AIMS Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 5(4): 3966–3989.
DOI: 10.3934/math.2020256
Received: 24 December 2019
Accepted: 17 April 2020
Published: 26 April 2020

Research article

An optimal resource allocation scheme for virtual machine placement of deploying enterprise applications into the cloud

Wei Sun, Yan Wang and Shiyong Li*

School of Economics and Management, Yanshan University, Qinhuangdao, 066004, China

* **Correspondence:** Email: shiyongli@ysu.edu.cn; Tel: +863358057025; Fax: +863358057025.

Abstract: The emergence of cloud computing can help enterprises reduce their hardware and software investment and save their own operation and maintenance costs, thus more and more enterprises deploy their applications into the cloud. Generally, components of enterprise applications are resided in virtual machines and then hosted by physical machines. In order to achieve the efficiency and utilization of physical machines, reasonable virtual machines placement becomes very important. In this paper we propose a scheme of resource allocation model for virtual machines placement and investigate it with convex optimization approach. We also present a heuristic algorithm to achieve the optimal resource allocation and discuss its equilibrium and stability by applying the asymptotic stability of the continuous dynamic system of Lyapunov stability theory. Finally, we give some numerical examples to illustrate the performance of the resource allocation scheme and confirm its convergence with a certain number of iterations.

Keywords: convex optimization; cloud computing; data center; virtual machine placement; resource allocation
Mathematics Subject Classification: 68M10, 68M20, 90C30

1. Introduction

Cloud computing is the integration of computer technologies such as distributed computing, network storage, utility computing, virtualization computing, load balancing, and modern network technologies. The cloud computing platform presents a variety of services by centralizing the pool of computing resources connected to the network, such as IaaS (Infrastructure as a Service), PaaS (Platform as a Service), and SaaS (Software as a Service). Regarding the definition of cloud computing, the literature [1] defines the hardware and software of the data center as "cloud", and defines the service as "utility computing", and the sum of software services and utility computing becomes cloud computing. According to the National Institute of Standards and Technology (NIST)

definition [2], cloud computing is a method to access computing resources (including networks, servers, storage, applications, services, etc.) in a convenient, pay-as-you-go manner over a network. It takes advantage of the decentralized resources to support the flexibility of resource sharing and access. Cloud computing enables users to take advantage of the computing resources of the cloud data center as a service, rather than to deploy those resources costly by themselves, and does not require the end users to calculate their exact values of the required resources, which can be calculated and granted by the cloud service providers according to the need of users [3]. Due to the attractive features of cloud computing, more and more enterprises migrate and deploy their applications into the cloud so as to reduce their own IT investments, and some interesting resource allocation schemes are also presented and implemented from the social welfare point of view [4, 5].

Data Center (DC) is used to host servers in the cloud and connects these servers through dedicated links and switches. A large number of distributed processing applications (such as cloud applications, search engines, social applications) are typically run in large data centers [6]. Data centers have become an important choice for data-intensive applications. Companies can reduce their own hardware deployment and maintenance costs, and achieve great revenue. Due to the different resource requirements of applications, many physical machines (PMs) in the data center are in low utilization at most of the time. A large number of running PMs are not being fully used, but at the same time they are occurring large management and maintenance costs, resulting in serious waste of resources. Virtualization is one of the most feasible ways to enhance data center utilization. Virtualization technologies can allocate resources of a single physical machine to multiple virtual machines (VMs) to improve scalability and resource utilization [7]. Virtualization technologies not only enable performance isolation between applications sharing the same computing node, but also enable dynamic or offline migration technology to migrate virtual machines from one node to another. In the actual environment, users are almost unaware of the impact of VM migration. On the other hand, real-time virtual machine reallocation can achieve dynamic load consolidation, so that virtual machines can be merged onto a smaller number of physical machines, thereby turning idle nodes into energy-saving mode. Virtualizing a physical machine can improve resource utilization, however, concentrating workloads on a subset of the physical machines can cause hot spots, and even degrade the actual performance of each application. Thus, how to achieve efficient VM placement has become a critical problem in the cloud [8]. Reasonable resource allocation for virtual machines can also reduce traffic transmission and management and maintenance costs.

Due to the importance of the placement of virtual machines on physical machines in the cloud computing environment, the efficient resource allocation for virtual machine has received a lot of research attention. Each virtual machine placement scenario has one or more targets or goals that are to achieve by designing placement algorithms. For example, some virtual machine placement schemes attempt to reduce network traffic and power consumption in the data center. In addition, the ideal VM placement scheme should reduce unnecessary migration of future VMs [9] and improve the utilization and availability of DC resources. The current research mainly focuses on optimal allocation of virtual machine resources in the cloud environment, so as to reduce energy consumption [10, 11], or achieve the goal of maximizing utility [4, 12]. At the same time, some researchers have concentrated their research on green cloud computing [13–15], minimizing the problem of data transmission time [16, 17], improving the scalability of the system [18], and achieving the fairness of network resource allocation [19, 20].

In this paper we consider enterprise applications deployment into the cloud for reducing its own IT investment, propose an optimization model for virtual machine resource allocation in the cloud environment, and present an optimal resource allocation algorithm. We try to place the virtual machines into the most suitable physical machines and realize the optimal resource allocation for these virtual machines under the constraints of physical machine resources, so as to achieve the goal of maximizing applications utility. We analyze the performance of the proposed resource allocation scheme by applying Lyapunov stability theory and find that all the trajectories along the scheme eventually converge to the optimum of the resource allocation problem. The simulation results show that the scheme can converge to the optimal resource allocation within a limited number of iteration times.

The remainder of this paper is organized as follows: Section 2 introduces the relevant research on virtual machine resource allocation; Section 3 presents the virtual machine resource allocation model and gives the corresponding analysis; Section 4 proposes the resource allocation algorithm and investigates its performance and implementation steps; Section 5 further discusses the utility functions for achieving fair resource allocation and presents the extended resource allocation scheme; Section 6 gives the simulation results of the proposed virtual machine resource allocation algorithm as well as the optimal solutions obtained by LINGO software; Section 7 finally summarizes this paper.

2. Related work

In recent years, cloud computing technologies have received a lot of research attention from academic and industrial researcher, and have been found to be useful and further implemented in many scenarios, such as ubiquitous industrial networks [21, 22], power Internet of Things [23, 24], social vehicular networks [25, 26], 5G networks [27, 28], space-terrestrial integrated networks [29, 30], et al. In many cloud computing systems servers are typically underutilized, resulting in a large amount of resources being idle. Therefore, placing more VMs to fewer PMs is an effective way to reduce power consumption and improve resource utilization. The basic principle of VM placement is to allocate as many VMs as possible on the physical machines while meeting the various constraints specified in the system requirements. VM placement is the process of mapping VMs to the most appropriate location and number of PMs based on the actual needs of users. The existing virtual machine placement algorithms can be divided into two categories according to the targets: one is based on energy consumption, these methods mainly consider server-side constraints; the other is based on application service quality and users' needs, the purpose is to maximize the satisfaction of users. Several virtual machine placement algorithms are presented recently, including constraint programming [31, 32], simulated annealing, boxing algorithm, stochastic programming [33], mixed integer linear programming [34, 35], genetic algorithm [36–38], ant colony algorithm [39, 40], et al., some of which involve virtual machine dynamic migration, while other algorithms only consider static placement.

The goal of the boxing approach is to minimize the number of PMs in operation, treating the VM as an object and the PM as a container, and the goal is to package those objects into as few containers as possible. Although traditional online boxing heuristics (such as Best Fit, BF) have been used to reduce the number of PMs used, there are problems with low utility. Wang et al. [41] established an energy consumption model in a cloud computing environment. Based on the analysis of energy

consumption in the model, the online packing algorithm was improved, and the energy efficiency of a cloud computing environment composed of virtual machines was proposed. The new energy-aware approach not only improves resource utilization, but also makes the data center more energy efficient. Kaaouache et al. [42] considered a one-dimensional packing problem and proposed a new method based on correcting infeasible chromosomes to prevent package overflow. The application of the proposed chromosome to genetic algorithms helps to reduce execution time and minimize energy consumption in cloud data centers. This optimization is done by VM placement, resulting in the use of a minimum number of physical machines and another server that can be turned to sleep or shut down.

For the case where multiple targets are considered, Xu et al. [43] defined the VM placement problem as a multi-objective optimization problem, and proposed an improved genetic algorithm based on fuzzy multi-objective evaluation to minimize the total resource loss, energy consumption and heat dissipation costs. Dörterler et al. [44] formulated the VM placement problem as a multi-objective ant colony optimization algorithm to minimize SLA collisions, total resource loss, and power consumption. Jayasinghe et al. [45] proposed a structural constraint-aware virtual machine placement scheme that supports the constraints of requirements, communication, and availability, and effectively improves the performance and availability of services hosted on the IaaS cloud. Based on the analysis of data center topology characteristics and traffic patterns, Fang et al. [46] proposed a new virtual data center network power consumption reduction method to shut down as many unnecessary network nodes as possible, thereby saving power.

In addition, some scholars investigate VM placement and resource allocation by applying queuing theory and game theory. Guo et al. [47] used the queuing theory to establish a comprehensive optimization model for cloud computing, which is to optimize the average waiting time, average queue length and number of customers of the cloud system. Xiao et al. [48] proposed a new algorithm based on evolutionary game theory. By dynamically adjusting the resource allocation of virtual machines, energy consumption can be effectively reduced. Song et al. [49] studied the energy-saving and scalability issues of modern data centers based on convex optimization theory, and proposed a new virtual machine placement scheme. The scheme takes into account the inherent dependencies and traffic between virtual machines, reduces the cost of communication between virtual machines and increases the efficiency and scalability of the data center.

3. Resource allocation for virtual machine placement

3.1. Background

In the cloud computing architecture, virtualization technologies are used to realize the flexibility and scalability of resources in the cloud data center. Virtualization technologies can map a virtual machine to one or more physical machines to meet the user's flexible demand for resources [49]. At the same time, multiple virtual machines can also parallelly reside on the same physical machine and each virtual machine occupies only part of physical resources, so as to make rational use of the resources of physical machines.

As shown in Figure 1, the VM scheduler is considered as a manager, which continuously monitors the physical resource mapping status provided by the hardware layer to the application service layer, thereby enabling efficient deployment of virtual machines for cloud applications. Generally, in order to improve the utilization of resources, the architecture sets the maximum and minimum thresholds.

When the resource utilization of a physical machine is less than the minimum threshold, the physical machine will be shut down to save power consumption and move its virtual machines to other physical machines; when the resource utilization of a physical machine is greater than the maximum threshold, the architecture will no longer place a new virtual machine to the physical machine to ensure that the quality of service does not drop. The task of the resource monitor is to solve the problem of matching the requirements of the application service with the virtual machine resources, obtain the optimal resource allocation for virtual machines, and communicate VM scheduler to place the virtual machine on a suitable physical machine or perform dynamic virtual machine migration. Then, how to allocate the resources of these physical machines is a key issue to improve server utilization and reduce costs. This paper proposes an optimization model for virtual machine resource allocation and introduce a resource allocation scheme to achieve the optimum.

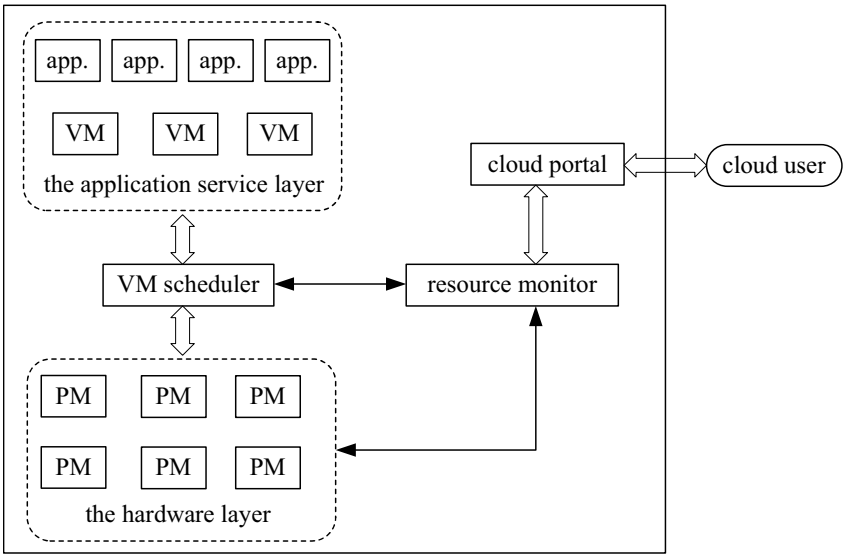


Figure 1. VM management architecture in the cloud center.

3.2. Model description

An enterprise user intends to reduce its own system infrastructure investment and maintenance cost, and wants to deploy its applications into the cloud. Indeed, each application is composed of a number of distributed components, generally including three functional tiers: A front-end tier, a business-logic tier, and a back-end tier [50]. A front-end tier mainly handles user requests such as web requests and consists of a number of components. A business-logic tier mainly performs specialized application logic. And a back-end tier generally stores some important operation data of an application, and comprises of various databases servers. In some scenarios the components of the back-end tier may not be deployed into the cloud and operate in enterprise local data center for data security consideration.

Consider there is a set P of physical machines in the cloud data center. Virtual machines will be effectively placed in the physical machines through the resource scheduler. Introduce the set S of applications that an enterprise user intends to deploy into the cloud and the set R of application components. Each application $s \in S$ is composed of a set $R(s)$ of components. Here we assume each component resides in one virtual machine. Multiple virtual machines can simultaneously reside on

the same physical machine so as to improve the resource utilization. Let $P(r)$ be the set of physical machines that provide resource for the corresponding virtual machine of component r. Introduce the set $S(p)$ of applications whose components reside on the physical machine p and obtain resource allocation for the virtual machines from physical machine p.

Define x_{srp}^{CPU}, x_{srp}^{ram} and x_{srp}^{sto} be the CPU, memory and storage resources offered by physical machine p to component r of application s, respectively. And denote y_s^{CPU}, y_s^{ram} and y_s^{sto} as the total CPU, memory and storage resources obtained by application s, respectively, that is, the summation of all its components for each type of resource. Introduce $U_s(\cdot)$ be the deployment utility of application s which is a continuously differentiable and bounded function with respect to its variables. Let C_p^{CPU}, C_p^{ram} and C_p^{sto} be the maximum capacity of CPU, memory and storage of resource provider p, respectively.

When an enterprise deploys its applications into the cloud, the goal of resource allocation for virtual machine placement in the cloud is indeed to maximize the aggregated deployment utility of all applications, so as to achieve effective utilization of the resources of physical machines. Then, we establish the following resource allocation optimization model for virtual machine placement in the cloud.

$$
\begin{aligned}
\textbf{Q:}\quad \max \quad & \sum_{s\in S} U_s(y_s^{CPU}\times y_s^{ram}\times y_s^{sto}) \\
\text{subject to} \quad & \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}^{CPU} = y_s^{CPU}, s\in S \\
& \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}^{ram} = y_s^{ram}, s\in S \\
& \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}^{sto} = y_s^{sto}, s\in S \\
& \sum_{s\in S(p)}\sum_{r\in R(s)} x_{srp}^{CPU} \le C_p^{CPU}, p\in P \\
& \sum_{s\in S(p)}\sum_{r\in R(s)} x_{srp}^{ram} \le C_p^{ram}, p\in P \\
& \sum_{s\in S(p)}\sum_{r\in R(s)} x_{srp}^{sto} \le C_p^{sto}, p\in P \\
\text{over} \quad & x_{srp}^{CPU}\ge 0, x_{srp}^{ram}\ge 0, x_{srp}^{sto}\ge 0, s\in S, r\in R, p\in P.
\end{aligned}
\tag{3.1}
$$

We investigate the resource allocation for virtual machines placement when enterprise applications are deployed into the cloud. Typically, tightly packing virtual machines onto a small number of working physical machines and turning off other idle machines is an effective way to maximize physical machine utilization and reduce energy consumption. However, concentrating workloads on a subset of the physical machines can cause hot spots, and even degrade the actual performance of each application resided and deployed in the cloud. An effective resource allocation strategy should achieve the tradeoffs between energy efficiency and fairness. Thus we adopt the fairness utility proposed for resource allocation in traditional IP Networks [51] and further in per-to-peer networks [52], which has the following logarithmic form.

$$
U_s(y_s^{CPU}\times y_s^{ram}\times y_s^{sto}) = w_s \log(y_s^{CPU}\times y_s^{ram}\times y_s^{sto}), \tag{3.2}
$$

where w_s is the payment that the enterprise user is willing to pay to its resource provider for its application s when the application s is deployed into the cloud.

We find that the application deployment utility function $U_s(\cdot)$ is the summation of deployment utilities from CPU, memory and storage resources, i.e., $U_s(y_s^{CPU}) + U_s(y_s^{ram}) + U_s(y_s^{sto})$. Then the resource allocation model (3.2) for virtual machines placement when applications are deployed into the cloud can be decomposed into three independent sub-problems for CPU, memory and storage resource allocation, respectively. For simplicity, in the following analysis, we do not further differentiate among the specic types of resource, and generalize these three sub-problems into the following one form

$$\begin{aligned} \textbf{QG:} \quad & \max \quad \sum_{s\in S} U_s(y_s) \\ & \text{subject to} \quad \sum_{r\in R(s)} \sum_{p\in P(r)} x_{srp} = y_s, s \in S \\ & \qquad\qquad \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp} \le C_p, p \in P \\ & \text{over} \qquad\quad x_{srp} \ge 0, s \in S, r \in R, p \in P. \end{aligned} \tag{3.3}$$

3.3. Model analysis

For the generalization-form resource allocation model (3.3), the objective function is a strict concave function with respect to variable y_s but not a strict concave function with respect to variable x_{srp}. At the same time, the constraints are linear, so the constraint set for this optimization problem is convex. Therefore, the following results can be obtained based on the convex optimization theory.

Theorem 1 When the enterprise user deploys its application into the cloud, the resource allocation model (3.3) for virtual machines placement is a convex optimization problem. The optimal resource allocation $x^* = (x^*_{srp}, r \in R, s \in S, p \in P)$ offered to each virtual machine that the component of an application resides on exists but is not unique. The optimal resource allocation $y^* = (y^*_s, s \in S)$ for each application exists and is unique.

In order to get the optimal solution of the resource allocation model, we firstly give the Lagrangian of this nonlinear optimization problem

$$\begin{aligned} L(\mathbf{x}, \mathbf{y}; \lambda, \mu) = & \sum_{s\in S} U_s(y_s) + \sum_{s\in S} \lambda_s \left(\sum_{r\in R(s)} \sum_{p\in P(r)} x_{srp} - y_s \right) \\ & + \sum_{p\in P} \mu_p \left(C_p - \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp} \right), \end{aligned} \tag{3.4}$$

where $\lambda = (\lambda_s, s \in S)$ is the price vector with element $\lambda_s \ge 0$, which can be considered as the price per unit cloud resource paid by the enterprise user for its application s; $\mu = (\mu_p, p \in P)$ is the price vector with element $\mu_p \ge 0$, which can be considered as the price per unit cloud resource charged by physical machine p.

We can also rewrite (3.4) with the following form

$$L(\mathbf{x}, \mathbf{y}; \lambda, \mu) = \sum_{s\in S} (U_s(y_s) - \lambda_s y_s) + \sum_{s\in S} \sum_{r\in R(s)} \sum_{p\in P(r)} x_{srp} \left(\lambda_s - \mu_p \right) + \sum_{p\in P} \mu_p C_p \tag{3.5}$$

It can be noted that the first expression in (3.5) is separable in variable y_s, and the second expression is separable in variables x_{srp}. Thus the objective function of the dual problem can be written as:

$$D(\lambda, \mu) = \max_{\mathbf{x},\mathbf{y}} L(\mathbf{x}, \mathbf{y}; \lambda, \mu) = \sum_{s\in S} S_s(\lambda_s) + \sum_{s\in S} \sum_{r\in R(s)} \sum_{p\in P(r)} P_{sp}(\lambda_s, \mu_p) + \sum_{p\in P} \mu_p C_p, \tag{3.6}$$

where

$$S_s(\lambda_s) = \max_{y_s} U_s(y_s) - \lambda_s y_s, \tag{3.7}$$

$$P_{sp}(\lambda_s, \mu_p) = \max_{x_{srp}} x_{srp}\left(\lambda_s - \mu_p\right). \tag{3.8}$$

In equation (3.7), the enterprise user tries to maximize its own deployment utility for its applications, which depends on the total obtained resource y_s of application s. λ_s is the price per unit resource that the enterprise user pays for application s, then $\lambda_s y_s$ is the total payment that the user provides to the resource providers for its application deployment into the cloud. In equation (3.8), component r of application s obtains the resource allocation x_{srp} from physical machine p, then $x_{srp}\lambda_s$ is the cost of the enterprise user paying for the physical machine p in the cloud deployment for the application s. Since μ_p is the price per unit resource charged by the physical machine p, then $x_{srp}\mu_p$ is the cost of physical machine p when offered resource x_{srp} to component r of application s, so equation (3.8) means that each physical machine is to maximize its own revenue.

Hence, the dual problem is

$$\begin{aligned} \textbf{D:}\quad & \min \quad D(\lambda, \mu) \\ & \text{over} \quad \lambda_s \geq 0, \mu_p \geq 0, s \in S, p \in P. \end{aligned} \tag{3.9}$$

The original resource allocation problem (3.1) or its generalization form (3.3) is to maximize the aggregation of deployment utility of applications when deployed into the cloud under the constraints of physical machines capacity, and the dual problem (3.9) is to minimize the overall price of the cloud computing data center.

3.4. Optimize resource allocation

In order to obtain the optimal resource allocation for virtual machine placement when the applications are deployed into the cloud, let $(\mathbf{x}^*, \mathbf{y}^*, \lambda^*, \mu^*)$ be the optimal solution to the generalization-form problem (3.3) and the dual problem (3.9). Let $\partial L(\mathbf{x}, \mathbf{y}; \lambda, \mu)/\partial y_s = 0$, we can obtain

$$y_s^* = \frac{w_s}{\lambda_s}. \tag{3.10}$$

Substituting (3.10) into (3.4), we obtain

$$\begin{aligned} \widetilde{L}(\mathbf{x}; \lambda, \mu) = & \sum_{s\in S}\left(w_s \log\left(\frac{w_s}{\lambda_s}\right) - w_s + \lambda_s \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}\right) \\ & + \sum_{p\in P}\mu_p\left(C_p - \sum_{s\in S(p)}\sum_{r\in R(s)} x_{srp}\right), \end{aligned} \tag{3.11}$$

Let $\partial\widetilde{L}(\mathbf{x}; \lambda, \mu)/\partial\lambda_s = 0$, then we obtain

$$\lambda_s^* = \frac{w_s}{\sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}}. \tag{3.12}$$

Substituting (3.12) into (3.11), we obtain

$$\widehat{L}(\mathbf{x}; \mu) = \sum_{s\in S}\left(w_s \log\left(\sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}\right)\right) + \sum_{p\in P}\mu_p\left(C_p - \sum_{s\in S(p)}\sum_{r\in R(s)} x_{srp}\right), \tag{3.13}$$

Assume that the virtual machine corresponding to the component r of application s obtains a certain amount of resource allocation from physical machine p, that is $x_{srp} > 0$, then let $\partial\widehat{L}(\mathbf{x};\mu)/\partial x_{srp} = 0$, we can obtain the optimal resource allocation

$$y_s^* = \sum_{r\in R(s)} \sum_{p\in P(r)} x_{srp} = \frac{w_s}{\mu_p}, p \in P(r), r \in R(s). \tag{3.14}$$

Then we can obtain the following result.

Theorem 2 For the resource allocation optimization model (3.1) or its generalization form (3.4) for virtual machines placement, if there are two components of an application residing on two different physical machines and obtaining non-zero resource allocations from these machines, then the prices charged by these physical machines are both equal to the price paid by the enterprise user for this application. That is, if $x^*_{sr_1p_1} > 0, p_1 \in P(r_1)$ and $x^*_{sr_2p_2} > 0, p_2 \in P(r_2)$, then $\mu^*_{p_1} = \mu^*_{p_2} = \lambda^*_s$.

Actually, from (3.14) we can find

$$\mu^*_{p_1} = \mu^*_{p_2} = \frac{w_s}{\sum\limits_{r\in R(s)} \sum\limits_{p\in P(r)} x_{srp}} = \lambda^*_s. \tag{3.15}$$

Therefore, a bipartite graph consisting of physical machines P and applications S can be constructed. Each edge represents the relationship between a physical machine and an application, indicating that the physical machine provides resources for the virtual machine of the application to complete the cloud deployment. If the graph is fully connected, then $\mu^*_{p_1} = \mu^*_{p_2} = \mu^*$, otherwise, it can be divided into multiple connected subgraphs for analysis. In the following discussion, we assume $\mu^*_{p_1} = \mu^*_{p_2} = \mu^*$. Then let $\partial\widehat{L}(\mathbf{x};\mu)/\partial\mu = 0$, we can obtain

$$\mu^* = \frac{\sum_{s\in S} w_s}{\sum_{p\in P} C_p}. \tag{3.16}$$

Substituting (3.16) into (3.14),

$$y_s^* = w_s \frac{\sum_{p\in P} C_p}{\sum_{s\in S} w_s}. \tag{3.17}$$

It can be found from (3.17) that when an enterprise application s is deployed into a cloud, the optimal resource allocation y_s^* obtained by the virtual machines in which the components of this application are located depends on the total capacity of its physical machines, and the total payment of all applications that are hosted by the physical machines weighted by the payment w_s for application s. At the same time, it can be seen that the total resource allocation y_s^* for each application s is unique, which has been discussed in Theorem 1.

4. Resource allocation algorithm

4.1. Resource allocation algorithm description

In order to enable enterprise users to deploy applications into the cloud, the virtual machines in which the components of each application are located should be provisioned with optimal resource allocation. Then this paper proposes the following resource allocation algorithm. Each physical

machine p updates its resource allocation $x_{srp}(t)$ for component r of application s according to the following rule

$$\frac{d}{dt}x_{srp}(t) = \kappa x_{srp}(t)(\lambda_s(t) - \xi_p(t))^+_{x_{srp}(t)-\varepsilon}, \tag{4.1}$$

$$\xi_p(t) = \frac{\sum_{j\in S(p)}\sum_{i\in R(j)} x_{jip}(t)\lambda_j(t)}{C_p}. \tag{4.2}$$

The price per unit resource that an enterprise user should pay to physical machine p when deploying application s into the cloud is

$$\lambda_s(t) = \frac{w_s}{\max\{\eta, y_s(t)\}}, \tag{4.3}$$

$$y_s(t) = \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}(t), \tag{4.4}$$

where $\kappa > 0$ is the step size of the algorithm, $\varepsilon > 0, \eta > 0$ are small constants to ensure the allocated resources $x_{srp}(t)$ not lower than ε and the price $\lambda_s(t)$ not higher than w_s/η. Here $a = (b)^+_c$ means $a = b$ if $c > 0$ and $a = \max\{0, b\}$ if $c = 0$. In the above algorithm, $\xi_p(t)$ can be regarded as the expected price when the physical machine p provides resources for virtual machines to host components of applications, and at the optimal resource allocation, the expected price ξ^*_p of the physical machine p will be equal to the actual paid price λ^*_s of the enterprise user for its applications.

4.2. Equilibrium

Now we study the proposed resource allocation scheme (4.1)–(4.4) and analyze its equilibrium. By substituting (4.2) into (4.1) and setting (4.1) to zero, we can obtain the equilibrium (x^*, λ^*), that is,

$$\lambda^*_s = \frac{\sum_{j\in S(p)}\sum_{i\in R(j)} x^*_{jip}\lambda^*_j}{C_p}, \tag{4.5}$$

Meanwhile, from (4.3) (4.4), at the equilibrium

$$\lambda^*_s = \frac{w_s}{y^*_s} = \frac{w_s}{\sum_{r\in R(s)}\sum_{p\in P(r)} x^*_{srp}}. \tag{4.6}$$

Then

$$\begin{aligned}\sum_{p\in P} C_p &= \sum_{p\in P}\frac{\sum_{j\in S(p)}\sum_{i\in R(j)} x^*_{jip}\lambda^*_j}{\lambda^*_s} = \frac{1}{\lambda^*_s}\sum_{p\in P}\sum_{j\in S(p)}\sum_{i\in R(j)} x^*_{jip}\lambda^*_j \\ &=^{(a)} \frac{y^*_s}{w_s}\sum_{j\in S}\lambda^*_j\sum_{i\in R(j)}\sum_{p\in P(i)} x^*_{jip} =^{(b)} \frac{y^*_s}{w_s}\sum_{j\in S} w_j,\end{aligned}$$

where (a) and (b) follow from (4.6). Thus the total resource allocation for application s is

$$y^*_s = \frac{w_s\sum_{p\in P} C_p}{\sum_{j\in S} w_j}, \tag{4.7}$$

and from (4.6) the price paid by the enterprise user for application s is

$$\lambda^*_s = \frac{\sum_{j\in S} w_j}{\sum_{p\in P} C_p}. \tag{4.8}$$

It is very important to emphasize that the optimal price paid by the enterprise user has no relationship with each application, but only depends on the ratio of the total payment of the enterprise user to the total capacity of physical machines. It is also equal to the optimal price charged by each physical machine. In the following stability analysis, we use $\lambda_s^* = \lambda^*$ and $\mu_p^* = \mu^*$ for simplicity.

Obviously, the equilibrium (4.7),(4.8) is equal to the optimum (3.16),(3.17) of the generalization-form resource allocation problem (3.3). At the same time, we can also find at the equilibrium point the following equality holds

$$\sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp}^* = C_p \tag{4.9}$$

Then, the summation of the allocated resources by each physical machine to the virtual machines of applications is exactly the resource capacity of the physical machine, which also indicates that the resources provided by the physical machine are fully utilized at the equilibrium of the dynamic system described by the algorithm.

4.3. Stability

From the analysis above, we know that the equilibrium point of the dynamic systems (4.1)-(4.4) described by the algorithm is consistent with the optimal resource allocation of the resource allocation problem (3.3). Based on the asymptotic stability of the continuous dynamic system of Lyapunov stability theory, we can obtain the following result.

Theorem 3 The equilibrium point (4.7),(4.8) of the dynamic systems (4.1)–(4.4) described by the proposed algorithm is asymptotically stable. Therefore, all the trajectories along (4.1)–(4.4) eventually converge to the optimum of the resource allocation problem (3.3).

Proof

Define the following Lyapunov function

$$V(t) = V_1(t) + V_2(t) = \sum_{s\in S} \int_{y_s(t)}^{y_s^*} \left(\frac{w_s}{\upsilon} - \lambda^*\right) d\upsilon + \sum_{p\in P} \lambda^*(C_p - \psi_p(t)),$$

where $\psi_p(t) = \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp}(t)$, $p \in P$, and $\lambda^* = \lambda_s^*, \forall s \in S$.

Since $y_s(t) \geq 0$, $y_s^* \geq 0$, the first part of the Lyapunov function is

$$\begin{aligned} \int_{y_s(t)}^{y_s^*} \left(\frac{w_s}{\upsilon} - \lambda^*\right) d\upsilon &= w_s(\log y_s^* - \log y_s(t)) - \lambda^*(y_s^* - y_s(t)) \\ &= w_s\left(\frac{y_s(t)}{y_s^*} - 1 - \log\frac{y_s(t)}{y_s^*}\right) \geq 0. \end{aligned}$$

Obviously, $V_1(t) = 0$ if and only if $y_s(t) = y_s^*$ (i.e., $\lambda_s(t) = \lambda_s^* = \lambda^*$). Meanwhile, the second part of the Lyapunov function is $V_2(t) \geq 0$ since $\psi_p(t) = \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp}(t) \leq C_p$, and $V_2(t) = 0$ if and only if $\psi_p(t) = \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp}^* = C_p$. Thus, the Lyapunov function $V(t)$ is a positive definite function, and it is zero only at the equilibrium point (x^*, λ^*) (i.e., $\lambda_s(t) = \lambda_s^* = \lambda^*$, $\psi_p(t) = \sum_{s\in S(p)} \sum_{r\in R(s)} x_{srp}^* = C_p$).

The derivative of $V(t)$ along the trajectories of the dynamic system (4.1)–(4.4) is

$$\begin{aligned}
\frac{dV(t)}{dt} &= \sum_{s\in S}\frac{\partial V(t)}{\partial y_s(t)}\frac{dy_s(t)}{dt} + \sum_{p\in P}\frac{\partial V(t)}{\partial \psi_p(t)}\frac{d\psi_p(t)}{dt} \\
&= -\sum_{s\in S}\left(\frac{w_s}{y_s(t)} - \lambda^*\right)\sum_{r\in R(s)}\sum_{p\in P(r)}\frac{dx_{srp}(t)}{dt} - \sum_{p\in P}\lambda^*\sum_{s\in S(p)}\sum_{r\in R(s)}\frac{dx_{srp}(t)}{dt} \\
&= -\sum_{s\in S}(\lambda_s(t) - \lambda^*)\sum_{r\in R(s)}\sum_{p\in P(r)}\frac{dx_{srp}(t)}{dt} - \sum_{p\in P}\lambda^*\sum_{s\in S(p)}\sum_{r\in R(s)}\frac{dx_{srp}(t)}{dt} \\
&= -\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s(t)x_{srp}(t)\left(\lambda_s(t) - \frac{\sum_{j\in S(p)}\sum_{i\in R(j)}x_{jip}(t)\lambda_j(t)}{C_p}\right) \\
&= -\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s^2(t)x_{srp}(t) \\
&+ \sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\frac{x_{srp}(t)\lambda_s(t)}{C_p}\sum_{j\in S(p)}\sum_{i\in R(j)}x_{jip}(t)\lambda_j(t) \\
&= -\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s^2(t)x_{srp}(t)\left(1 - \frac{x_{srp}(t)}{C_p}\right) \\
&+ \sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\frac{x_{srp}(t)\lambda_s(t)}{C_p}\sum_{j\in S(p)\setminus\{s\}}\sum_{i\in R(j)}x_{jip}(t)\lambda_j(t).
\end{aligned}$$

Add $\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s^2(t)x_{srp}(t)\frac{1}{C_p}\sum_{j\in S(p)\setminus\{s\}}\sum_{i\in R(j)}x_{jip}(t)$ to the first part of the derivative above, and subtract the same term from the second part, then

$$\begin{aligned}
\frac{dV(t)}{dt} &= -\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s^2(t)x_{srp}(t)\left(1 - \frac{\sum_{j\in S(p)}\sum_{i\in R(j)}x_{jip}(t)}{C_p}\right) \\
&+ \sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\sum_{j\in S(p)\setminus\{s\}}\sum_{i\in R(j)}\frac{\kappa}{C_p}\left(x_{srp}(t)\lambda_s(t)x_{jip}(t)\lambda_j(t) - \lambda_s^2(t)x_{srp}(t)x_{jip}(t)\right).
\end{aligned}$$

Hence

$$\begin{aligned}
\frac{dV(t)}{dt} &= -\sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\kappa\lambda_s^2(t)x_{srp}(t)\left(1 - \frac{\sum_{j\in S(p)}\sum_{i\in R(j)}x_{jip}(t)}{C_p}\right) \\
&+ \sum_{s\in S}\sum_{r\in R(s)}\sum_{p\in P(r)}\sum_{j\in S(p)\setminus\{s\}}\sum_{i\in R(j)}\frac{\kappa x_{srp}(t)x_{jip}(t)}{2C_p}\left(\lambda_s(t) - \lambda_j(t)\right)^2.
\end{aligned}$$

Thus, $dV(t)/dt \le 0$ since $\sum_{j\in S(p)}\sum_{i\in R(j)}x_{jip}(t) \le C_p, p \in P$, and $dV(t)/dt = 0$ if and only if $\lambda_s(t) = \lambda_j(t) = \lambda^*$, $\sum_{j\in S(p)}\sum_{i\in R(j)}x^*_{jip} = C_p$ (i.e., the equilibrium point (x^*, λ^*)). Therefore, from the Lyapunov stability theory [53] the equilibrium point (4.7),(4.8) of the dynamic system (4.1)–(4.4) is asymptotically stable. All the trajectories along (4.1)–(4.4) eventually converge to the optimum of the resource allocation problem (3.3). The result is obtained.

4.4. Implementation

In the practical implementation, the resource allocation scheme is updated according to the discrete form of the proposed algorithm. That is, at time $t = 1, 2...$, each physical machine p updates its resource

allocation for component r of application s according to the following expression.

$$x_{srp}[t+1] = ((1-\theta)x_{srp}[t] + \theta\widetilde{x}_{srp}[t] + \theta\kappa x_{srp}[t](\lambda_s[t] - \xi_p[t]))^+_{x_{srp}[t]-\varepsilon}, \tag{4.10}$$

$$\widetilde{x}_{srp}[t+1] = (1-\theta)\widetilde{x}_{srp}[t] + \theta x_{srp}[t], \tag{4.11}$$

$$\xi_p[t] = \frac{\sum_{j\in S(p)}\sum_{i\in R(j)} x_{jip}[t]\lambda_j[t]}{C_p}. \tag{4.12}$$

Here, we introduce an augmented variable $\widetilde{x}_{srp}[t]$, which is considered to be the optimal allocation estimation for resource allocation $x_{srp}[t]$. Therefore, a low-pass filtering scheme is added to the original algorithm, where θ is the low-pass filtering parameter. By applying low-pass filtering approach, the augmented variable is assisted to remove possible oscillation due to the fact that the model is not strictly concave and optimal resource allocation is not necessarily unique, but not to change optimal resource allocation.

The price paid by the enterprise user for each application follows the rule

$$\lambda_s[t] = \frac{w_s}{\max\{\eta, y_s[t]\}}, \tag{4.13}$$

$$y_s[t] = \sum_{r\in R(s)}\sum_{p\in P(r)} x_{srp}[t]. \tag{4.14}$$

The practical implementation of the proposed resource allocation algorithm is summarized as follows:

Step 1: Initialize variables and parameters. Select step size parameter κ and small constants ε, η. At time t, initialize the resource allocation $x_{srp}[t]$ for component r of application r by physical machine p.

Step 2: Calculate the price that the enterprise user should pay to the physical machine p. At time t, application s obtains the total resource allocation $y_s[t]$ for all its components from its available physical machines, and calculates the price $\lambda_s[t]$ that should be paid for using the resource $y_s[t]$.

Step 3: Calculate the expected price of each physical machine. At time t, physical machine p calculates its expected price $\xi_p[t]$.

Step 4: Update the resource allocation. At time $t+1$, physical machine p updates its resource allocation $x_{srp}[t+1]$ for component r of application s.

Step 5: Set the stop criteria. When the algorithm reaches equilibrium, the iterative process can be stopped to obtain the optimal resource allocation.

In each iteration, the enterprise user separately calculates the price it should pay for each application when obtaining resources from its physical machines and communicates the price to the resource scheduler. Each physical machine updates its resource allocation based on the price paid by the enterprise for each application and reports the optimal resource allocation to the resource scheduler. The iterative process described above is repeated many times until the equilibrium is finally reached.

5. Further discussion

5.1. Utility functions and resource allocation

In this section we consider other forms of utility functions and discuss the proposed resource allocation scheme. We follow the well known α-fairness utility functions used for resource allocation

in IP wired and wireless networks [54], which are summarized as the following form

$$U_s(y_s) = \begin{cases} w_s \log y_s, & \text{if } \alpha = 1, \\ w_s \dfrac{y_s^{1-\alpha}}{1-\alpha}, & \text{if } \alpha > 0 \quad \text{and} \quad \alpha \neq 1, \end{cases} \tag{5.1}$$

where α is the well known fairness parameter. If we choose different value of parameter α, we can achieve different objective of fair resource allocation. For example, we have achieved proportional fairness of resource allocation among applications when choosing $\alpha = 1$. We can also achieve harmonic mean fairness if we choose $\alpha = 2$. And, if $\alpha \to \infty$, max-min fairness for resource allocation of the physical machines can be achieved when the enterprise user deploys its applications into the cloud.

As for the generalization-form resource allocation model (3.3) with the above utility functions, the objective is also concave but not strictly concave with respect to variables x_{srp}, thus the optimal resource allocation can also be obtained by following the aforementioned analysis approach. In details, the optimal resource allocation y_s^* for each application i is

$$y_s^* = w_s^{1/\alpha} \frac{\sum_{p \in P} C_p}{\sum_{s \in S} w_s^{1/\alpha}}, \tag{5.2}$$

and the optimal price μ_p^* charged by each physical machine p is also equal to the optimal price λ_s^* paid by the enterprise user for its physical machines, that is,

$$\lambda_s^* = \mu_p^* = \left(\frac{\sum_{s \in S} w_s^{1/\alpha}}{\sum_{p \in P} C_p} \right)^{\alpha}. \tag{5.3}$$

Then the optimal resource allocation depends not only on the total payment for applications and the total resource capacity of all physical machines, but also on the the fairness parameter α, i.e., other fairness goals of resource allocation can also be achieved if we choose different fairness parameters accordingly.

5.2. Extended resource allocation scheme

As for the extended form of utility function (5.1), we modify the original resource allocation scheme and propose an improved resource allocation algorithm. When an enterprise user deploys its applications into the cloud, each physical machine p updates its resource allocation for component r of application s with the following rule

$$\frac{d}{dt} x_{srp}(t) = \kappa x_{srp}(t) (\lambda_s^{\frac{1}{\alpha}}(t) - \xi_p(t))^+_{x_{srp}(t)-\varepsilon}, \tag{5.4}$$

$$\xi_p(t) = \frac{\sum_{j \in S(p)} \sum_{i \in R(j)} x_{jip}(t) \lambda_j^{\frac{1}{\alpha}}(t)}{C_p}. \tag{5.5}$$

And the price that the enterprise user pays to the physical machines is

$$\lambda_s(t) = \frac{w_s}{\max\{\eta, y_s^{\alpha}(t)\}}, \tag{5.6}$$

$$y_s(t) = \sum_{r \in R(s)} \sum_{p \in P(r)} x_{srp}(t). \tag{5.7}$$

Substituting (5.5) into (5.4) and setting (5.4) to zero, we can obtain the equilibrium of the dynamic system described by the improved algorithm, which is also equal to optimum (5.2),(5.3) of the generalization-form resource allocation model with the extended form of utility function (5.1). Furthermore, by applying the asymptotic stability approach for the continuous dynamic system, we also deduce that the equilibrium (5.1),(5.3) of the dynamic systems (5.4)–(5.7) described by the proposed algorithm is also asymptotically stable. All the trajectories along (5.4)–(5.7) eventually converge to the optimum of the resource allocation problem (3.3) with utility functions (5.1).

6. Numerical examples and analysis

In this section we will analyze the performance of the proposed resource allocation scheme for virtual machines placement. We first consider a simple scenario and discuss the performance of resource allocation scheme, then we will further investigate the performance in some large scale scenarios.

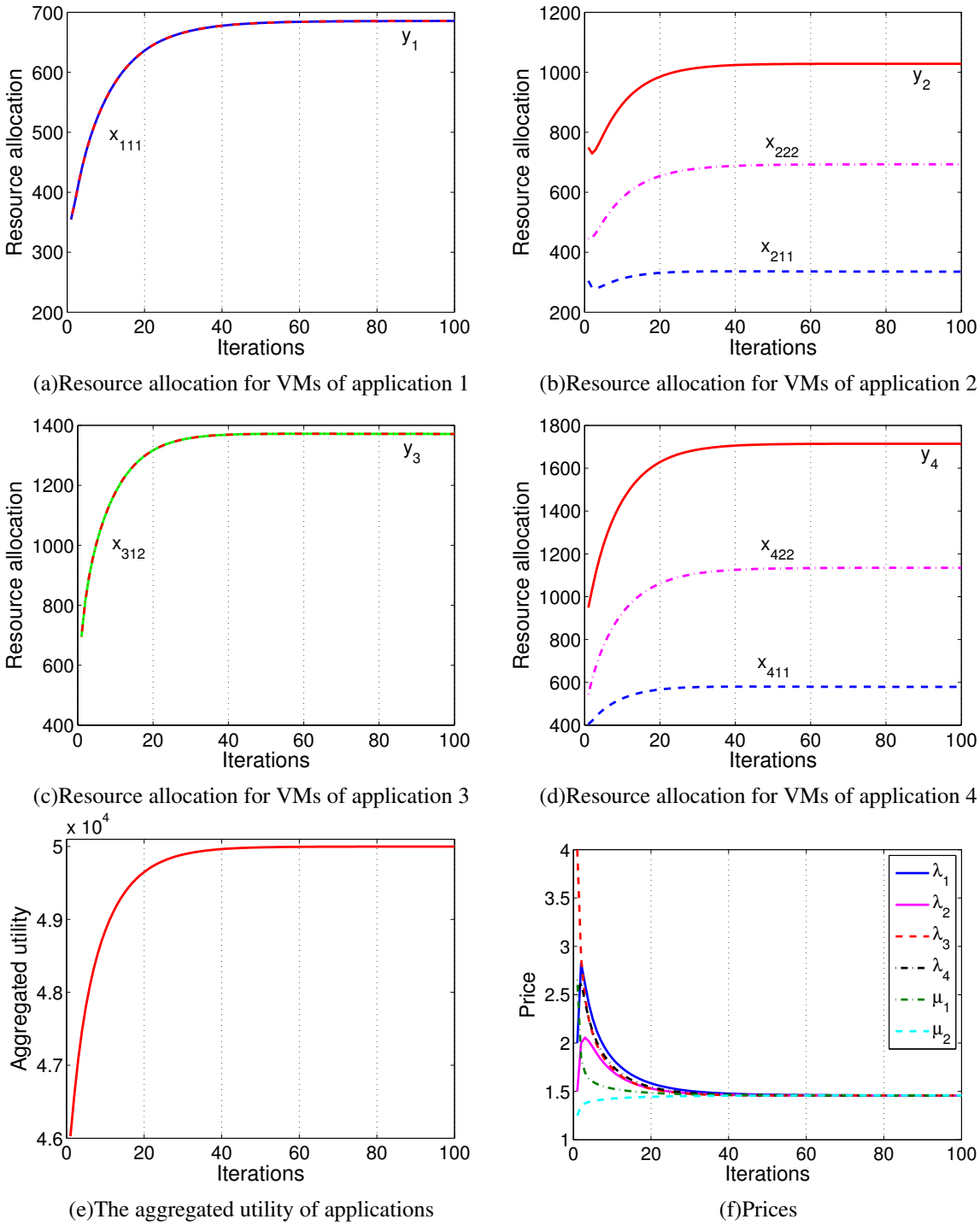
6.1. A simple scenario

Consider an enterprise user deploying its four applications into a cloud data center, where application 1 and application 3 are both composed of only one component, and application 2 and application 4 both consist of two components. Each component of an application is packaged separately into a virtual machine. There are two physical machines in the cloud data center to complete the deployment of virtual machines.

We all know that CPU is an critical resource of each physical machine. Therefore, we consider the performance of the resource allocation scheme by taking the CPU resource of the physical machines as an example. Assume that the CPU resource capacity of the physical machines is $C = (C_1, C_2) = (1600, 3200)$MIPS, where MIPS (million instructions per second) is an indicator of CPU operation speed. The payment of this enterprise user for its four applications to deploy into the cloud is $w = (w_1, w_2, w_3, w_4) = (1000, 1500, 2000, 2500)$. The step size of the resource allocation scheme is $\kappa = 0.2$, the filtering parameter is $\theta = 0.2$, and the positive constants are $\eta = \varepsilon = 0.01$.

Here we consider the proportional fairness (i.e., $\alpha = 1$) of resource allocation among applications, and depict the performance of the proposed resource allocation scheme in Figure 2. It is not hard to find that the resource allocation scheme can converge to the equilibrium within a certain number of iterations, which is also verified through theoretical analysis in Theorem 3. The optimal prices that the enterprise user has to pay for the physical machines when placing virtual machines and obtaining resource are also illustrated in this figure. We can find they are all equal to the optimal price that each physical machine charges, which has been discussed in Theorem 2. Indeed, the optimal price obtained from this scheme is also equal to the value which is derived from the equation (3.16) (i.e., $\mu^* = \sum_{s \in S} w_s / \sum_{p \in P} C_p = 7000/4800 = 1.4583$).

Meanwhile, we give the equilibrium value obtained by the resource allocation scheme in Table 1. We also list the optimal solution solved by nonlinear programming software LINGO in the table. We can observe that the proposed scheme is efficient to achieve the optimal resource allocation for virtual machines placement into the cloud data center.



(a)Resource allocation for VMs of application 1

(b)Resource allocation for VMs of application 2

(c)Resource allocation for VMs of application 3

(d)Resource allocation for VMs of application 4

(e)The aggregated utility of applications

(f)Prices

Figure 2. Performance of the resource allocation for VMs placement of applications in case 1.

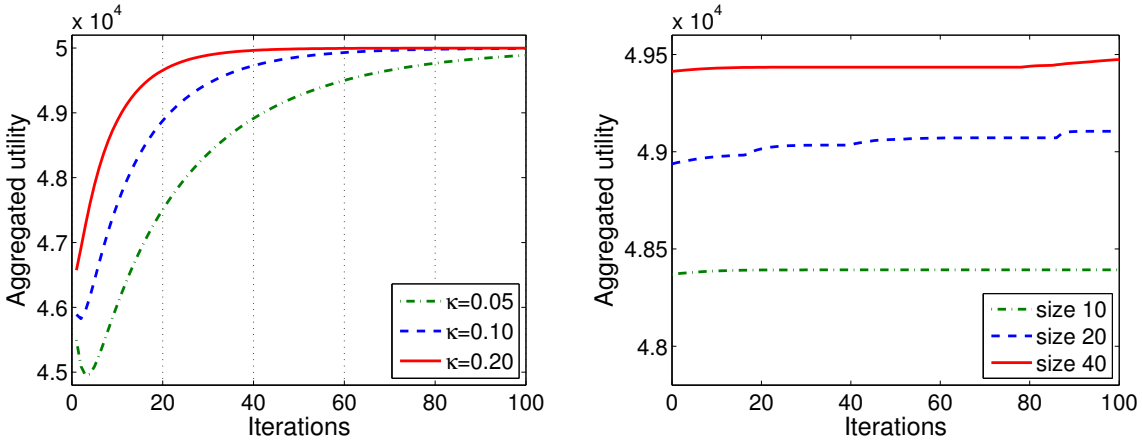
We know there are also some efficient VM placement methods for cloud data centers by applying genetic algorithms. For example, in [37] the authors present an energy-efficient VM placement method for cloud data centers by using a hybrid genetic algorithm. We also investigate the resource allocation

for VM placement in the cloud center using Particle Swarm Optimization (PSO), a genetic algorithm which was first proposed by Kennedy and Eberhart [55]. In the PSO-based resource allocation scheme, the fitness function can be formed by the objective of the resource allocation model and its constraints, which is similar to that in [56]. The PSO strategy parameters are chosen as $c1 = c2 = 2$ and $\omega = 1$ so as to guarantee the convergence [57].

Table 1. Optimal resource allocation for VMs placement of applications.

variable	x^*_{111}	y^*_1	x^*_{211}	x^*_{222}	y^*_2
algorithm	685.64	685.64	335.30	693.27	1028.57
LINGO	685.71	685.71	239.96	788.61	1028.57
variable	x^*_{312}	y^*_3	x^*_{411}	x^*_{422}	y^*_4
algorithm	1371.51	1371.51	579.09	1135.20	1714.29
LINGO	1371.43	1371.43	674.32	1039.96	1714.29

In order to compare the performance, we first discuss the performance of the proposed resource allocation algorithm, obtain the simulation results with different step sizes, and depict them in the Figure 3(a). It is not difficult to find that as the step size of the proposed scheme increases, the convergence speed increases significantly. Indeed, the convergence performance of the proposed scheme mainly depends on the step size not the number of applications or physical machines. Similarly, we also select different particle swarm sizes and analyze the performance of the PSO-based resource allocation scheme, as shown in Figure 3(b). We can find that the PSO-based scheme converges faster at the beginning, but converges slower near the optimal value. At the same time, as the swarm size increases from 10 to 40, although the convergence speed of the scheme increases, it still cannot effectively converge to the global optimal value within a certain number of iterations.



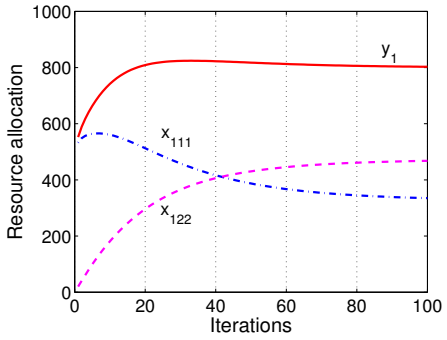
(a)The proposed scheme with different step sizes (b)The PSO-based scheme with different swarm sizes

Figure 3. Optimal objective of the resource allocation model in case 1.

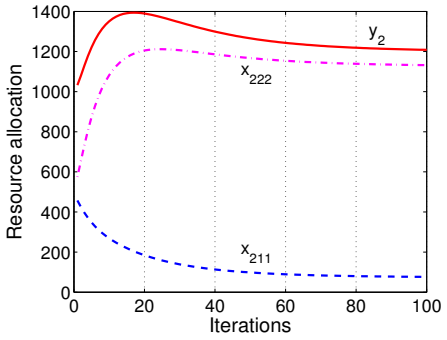
6.2. Large scale scenarios

Following the scenario above, we assume the four applications are all composed of two components. The first one component of these four applications is separately packaged into a virtual

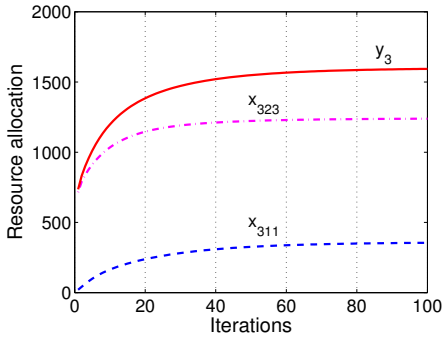
machine residing on physical machine 1, the second component of applications 1 and 2 is packaged into the corresponding virtual machine residing on physical machine 2, and the second component of applications 3 and 4 is packaged into the virtual machine residing on physical machine 3. The three physical machines have a capacity $C = (C_1, C_2, C_3) = (2400, 1600, 1600)$MIPS of CPU resource to complete the deployment of virtual machines. The resource allocation scheme has the same algorithm parameters as the aforementioned simple scenario. the simulation results are illustrated in Figure 4. We can observe that the optimal resource allocation can be also achieved within a certain number of iterations.



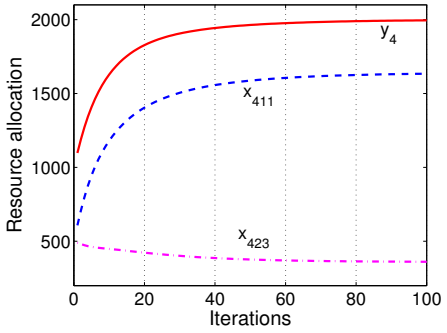
(a)Resource allocation for VMs of application 1



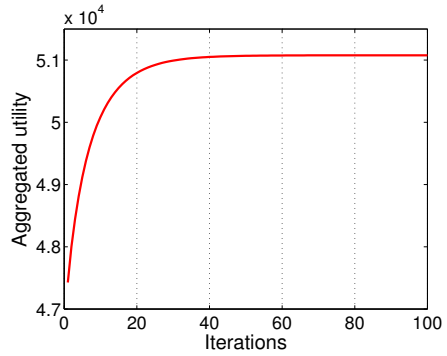
(b)Resource allocation for VMs of application 2



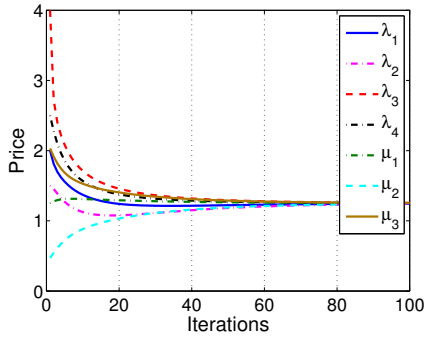
(c)Resource allocation for VMs of application 3



(d)Resource allocation for VMs of application 4



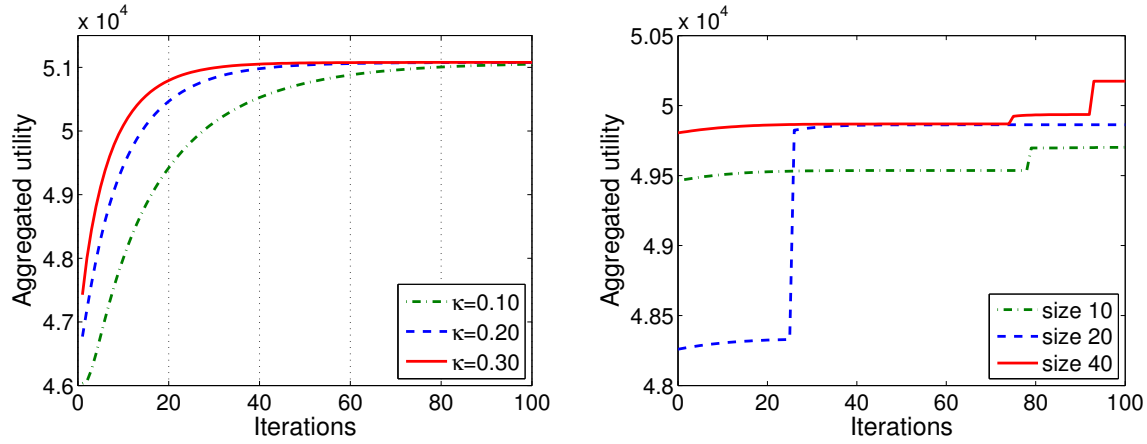
(e)The aggregated utility of applications



(f)Prices

Figure 4. Performance of the resource allocation for VMs placement of applications in case 2.

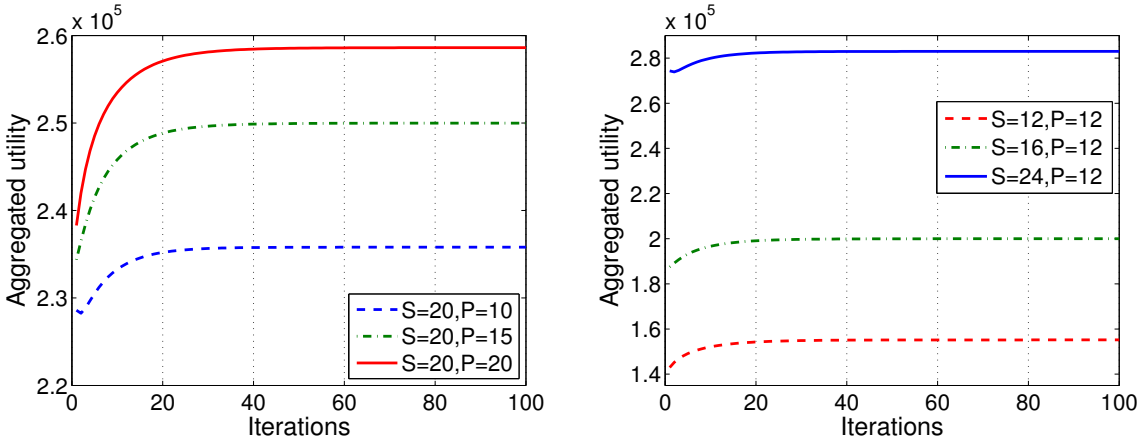
We also analyze the performance of the proposed scheme with different step sizes, and depict the simulation results in Figure 5(a). We find that the convergence speed of the proposed scheme increases significantly as the step size increases. Meanwhile, we also study the performance of PSO-based resource allocation scheme with different swarm sizes and depict the simulation results in Figure 5(b). Similar to Figure 3, the PSO-based scheme converges faster at the beginning, but converges slower when it approaches the optimal value as it is compared with our scheme.



(a)The proposed scheme with different step sizes (b)The PSO-based scheme with different swarm sizes

Figure 5. Optimal objective of the resource allocation model in case 2.

Now we consider the performance of the proposed resource allocation scheme for virtual machines placement of applications in the cloud data center in large scale scenarios, e.g., large numbers of applications as well as physical machines. Here, we assume each application is composed of two components. Each component is packaged into a separate virtual machine which is then placed into the physical machines in the cloud data center. We also take the CPU resource of the physical machines as an example to investigate the performance of the scheme. Each physical machine has a capacity 1600MIPS of its CPU resource. The applications are classified into four categories. The resource allocation scheme has the same parameters as the aforementioned simple scenario. We depict the evolution of aggregated utility of applications in Figure 6. We observe that the size of applications or physical machines in the cloud data center has no obvious affect on the convergence of the proposed resource allocation scheme. The optimal objective (i.e., the aggregated utility) increases with the number of applications or physical machines but, in all scenarios, the optimal value is obtained within almost the same number of iterations (e.g., 100 iterations).



(a)Different number of physical machines (b)Different number of applications

Figure 6. Performance of the resource allocation scheme in different scenarios.

7. Conclusions

With rapid development of the Internet and computer technology, the convenience and cost advantages brought by cloud computing have been adopted by more and more enterprises. At the same time, the resource allocation for virtual machine has become an important key to improve the efficiency and utilization of physical machines. Then, in this paper we establish a resource allocation for virtual machines placement in the cloud data center, deduce the expression of optimal resource allocation for each application, and propose a heuristic algorithm to achieve the optimal resource allocation within a certain number of iterations. We discuss the performance of the proposed resource allocation scheme through theoretical approach and further confirm it with simulation results.

Acknowledgements

The authors would like to thank the support from the National Natural Science Foundation of China (Nos.71971188, 71671159), the Humanity and Social Science Foundation of Ministry of Education of China (No.16YJC630106), the China Postdoctoral Science Foundation (No.2018T110205), the Natural Science Foundation of Hebei Province (Nos.G2018203302, G2020203005), and the project Funded by Hebei Education Department (No.BJ2017029).

Conflict of interest

The authors declare no conflict of interest.

References

1. N. Alsaeed, M. Saleh, *Towards cloud computing services for higher educational institutions: Concepts & literature review*, IEEE International Conference on Cloud Computing (ICCC), 2015, 1–7, Riyadh, Saudi Arabia.

2. P. M. Mell, T. Grance, *The NIST definition of cloud computing*, National Institute of Standards & Technology, 2011.
3. M. Reza, *Framework on large public sector implementation of cloud computing*, IEEE International Conference on Cloud Computing and Social Networking (ICCCSN), 2012, 1–4, Bandung, Indonesia.
4. S. Li, Y. Zhang, W. Sun, *Optimal resource allocation model and algorithm for elastic enterprise applications migration to the cloud*, Mathematics, **7** (2019), 1–20.
5. S. Li, W. Sun, *Utility maximisation for resource allocation of migrating enterprise applications into the cloud*, Enterp. Inf. Syst., 2020.
6. P. D. Bharathi, P. Prakash, M. V. K. Kiran, *Energy efficient strategy for task allocation and VM placement in cloud environment*, IEEE Innovations in Power and Advanced Computing Technologies (i-PACT), 2017, 1–6, Vellore, India.
7. B. Zhang, Z. Qian, W. Huang, et al. *Minimizing communication traffic in data centers with power-aware VM placement*, IEEE Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2012, Palermo, Italy.
8. F. B. Hassen, Z. Brahmi, H. Toumi, *VM placement algorithm based on recruitment process within ant colonies*, IEEE International Conference on Digital Economy (ICDEc), 2016, Carthage, Tunisia.
9. M. Sindelar, P. K. Sitaraman, P. Shenoy, *Sharing-aware algorithms for virtual machine colocation*, ACM Symposium on Parallelism in Algorithms and Architectures, June 04–06, 2011, 367–378, San Jose, California, USA.
10. A. Beloglazov, J. Abawajy, R. Buyya, *Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing*, Futur. Gener. Comp. Syst., **28** (2012), 755–768.
11. S. B. Shaw, A. K. Singh, *Use of proactive and reactive hotspot detection technique to reduce the number of virtual machine migration and energy consumption in cloud data center*, Comput. Electr. Eng., **47** (2015), 241–254.
12. N. K. Sharma, G. R. M. Reddy, *Multi-objective energy efficient virtual machines allocation at the cloud data center*, IEEE Trans. Serv. Comput., **12** (2019), 158–171.
13. Z. Xiao, W. Song, Q. Chen, *Dynamic resource allocation using virtual machines for cloud computing environment*, IEEE Trans. Parallel Distrib. Syst., **24** (2013), 1107–1117.
14. A. Khosravi, L. L. H. Andrew, R. Buyya, *Dynamic VM placement method for minimizing energy and carbon cost in geographically distributed cloud data centers*, IEEE Trans. Sustainable Comput., **2** (2017), 183–196.
15. S. K. Mishra, D. Puthal, B. Sahoo, et al. *An adaptive task allocation technique for green cloud computing*, J. Supercomput., **74** (2018), 370–385.
16. E. Mohammadi, M. Karimi, S. R. Heikalabad. *A novel virtual machine placement in cloud computing*, Aust. J. Basic Appl. Sci., **5** (2011), 1549–1555.
17. S. Rahman, A. Gupta, M. Tornatore, et al. *Dynamic workload migration over backbone network to minimize data center electricity cost*, IEEE Trans. Green Commun. Netw., **2** (2018), 570–579.

18. X. Meng, V. Pappas, L. Zhang, *Improving the scalability of data center networks with traffic-aware virtual machine placement*, Proceedings IEEE INFOCOM, 2010, 1–9, San Diego, CA, USA.

19. W. Wang, B. Liang, B. Li, *Multi-resource fair allocation in heterogeneous cloud computing systems*, IEEE Trans. Parallel Distrib. Syst., **26** (2015), 2822–2835.

20. G. Wei, A. V. Vasilakos, Y. Zheng, et al. *A game-theoretic method of fair resource allocation for cloud computing services*, J. Supercomput., **54** (2010), 252–269.

21. K. Wang, W. Quan, N. Cheng, et al. *Betweenness centrality based software defined routing: Observation from practical Internet datasets*, ACM Trans. Internet. Technol., **19** (2019), 1–19.

22. F. Song, Z. Ai, Y. Zhou, et al. *Smart collaborative automation for receive buffer control in multipath industrial networks*, IEEE Trans. Ind. Inform., **16** (2020), 1385–1394.

23. Z. Ai, Y. Zhou, F. Song, *A smart collaborative routing protocol for reliable data diffusion in IoT scenarios*, Sensors, **18** (2018), 1–21.

24. F. Song, M. Zhu, Y. Zhou, et al. *Smart collaborative tracking for ubiquitous power IoT in edge-cloud interplay domain*, IEEE Int. Things J., 2020.

25. K. Wang, H. Yin, W. Quan, et al. *Enabling collaborative edge computing for software defined vehicular networks*, IEEE Netw., **32** (2018), 112–117.

26. F. Lin, X. Lv, I. You, et al. *A novel utility based resource management scheme in vehicular social edge computing*, IEEE Access, **6** (2018), 66673–66684.

27. G. H. S. Carvalho, I. Woungang, A. Anpalagan, et al. *Intercloud and hetNet for mobile cloud computing in 5G systems: Design issues, challenges, and optimization*, IEEE Netw., **31** (2017), 80–89.

28. Z. Ai, Y. Liu, F. Song, et al. *A smart collaborative charging algorithm for mobile power distribution in 5G networks*, IEEE Access, **6** (2018), 28668–28679.

29. F. Song, Y. Zhou, L. Chang, et al. *Modeling space-terrestrial integrated networks with smart collaborative theory*, IEEE Netw., **33** (2019), 51–57.

30. F. Song, Y. Zhou, Y. Wang, et al. *Smart collaborative distribution for privacy enhancement in moving target defense*, Inf. Sci., **479** (2019), 593–606.

31. C. Helene, G. L. Louet, J. M. Menaud, *Virtual machine placement for hybrid cloud using constraint programming*, IEEE 23rd International Conference on Parallel and Distributed Systems (ICPADS), 2017, 326–333.

32. M. S. P. Mohamed, S. R. Swarnammal, *An efficient framework to handle integrated VM workloads in heterogeneous cloud infrastructure*, Soft Comput., **21** (2017), 3367–3376.

33. S. Chaisiri, B. S. Lee, D. Niyato, *Optimization of resource provisioning cost in cloud computing*, IEEE Trans. Serv. Comput., **5** (2012), 164-177.

34. X. Zheng, Y. Xia, *Exploring mixed integer programming reformulations for virtual machine placement with disk anti-colocation constraints*, Perform. Eval., **135** (2019), 1–18.

35. B. Xu, Z. Peng, F. Xiao, et al. *Dynamic deployment of virtual machines in cloud computing using multi-objective optimization*, Soft Comput., **19** (2015), 2265–2273.

36. D. Zhao, J. Zhou, K. Li, *An energy-aware algorithm for virtual machine placement in cloud computing*, IEEE Access, **7** (2019), 55659–55668.

37. M. A. Kaaouache, S. Bouamama, *An energy-efficient VM placement method for cloud data centers using a hybrid genetic algorithm*, J. Syst. Inf. Technol., **20** (2018), 430–445.

38. F. Stefanello, V. Aggarwal, L. S. Buriol, et al. *Hybrid algorithms for placement of virtual machines across geo-separated data centers*, J. Comb. Optim., **38** (2019), 748–793.

39. X. Liu, Z. Zhan, J. D. Deng, et al. *An energy efficient ant colony system for virtual machine placement in cloud computing*, IEEE Trans. Evol. Comput., **22** (2018), 113–128.

40. F. Alharbi, Y. Tian, M. Tang, et al. *An ant colony system for energy-efficient dynamic virtual machine placement in data centers*, Expert Syst. Appl., **120** (2019), 228–238.

41. X. Wang, Z. Liu, *An energy-aware VMs placement algorithm in cloud computing environment*, IEEE Second International Conference on Intelligent System Design and Engineering Application, 2012, 627–630, Sanya, Hainan, China.

42. M. A. Kaaouache, S. Bouamama, *Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud*, Procedia Comput. Sci., **60** (2015), 1061–1069.

43. J. Xu, J. A. B. Fortes, *Multi-objective virtual machine placement in virtualized data center environments*, IEEE/ACM International Conference on Green Computing and Communications & International Conference on Cyber, Physical and Social Computing, 2010, 179–188, Hangzhou, China.

44. S. Dörterler, M. Dörterler, S. Ozdemir, *Multi-objective virtual machine placement optimization for cloud computing*, IEEE International Symposium on Networks, Computers and Communications (ISNCC), 2017, Marrakech, Morocco.

45. D. Jayasinghe, C. Pu, T. Eilam, et al. *Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement*, IEEE International Conference on Services Computing, 2011, 72–79, Washington, DC, USA.

46. W. Fang, X. Liang, S. Li, et al. *VMPlanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers*, Comput. Netw., **57** (2013), 179–196.

47. L. Guo, T. Yan, S. Zhao, et al. *Dynamic performance optimization for cloud computing using M/M/m queueing system*, J. Appl. Math., **2014** (2014), 1–8.

48. Z. Xiao, J. Jiang, Y. Zhu, et al. *A solution of dynamic VMs placement problem for energy consumption optimization based on evolutionary game theory*, J. Syst. Softw., **101** (2015), 260–272.

49. F. Song, D. Huang, H. Zhou, et al. *An optimization-based scheme for efficient virtual machine placement*, Int. J. Parallel Program., **42** (2014), 853–872.

50. D. Huang, L. Yi, F. Song, et al. *A secure cost-effective migration of enterprise applications to the cloud*, Int. J. Commun. Syst., **27** (2014), 3996–4013.

51. M. Chiang, S. H. Low, A. R. Calderbank, et al. *Layering as optimization decomposition: a mathematical theory of network architectures*, Proc. IEEE, **95** (2007), 255–312.

52. S. Li, W. Sun, Q. L. Li, *Utility maximization for bandwidth allocation in peer-to-peer file-sharing networks*, J. Ind. Manag. Optim., **16** (2020), 1099–1117.

53. W. E. Boyce, R. C. DiPrima, *Elementary Differential Equations and Boundary Value Problems*, Hoboken: John Wiley & Sons, 2005.

54. Q. V. Pham, W. J. Hwang, *Network utility maximization based congestion control over wireless networks: A survey and potential directives*, IEEE Commun. Surv. Tut., **19** (2017), 1173–1200.

55. J. Kennedy, R. C. Eberhart, *Particle swarm optimization*, Proceeding of the 1995 IEEE International Conference on Neural Networks (ICNN), 1995, 1942–1948.

56. S. Li, W. Sun, J. Liu, *A mechanism of bandwidth allocation for peer-to-peer file-sharing networks via particle swarm optimization*, J. Intell. Fuzzy Syst., **35** (2018), 2269–2280.

57. M. Clerc, J. Kennedy, *The particle swarm-explosion, stability, and convergence in a multidimensional complex space*, IEEE Trans. Evol. Comput., **6** (2002), 58–73.

AIMS Press