang, K. S. Wong, and M. H. Kim, "Skyline service selection approach based on QoS prediction," *International Journal of Web and Grid Services*, vol. 13, no. 4, pp. 425–447, 2017.

[35] Y. Mansouri and R. Buyya, "To move or not to move: Cost optimization in a dual cloud-based storage architecture," *Journal of Network and Computer Applications*, vol. 75, pp. 223–235, 2016.

[36] Microsoft Azure Price, https://azure.microsoft.com/en-us/pricing/.

[37] Amazon EC2 Price, https://aws.amazon.com/ec2/pricing/.

[38] Google Cloud Platform Price, https://cloud.google.com/pricing/.

[39] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical Programming*, vol. 1, pp. 127–136, 1971.