Copyright © 2018 Tech Science Press CMC, vol.56, no.1, pp.123-135, 2018

Machine Learning Based Resource Allocation of Cloud Computing in Auction

Jixian Zhang[1], Ning Xie[1], Xuejie Zhang[1], Kun Yue[1], Weidong Li[2, *] and Deepesh Kumar[3]

Abstract: Resource allocation in auctions is a challenging problem for cloud computing. However, the resource allocation problem is NP-hard and cannot be solved in polynomial time. The existing studies mainly use approximate algorithms such as PTAS or heuristic algorithms to determine a feasible solution; however, these algorithms have the disadvantages of low computational efficiency or low allocate accuracy. In this paper, we use the classification of machine learning to model and analyze the multi-dimensional cloud resource allocation problem and propose two resource allocation prediction algorithms based on linear and logistic regressions. By learning a small-scale training set, the prediction model can guarantee that the social welfare, allocation accuracy, and resource utilization in the feasible solution are very close to those of the optimal allocation solution. The experimental results show that the proposed scheme has good effect on resource allocation in cloud computing.

Keywords: Cloud computing, resource allocation, machine learning, linear regression, logistic regression.

1 Introduction

Auction-based resource allocation can effectively improve the social welfare and resource utilization of resource providers [Nisan, Roughgarden, Tardos et al. (2007)], which is an important research direction in cloud computing. A single-dimensional resource allocation problem has been proven to be NP-hard [Kellerer, Pferschy and Pisinger (2004)], so it cannot be solved in polynomial time. In multi-dimensional resource allocation problems, different types of resources (e.g. CPU, memory, and storage) must be considered. Thus, multi-dimensional resource allocation problems are strongly NP-hard. The resource allocation problem can be solved by using an optimal algorithm such as integer programming and dynamic programming, or it can be solved by an approximate algorithm such as PTAS or heuristic algorithms. Because the problem of resource allocation is NP-hard, the optimal algorithm can only be used in small-scale data sets; on the other hand, the approximate algorithm or the heuristic algorithm will cause the problems of low

[1] School of Information Science and Engineering, Yunnan University, Kunming 650504, PR China.

[2] School of Mathematics and Statistics, Yunnan University, Kunming 650504, PR China.

[3] Computer Science & Engineering SSITM, Aktu, Lucknow 226014, India.

[*] Corresponding Author: Weidong Li. Email: weidong@ynu.edu.cn.

computational efficiency and low allocation accuracy compared with the optimal solution.

1.1 Our contribution

By analyzing the optimal allocation solution of small-scale data sets, we speculate that there is a potential model in optimal resource allocation that can be represented by machine learning algorithms. Thus, we can allocate resources in a way that is very close to the optimal solution. In this paper, we transform the problem of multi-dimensional cloud resource allocation in an auction into machine learning classification or regression problems and propose two resource allocation prediction algorithms based on linear and logistic regressions. The main idea is solve the optimal allocation solution for a user's requirements in small-scale training sets and then use machine learning algorithms to fit the optimal allocation. The final predict model can select winner users and guarantee social welfare, allocation accuracy, resource utilization results that are very close to the optimal solution allocation.

1.2 Related works

The Resource Allocation Problem (RAP) in an auction is an optimization problem [Darmann, Pferschy and Schauer (2010)] in which m users submit their resources requirements and provide the corresponding value; the problem is how to maximize the social welfare and revenue for resource providers without exceeding the total amount of resources. Resource allocation can be equivalent to the knapsack problem, which is NP-hard. There are many ways to solve for the exact optimal solution of resource allocation problems. Nejad et al. [Nejad, Mashayekhy and Grosu (2015)] used integer programming to solve the resource allocation problem. Mashayekhy et al. [Mashayekhy, Fisher and Grosu (2016)] used a dynamic programming method to solve the same problem. Wu et al. [Wu and Hao (2016)] transformed a combinatorial resource auction into a winner decision problem (WDP) and used a clique-based exact method to obtain the exact solution. Lai et al. [Lai and Parkes (2012)] obtained the optimal solution by utilizing a monotone branch-and-bound search method. When the problem scale is small, these methods can be used to solve the optimal solution, but the required computational time increases exponentially with an increasing number of users, resource types, and user requirements. Thus, a more efficient algorithm is needed for practical use. There are many methods for solving the approximate solution of resource allocation, and they can be mainly divided into Polynomial Time Approximation Scheme (PTAS) algorithms and heuristic algorithms. Liu et al. [Liu, Li and Zhang (2017)] proposed an n-approximation mechanism to solve the problem of heterogeneous physical machines resource management. Mashayekhy et al. [Mashayekhy, Nejad and Grosu (2015)] proposed a PTAS algorithm to solve the multitask scheduling problem. Shi et al. [Shi, Zhang, Wu et al. (2016)] transformed the online auction resource allocation problem into a continuous static resource allocation problem for a period of time; they used the primal-dual algorithm to obtain the approximate resource allocation solution. In terms of heuristic algorithms, Zaman et al. [Zaman and Grosu (2011); Sharrukh and Daniel (2013)] proposed the CA-GREEDY and CAPROVISION algorithms, Nejad et al. [Nejad, Mashayekhy and Grosu (2015); Mashayekhy, Nejad and Grosu (2015)] proposed the G-

VMPAC-X algorithm, and Zhang et al. [Zhang, Xie, Li et al. (2018)] proposed the VRAP_A algorithm, which is based on greedy theory, to solve the allocation problem. Lin et al. [Lin, Xu, He et al. (2017); Lin, Zhu, Li et al. (2015); Lin, Xu, Li et al. (2017); Liu, Li and Zhang (2017)] have extended CloudSim with a multi-resource scheduling and power consumption model, also proposed a VM placement algorithm based on the peak workload characteristics, which models the workload characteristics of VMs with mathematical method, and measures the similarity of VMs' workload with VM peak similarity. Wang et al. [Wang, Li and Li (2017)] present a new partition scheduling algorithm called Heterogeneous Multiprocessor Partition (HMP) based on the prefetching technique for heterogeneous multicore processors, which can hide memory latencies for applications with multi-dimensional loops. Generally, integer or dynamic programming is used to solve the exact solution or optimal solution of the allocation problem, but the problem of resource allocation is NP-hard and, thus, cannot be solved in polynomial time. The PTAS algorithm time complexity is $O(n^{1/\varepsilon})$, which often causes the execution time of the algorithm to increase rapidly with a decrease in ε. Heuristic algorithms such as greedy algorithms cannot achieve good allocation accuracy and resource utilization compared with the optimal solution. On the other hand, machine learning techniques are used in many fields to solve classification and prediction problems. Auction-based resource allocation will produce winners and losers, so it can be seen as a machine learning classification or regression problem; however, there are very few prior results.

In summary, for the resource allocation of cloud computing in an auction, the current research has achieved positive results; however, it still faces enormous challenges in terms of solving the multi-dimensional resource allocation accuracy and algorithm performance.

1.3 Organization

The remainder of this paper is organized as follows. In Section 2, we discuss the resource allocation mathematical programming model. In Section 3, we propose two resource allocation algorithms, Linear-ALLOC and Logistic-ALLOC, based on machine learning and theoretical analysis. In Section 4, we present the experimental results and performance analysis. In Section 5, we summarize our results and discuss possible directions of future work.

2 Resource allocation problem in an auction

At present, most cloud computing corporations provide virtual machines to the user, such as Amazon EC2, which can classify virtual machines as general, computationally intensive or storage-intensive. Amazon provides the CPU, memory, and SSD storage for each type of virtual machine. This method makes it more convenient for the user to select a suitable virtual machine, but from the perspective of resource allocation, the requested virtual machine must be represented as the number of various resources such as CPU, memory and storage.

Resource model: We denote a cloud provider offering *n* types of resources, such as CPU, memory, and storage. The capacity of each resource is represented by the vector $\mathbf{C} = (c_1\ c_2\ \dots c_n)$. The unit cost for each type of resource is defined by the vector

$\mathbf{V} = (v_1\, v_2 \dots v_n)$; for example, v_1 means 1 CPU core cost per hour.

User information model: We assume that there are *m* users in set $U = \{1, 2, ..., m\}$, User $i \in U$. User *i* can submit his requirements, which are denoted by the vector $\mathbf{k}^{(i)} = (k_1^{(i)}, k_2^{(i)} ..., k_n^{(i)})$, where $k_r^{(i)}$ $r = (1, 2, ..., n)$, represents the amount of the *r*-th resource that is requested by user *i*. User *i* values a bid b_i for his requirement $\mathbf{k}^{(i)}$. The final submission information of user *i* is represented by vector $\mathbf{R}^{(i)} = (\mathbf{k}^{(i)}, b^{(i)})$. For example, user *I*'s requirement is 2 CPU cores, 4 GB of memory, and 50 GB of storage, and the user is willing to pay 9. We define $\mathbf{R}^{(1)} = (\mathbf{k}^{(1)}, b^{(1)})$, $\mathbf{k}^{(1)} = (2, 4, 50)$, $b^{(1)} = 9$.

In the auction mechanism, the resource provider is pursuing the maximization of social welfare. Each user's valuation $b^{(i)}$ can be considered a part of social welfare. We denote the total social welfare as *V* and formulate the problem of resource allocation in cloud computing as an integer program as follows:

$$V = \max\left(\sum_{i \in U} (b^{(i)} - cp^{(i)}) \cdot x^{(i)} \right) \tag{1}$$

$$\text{s.t.: } \sum_{i \in U} k_r^{(i)} \cdot x^{(i)} \le c_r, \forall r = 1, 2, ..., n \tag{1a}$$

$$x^{(i)} \in \{0, 1\}, \forall i \in U \tag{1b}$$

where *V* represents the total social welfare and $cp^{(i)} = \sum_{r=1}^{n} k_r^{(i)} v_r$ is the cost of user *I*'s requirement. Eq. (1a) indicates that the resource allocation cannot exceed the capacity of any type of resource. Eq. (1b) indicates that the satisfaction of user *I*'s requirement is represented by $x^{(i)} = 1$ and is $x^{(i)} = 0$ otherwise. The above problems can be transformed into a multidimensional knapsack problem (MKP) problem, which is NP-hard.

We assume that $\mathbf{R} = \{\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, ..., \mathbf{R}^{(m)}\}$ is the set of requirements of all users and that $\mathbf{R}^{(-i)} = \{\mathbf{R}^{(1)}, \mathbf{R}^{(2)}, ..., \mathbf{R}^{(i-1)}, \mathbf{R}^{(i+1)}, ..., \mathbf{R}^{(m)}\}$ is the set of requirements of all users except user *i*.

When the user requirement data set is not large, we can use integer programming or dynamic programming to solve the optimal allocation solution $A(\mathbf{R})$ and $A(\mathbf{R}^{(-i)})$, and then use the VCG algorithm to solve for the optimal payment price as follows:

$$p^{(i)} = \sum_{j \in A(\mathbf{R}^{(-i)})} b^{(j)} - \sum_{j \in A(\mathbf{R}), j \ne i} b^{(j)} \tag{2}$$

$\sum_{j \in A(\mathbf{R}^{(-i)})} b^{(j)}$ is the optimal social welfare when user *i* does not participate, and $\sum_{j \in A(\mathbf{R}), j \ne i} b^{(j)}$ is the optimal social welfare, except for user *I*'s bid. $p^{(i)}$ is user *I*'s final payment price.

3 Multi-dimensional cloud resource allocation algorithm based on machine learning

We use machine learning regression and classification to design multi-dimensional cloud resource allocation algorithms in auctions. The main idea is to select part of requirements from all users, solve for the optimal allocation solution and optimal payment price, use linear regression and logistic regression to fit the optimal allocation solution, and finally apply the learned model to predict all user requirements. In this paper, we propose two kinds of machine learning algorithms to fit the optimal allocation solution. They are the linear regression-based resource allocation prediction algorithm (Linear-ALLOC) and the logistic regression-based resource allocation prediction algorithm (Logistic-ALLOC).

3.1 Linear regression-based resource allocation prediction algorithm (Linear-ALLOC)

In the auction, the user submits various resource requirements $\mathbf{k}^{(i)}$ and the corresponding bids $b^{(i)}$. We consider the user requirements for different types of resources as the features of hypothesis function. Define the learning parameters $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_{n+1})$, and the hypothesis function is as follows:

$$h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}) = \theta_0 + \theta_1 k_1^{(i)} + \theta_2 k_2^{(i)} + ... + \theta_n k_n^{(i)} \tag{3}$$

For ease of understanding, we can think of the $\theta_1, \theta_2, ..., \theta_n$ as the potential unit value of *n* types of resources, θ_0 is the prediction noise, and $h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)})$ can be understood as a price expression of the resources required by user *i*. Our goal is to determine the winning rules of user's, which are $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_n)$.

Assuming that there are *m* users submitting requirements, because of the limited resources, the number of winning users is definitely smaller or equal than *m*. We can use IBM CPLEX to determine the optimal allocation solution and the VCG mechanism to obtain the optimal payment price of each winning user. $p^{(i)}$ represents the final payment price of user *i*. If user *i* wins, $p^{(i)}$ is greater than 0; otherwise, it is equal to 0. According to all user requirements and optimal solutions, we define the user requirement matrix $\mathbf{K} = [\mathbf{k}^{(1)}, \mathbf{k}^{(2)}, ..., \mathbf{k}^{(m)}]^T$, the optimal allocation vector $\mathbf{X} = (x^{(1)}, x^{(2)}, ..., x^{(m)})^T$, the user bid vector $\mathbf{B} = (b^{(1)}, b^{(2)}, ..., b^{(m)})^T$, and the optimal payment price vector $\mathbf{P} = (p^{(1)}, p^{(2)}, ..., p^{(m)})^T$; the cost function is defined as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{2m} \sum_{i=0}^{m} x^{(i)} (h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}) - p^{(i)})^2 \tag{4}$$

With the goal of minimizing $J(\boldsymbol{\theta})$, solve $\boldsymbol{\theta}$ to determine the function representation of $h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)})$. $J(\boldsymbol{\theta})$ is a quadratic function of the parameter vector $\boldsymbol{\theta}$, and there is a theoretically global optimal solution. We can use the normal equation method or gradient descent method to solve it. Because we have fewer features of the hypothesis function, it is more suitable to use the normal equation. Another advantage of the normal equation is that we do not need to normalize the features. The normal equation for solving $\boldsymbol{\theta}$ is as follows:

$$\boldsymbol{\theta} = (\mathbf{K}^T \mathbf{K})^{-1} \mathbf{K}^T \mathbf{P} \tag{5}$$

The significance of solving minimizing $J(\boldsymbol{\theta})$ is that we need to find the most reasonable representation that can reflect the potential value of various resources in the current auction. Actually, the relationship between the user requirements and payment prices is not strictly linear. We find that in optimal allocation solutions, if a user requests a large amount of resources, the optimal payment price is lower than the same proportion of small-scale resource requests. This indicates that resource providers have appropriate preferences for user requests with large amounts of resources, result in the payment prices reduced. So we use a more reasonable hypotheses function $h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)})$ to adapt to this situation, and the $\boldsymbol{\theta}=(\theta_0,\theta_1,...,\theta_{2n})\in\mathbb{R}^{2n+1}$. By adding the 1/2 power term of $\mathbf{k}^{(i)}$, $h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)})$ changes from a linear function to a concave function, which realizes a preference for a large number of resource requirements. Thus, the prediction model will be more accurate. However, an increase in features will also bring an overfitting problem to the learning model; we need to regularize the cost function as follows:

$$J(\boldsymbol{\theta})=\frac{1}{2m}[\sum_{i=0}^{m}x^{(i)}(h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)})-p^{(i)})^2+\lambda\sum_{j=1}^{n}\theta_j^2] \tag{6}$$

The normal equation for solving $\boldsymbol{\theta}$ as follows:

$$\boldsymbol{\theta}=(\mathbf{K}^T\mathbf{K}-\lambda\mathbf{L})^{-1}\mathbf{K}^T\mathbf{P},\ \mathbf{L}=\begin{bmatrix}0 & & & \\ & 1 & & \\ & & \cdots & \\ & & & 1\end{bmatrix} \tag{7}$$

Once $\boldsymbol{\theta}$ is solved, for the new user requirement $\mathbf{R}^{(j)}=(\mathbf{k}^{(j)},b^{(j)})$, we use the sigmoid function to predict whether it is selected as follows:

$$predict^{(j)}=\frac{1}{1+e^{-(b^{(j)}-h_\theta(k^{(j)}))}},\ predict^{(j)}\in(0,1) \tag{8}$$

According to the machine learning classification, $predict^{(j)}$ can be understood as the probability that user *j* is a winning user when his bid is $b^{(j)}$, and $b^{(j)}-h_{\boldsymbol{\theta}}(\mathbf{k}^{(j)})$ indicates the relationship between the user bid and the potential price of the required resource. It can be seen that if $b^{(j)}-h_{\boldsymbol{\theta}}(\mathbf{k}^{(j)})>0$, that is, $predict^{(j)}>=0.5$, user *j* may be allocated the resource.

3.2 Logistic regression-based resource allocation prediction algorithm (Logistic-ALLOC)

Unlike Linear-ALLOC, Logistic-ALLOC transforms the resource allocation problem into a two classification problem and uses logistic regression to train the model.

We first use CPLEX to find the optimal allocation solution, mark the winning user ($x^{(i)}=1$) and failed user ($x^{(i)}=0$) as positive and negative samples, respectively, and then learn a decision boundary to separate the user's to the greatest extent possible. It can be assumed that the existence of a hyperplane in high-dimensional space can separate the winning user from others.

In Logistic-ALLOC, the user's final payment price $p^{(i)}$ need not be calculated. This avoids the time-consuming problem of the VCG algorithm. However, the user's bid $b^{(i)}$ will be an important feature that has a great influence on the decision boundary, so we need to enhance the role of user's bid in the hypothesis function $h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)})$, where $\boldsymbol{\theta} = (\theta_0, \theta_1, ..., \theta_{n+1})$.

$$f_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}) = \theta_0 + \theta_1 k_1^{(i)} + \theta_2 k_2^{(i)} + ... + \theta_n k_n^{(i)} + \theta_{n+1} (b^{(i)})^2$$

$$g(z) = \frac{1}{1+e^{-z}} \tag{9}$$

$$h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}) = g(f_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}))$$

The cost function of Logistic-ALLOC as follows:

$$J(\boldsymbol{\theta}) = \frac{1}{m} \sum_{i=1}^{m} [-x^{(i)} \log(h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)})) - (1 - x^{(i)}) \log(1 - h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}))] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2 \tag{10}$$

In Logistic-ALLOC, the normal equation cannot be used to minimize the cost function. Therefore, the features need to be normalized to ensure that the range of each feature is not overly different. Finally, gradient descent is used to minimize the cost function, where the partial derivatives of the parameters used in the gradient descent method are as follows:

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}) - x^{(i)}) k_j^{(i)} \quad \text{for } j = 0$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = (\frac{1}{m} \sum_{i=1}^{m} ((h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}) - x^{(i)})\, k_j^{(i)}) + \frac{\lambda}{m} \theta_j \quad \text{for } j = 1, 2, ..., n \tag{11}$$

$$\frac{\partial J(\boldsymbol{\theta})}{\partial \theta_j} = (\frac{1}{m} \sum_{i=1}^{m} ((h_{\boldsymbol{\theta}}(\mathbf{k}^{(i)}, b^{(i)}) - x^{(i)})\, b^{(i)}) + \frac{\lambda}{m} \theta_j \quad \text{for } j = n+1$$

Here, we need to try a different normalization parameter λ and different numbers of gradient descent iterations to achieve the best effect. Once $\boldsymbol{\theta}$ is solved, for the new user requirement $\mathbf{R}^{(j)} = (\mathbf{k}^{(j)}, b^{(j)})$, we use the following sigmoid function to predict whether it is selected:

$$predict^{(j)} = \frac{1}{1+e^{-(h_{\boldsymbol{\theta}}(\mathbf{k}^{(j)}, b^{(j)}))}}, \ predict^{(j)} \in (0,1) \tag{12}$$

Note that the Logistic-ALLOC prediction function is slightly different from the Linear-ALLOC prediction function.

3.3 Apply the algorithm to machine learning

3.3.1 Sample set processing

There are certain similarities and differences between machine learning-based resource allocation and traditional machine learning classification problems. A similarity is that the data set must be classified and predicted. A difference is that the traditional machine learning classification is objective and does not change. For example, a sample is marked positive in training set 1, and if this sample is placed in training set 2, it is still positive. However, the classification of user requirements in resource allocation is uncertain. For example, user *i* is the winning user in training set 1, but if user *i* is placed in training set 2, it may be fail to be allocated. Thus, if we only use one training set to train the prediction model, it may not be able to reflect the real situation. For this problem, we have made the following improvements. Assuming a single-round auction, there are *m* users who submit their requirements and corresponding bids; we first obtain the *resource density* $d^{(i)}$ for each user as follows:

$$d^{(i)} = \frac{b^{(i)}}{\sqrt{\sum_{r=1}^{n}(\frac{1}{c_r} \cdot k_r^{(i)})}}, \forall i = 1, 2, ..., m \tag{13}$$

We sorted all users' requirements in descending order to sample space *M* according to the *resource density* $d^{(i)}$ and set the proportion $\alpha \in (0,1)$. Every $\alpha \cdot m$ samples were selected as a sample set; that is, there are a total of $1/\alpha$ sample sets. The select rule is to build a sample set in the way of user position in sample space *M* modulo $1/\alpha$. For example, there are 1000 user requirements; we first sort the requirements according to *resource density* and then select 100 user requirements as a sample set. For a total of 10 sample sets, sample set 1 is $\mathbf{R}^{(1)} = [R^{(1)}, R^{(11)}, ..., R^{(991)}]$, sample set 2 is $\mathbf{R}^{(2)} = [R^{(2)}, R^{(12)}, ..., R^{(992)}]$, and so on. Select 80% of the sample set as the training set, and 20% as the cross-validation set; each training set will output a prediction model that will be brought into each cross-validation set for verification. Finally, the best-performing prediction model is selected. Fig. 1 shows the learning process of the algorithm.

3.3.2 Feature scaling

In the data set, different type of resources has different data ranges, which affects the learning performance. So it is necessary to normalize the features. We use 1 core CPU as a unit, 1 GB memory as a unit and 50 GB storage as a unit, so the range of each type of feature will not be much different, the CPU request range is $[1,128]$, The memory requests range is $[1,64]$, and the storage request range is $[1,200]$. Here we do not use mean normalization to deal with features, because if using mean normalization, the resources will have a negative value, which is not conducive to the visual representation.

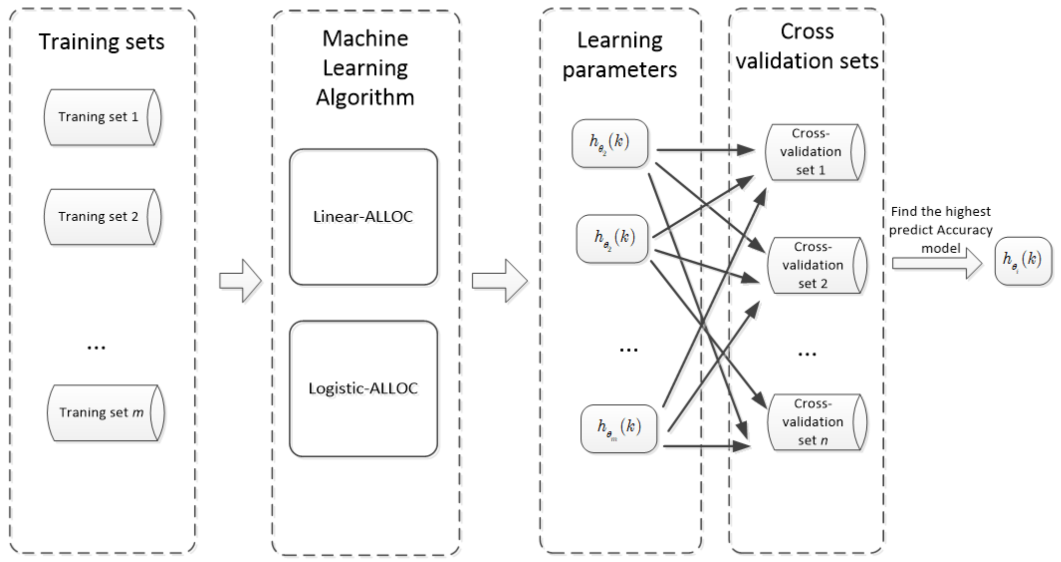

Figure 1: Learning process of the algorithm

3.3.3 Allocation terminate decision

The Linear-ALLOC or Logistic-ALLOC output the prediction value that represents the user's winning probability. We select the winning user according to the predicted value sorted in descending order; at the same time, the requirements of different resources are accumulated. When the cumulative amount of any type of resources exceeds the capacity, the resource allocation algorithm will be terminated, and all users selected before termination are considered winning users and can be allocated resources.

3.3.4 Prediction accuracy and error rate

Finally, we predict whether a user is allocated resources using $predict^{(i)}=1$ or 0, and the prediction accuracy is defined as follows:

$$\begin{aligned} &predict^{(i)}=1, \text{ where } predict^{(i)} \geq \varepsilon_{last} \\ &predict^{(i)}=0, \text{ where } predict^{(i)} < \varepsilon_{last} \end{aligned} \tag{14}$$

$$predict\ accuracy = \frac{\sum_{i=1}^{m}(predict^{(i)}=x^{(i)})}{m}$$

ε_{last} is the predicted value of the last allocated user, and $predict^{(i)}$ represents the probability that the user is selected in the Linear-ALLOC or Logistic-ALLOC algorithm. The predict error rate is 1 minus the prediction accuracy.

4 Experimental results

We use the DAS-2 [ASCI (2017)] from Grid Workloads as test data to simulate user requirements. DAS-2 is provided by the Advanced School for Computing and Imaging (ASCI). The DAS-2 data set contains user job IDs and the corresponding resource requirements. To ensure reasonable simulation data, we remove the jobs that have zero value for the CPUs, memory and storage requirements from the data set. The experimental platform hardware is configured as follows: Intel Core I7 6500U CPU, 8 GB of memory, and 1 TB of storage.

4.1 Experimental setup

We selected 5,000 user request records in the DAS-2 data set as the total sample set and divided them into 8 training sets and 2 cross-validation sets, as described in 3.3. Each set contains 500 user request records. It is worth noting that the algorithm can only be used for single round auctions because different numbers of users participating in the auction will produce different allocation results. We set total resource capacity to (CPU: 5000 core, Memory: 10000 G, Storage: 20000 G), which is scaled down to (CPU: 500 core, Memory: 1000 G, Storage: 2000 G) in the training and cross-validation sets. Details of the experimental settings are as follows:

(1) In each job, we use the CPU, memory, and storage requests to simulate the user requirements.

(2) We randomly generate a value from 1 to 100 to simulate the user's bid $b^{(i)}$ and pre-set the capacity of various types of resources in C and the unit price of various types resources in V.

(3) We use IBM CPLEX 12 to program the optimal allocation algorithm.

(4) We use C++ programming language to program the optimal payment algorithm based on VCG.

(5) We use GNU Octave 4.2.1 to program the Linear-ALLOC, Logistic-ALLOC and G-VMPAC-II-ALLOC algorithms.

4.2 Resource allocation prediction experiment

After obtaining the optimal prediction models for the two algorithms, we randomly generated four test sets that included 1000, 2000, 3000, or 5000 user requirements and the corresponding bid values and used CPLEX to determine the optimal allocation solution (OPT_ALLOC) for all test sets, which was used to compare the Linear-ALLOC and Logistic-ALLOC algorithms of this paper. We also used the heuristic algorithm G-VMPAC-II-ALLOC proposed in the literature [Nejad, Mashayekhy and Grosu (2015)] for comparison.

From Fig. 2, we can see that in solving social welfare, the Linear-ALLOC and Logistic-ALLOC algorithms are better than the existing algorithm G-VMPAC-II-ALLOC. It is proved that the optimal allocation solution has a potential model that can be fitted by a machine learning algorithm. The social welfare derived from the Linear-ALLOC algorithm is very close to the optimal allocation solution.

Fig. 3 shows the prediction accuracy. It can be seen that the prediction accuracy of G-VMPAC-II-ALLOC based on the greedy method is lower than 90%, which means that more than 10% of users should be allocated resources but are not. The prediction allocation algorithms based on machine learning has very high accuracies, all over 95%. Among them, the prediction accuracy of the Linear-ALLOC algorithm is more than 98%. The accuracy rate can reflect the fairness of the algorithm, which is a very important indicator in resource allocation.

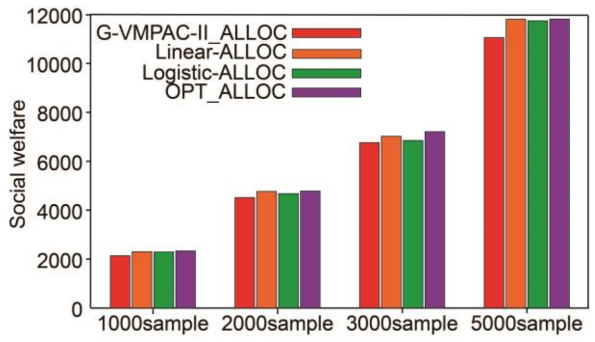

Figure 2: Social welfare comparison

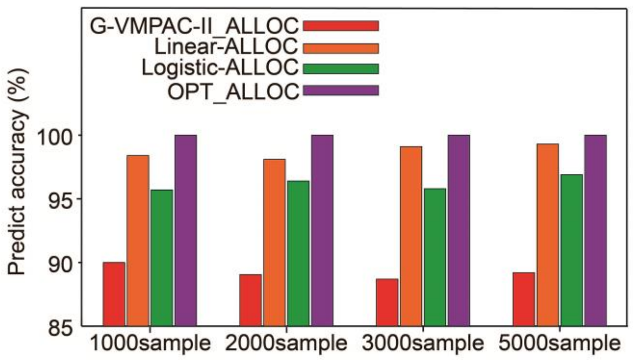

Figure 3: Prediction accuracy comparison

Fig. 4 shows the comparison of resource utilizations. Under a given resource capacity, the optimal solution exhausts the CPU and storage resources. The utilization of the machine learning-based algorithm is very close to the optimal solution resource utilization, which is better than the greedy-based allocation algorithm G-VMPAC-II-ALLOC.

We can see that the performance of the Linear-ALLOC algorithm is higher, mainly because the cost function of Linear-ALLOC contains the optimal payment price feature, which is one more dimension than the Logistic-ALLOC algorithm contains; however, it is worth noting that Linear-ALLOC needs to use VCG to solve the optimal payment price, which also brings a certain amount of time. In general, the Linear-ALLOC algorithm achieved the best performance in the experiment, which is consistent with the previous theoretical analysis above.

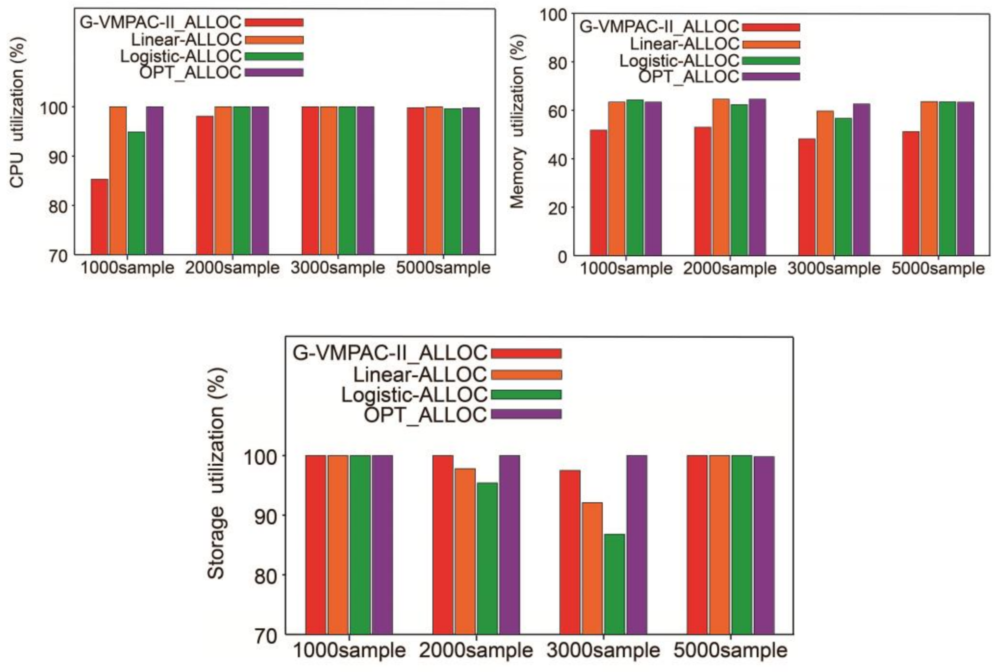

Figure 4: Resource utilization comparison

5 Conclusion

It is an innovative idea to transform the multi-dimensional cloud computing resource allocation problem into a regression or classification machine learning problem. By learning the training data, the algorithm can make the feasible solution very close to the optimal solution in terms of social welfare allocation accuracy and resource utilization. It is verified that there is indeed a potential model in optimal resource allocation that presents a new solution for multi-dimensional cloud computing resource allocation. In this paper, we did not discuss whether the resource allocation algorithm based on machine learning satisfies the strategy proof of the auction mechanism. This is one of our major tasks for future research.

Acknowledgement: This research is supported by the National Natural Science Foundation of China (No. 61472345, 61762091 and 11663007), the Scientific Research Foundation of the Yunnan Provincial Department of Education (No. 2017ZZX228), IRTSTYN, and Program for Excellent Young Talents, Yunnan University.

References

ASCI (2017): The grid workloads archive. http://gwa.ewi.tudelft.nl/.

Darmann, A.; Pferschy, U.; Schauer, J. (2010): Resource allocation with time intervals. *Theoretical Computer Science*, vol. 411, no. 49, pp. 4217-4234.

Kellerer, H.; Pferschy, U.; Pisinger, D. (2004): *Knapsack Problems*. Springer Berlin

Heidelberg.

Lai, J.; Parkes, D. (2012): Monotone branch-and-bound search for restricted combinatorial auctions. *ACM Conference on Electronic Commerce*, pp. 705-722.

Lin, W.; Xu, S.; He, L.; Li, J. (2017): Multi-resource scheduling and power simulation for cloud computing. *Information Sciences*, vol. 397, pp. 168-186.

Lin, W.; Xu, S.; Li, J.; Xu, L.; Peng, Z. (2017): Design and theoretical analysis of virtual machine placement algorithm based on peak workload characteristics. *Soft Computing*, vol. 21, no. 5, pp. 1301-1314.

Lin, W.; Zhu, C.; Li, J.; Liu, B.; Lian, H. (2015): Novel algorithms and equivalence optimisation for resource allocation in cloud computing. *International Journal of Web and Grid Services*, vol. 11, no. 2, pp. 193-210.

Liu, X.; Li, W.; Zhang, X. (2017): Strategy-proof mechanism for provisioning and allocation virtual machines in heterogeneous clouds. *IEEE Transactions on Parallel and Distributed Systems*, pp. 1.

Mashayekhy, L.; Fisher, N.; Grosu, D. (2016): Truthful mechanisms for competitive reward-based scheduling. *IEEE Transactions on Computers*, vol. 65, no. 7, pp. 2299-2312.

Mashayekhy, L.; Nejad, M. M.; Grosu, D. (2015): A ptas mechanism for provisioning and allocation of heterogeneous cloud resources. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 9, pp. 2386-2399.

Nejad, M. M.; Mashayekhy, L.; Grosu, D. (2015): Truthful greedy mechanisms for dynamic virtual machine provisioning and allocation in clouds. *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 2, pp. 594-603.

Nisan, N.; Roughgarden, T.; Tardos, E.; Vazirani, V. V. (2007): *Algorithmic Game Theory*. Cambridge University Press.

Sharrukh, Z.; Daniel, G. (2013): Combinatorial auction-based allocation of virtual machine instances in clouds. *Journal of Parallel and Distributed Computing*, vol. 73, no. 4, pp. 495-508.

Shi, W.; Zhang, L.; Wu, C.; Li, Z.; Lau, F. C. M. (2016): An online auction framework for dynamic resource provisioning in cloud computing. *IEEE/ACM Transactions on Networking*, vol. 24, no. 4, pp. 2060-2073.

Wu, Q.; Hao, J. K. (2016): A clique-based exact method for optimal winner determination in combinatorial auctions. *Information Sciences*, vol. 334, pp. 103-121.

Zaman, S.; Grosu, D. (2011): Combinatorial auction-based dynamic vm provisioning and allocation in clouds. *IEEE Third International Conference on Cloud Computing Technology and Science*, pp. 107-114.

Zhang, J.; Xie, N.; Li, W.; Yue, K.; Zhang, X. (2018): Truthful multi requirements auction mechanism for virtual resource allocation of cloud computing. *Journal of Electronics & Information Technology*, vol. 40, no. 1, pp. 25-34.