Dynamic Programming based Allocation of Virtual Machine Instances in Clouds

Hejing Geng^a, Zeyu Zhu^b, Yi Jiang^a, Junwu Zhu^{a,c,*}, Bin Li^{a,d}, Zhiwei Zhou^a

^aInformation Engineering School, Yangzhou University, China

^bSchool of Computer and Information Engineering, Guangxi Teachers Education University, China

^cDepartment of computer science and Engineering, Guelph University, Canada

^dState Key Laboratory of new software technology, Nanjing University, China

*Corresponding Author: jwzhu@yzu.edu.cn

Abstract

Auction-based mechanism for dynamic virtual machine provisioning and allocation (VMPA) can help cloud providers configure the resources efficiently based on the user demand and yield high revenue. The existing auction-based mechanisms mostly use greedy method to allocate the VM resources, which give priority to the users with high bid density. However, this kind of local optimal selection does not always bring the overall optimal solution. We propose dynamic programming based mechanism for solving VMPA problem (DP-VMPA), which takes the maximum social welfare as the objective function, and uses the combinatorial auction-based dynamic programming (CA-DP) allocation algorithm to solve the Winner Determination Problem (WDP). Finally, the Vickrey-Clarke-Groves (VCG) mechanism is used to decide the payment of each user. We perform simulation experiments to compare our proposed mechanism with CA-PROVISION mechanism and show that the DP-VMPA mechanism can allocate VM resources more effectively, and bring higher profits to the auctioneer.

Keywords: combinatorial auctions, dynamic resource allocation, mechanism design.

1. Introduction

Cloud computing enables individuals and small medium enterprises satisfy their computational needs, while cloud providers provide computing resources through cloud computing platform to obtain profits. A cloud computing platform abstracts the underlying physical resources with the view of virtual machines (VMs). This enables easy

management and pricing of the resources. Different types of VM instance represent different types of processors, memory, and *etc.* A user bid for a bundle of VM instances of different types required for executing her application.

The current cloud computing platforms (e.g., Microsoft Azure⁽¹⁾, Amazon EC2⁽²⁾) allocate and price the resource based on fixed-price mechanism. Obviously this allocation and pricing system has many shortcomings, first of all it can not guarantee the effective allocation of resources, those who has higher value of the resources do not have more chance to obtain the requested resources, and secondly, it leads to lower than optimal revenue for the cloud provider^(3,4). Combinatorial auction-based mechanisms can effectively solve the above problem, which allocate items based on the bundle size and the perceived value of the users.

Cloud providers provision their resources statically or dynamically and then allocate them in the form of VM instances to their customers. In the case of static provisioning, they require that the VM instances are already provisioned and would not change, while in the case of dynamic provisioning, the providers provision a set of VM instances according to the current users' demand. Due to virtualization technology, dynamic provisioning can configure available resource into different numbers and types of VM instances, and lead to efficient resource utilization. Existing methods, mostly use the greedy algorithm to determine the allocation. It sorts the users' bid in descending order of density of valuation, determines the allocation greedily by selecting bids in this order within the resource restriction. But this greedy method does not always get the optimal solution.

In this paper, we consider combinatorial auction-based mechanism that dynamically provisions and allocates VM

instances. Though the WDP is NP-complete, we can objectively formulate the VMPA problem, give the objective function, and then use the CA-DP allocation algorithm to find the optimal solution. The main idea of the CA-DP algorithm is to find the analytic construction of optimal solution, and then recursively find the winner set. We price the allocated resources by the VCG mechanism, which charges each individual the harm they cause to other bidders.

2. Related Work

Zaman⁽³⁾ first introduced the fixed-price mechanism, and then designed two combinatorial auction-based approximation mechanisms that require static provisioning: CA-GREEDY and CA-LP. They performed extensive simulation experiments to compare them with the fixed-price mechanism to show that auction-based mechanism can significantly improve the allocation efficiency while generating higher revenue for the cloud providers. The CA-LP mechanism is a better choice when the objective is to obtain higher revenue and higher utilization of resources and the CA-GREEDY mechanism is widely recognized for its low execution time and efficient allocation performance.

Zaman⁽⁴⁾ presented the CA-PROVISION mechanism to address the VMPA problem dynamically. They compared that with CA-GREEDY mechanism and the results showed that dynamic provisioning considering the current demand from the users can ensure the efficient use of resources and increase the profit of the auctioneer.

An effective bidding strategy⁽⁵⁾ was proposed to help the cloud computing users generate the best bid within users' budgets. This bidding strategy can help users to generate efficient bids and thus, ensure good completion times and higher utility for their applications. This also helps the provider generate a fair amount of revenue while maintaining user satisfaction.

Nejad⁽⁶⁾ proposed a heuristic greedy algorithm for dynamic VMPA which considered several types of resources including cores, memory, storage, *etc.* They used relevance fact characterizing the scarcity of each type of resources to re-define bid density which was then used for greedy allocation. They^(3,4,6) all used the greedy method to allocate resources, which found the approximate solution through constructing the local optimal solution step by step and would yield more benefit for the cloud provider practically.

Garfinkel⁽⁷⁾ proposed a set partitioning algorithm to solve general version of winner determination problem. Nisan⁽⁸⁾ expressed the WDP as a standard mixed integer programming problem and showed that could be managed by use of standard algorithms and commercially available software. Sandholm⁽⁹⁾ showed that CPLEX performed well to manage large bid sets. Fujisima⁽¹⁰⁾ recommended CASS software to solve the large-scale WDP. Their work increased the efficiency of the auction but they ignore the price problem.

Feng⁽¹¹⁾ presented four possible mechanisms: opt, greedy, VCG and TruTeamin to solve the allocation and pricing problems in set packing, and proved that VCG mechanism and TruTeam mechanism can efficiently allocate the resources with truthful and individual rational properties. The author proposed a strategy-proof incentive mechanism based on the VCG mechanism to deal with Mobile Crowdsourcing problem and achieved the desired design objectives. Le⁽¹³⁾ presented two modifications of the Vickrey-Clark-Groves mechanism to accommodate bidders' budget constraints in the combinatorial auction setting. Conitzer⁽¹⁴⁾ studied two related problems concerning the VCG mechanism: the problem of revenue guarantees, and that of collusion. They all studied the VCG mechanism and proved that it can satisfy the incentive compatible property and the over payment is modest.

Boughaci⁽¹⁵⁾ presented a computational experience regarding four well-known meta-heuristics (stochastic local search, tabu search, genetic algorithms and memetic algorithms) for solving WDP. They showed that the metaheuristic approaches could provide competitive results comparing with the hybrid simulated annealing and Casanova. Parallel genetic algorithm uses two major modifications compared to the genetic algorithm⁽¹⁶⁾. They found solutions of very large problems, which are comparable or even better than any other solution found by other heuristics. An improved ant colony (IAC) algorithm considering the dynamic transition strategy and the Max-Min pheromone strategy was proposed for the allocation problem⁽¹⁷⁾. The study of approximate optimal solution to the WDP problem in combinatorial auctions can help the auctioneer to make a quick and efficient decision to increase the profits for the auctioneer.

3. Dynamic VM Provision and Allocation Problem

Virtualization technology allows the cloud computing

providers to configure computational resources into virtually any combination of different types of VMs. A cloud provider offers computing services to users through m different types of VM instances $VM_1, \dots, VM_m$. The computing power of a VM instance of type VM_i , $i = 1, \dots, m$ is w_i , where $w_1 = 1$ and $w_1 \leq w_2 \leq \dots \leq w_m$. We denote the vector of computing powers of the m VMs as $w = (w_1, \dots, w_m)$. To make this clear, we consider the types of instances currently offered by Microsoft Azure Platform: Small (CPU 1.6 GHz, Memory: 1.75 GB, Storage: 225 GB), Medium (CPU 2x1.6 GHz, Memory: 3.5 GB, Storage: 490GB), Large (CPU 4x1.6 GHz, Memory: 7 GB, Storage: 1TB), and Extra large (CPU 8x1.6 GHz, Memory: 14 GB, Storage: 14 GB). In this example, VM_1, VM_2, VM_3, VM_4 are the Small, Medium, Large and respectively Extra large VM instances. The weight vector characterizing the VM instances is $w = (1, 2, 4, 8)$. We assume that the cloud provider has enough resources to create a maximum of M VM instances of the least powerful type, VM_1 . The cloud provider can provision the VM instances in several ways according to the specified types given by $VM_1, \dots, VM_m$. We denote the number of VM_i instances provisioned by the cloud provider as k_i . The cloud provider can provision any combination of instances given by the vector $(k_1, k_2, \dots, k_m)$ as long as $\sum_{i=1}^m w_i k_i \leq M$.

We consider n users $u_1, \dots, u_n$ who request computing resources from the cloud provider specified as bundles of VM instances. A user u_j requests VM instances by submitting a bid $B_j = (r_1^j, \dots, r_m^j, v_j)$ to the cloud provider, where r_i^j is the number of instances of type VM_i and v_j is the maximum price user j is willing to pay to use the requested bundle of VMs for a unit of time. We assume that the allocation and prices are decided periodically by the cloud provider and the interval between two such decisions is a unit of time. Let's denote by $s_j = \sum_{i=1}^m w_i r_i^j$ the amount of unit computing resources requested by u_j . In order to define the profit obtained by the cloud provider, we denote by $P = \{p_1, p_2, \dots, p_n\}$ the payment vector, where p_j is the amount paid by the user u_j for using her requested bundle of VMs; Let's denote by $x = \{x_1, x_2, \dots, x_n\}$ the allocation vector, where $x_j = 1$ if the bundle u_j requested was allocated to her and $x_j = 0$, otherwise. We define the winner set as $W = \{u_j \mid 1 \leq j \leq n, x_j = 1\}$.

Definition 1 : The VMPA problem in clouds is defined as follows

$$\max \sum_{j=1}^n x_j v_j \quad (1)$$

Subject to: (i) $\sum_{j=1}^n x_j s_j \leq M$; (ii) $p_j \leq v_j$, if $x_j = 1$; (iii) $p_j = 0$, if $x_j = 0$; (iv) $x_j \in \{0, 1\}$.

The constraint in Equation (i) ensures that the total allocation should not exceed the number of available VM instances. Equation (ii) and (iii) are to make sure that the winners' payment should not exceed their valuations and the losers do not pay at all. The solution to this problem consists of allocation vector x and payment vector P . Since we know little about revenue maximization in the context of Combinatorial auctions⁽¹⁸⁾, trying to maximize the overall payment is not feasible. Usually we take maximizing the social welfare which is the sum of valuations of all winners as the most reasonable objective function. The allocation will determine the number of VMs of each type that needs to be provisioned as follows. We just need to compute $k_i = \sum_{j=1}^n x_j r_i^j$ for each type VM_i and provision k_i VM instances of type VM_i .

Definition 2 : Truthful (incentive-compatible)

A mechanism is truthful if the participants benefit most when they reveal their true valuation to the mechanism.

A truthful mechanism provides the access to find out the most suitable winners to utilize the resources. And it also helps the bidders in that they do not need compute a complex strategy or assume other users' strategies while making their own bids.

Definition 3 : Individual Rationality

Individual Rationality means that each user's utility is non-negative, i.e., for each user j , the utility $U_j = v_j - p_j$ is greater than or equal to 0. Obviously, user is willing to participate in the auction if the mechanism satisfies individual rationality.

4. Dynamic Programming based Mechanism

4.1 DP-VMPA mechanism

We propose DP-VMPA Mechanism, that computes an optimal solution to the VMPA problem. That is, it determines the price each winner has to pay, and the set of VM instances that need to be provisioned to meet the winners' demand. We now describe the mechanism in detail.

Algorithm 1 DP-VMPA mechanism

Input: $M; m; w_i : 1, \dots, m;$ **Output:** $W; P; k_i : 1, \dots, m;$

```
1. { phase 1 : Collect Bids }
   a. Initialize  $BN \leftarrow \emptyset, BestValue \leftarrow 0$ 
   b. For  $j = 1 \dots n$ ,
       Collect bid  $B_j = (r_1^j, \dots, r_m^j, v_j)$  from  $u_j$ 
        $BN \leftarrow BN \cup \{B_j\}$ 
2. { phase 2 : Winner Determination and Provision }
    $(W, BestValue) = \text{CA-DP}(BN, M)$ 
   //Algorithm 2
   For  $i = 1 \dots m$ ,
        $k_i \leftarrow \sum_{j: u_j \in W} r_i^j$ 
3. { phase 3 : Payment }
   For all  $j \in W$ 
        $(W', BestValue') = \text{CA-DP}(BN - \{u_j\}, M)$ 
       //Algorithm 2
        $p_j \leftarrow BestValue' - (BestValue - v_j)$ 
   For all  $j \notin W$ ,
        $p_j \leftarrow 0$ 
Return  $(W; P; k_i : 1, \dots, m)$ 
```

DP-VMPA mechanism is given in Algorithm 1. The mechanism requires some information from the auctioneer such as the total amount of computing resources M expressed as the total number of VMs of type VM_1 that can be provisioned by the cloud provider. The mechanism also requires as input the number of available VM types, m , and their weight vector w .

The mechanism works in three phases. In phase 1, it collects the users' bids, and all bids constitute the set BN . In phase 2, the mechanism use CA-DP allocation algorithm to determine the winning bidders and the VM configuration that needs to be provisioned by the cloud provider. In phase 3, the mechanism determines the payment of each user. For each winning bidder j , the VCG mechanism is used to price the requested bundle the social cost of her winning that is incurred by the rest of the users. We first calculate the maximal welfare $BestValue'$ with bidders other than u_j , and then minus the total sum of winners' valuation except u_j . That is the payment of u_j . Obviously all losing bidders pay zero. This type of payment is known in the mechanism design literature as the critical payment⁽¹⁸⁾.

4.2 CA-DP allocation algorithm

Combinatorial auction-based dynamic programming allocation algorithm (CA-DP) is given in algorithm 2. The

Algorithm 2 CA-DP allocation algorithm

Input: $Bids, M$ **Output:** $W, BestValue$

```
1. Initialize  $W \leftarrow \emptyset, n \leftarrow \text{size}(Bids), V[n+1][M+1]$ 
2. For  $j = 0 \dots n$ ,
    $s_j \leftarrow \sum_{i=1}^m w_i r_i^j$ 
   For  $h = 0 \dots M$ ,
       If  $h = 0 \mid j = 0$  then  $V[j][h] \leftarrow 0$ 
       Else
           If  $s_j > h$  then  $V[j][h] \leftarrow V[j-1][h]$ 
           Else  $V[j][h] \leftarrow \max\{V[j-1][h], V[j-1][h-s_j] + v_j\}$ 
3.  $h \leftarrow M$ 
4. For  $j = n \dots 1$ 
   If  $V[j][h] > V[j-1][h]$ 
   then  $W \leftarrow W \cup \{u_j\}, h \leftarrow h - s_j$ 
5.  $BestValue \leftarrow V[n][M]$ 
Return  $(W, BestValue)$ 
```

algorithm has two input parameters: the vector of users' bids $Bids$ and the total amount of VM resources M . It has two output parameters: the optimal winner set W and total sum of winners' valuation $BestValue$.

The structure of optimal solution for VMPA problem can be considered as a series of decision-making process, which decides weather each user should obtain the requested VM instances or not. If the optimal winner set contains the user u_n , that is, then the rest users $x_1, x_2, \dots, x_{n-1}$ and computing resources $M - s_j$ constitute a new VMPA problem. And if the optimal solution do not contain u_n , then we just need to solve the VMPA problem with users $x_1, x_2, \dots, x_{n-1}$ and resources M .

Then we can recursively find the optimal solution according to the structure properties. We denote by $V[j][h]$ the maximum sum of winners' valuation where users participated are $u_1, \dots, u_j$ and the amount of VM resource is h , then,

$$V[j][h] = \begin{cases} V[j-1][h] & s_j > h \\ \max\{V[j-1][h], V[j-1][h-s_j] + v_j\} & \text{otherwise} \end{cases}$$

When the amount of resources requested by the user u_j is greater than that provider could provision, $V[j][h]$ is determined by the rest users without u_j and resource amount h ; Otherwise we compute the maximal social welfare in which situation user u_j get the requested resources $V[j-1][h-s_j] + v_j$ and that in which case u_j is a loser $V[j-1][h]$, and let $V[j][h]$ equal to the higher social welfare of the two.

Obviously the ultimately we need to solve $V[n][M]$.

After we find the optimal social welfare, we determine the winner set through tracing back, that is, comparing the maximal sum of winners' valuation $V[j-1][h]$ and $V[j][h]$ to test if user u_j belongs to winner set W . We start to trace when $j = n$, $h = M$, and stop when $j = 1$ to find all the winners.

Claim 1. DP-VMPA mechanism is truthful in expectation.

Proof: (Sketch) We first show that the proposed mechanism is monotone. Since the CA-DP allocation algorithm aims to find the optimal solution, a user can increase the chance of winning her requested bundle by increasing her valuation. Therefore the allocation is monotone. Secondly, we will prove the payment calculated by the VCG mechanism is the critical value. We all know that VCG mechanism uses the social cost of user's winning that incurred by the rest of the bidders to decide the payment. And for all losers, the payment is 0. Thus, monotone allocation and critical payment guarantee that our proposed mechanism is truthful.

Claim 2. DP-VMAP mechanism is individually rational.

For each winner u_j , the payment $p_j = \text{BestValue}' - (\text{BestValue} - v_j)$, where $\text{BestValue}' \leq \text{BestValue}$ because CA-DP algorithm decides the maximal the social welfare. Thus, we can easily prove that the utility $U_j = v_j - p_j$ of u_j is greater than or equal to 0. Utility of Ones who do not get the requested bundles is 0. So the claim is proved true.

Claim 3. Complexity

Next we analyze the complexity of the DP-VMPA mechanism. CA-DP allocation algorithm for the optimal solution use two loops to find out the optimal solution: for $j = 0, \dots, n$ and for $h = 0, \dots, M$. The loops has a worst case complexity of $O(nM)$. It also use a loop to search the winner set W by tracing back method and the execution time of the loop is $O(n)$. Thus, the total time complexity of CA-DP is $O(nM)$. When we use VCG mechanism to calculate the payment, for all winners we invoke the CA-DP algorithm again and the complexity is $O(nnM)$. Hence, the overall worst time complexity is $O(nnM)$.

5. Experimental Results

5.1 CA-PROVISION mechanism

We compare our proposed mechanism with CA-PROVISION mechanism that computes an

Algorithm 3 CA-PROVISION mechanism

Input: $M; m; w_i : 1, \dots, m;$

Output: $W; P; k_i : 1, \dots, m$

1. { *phase 1 : Collect Bids* }

For $j = 1 \dots n$,

Collect bid $B_j = (r_1^j, \dots, r_m^j, v_j)$ from u_j

2. { *phase 2 : Winner Determination and Provision* }

a. Initialize $W \leftarrow \emptyset$

b. For $j = 1 \dots n$, $s_j \leftarrow \sum_{i=1}^m w_i r_i^j$, $d_j \leftarrow v_j / s_j$

c. Add u_{n+1} with $B_{n+1} = (1, 0, \dots, 0, v_{res})$,

Re-order users $u_1, u_2, \dots, u_{n+1}$ such that

$d_1 \geq d_2 \geq \dots \geq d_{n+1}$

d. Let $l+1$ be index of u_{n+1} , discard

$u_{l+1}, \dots, u_{n+1}$,

$n \leftarrow l$, $R \leftarrow M$

e. For $j = 1 \dots n$,

If $s_j \leq R$

then $W \leftarrow W \cup \{u_j\}$, $R \leftarrow R - s_j$

f. For $i = 1 \dots m$,

$k_i \leftarrow \sum_{j: u_j \in W} r_i^j$

3. { *phase 3 : Payment* }

a. For each $u_j \in W$,

• $W_j' \leftarrow \{u_l : u_j \notin W \Rightarrow u_l \in W\}$

$l \leftarrow \text{minimum index in } W_j'$

• If $W_j' \neq \emptyset$ then $p_j \leftarrow d_l * s_j$

• Else $p_j \leftarrow 0$

b. For each $u_j \notin W$, $p_j \leftarrow 0$

Return ($W; P; k_i : 1, \dots, m$)

approximate solution to the VMPA problem. That is, it determines the price the winning users have to pay, and the set of VM instances that needs to be provisioned by the cloud provider.

CA-PROVISION mechanism is given in Algorithm 3. The mechanism determines the winners by first setting a dummy user u_{n+1} and ranking all users in decreasing order of their bid density (i.e., v_j / s_j), then discarding users for which $d_j < d_{u+1}$ to prevent losses, finally greedily allocating them starting from the top of the list. For each winning bidder u_j , it finds the losing bidder u_l who would win if u_j would not participate. User u_j 's payment is then calculated by multiplying her bundle size s_j with the bid density d_l of u_l . Otherwise she pays zero.

5.2 Experimental Setup

Table 1. Simulation Parameters

Parameter	Description	Value(s)
m	Types of VMs	4
M	Available computing resource	2000
w	Relative weight of VMs	(1,2,4,8)
N	Maximum number of users	1000
r^{min}, r^{max}	Min. & Max. VM instances of each type in a bundle	0,5
v^{max}	Maximum valuation	1,2,5,10
t^{min}, t^{max}	Min. & Max. execution time	1,2
d^{min}, d^{max}	Min. & Max. deadline	2,4

We perform two set of simulation experiments, Each user takes part in two separate auctions which use CA-PROVISION mechanism and DP-VMPA mechanism respectively to decide the allocation and the payment. Users are randomly generated and they simultaneously take part in two set of experiments. Considering the time complexity of DP-VMPA mechanism, we cut the size of experiment by setting longer intervals and decreasing the experimental parameters.

Each experiment consists of 5 sequential auctions. A total of about 1000 people are generated, that is, at the beginning 600 users are involved in the auction, after each round of the auction 100 new users are created to take part in the auction. Users are generated during that auction and all users from previous time slots that are still active. An active user is one whose task has not been finished or the task deadline has not been reached.

We assume that the cloud provider offers four types of VM instances: small, medium, large, and huge (VM_1, VM_2, VM_3, VM_4) and their relative weight vector is $w = (1, 2, 4, 8)$. The total available resource (i.e., all VM_1 instances) is 2000. Each user u_j 's bid is a 5-tuple $(r_1^j, r_2^j, r_3^j, r_4^j, v_j)$, where r_i^j is the number of instances of type VM_i requested and v_j is the valuation. User u_j 's task is characterized by the tuple (t_j, d_j) , where t_j is the unit of time the resources are requested for and d_j is the time by which u_j 's job needs to be completed.

We list all the simulation parameters and their values in Table 1. These parameters r_i^j, v_j, t_j , and d_j are randomly chosen from their respective minimum and maximum ranges. To create different instances of VMPA, we consider four values of v_j , which give four ranges of valuations (0-1), (0-2), (0-5), and (0-10). Combining all these valuations with each other, our simulation experiment runs 50 different instances of VMPA.

5.3 Analysis of Results

Fig. 1. Percent of user Served

Figure (1) shows the percent of users who finish their tasks with different ranges of user valuation. We see that the served users served by DP-VMPA mechanism are more than twice that served by CA-PROVISION mechanism. This is because CA-PROVISION gives priority to users which is related to bid density, while DP-VMPA fully considers all users' bids to maximize social welfare and this makes resources utilized efficiently and serves more users.

Fig. 2. Utility of users Served vs. Valuation Ranges

Figure (2) describes utility of user who finished her task with respect to users' valuation. It shows that the two mechanisms can make users receive more utility when the valuation is larger, and CA-PROVISION mechanism has better performance. It also means that VCG mechanism used by our proposed mechanism charges more than CA-PROVISION mechanism, which will yield more revenue for the cloud provider.

Fig. 3. Revenue vs. Valuation Ranges

Figure (3) shows the revenue generated for different ranges of user valuation. We can see that, the use of DP-VMPA mechanism can always make the cloud provider get higher revenue. This is determined by the nature of the two mechanisms. CA-PROVISION mechanism using greedy algorithm, selects the most suitable user with the highest bid density and allocates the VM bundles requested within resource capacity. This method is easy to implement but is not optimal. CA-DP allocation algorithm finds the optimal solution to maximize the social welfare. This mechanism achieves the highest sum of winners' valuation and brings more benefits to the auctioneer. Furthermore, VCG mechanism can also help to bring more revenue for the cloud provider.

6. Conclusion

In order to solve the VMPA problem, we proposed dynamic programming based mechanism. This mechanism takes maximizing social welfare as the objective function, and recursively solves the optimal solution to determine the winner set, and uses the VCG mechanism to price the user for resources. Experiments comparing with the CA-PROVISION mechanism showed that the DP-VMPA mechanism can enable more users to complete the application, improve the allocation efficiency, and significantly increase the cloud provider's profits. However the mechanism is more complex, we have to limit the application of DP-VMPA to instances with a small number of users. DP-VMPA can also be a choice when auctions are run at longer intervals. Future work will focus on the optimization of CA-DP allocation algorithm, and improve the VCG mechanism to make the pricing mechanism more convenient and efficient.

References

- (1) Microsoft, "Windows Azure platform, <http://www.microsoft.com/windowsazure/>."
- (2) Amazon, "Amazon Elastic Compute Cloud (Amazon EC2), <http://aws.amazon.com/ec2/>."
- (3) Zaman, S., and D. Grosu. "Combinatorial Auction-Based Allocation of Virtual Machine Instances in Clouds." 2013 IEEE 5th International Conference on Cloud Computing Technology and Science IEEE, pp.127-134, 2010.
- (4) Zaman, Sharrukh, and D. Grosu. "Combinatorial Auction-Based Dynamic VM Provisioning and Allocation in Clouds." 2011 Third IEEE International Conference on Cloud Computing Technology and Science IEEE Computer Society, pp.107-114, 2011.
- (5) Zaman, Sharrukh, and D. Grosu. "Efficient Bidding for Virtual Machine Instances in Clouds." 2013 IEEE Sixth International Conference on Cloud Computing IEEE, pp.41-48, 2011.
- (6) Nejad, Mahyar Movahed, L. Mashayekhy, and D. Grosu. "Truthful Greedy Mechanisms for Dynamic Virtual Machine Provisioning and Allocation in Clouds." IEEE Transactions on Parallel & Distributed Systems 26.2, pp.594-603, 2014.
- (7) Garfinkel, R. S., and G. L. Nemhauser. "The Set Partitioning Problem: Set Covering with Equality Constraints." Operations Research 17.1, pp.S40-S47, 1969.
- (8) Nisan, Noam. "Bidding and Allocation in Combinatorial Auctions." Ec 00 Acm Conference on Electronic Commerce, pp.1-12, 2001.
- (9) Sandholm, Tuomas. "Algorithm for optimal winner determination in combinatorial auctions." Artificial Intelligence 135.3, pp.1-54, 2002.
- (10) Fujisima, Yuzo, K. Leyton-Brown, and Y. Shoham. "Taming the computational complexity of combinatorial auctions." In Proceedings of the 16th Int'l Joint Conf on AI, pp.548-553, 1999.
- (11) Liu, Qing, *et al.* "An efficient and truthful pricing mechanism for team formation in crowdsourcing markets." Communications (ICC), 2015 IEEE International Conference on IEEE, 2015.
- (12) Feng, Zhenni, Y. Zhu, and L. M. Ni. iMac: Strategy-Proof Incentive Mechanism for Mobile Crowdsourcing. Wireless Algorithms, Systems, and Applications. Springer Berlin Heidelberg, pp.337-350, 2013.
- (13) Le, Phuong. "Modified VCG Mechanisms in Combinatorial Auctions with Budget Constraints." Social Science Electronic Publishing, 2015.
- (14) Conitzer, Vincent, and T. Sandholm. "Failures of the VCG mechanism in combinatorial auctions and exchanges." Fifth International Joint Conference on Autonomous Agent & Multiagent System, pp.521-528, 2006.
- (15) Boughaci, Dalila. Metaheuristic Approaches for the Winner Determination Problem in Combinatorial Auction. Artificial Intelligence, Evolutionary Computing and Metaheuristics. Springer Berlin Heidelberg, pp.775-791, 2013.
- (16) Mühlenbein, Heinz. "PARALLEL GENETIC ALGORITHMS IN COMBINATORIAL OPTIMIZATION - Computer Science and Operations Research." Computer Science & Operations Research pp.441-453, 1992.
- (17) Qian, Xiaohu, *et al.* "An improved ant colony algorithm for winner determination in multi-attribute combinatorial reverse auction." Evolutionary Computation (CEC), 2014 IEEE Congress on IEEE, pp.1917-1921, 2014.
- (18) Nisan, Noam, *et al.* "Algorithmic Game Theory." Noam Nisan 53.7, pp.359, 2007.