Efficient Algorithms for VM Placement in Cloud Data Center

Jiahuai Wu[1] and Hong Shen[1,2](✉)

[1] School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China
849197033@qq.com, hongsh01@gmail.com
[2] School of Computer Science, University of Adelaide, Adelaide, Australia

Abstract. Virtual machine (VM) placement problem is a major issue in cloud data center. With the rapid development of cloud computing, efficient algorithms are needed to reduce the power consumption and save energy in data centers. Many models and algorithms are designed with an objective to minimize the number of physical machines (PMs) used in cloud data center. In this paper, we take into account the execution time of the PM, and formulate a new optimization problem of VM placement, which aims to minimize the total execution time of the PMs. We discuss the NP-hardness of the problem, and present heuristic algorithms to solve it under both offline and online scenario. Furthermore, we conduct experiments to evaluate the performance of the proposed algorithms and the result show that our methods are able to perform better than other commonly used algorithms.

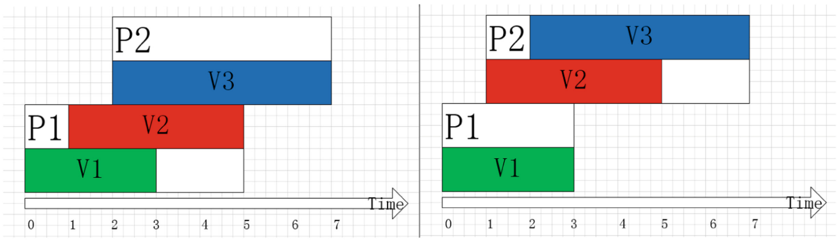
Keywords: Cloud data center · Virtual machine placement · Bin packing · Heuristic algorithm

1 Introduction

Cloud computing has become an emerging technology that transforms the IT industry and affects people's lives in recent years. Today in most modern cloud data centers, such as Amazon EC2 [16] and Google data center, there are a large number of physical machines (PMs), also called servers or hosts, and the total number of the PMs in each data center can reach hundreds of thousands. However, due to the uneven resource demand of the applications, a good few of the PMs have very low utilization most of the time. An unnecessarily great number of PMs have to be opened with high cost for the management and maintenance, which results in a serious waste of resources. Hence, virtualization technology [2] is applied to settle these issues.

With the tremendous benefits of virtualization, applications are running on the VMs, not directly on the PMs. Furthermore, a single PM can accommodate multiple VMs as long as their resource demands are satisfied. In another word, the applications may be able to share the resources on the PMs in an isolated way. And in the majority of cases the load of a VM has almost no effect on the performance of the co-located VMs [15]. In some cloud case, particularly in the Infrastructure-as-a-Service (IaaS), the VMs are provided directly to the customers. Typically, the VM placement problem is a serious challenge for the data centers. Usually, the customers submit their resource requirements in terms of the basic resource, including CPU, memory, network, etc., to the cloud system, and the cloud system needs to decide the resource allocation. As the cost caused by PMs is proportional to the number of running PMs [10], lots of research aim to use the virtualization technology to consolidate the VMs onto a smaller number of PMs so that the saved PMs can be switched to a low power mode or shut down. Therefore, the objective is to minimizing the number of the PMs that hosts the requested VMs.

To study the VM placement problem for PM cost minimization. We explain that the cost caused by PMs is proportional to not only the number but also the execution time of the PMs. We further assume that the VM requests from the tenants contain both resources demand and the running time of VMs, and the PM cost is mainly determined by the total execution time of all the used PMs accordingly. For example, in Fig. 1, there are three VMs submitted to the cloud system with the same resource demand. Their running time are 3, 4 and 5 respectively, Fig. 1a and b are two different VM placements. As a result, the placement in Fig. 1b has fewer PM cost since the total execution time of the PMs are fewer.



(a) placement with cost 10. (b) placement with cost 9.

Fig. 1. Two placements for the same request (V1, V2, V3). All the three VMs are of half size of the PM and different in the running time.

Obviously, packing VMs onto a number of PMs and minimizing the total execution time of the PMs is an effective way to reduce the cost in cloud data center. In this paper, we formulate the VM placement problem for minimizing the total execution time of the used PMs under both off-line and on-line scenarios. Due to the NP-hardness of this problem, we propose heuristic algorithms to give an efficient method.

The remainder of this paper is organized as follows. First, we make a summary of the related work in Sect. 2. Then we introduce the problem of minimizing the total execution time of the PMs for VM placement in Sect. 3. In Sect. 4, we propose heuristic algorithms for both off-line and on-line scenarios. Experiments and performance evaluation follows in Sect. 5, and Sect. 6 concludes this paper finally.

2 Related Work

Virtual machine placement problem is one of the major issues in cloud data center. An ideal approach can be greatly beneficial to both cloud users and service providers. Recently, a great deal of algorithms and models have been proposed. There are various objectives of these works, which play important roles in the data centers, the VMs and their executions [13]. Due to the features of cloud computing, there have been lots of new models and algorithms with different constraints and objects, for example, availability [7], fairness [17], energy [3] and the communication cost between VMs [4, 11]. In addition, as the result in [5] shows that the physical machines will consume about forty-five percent overall cost in cloud data center, the PM cost is another important objective being considered in the literature, and the goal is to minimizing the number of the used PMs.

Many VM placement schemes consider the VM placement problem as the classical bin packing problem, which is the common way to deal with this problem. In [6], the authors design a new model for saving the power in real-life cloud computing, and they present three different bin packing algorithms with different aspects of the cloud as constraints, it is concluded that simple bin packing algorithms like 2D and 3D bin packing algorithms can reduce power consumption. Due to hardness of bin packing problem, other approach should be applied to solve it with an acceptable complexity, so evolutionary algorithms are very common. In [1], the authors use the modeling of multiple knapsack problem for VM placement problem, they give an algorithm based on ant colony and compare the performance with other solutions. And in [8], a hybrid genetic algorithm using Best Fit Decreasing is designed to deal with infeasible solution due to the bin-used representation, they also conduct experiments to show better performance of their algorithm.

When considering the online scenario, other heuristic methods are proposed. In [18], the authors propose a new energy-aware approach based on the online bin packing algorithm to improve the energy efficiency and resource utilization in cloud data center. In order to deal with the varying resource demands from users, they present an over-provision method. Another online-bin-packing-based research is discussed in [14]. In their work, the VMs, regarded as items in bin packing problem, are divided into four types based on their sizes, and the PMs (called the bins) can be also divided into different types. Their main idea is to keep the gap of most bins within 1/3, and the algorithm can achieve an approximation ratio of 3/2 while the number of movements of the primitive operations, insert and change, are at most 6, which dynamically based on application demands and support green computing. [9] extends the previous one's work and achieve a better approximation ratio of 4/3 with an even finer division of the item types, the proposed approach supports all operations that [14] has constructed but moves at most ten items per operation, which may be a little worse than the 3/2 approximation one. Considering different factors of the cloud environment, there can be different aspects that make VM placement more complex than bin packing. Besides the above works, there are other variants of bin packing problem for modeling VM placement problem [12]. When PM capacities and VM sizes account for different resource types, the problem becomes the vector bin packing problem, when PMs are characterized in different sizes which is motivated by its heterogeneity, the problem becomes the variable sized bin packing, when the VM requests change during the time, the problem becomes the dynamic bin packing problem.

As a result, the goal in most of the work is to minimize the number of the PMs, taking into account the resource demands of the VM, the capacity of the PMs and other factors in cloud computing environment. None of them consider the execution time. In this paper, we aim to reduce the number of PMs and minimize the total execution time of all the PMs.

3 Problem Description

In this section, we study the VM placement problem under the offline scenario, and there are some differences between offline and online VM placement. Under the offline scenario, we know all the information about the requested VMs set a priori, such as the number of VMs and the resource requirement of each VM, while under online scenario, the VM requests are coming one by one without knowing the information beforehand.

Suppose that there are infinite PMs in the cloud system, where the capacity of each PM is C. And n VMs are waiting to be allocated to the PMs. Different from most of the exist works, we put emphasize on not only the total number of PMs but also the duration each PM stay active when calculating the energy consumption by PMs. In this paper, therefore, we aim to find a VM placement that minimize the total cost of the PMs actively used over the whole period.

We first define the notations used in this problem (Table 1).

Table 1. Notations

Notation	Description
V	The set of VMs
n	The number of VMs
v_i	The ith VM in V
R_i	The resource demand of v_i
Su_i	The submission time of v_i
L_i	The running time of v_i
P	The set of PMs
p_i	The ith PM in P
St_i	The start time of p_i
E_i	The execution time of p_i
y_{jk}	p_i is used in time k if $y_{jk} = 1$
x_{ijk}	v_i is allocated on p_i in time k

We study the VM placement problem for minimizing the cost caused by the utilization of PM, we called PM cost. We assume that all PMs are homogeneous with unit capacity and normalize the resource demand of the VM to be a fraction of the capacity. For example, when VM needs 10% of the memory of the host PM, its size can be defined as 0.1.

Obviously, the PM cost is not only proportional to the number of PMs but also the total execution time of all PMs. In this problem, the customers submit their request of a set of VMs, where each VM is define as the resource size and the running time length.

Without loss of generality, we assume that the request from customers and the response measures from cloud system will be completed in every unit time. For simplicity, we set the cost rate of unit as 1, and the unit time is also indicated by 1. The problem seeks to determine a set of PMs that can accommodate the requested VMs with the minimum PM cost. Accordingly, the object of the VM placement problem is to minimize the total execution time of all the PMs. We use E_i to represent the executing time of p_i, and the problem can be formulated as:

$$\text{minimize} \sum\nolimits_{i=1}^{n} \sum\nolimits_{j=1}^{n} \sum\nolimits_{k=\min\limits_{v_i \in V} Su_i}^{\max\limits_{v_i \in V}(Su_i + E_i)} x_{ijk}$$

Subject to:

$$\sum_{i=1}^{n} R_i \cdot x_{ijk} \le y_j, j = \{1, 2, 3\ldots n\}, k = \{St_j, St_j + 1, \ldots St_j + E_j\}$$
$$x_{ijk} \in \{0, 1\}, y_i \in \{0, 1\}$$

This problem is NP-hard because, when the submission time and the running time of each VM are identical, the problem then becomes a variant of the classical one dimensional bin packing problem, which is one of the well-studied NP-hard problem. Note that here we are minimizing the total execution time of the PMs, rather than the number of the PMs in bin packing problem. The result is that our problems are more complicated. Therefore, we prepare to present heuristic algorithms to find an acceptable solution.

4 Algorithm

4.1 Offline Problem Analysis

In this section, we analyze the offline VM placement, which aims to minimize the total execution time of all the PMs. We know all the information about the requested VMs set a priori, including the number of VMs, the resource requirement and the submission time of each VM. Because of the order in submission time, the VMs may not be placed arbitrarily. Thus, we need to sort the VMs increasingly by submission time at the beginning of our algorithm. Then we describe the placement strategy to place the VMs one by one in the time order.

Our basic heuristic idea is based upon the resource utilization rate in a period of time. In order to explain the idea more clearly, we use an example here. We assume that in the time T, there is an active PM p. Let the residual capacity of resource in p be 0.4, and the planned execution time is 4. At the same time two requested VMs v_1 and v_2 have same resource demand of 0.3. Then we have two possible options below according to their running time.

(1) The running time of v_1 is 3, which is smaller than the planned execution time of p, the resource utilization rate is *(0.6 * 4 + 0.3 * 3)/4 = 0.825.*

(2) The running time of v_2 is 5, which is larger than the planned execution time of p, the resource utilization rate is *(0.6 * 4 + 0.3 * 5)/5 = 0.78.*

Thus, under this circumstance we will place v_1 onto p, since that it can achieve a higher resource utilization rate in the coming period of time.

Let *U(p, v)* represents the resource utilization rate when placing v onto p, C represents the residual capacity of resource in p, E represents the planned execution time of p, R represents the resource demand of v, L represents the running time of v. Then the calculation formula of resource utilization rate is given.

$$\mathrm{U(p,v)} = \frac{(1-C)*E+R*\max(E,L)}{\max(E,L)}$$

The resource utilization rate *U(p, v)* can describe the resource usage in the next period of time. Higher rate means that more resources are more likely to be fully utilized. Based on this we propose a heuristic algorithm to reduce the total execution time of all the PMs (Table 2).

Table 2. Offline algorithm.

```
Algorithm 1 Heuristic algorithm for offline VM placement.

Input: V={$v_1$, $v_2$, $v_3$...$v_n$}
Output: P={$p_1$, $p_2$, $p_3$...$p_m$}, x
  1: P'← Ø
  2: Sort V increasingly by Su and then decreasingly by R and L
  3: for i =1 to n do
  4:     if $\exists p_i \in P', C_i \geq R_1$ then
  5:         $p \leftarrow arg_{p_i \in P} max\ U(p_i, v_i)$
  6:         allocate $v_i$ to p
  7:         update P' and x
  7:     else
  8:         open new_p and allocate $v_i$ to new_p
  9:         add new_p to P'
  10:        update P' and x
  11:    end if
  12: end for
  13: return P'and x
```

4.2 Online Problem Analysis

In this section, we present a greedy strategy for VM placement under the online scenario. In this problem, the requests of VMs are submitting one by one. Hence, we should place VM one after another according to the current usage of the PM. Similar to the dynamic bin packing problem, we first study two commonly used packing algorithms, First Fit and Best Fit. Their main idea will be described below.

Each time when there is a new VM request submitted, First Fit seeks to allocates it to earliest opened PM that can accommodate it, while Best Fit seeks to allocate it to the suitable PM with the minimum residual capacity. And if all of the opened PMs are available for the new requested VM, the cloud will open a new PM to accept it. When there are no VMs running, the PM can be switched to a low power mode or shut down.

It should be noted that under the online scenario, the opened PM may have a planned execution time obtained by the allocated VMs. Our greedy algorithm is learned from the Best Fit algorithm, and the basic idea is discussed then. When a new requested VM submits, suppose that there are a set of opened PMs with the information of remaining resources *R* and planned execution time *E*. The greedy algorithm helps to develop the strategy to allocate the VM. We have three cases to analyze: (1) There are opened PMs that can accommodate the VM and the planned execution time of the PMs is longer than its running time, then we allocate the VM to the suitable PM that with the minimum residual capacity. (2) There are opened PMs that can accommodate the VM but none of them has a longer execution time than the VM's running time, then we allocate the VM to the PM with a longest planned execution time, if there are multiple eligible PMs, choose the one with the minimum residual capacity. (3) Otherwise, open a new PM and put the VM in it.

Table 3. Online algorithm.

```
Algorithm 2 Greedy strategy for online VM placement.
Input: v = (R, L): The new requested VM.
        $P=\{p_1, p_2, p_3 ... p_m\}$: The set of opened PMs.
Output: p: A suitable PM to accommodate v.
1: Find the VM $v_i$ with the maximum residual capacity.
2: if $R > C_i$ then
3:     open $p_{m+1}$ and allocate v to $p_{m+1}$
4:     return $p_{m+1}$
5: else
6:     $P' = arg_{p_i \in P}\, min\ Inc(v, p_i)$
7:     $p = arg_{p_i \in P'}\, min\ C_i$
8: end if
9: return p
```

According to above analysis, we let *Inc (p, v)* represents the cost increment when VM v is allocated to PM *p*, where the resource demand is *R*, running time is *L* and planned execution time of *p* is *E*. In case (1) and (2), *Inc (p, v)* is 0 and *L-E*, while in case (3) it is undefined. Based on this strategy we design a greedy algorithm and the details can be described as following (Table 3):

5 Experiments

In this section, simulation experiments are conducted to evaluate the performance of the proposed algorithms under offline and online scenario. And we evaluate the performance of the algorithms by comparing the results to other common used bin packing algorithms.

5.1 Simulation Settings

The experiments are conducted on Intel(R) Core(TM) i7 processor @ 12.0 GB using C++. Originally, we generate several problem instances with different sizes of requested VMs. We also conduct groups of experiments for VM placement under offline and online scenario respectively. In our simulations, the size of the VMs range from 0 to 1, the submission time range from 1 to 10, and the running time range from 1 to 20. Our goal is to find a VM placement in this period to reduce the total execution time of all the PMs.

For the purpose of comparison, we also implement the common bin packing algorithms, such as FF (First Fit), BF (best fit), FFD (First Fit Decreasing), BFD (Best Fit Decreasing). And there are two evaluation standards for the algorithms in this problem. One is the number of used PMs, and the other is the total execution time.

5.2 Simulation Results

The simulations are divided into three case studies. In Case 1 and Case 2, we evaluate the performance of the VM placement algorithms under offline and online scenario respectively. We compare the offline placement algorithm with FFD and BFD, while the online placement algorithm with FF and BF. And the comparison sets of simulations are conduct in Case 3.

Case 1: Offline Placement Algorithm
In the first case, we conduct four groups of simulations, in which the default number of VMs are 200, 400, 600, and 800 respectively. We compare our proposed heuristic algorithm (PHA) with FFD and BFD by the two standards.

Figure 2 shows the number of PMs that the three different algorithms produce respectively under the offline scenario. The x-coordinate is the number of the requested VMs, while the y-coordinate indicates the number of PMs used to host the VMs. We can see that in the same case, BFD has fewer PMs than FFD, while our algorithm has the minimum PMs.

Figure 3 shows the total execution time of all the used PMs that the three different algorithms produce respectively under the offline scenario. The x-coordinate is the total execution time, besides, Fig. 3 has the same format as Fig. 2 As the result, BFD and FFD produce more execution time than our proposed algorithm.

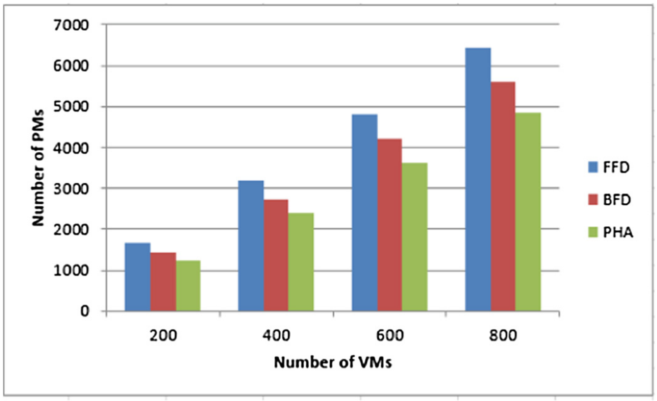


Fig. 2. Number of PMs under offline scenario

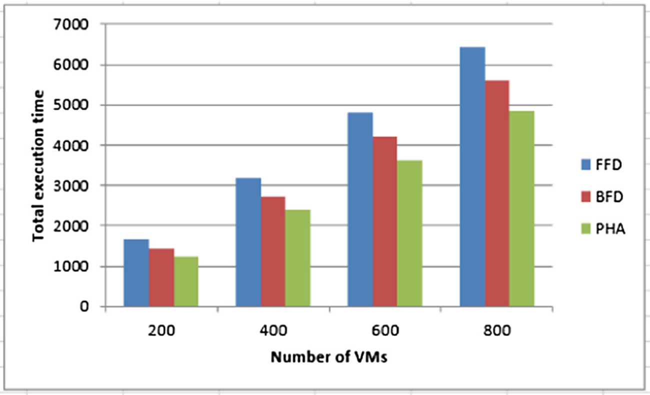


Fig. 3. Total execution time of PMs under offline scenario

According to the results, we have the observation that our proposed heuristic algorithm has a better performance than both BFD and FFD. This is because a request VM is more likely to be placed on the PM with less resource left while taking the running time into account, which enhances the resource utilization rate during the placement period.

Case 2: Online Placement Algorithm

In the second case, we evaluate our proposed algorithm under the online scenario, which is also a modified best fit algorithm (MBF). We compare MBF with FF and BF by the two standards. We use the same problem instances in case 1 to evaluate the performance of the online placement algorithm.

Figure 4 has the same format as Fig. 2 It shows the number of PMs that the three different algorithms produce respectively under the online scenario. We can see that our proposed algorithm always has the smallest number of PMs in the four instances, and FF has a bit more PMs than BF.

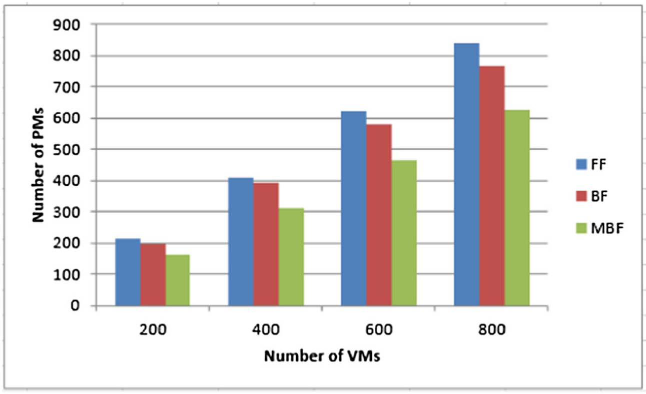


Fig. 4. Number of PMs under online scenario.

Figure 5 has the same format as Fig. 3. It shows that BF and FF produce more execution time than our proposed algorithm under the online scenario, and FF always produce the most execution time.

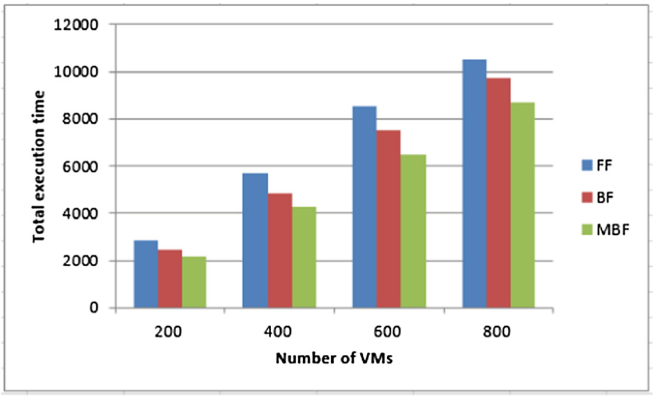


Fig. 5. Total execution time of the PMs under online scenario.

According the results, we can make a conclusion that our proposed heuristic algorithm has a better performance than both BF and FF under the online scenario. This is because when a new requested VM is coming, our placement strategy tries to choose a PM to place on with the minimum execution time increasing.

Case 3: Analysis of Offline and Online Algorithm
Finally, in case 3, we conduct the experiments to compare both of our proposed algorithms. Figures 6 and 7 shows the number and the total execution time of the PMs of our proposed offline and online algorithms. The online algorithm has more (more than about 1.8 times) PMs and execution time than the offline algorithm. Furthermore, the online algorithm brings more PMs and execution time than the offline algorithm as the number of requested VMs increasing. This is due to the characteristic of the problem that we know all the information about the requested VMs under the offline scenario, while the online algorithm only gives the only coming VM.

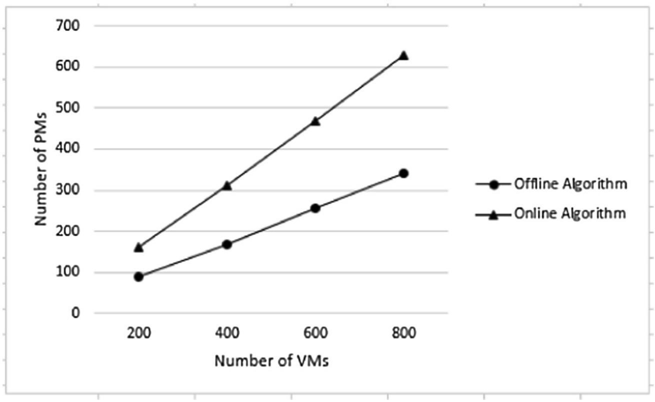


Fig. 6. Number of PMs.

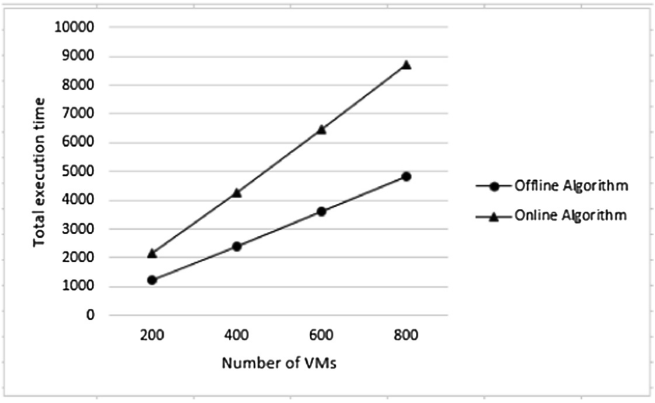


Fig. 7. Total execution time of the PMs

6 Conclusion

In this paper, we study the VM placement problem for minimizing the cost caused by the PMs in cloud data center. We aim to reduce the total execution time of the used PMs. We formulate a new optimization problem for deciding the placement of VMs in cloud data center. Moreover, we propose heuristic algorithms for both off-line and on-line scenarios to solve the problem. To evaluate the performance, simulation experiments are conducted to observe the placement of VMs. In the future, we plan to further optimize the performance of our algorithm, and give some theoretical proof.

Acknowledgement. This work is supported by Research Initiative Grant of Sun Yat-sen University under Project 985 and Australian Research Council Discovery Project DP150104871.

References

1. Amarante, S.R.M., Roberto, F.M., Cardoso, A.R., Celestino, J.: Using the multiple knapsack problem to model the problem of virtual machine allocation in cloud computing. In: 2013 IEEE 16th International Conference on Computational Science and Engineering (CSE), pp. 476–483. IEEE (2013)
2. Anderson, T., Peterson, L., Shenker, S., Turner, J.: Overcoming the internet impasse through virtualization. Computer **38**(4), 34–41 (2005)
3. Dong, J., Jin, X., Wang, H., Li, Y., Zhang, P., Cheng, S.: Energy-saving virtual machine placement in cloud data centers. In: 2013 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), pp. 618–624. IEEE (2013)
4. Fukunaga, T., Hirahara, S., Yoshikawa, H.: Virtual machine placement for minimizing connection cost in data center networks. In: 2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 486–491. IEEE (2015)
5. Greenberg, A., Hamilton, J., Maltz, D.A., Patel, P.: The cost of a cloud: research problems in data center networks. ACM SIGCOMM Comput. Commun. Rev. **39**(1), 68–73 (2008)
6. Hage, T., Begnum, K., Yazidi, A.: Saving the planet with bin packing-experiences using 2D and 3D bin packing of virtual machines for greener clouds. In: 2014 IEEE 6th International Conference on Cloud Computing Technology and Science (CloudCom), pp. 240–245. IEEE (2014)
7. Jayasinghe, D., Pu, C., Eilam, T., Steinder, M., Whally, I., Snible, E.: Improving performance and availability of services hosted on IaaS clouds with structural constraint-aware virtual machine placement. In: 2011 IEEE International Conference on Services Computing (SCC), pp. 72–79. IEEE (2011)
8. Kaaouache, M.A., Bouamama, S.: Solving bin packing problem with a hybrid genetic algorithm for VM placement in cloud. Procedia Comput. Sci. **60**, 1061–1069 (2015)
9. Kamali, S.: Efficient bin packing algorithms for resource provisioning in the cloud. In: Karydis, I., Sioutas, S., Triantafillou, P., Tsoumakos, D. (eds.) ALGOCLOUD 2015. LNCS, vol. 9511, pp. 84–98. Springer, Cham (2016). doi:10.1007/978-3-319-29919-8_7
10. Li, X., Qian, Z., Sanglu, L., Jie, W.: Energy efficient virtual machine placement algorithm with balanced and improved resource utilization in a data center. Math. Comput. Model. **58** (5), 1222–1235 (2013)

11. Li, X., Wu, J., Tang, S., Lu, S.: Let's stay together: towards traffic aware virtual machine placement in data centers. In 2014 Proceedings IEEE INFOCOM, pp. 1842–1850. IEEE (2014)
12. Mann, Z.Á.: Approximability of virtual machine allocation: much harder than bin packing (2015)
13. Masdari, M., Nabavi, S.S., Ahmadi, V.: An overview of virtual machine placement schemes in cloud computing. J. Netw. Comput. Appl. **66**, 106–127 (2016)
14. Song, W., Xiao, Z., Chen, Q., Luo, H.: Adaptive resource provisioning for the cloud using online bin packing. IEEE Trans. Comput. **63**(11), 2647–2660 (2014)
15. Verma, A., Ahuja, P., Neogi, A.: pMapper: power and migration cost aware application placement in virtualized systems. In: Issarny, V., Schantz, R. (eds.) Middleware 2008. LNCS, vol. 5346, pp. 243–264. Springer, Heidelberg (2008). doi:10.1007/978-3-540-89856-6_13
16. Wang, G., Ng, T.E.: The impact of virtualization on network performance of amazon EC2 data center. In: 2010 Proceedings IEEE INFOCOM, pp. 1–9. IEEE (2010)
17. Wang, W., Li, B., Liang, B.: Dominant resource fairness in cloud computing systems with heterogeneous servers. In: 2014 Proceedings IEEE INFOCOM, pp. 583–591. IEEE (2014)
18. Wang, X., Liu, Z.: An energy-aware VMs placement algorithm in cloud computing environment. In: 2012 Second International Conference on Intelligent System Design and Engineering Application (ISDEA), pp. 627–630. IEEE (2012)