Efficient Multi-Tenant Virtual Machine Allocation in Cloud Data Centers

Jiaxin Li
the National Key Laboratory of Parallel and Distributed Processing (PDL), College of Computer, National University of Defense Technology, Changsha 410073, China.

Dongsheng Li
the National Key Laboratory of Parallel and Distributed Processing (PDL), College of Computer, National University of Defense Technology, Changsha 410073, China.

Yuming Ye
the National Key Laboratory of Parallel and Distributed Processing (PDL), College of Computer, National University of Defense Technology, Changsha 410073, China.

Xicheng Lu
the National Key Laboratory of Parallel and Distributed Processing (PDL), College of Computer, National University of Defense Technology, Changsha 410073, China.

Follow this and additional works at: https://tsinghuauniversitypress.researchcommons.org/tsinghua-science-and-technology

Part of the Computer Sciences Commons, and the Electrical and Computer Engineering Commons

Recommended Citation

This Research Article is brought to you for free and open access by Tsinghua University Press: Journals Publishing. It has been accepted for inclusion in Tsinghua Science and Technology by an authorized editor of Tsinghua University Press: Journals Publishing.
Efficient Multi-Tenant Virtual Machine Allocation in Cloud Data Centers

Jiaxin Li*, Dongsheng Li, Yuming Ye, and Xicheng Lu

Abstract: Virtual Machine (VM) allocation for multiple tenants is an important and challenging problem to provide efficient infrastructure services in cloud data centers. Tenants run applications on their allocated VMs, and the network distance between a tenant’s VMs may considerably impact the tenant’s Quality of Service (QoS). In this study, we define and formulate the multi-tenant VM allocation problem in cloud data centers, considering the VM requirements of different tenants, and introducing the allocation goal of minimizing the sum of the VMs’ network diameters of all tenants. Then, we propose a Layered Progressive resource allocation algorithm for multi-tenant cloud data centers based on the Multiple Knapsack Problem (LP-MKP). The LP-MKP algorithm uses a multi-stage layered progressive method for multi-tenant VM allocation and efficiently handles unprocessed tenants at each stage. This reduces resource fragmentation in cloud data centers, decreases the differences in the QoS among tenants, and improves tenants’ overall QoS in cloud data centers. We perform experiments to evaluate the LP-MKP algorithm and demonstrate that it can provide significant gains over other allocation algorithms.

Key words: virtual machine allocation; cloud data center; multiple tenants; multiple knapsack problem

1 Introduction

Virtual Machine (VM) allocation[1] is a fundamental issue in Infrastructure-as-a-Service cloud computing systems[2]. In general, before tenants deploy their applications in a cloud computing system, they need to request VMs from the cloud data centers to meet the requirements of applications. Then, the resource manager of the cloud system selects and assigns appropriate physical resources for each requested VM. Therefore, VM allocation algorithms have a direct impact on many aspects of cloud systems, including resource utilization, application performance, and the ability to satisfy tenant requirements.

Most studies[3] primarily focus on VM allocation for individual tenants without considering multiple tenants simultaneously. Moreover, existing VM allocation methods usually have some serious defects. For example, during allocation, they tend to place VMs on racks or Physical Machines (PMs) having more free resources[4]. However, this results in large amounts of resource fragmentation in cloud data centers, making them difficult to be used and thereby reducing overall resource utilization. In addition, when dealing with a tenant request, current VM allocation algorithms usually focus on the current best solution (i.e., local optimal solution) for the individual tenant request, ignoring the impact on tenant requests that arrive later. Hence, current algorithms often result in poor Quality of Service (QoS) for subsequent requests and fail to achieve a global optimal solution, and this leads to significant differences in the QoS among tenants[5].

Tenants run applications on their allocated VMs, and the network distance between a tenant’s VMs may significantly impact the tenant’s QoS. We define
network diameter as the maximum network distance between the allocated VMs for a tenant request. In this paper, for a tenant request, we suppose that the QoS is determined mainly by its network diameter and a smaller network diameter is better\cite{4}. A greater network distance between VMs can lead to longer communication time and longer job completion time, which are determined by the maximum task completion time. Therefore, for all tenant requests in a given time period, we aim to minimize the sum of the VMs’ network diameters of all tenants, i.e., to get the minimum sum of all tenants’ network diameters.

For solving the multi-tenant virtual machine allocation problem, we take into account several aspects: (1) examining the VM requirements of multiple tenants and their relationship, (2) reducing resource fragmentation and utilizing it in cloud data centers, (3) decreasing the differences in the QoS among tenants and improving the tenants’ overall QoS, and (4) attempting to achieve a local optimal solution for each tenant request and a global optimal solution across all tenant requests. Finally, we propose a Layered Progressive resource allocation algorithm for multi-tenant cloud data centers based on the Multiple Knapsack Problem (LP-MKP). Specifically, this paper makes the following contributions:

- We define and formulate the multi-tenant VM allocation problem in cloud data centers.
- We design and implement LP-MKP, a layered progressive resource allocation algorithm based on the multiple knapsack problem.
- We compare our approach with two other algorithms. The experimental results show that LP-MKP is significantly superior to the greedy algorithm based on maximum idle resources and better than the heuristic algorithm based on minimum subtrees. Thus, LP-MKP can be efficiently applied to VM allocation for multi-tenant cloud data centers.

The rest of the paper is organized as follows. Section 2 describes the background of multi-tenant VM allocation and formulates the problem. Section 3 proposes the LP-MKP resource allocation algorithm. Section 4 presents the experimental evaluation. Section 5 describes the related work. Finally, the paper concludes in Section 6.

2 Problem Description

In this section, we first provide background on the multi-tenant VM allocation problem followed by the problem formulation.

2.1 Background

Cloud computing uses Internet Data Centers (IDCs) as the basic infrastructure to provide tenants with various applications. A large IDC may consist of over one hundred racks, and each rack usually comprises about twenty PMs. These PMs are usually interconnected by dedicated high-speed data center networks\cite{6}. Tenants’ requests for cloud system resources may arrive at any time, e.g., they may arrive at the same time or over a period of time. Cloud resource allocation algorithms select appropriate racks and PMs for strategically placing the VMs to minimize the communication costs between VMs, maximize the resource utilization, and improve application performance.

The typical architecture of a hierarchical data center is shown in Fig. 1. The topmost router interacts with the outside data centers. The bottom PMs are placed in racks. Each rack has a top switch. Typically, there are several layers of switching devices between the top and bottom layers of the data center. The ideal approach is to place the VMs requested by a tenant on the same PM or rack. However, the resources on the same PM or rack may be insufficient to meet the tenant’s needs. Moreover, tenants may choose to reduce some resources or log out of the cloud system after completion of jobs, which results in large amounts of resource fragmentation in the cloud system. Therefore, the requested VMs may be distributed among several PMs or several racks.

Given the number of VMs requested by each tenant within a certain period of time, we aim to minimize the sum of all tenants’ network diameters. To solve this, we formulate the multi-tenant virtual machine allocation problem.
2.2 Problem formulation

Given the current idle resources in a data center, we build a model for VM resource allocation in the cloud system. Assuming that there are \( n \) tenant requests, the \( t \)-th (\( 1 \leq t \leq n \)) request needs \( K(t) \) VMs. After VM allocation, we use \( G(t) = (V(t), E(t)) \) to denote the relationship diagram between multiple VMs for the \( t \)-th tenant, while \( V(t) = \{v_{ij}^{(t)} | 1 \leq i \leq K(t) \} \) denotes the allocated \( K(t) \) VMs, and \( E(t) = \{e_{ij}^{(t)}(v_i^{(t)}, v_j^{(t)}) | 1 \leq i, j \leq K(t) \} \) denotes the network distances between \( K(t) \) VMs. The process of VM resource allocation is modeled and formulated in detail in Section 3, taking into account the VM requirements of the tenant requests and the constraints of the idle physical resources in the data center.

We define \( R(t) \) as the network diameter of the \( t \)-th tenant, which denotes the maximum network distance among all allocated VMs, and is given by

\[
R(t) = \max \{d_{ij}^{(t)} : 1 \leq i, j \leq K(t) \}
\] (1)

The optimization goal is to minimize the Sum of all tenants’ network Diameters (SD).

Minimize \( SD = \sum_{t=1}^{n} R(t) \) (2)

The network distance can be represented by the network hop, network delay, and so on. In this study, we consider the network hop as an example.

2.3 Problem analysis

Many data centers have adopted the hierarchical architecture shown in Fig. 1. Each data center of the typical hierarchical architecture (e.g., VL2[7] and Fat-Tree[8]) can be modeled as a full multi-branched information tree (Fig. 2) with the same number of layers as the data center. The information tree for a data center can be constructed as follows. From the bottom up, the first layer is a leaf node layer, representing the resources of the PMs; the second layer consists of racks; the third and above layers represent switching layers. Each node on the information tree can record available resources (e.g., the number of available VMs) of the subtree rooted at itself and other related information.

Figure 2 shows an example of the information tree for a hierarchical data center, which is a full multi-branched tree with a height of four. Each node has been marked with two values: the number of available VMs (i.e., weight) and the height of the subtree rooted at itself. For example, the total number of available VMs in this information tree is 105 and its height is 4, thus the root node of the tree is marked by (105, 4). In practice, the height of the information tree may increase because of multiple switching layers.

The VMs requested by tenants will be placed into the leaf nodes of the information tree. For an individual tenant, the more decentralized the leaf nodes where the VMs are placed, the higher the height of the subtree rooted at the Least Common Ancestor (LCA) of these leaf nodes and the greater the network diameter. For example, if we place VMs into nodes \( a \) and \( b \) (Fig. 2), we will obtain a network diameter of 2, because there are 2 network hops from node \( a \) to \( b \). But if we place VMs into nodes \( a \), \( b \), and \( c \), there are 6 network hops from node \( a \) to \( c \) or node \( b \) to \( c \); thus, the maximum network distance (i.e., network diameter) is 6.

Therefore, the optimal placement strategy is to select the subtree with the minimum height whose available VM resources are sufficient to meet the tenant request. However, for multiple tenants, we need to find a global optimal solution. Since selecting a feasible solution for the current tenant may affect the solutions of other tenants, we may not select the minimum subtree (i.e., the subtree with a minimal height) from all available ones for a tenant, in order to minimize the sum of all tenants’ network diameters.

To achieve the goal of minimizing the sum of all tenants’ network diameters, we investigate the effect that nodes in different layers of the information tree toward the goal, without distinguishing nodes in the same layer. By placing VMs into the subtrees, whose roots are in the same layer, we can achieve the same network diameter. As can be seen in Fig. 2, from the bottom layer to the top, the network diameters in order are 0, 2, 4, and 6.

3 LP-MKP Algorithm

The problem of multi-tenant VM allocation,
formulated and analyzed in Section 2, is an NP-hard problem. Owing to space considerations, we omit the proof in this paper.

In this section, we propose LP-MKP, an approximate solution to the multi-tenant VM allocation problem. LP-MKP is a multi-stage layered progressive VM allocation algorithm for multiple tenant requests, in order to deal with unprocessed tenant requests at each stage as efficiently as possible.

To reduce resource fragmentation in cloud data centers, decrease the differences in the QoS among tenants, and achieve the goal of minimizing the sum of all tenants’ network diameters, LP-MKP considers these four aspects: (1) examining the VM requirements of multiple tenants and their relationship, (2) reducing resource fragmentation and utilizing it in cloud data centers, (3) decreasing the differences in the QoS among tenants and improving the overall QoS across all tenants, and (4) attempting to achieve both a local optimal solution for each tenant request and a global optimal solution across all tenant requests.

Specifically, LP-MKP first divides the information tree into several layers from the bottom up and deals with them in order. Each layer corresponds to a stage of VM resource allocation. Placing VMs into subtrees whose root nodes are in different layers results in different network diameter, and we tend to place VMs into the subtrees whose root nodes are in the lower layers as it will result in a smaller diameter. At each stage of VM resource allocation, all unprocessed tenant requests and the remaining VM resources of this layer are modeled as an allocation model based on the multiple knapsack problem, which is solved with an approximation algorithm. The model based on the multiple knapsack problem can effectively deal with tenant requests by considering the VM requirements of different tenant requests and their relationship. Then, LP-MKP is completed when all tenant requests are processed. So LP-MKP may not need to deal with the upper layers of the information tree.

As shown in Fig. 3, taking a full ternary information tree, with a height of four, as an example. The upper part of the figure shows available resources of the information tree. Label 1 represents leaf nodes, label 2 represents intermediate nodes just above the leaf nodes, and so on. We divide the process of VM resource allocation into four stages from the bottom up, and each stage corresponds to the nodes in one layer.

The lower part of the figure indicates the tenant requests. We assume that there are \( n \) tenant requests, where \( K^{(i)} \) denotes the number of resources for the \( i \)-th (\( 1 \leq i \leq n \)) tenant request. In the first stage, we take 27 leaf nodes (labelled 1) as multiple knapsacks and take \( K^{(i)} (1 \leq i \leq n) \) as \( n \) items to be placed. As seen in Fig. 3, requests \( K^{(1)}, K^{(4)}, K^{(6)}, K^{(7)}, K^{(8)}, \ldots, K^{(n)} \) are placed into nodes labelled 1. In the second stage, we take 9 intermediate nodes (labelled 2) as multiple knapsacks and continue to place the remaining items that were not placed in the first stage. Finally requests \( K^{(2)}, K^{(3)}, K^{(5)}, \) and so on, are placed into nodes labelled 2. We repeat this process in the third and fourth stages, which results in request \( K^{(9)} \) being placed into the node labelled 3. Through the layered progressive allocation process, we complete the VM resource allocation for all tenant requests. At the beginning of each stage, we aggregate the remaining resource fragmentation from the nodes in the previous stage to obtain the available resources for the current stage.

Each stage of VM resource allocation corresponds to the nodes in one layer. We assume that the number of nodes is \( m \), the number of available VM resources for the \( i \)-th (\( 1 \leq i \leq m \)) node is \( c_i \), the number of current requests is \( n \), and the number of VM resources required by the \( j \)-th (\( 1 \leq j \leq n \)) tenant request is \( w_j \). We add
a corresponding profit value $p_j$ for each request, which helps build up the model. Consequently, it becomes the 0-1 MKP. To maximize the total profits, we need to select $m$ disjoint sets from $n$ requests and place them into $m$ nodes separately under the condition that the total number of VM resources for each set is not more than that of the corresponding node. Formulating the problem, we obtain the following.

MKP’s objective:

$$\text{Maximize } z = \sum_{i=1}^{m} \sum_{j=1}^{n} p_j x_{ij}$$

Constraints:

$$\sum_{j=1}^{n} w_j x_{ij} \leq c_i, 1 \leq i \leq m$$

$$\sum_{i=1}^{n} x_{ij} \leq 1, 1 \leq j \leq n$$

$$x_{ij} = 0 \text{ or } 1, 1 \leq i \leq m, 1 \leq j \leq n$$

where

$$x_{ij} = \begin{cases} 1, & \text{if request } j \text{ is assigned to node } i; \\ 0, & \text{otherwise.} \end{cases}$$

To solve the problem, we define the profit value $p_j (1 \leq j \leq n)$. When $p_j = 1$, the goal is to achieve the maximum number of requests, which we use in our experiments. When $p_j = w_j$, the goal is to achieve the maximum number of VM resources.

Let set $C = \{c_i, 1 \leq i \leq m\}$, $W = \{w_j, 1 \leq j \leq n\}$, and $P = \{p_j, 1 \leq j \leq n\}$. We use $T$ to denote the information tree, where $T(i)$ is the set of all nodes in the $i$-th layer, $C$ is the set of the number of available VM resources in $T(i)$. Let request $R(W, P) = \{\text{request}_j(w_j, p_j), 1 \leq j \leq n\}$.

Algorithm 1 presents an approximation algorithm for solving MKP called the Martello and Toth heuristic method, whose time complexity is $O(n^3)$. First, we sort node set $C$ in ascending order by $c_i$ and sort request set $R$ in descending order by $p_j/w_j$. Second, for each node in set $C$, we use a greedy strategy to choose satisfiable requests in order until there are no more satisfiable requests. Third, if possible, we insert additional requests into the nodes by swapping any two requests that are on different nodes. Finally, for each request in the nodes, if we can replace it with other requests that are in the nodes to increase the profit value of its node, then do it.

Algorithm 2 presents the layered progressive resource allocation algorithm based on MKP for multi-tenant cloud data centers, named LP-MKP. It uses a multi-stage layered progressive method for VM placement to deal with unprocessed requests at each stage as efficiently as possible. LP-MKP deals with the information tree from the bottom layer to the top in order. For each stage, it calls the approximation algorithm for MKP (Algorithm 1) to compute a VM allocation scheme. Then, it gets rid of the processed requests according to the allocation scheme and updates the current available VM resources. Subsequently, it moves into the next stage and repeats the same process until the algorithm ends.
Algorithm 2 LP-MKP($T$, $R$), Layered progressive resource allocation algorithm based on MKP.

Input: Information tree $T$, where $T(i).C$ denotes the node set of the $i$-th layer; Request set $R$;
Output: The sum of all tenants’ network diameters SD;

1. $SD \leftarrow 0$;
2. for $i = \text{layer}(T)$ downto 1 do
3.  $\text{MKP}(T(i).C, R)$;
4.  $U \leftarrow \{\text{request}_y : y_u > 0\}$;
5.  $SD \leftarrow SD + |U| \cdot (\text{layer}(T) - i)$;
6.  $R \leftarrow R - U$;
7.  for each node $v$ in $T(i - 1)$ do
8.   $T(i - 1).c(v) \leftarrow \sum_{u \in \text{son}(v)} T(i).c(u)$
9.  end for
10. end for
11. return $SD$

4 Experimental Evaluation

In this section, we present the experimental evaluation for the LP-MKP algorithm. We first present our experimental setup followed by experimental results.

4.1 Experiment setup

We compare our approach with (1) the greedy algorithm based on maximum idle resources (Greedy) that chooses the branch with the maximum available resources in the information tree from the top down to allocate VMs for each tenant request, and (2) the heuristic allocation algorithm based on minimum subtrees (MinTree) that selects the subtree with the minimum height in the information tree to allocate VMs for each tenant request. Both the Greedy and MinTree algorithms deal with tenant requests according to the sequence of the tenant requests’ arrival.

We perform our experiments on four different types of full $k$-ary information trees ($k = 2, 3, 4, 5$). As shown in Table 1, the height of the information trees is 5, and the number of VMs of each leaf node is distributed randomly in the range of 0 to 100. The number of leaf nodes and total number of available VMs in the information trees are detailed in Table 1. The experimental data of tenant requests uses four types of random integer sequences with different ranges (in the range $[1, n]$, $n = 100, 200, 500, 1000$). The length of every sequence is about $2^{(k-1)} \frac{50}{n/2}$, so that the total number of VMs requested by tenants is equal to the corresponding information tree’s available VM resources. We report the results as an average of 100 runs.

4.2 Results and analysis

Figure 4 shows the results of the three algorithms in four different experimental settings (request sequences in the range $[1, n]$, $n = 100, 200, 500, 1000$). As the scale of the information trees increases (i.e., the increasing value of variable $k$), we see that all three algorithms get a larger sum of all tenants’ network diameters (i.e., a larger SD value); because the more available VMs the information trees have, the more tenant requests they will have. Among these three algorithms, LP-MKP is significantly superior to the Greedy algorithm and better than the heuristic allocation algorithm MinTree. Figure 4a shows that LP-MKP is particularly efficient. The reason is that request sequences in the range $[1, 100]$ match the information trees whose number of available VMs in leaf nodes ranges from 0 to 100 at random. In such cases, LP-MKP can make good use of the advantages of MKP.

We analyze one of the experimental results (for one of the 100 runs) in Fig. 4a. The QoS differences among tenant requests are shown in Fig. 5. We sort the tenant requests in ascending order by the number of VMs requested by each tenant. Thereby, we can compare the network diameters of similar tenant requests that require a same or an approximate number of VMs. We hold that the QoS of VM resources is determined by the network diameter of VMs. As we can see from the figure, as the number of the requested VMs increases, the results of the Greedy and MinTree algorithms present a trend of obvious fluctuation and show significant differences in the QoS. By contrast, LP-MKP presents a trend of increase except for a few abnormal cases and shows fewer differences in the QoS. Overall, LP-MKP can ensure that the request with less VMs gets a smaller diameter efficiently, i.e., it guarantees the fairness of resource allocation for similar tenant requests.

Figure 6 shows the increased percentage of performance of LP-MKP compared with the MinTree

<table>
<thead>
<tr>
<th>$k$</th>
<th>Height of the information tree</th>
<th>Number of leaf nodes</th>
<th>Value range of each leaf node (#VMs)</th>
<th>Total number of available VMs</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>5</td>
<td>$2^2=16$</td>
<td>[0,100]</td>
<td>about 800</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>$3^4=81$</td>
<td>[0,100]</td>
<td>about 4050</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>$4^4=256$</td>
<td>[0,100]</td>
<td>about 12 800</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>$5^4=625$</td>
<td>[0,100]</td>
<td>about 31 250</td>
</tr>
</tbody>
</table>
algorithm. As seen from the results, the performance improvement of LP-MKP is obvious, with an average increase in performance from 5% to 10%. Theoretically, with an increasing range \([1, n]\) for request sequences (i.e., \(n\) gets bigger), the number of requests is less. This will cause a gradual decrease in the increased percentage of performance for the same scale of information trees (i.e., \(k\) remains constant). However, the increased percentage of performance presents a state of intersection (e.g., when \(k = 3, 4\)). This is because we use an approximation algorithm for MKP, which is used in LP-MKP. With the increasing scale of information trees, the increased percentage of performance decreases (except for the range \([1, 100]\)), because the degree of approximation of the approximation algorithm reduces with the increasing scale of information trees. However, due to the fact that the increasing number of tenant requests will facilitate VM resource allocation for LP-MKP by taking more tenant requests into account, LP-MKP can still efficiently solve the VM resource allocation problem for multi-tenant data centers.

From the above experimental evaluation, we can see that LP-MKP can be efficiently applied to VM allocation for multi-tenant in cloud data centers.


## 5 Related Work

Currently, there are a lot of research works on VM resource allocation. Several studies modeled the issue as a one-dimensional or multi-dimensional bin packing problem \[10, 11\]. Nakada et al. \[10\] defined the issue as a multi-objective optimization problem and proposed a genetic algorithm that took the Service-Level Agreement (SLA) and the minimum number of servers required into account. Gao et al. \[12\] designed a multi-objective ant colony algorithm for virtual machine placement.

Furthermore, Song et al. \[13\] abstracted the VM placement as an optimization problem that considered the inherent dependencies and traffic between VMs. Gupta et al. \[14\] considered the server consolidation problem with item-item and bin-item incompatibility constraints and proposed a two-stage heuristic allocation algorithm. Zhu et al. \[15\] studied the reliable virtual machine allocation problem with the objective of minimizing the total failure probability.

Some studies considered on-line or on-demand resource allocation in cloud data centers. For example, Song et al. \[16\] proposed a two-tiered on-demand resource allocation mechanism consisting of local and global resource allocation with feedback to provide on-demand capacity to concurrent applications. Hao et al. \[17\] proposed a generalized resource placement methodology that can work across different cloud architectures and resource request constraints, together with real-time request arrivals and departures. Ahmed and Wu \[18\] proposed a novel approach using the hidden Markov model to estimate the future demand of cloud nodes for resource allocation.

Moreover, some studies investigated the resource allocation problem from other perspectives. Alicherry and Lakshman \[19\] studied network-aware resource allocation and proposed a heuristic allocation algorithm based on minimum subtrees. Beloglazov et al. \[20\] presented resource provisioning and allocation algorithms for energy-efficient management of cloud computing environments. Meng et al. \[21\] proposed a traffic-aware virtual machine placement method to improve network scalability.

In this study, however, we choose the multi-tenant issue in cloud resource allocation as the researching point, and consider the relationship between multiple tenants to reduce resource fragmentation in cloud data centers, decrease the differences in the QoS among tenants, and improve overall service quality in cloud data centers. We first define and formulate the VM resource allocation problem for multi-tenant data centers, and then take the minimum sum of all tenants’ network diameters as the optimization goal.

## 6 Conclusions

Multi-tenant VM allocation in cloud data centers is a type of NP-hard problem. Existing methods usually result in low utilization of cloud data centers and significant differences in the QoS among multiple tenants. To solve this problem, we propose a layered progressive resource allocation algorithm based on the multiple knapsack problem called LP-MKP. The LP-MKP algorithm considers the VM requirements of different tenants and their relationship and takes the minimum sum of all tenants’ network diameters as the optimization goal, in order to reduce the resource fragmentation in cloud data centers, decrease the differences in the QoS among tenants, and improve the overall QoS across all tenants for cloud data centers. The experimental results show that LP-MKP can efficiently deal with the VM resource allocation problem for multi-tenant in cloud data centers. Our future study will focus on integrating the LP-MKP algorithm into open-source cloud computing platforms, such as OpenStack \[21\] and CloudStack \[22\].

## Acknowledgements

This work was supported in part by the National Key Basic Research and Development (973) Program of China (No. 2011CB302600), the National Natural Science Foundation of China (No. 61222205), the Program for New Century Excellent Talents in University, and the Fok Ying-Tong Education Foundation (No. 141066).

## References


Jiaxin Li et al.: Efficient Multi-Tenant Virtual Machine Allocation in Cloud Data Centers


Jiaxin Li is a PhD candidate in the College of Computer at National University of Defense Technology (NUDT). He received his master degree in computer science and technology from NUDT in 2013. His research interests include distributed systems, and computer networks & communications.

Dongsheng Li is a professor and doctoral supervisor in the College of Computer at National University of Defense Technology (NUDT). He received his PhD degree in computer science and technology from NUDT in 2005. He was awarded the Chinese National Excellent Doctoral Dissertation in 2008. His research interests include distributed systems, cloud computing, and big data processing.

Yuming Ye is currently a master student in the College of Computer at National University of Defense Technology (NUDT). He received his bachelor degree from Shanghai Jiaotong University in 2013. His research interests include distributed systems and big data processing.

Xicheng Lu is a professor and doctoral supervisor in the College of Computer at National University of Defense Technology (NUDT). He is a member of the Chinese Academy of Engineering since 1999. His research interests include parallel and distributed processing, and computer networks.