Journal of Parallel and Distributed Computing 149 (2021) 1–12

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Coalition formation for deadline-constrained resource procurement in cloud computing

Junyan Hu [a,b], Kenli Li [a,b,*], Chubo Liu [a,b], Jianguo Chen [a,b], Keqin Li [a,b,c,*]

[a] College of Information Science and Engineering, Hunan University, Hunan, 410082, China
[b] National supercomputing center in Changsha, Hunan, 410082, China
[c] Department of Computer Science, State University of New York, New Paltz, NY 12561, USA

ARTICLE INFO

Article history:
Received 26 June 2019
Received in revised form 4 March 2020
Accepted 13 October 2020
Available online 28 October 2020

Keywords:
Cloud computing
Coalition game
Rectangular packing
Resource procurement

ABSTRACT

To attract more customers, a cloud provider tends to give some discounts to a customer if he/she rents a plenty of resources. Under this situation, a group of customers who need homogeneous cloud instances with various deadlines are prone to purchasing resources in a collaborative manner, i.e., using a coalition game, to reduce purchase costs. It is essential to design a mechanism that enables all customers to voluntarily and happily collaborate while ensuring that each customer pays at the lowest cost possible. To address this issue, we propose a mechanism to show collaborative interactions between customers and determine the number of service programs purchased from each provider to charge each cloud customer a minimum cost. We establish a coalition game based on multi-customer resource procurement and prove that there exists a unique optimal solution in the coalition game, while satisfying individual stability and group stability. In addition, the optimal solution is a solution in which the selected service program of each coalition optimizes the cost per customer and maximizes resource utilization. We propose a heuristic Deadline-constrained Resource Coalition Allocation (DRCA) algorithm to calculate the near-optimal solution. A backtracking algorithm is proposed to calculate the pseudo maximum resource utilization of the provided programs by improving the rectangular packing. Extensive experiments are performed to verify the feasibility and effectiveness of the proposed algorithm.

1. Introduction

1.1. Motivation

Benefiting from excellent computing power and elastic resource allocation, cloud computing is widely applied in a variety of applications, such as Amazon EC2, Microsoft Azure and Google AppEngine [6]. It offers an attractive paradigm of dynamic provisioning computing services in a *pay-as-you-go* manner [33]. In the real world, providers offer cloud service/computing resources to customers at various prices, depending on the amount of resources customers need and the length of the rental service. In general, the longer the cloud service time or the more cloud instances the customer rents, the lower the rental price per unit. Faced with a variety of service programs offered by providers at various prices, it is important to configure appropriate resource procurement strategies for multiple customers to minimize overall purchasing costs.

In cloud scenario, cloud customers submit the computing resource requests to the cloud providers, and cloud providers make resource allocation according to the resource requirements of multiple cloud customers. According to customers' different types of task requirements, it is normal for each customer to have an expected completion time, *i.e.*, deadline constrain. Many customers only need to rent resource services within a certain deadline, which is not less than the resource lease time. We can find that many researchers studied the issues related to cloud customers with different deadline constrains [11,15,26]. Besides, each customer wants to reduce costs while his requirements are met. Many customers reduce their costs through purchasing cloud resources provided by the brokers [5,22,40], comparing the prices of cloud resources provided by multiple providers, or purchasing cloud resources in combination [39].

Therefore, suppose a customer wants to rent a relatively short-term cloud service or a small amount of cloud instances before a specified deadline. In this scenario, if he rents cloud instances as an individual, the rental cost is high, and the customer is not

* Corresponding author at: College of Information Science and Engineering, Hunan University, Hunan, 410082, China.
E-mail addresses: junyanhu@hnu.edu.cn (J. Hu), lkl@hnu.edu.cn (K. Li), liuchubo@hnu.edu.cn (C. Liu), cccjianguo@163.com (J. Chen), lik@newpaltz.edu (K. Li).

https://doi.org/10.1016/j.jpdc.2020.10.004
0743-7315/© 2020 Elsevier Inc. All rights reserved.

Table 1
Comparison of cloud-computing-resource purchase models.

Model	Object	Deadline	Incentive	Method
ACR − MSC [31]	MP	No	No	CF
CCF [37]	MC	No	No	CF
C-DSIC,C-BIC,C-OPT [25]	SCU	No	Yes	auction
$\mathcal{NPBA}$ [17]	MC	No	No	NC
IEDA [36]	MP	No	Yes	auction
DRCA (our work)	**MC**	**Yes**	**Yes**	**CF**

MP: multi-provider, MC: multi-customer, SCU: single customer, CF: coalition formation, NC: non-cooperative.

entitled to preferential treatment. The customer also wants to pay a lower price as long as available service programs meet his demands. As a solution, several customers can purchase a low-cost service program in a cooperative way to reduce their individual costs. It is essential for each customer to choose the most appropriate customers to collaborate with and purchase service programs from appropriate providers to minimize the cost per customer.

In cloud computing, various mechanisms were have been proposed to optimize the cost or benefit of providers [12,18,31,32,36]. However, far fewer studies focus on the cost or benefits of the customers [17,25]. Our work focuses on optimizing cost for customers using a coalition game. In Table 1, the comparison of the related cloud computing resource purchase models with some properties intuitively shows our motivation.

1.2. Our contributions

In this paper, we focus on a multi-customer cooperative mechanism for leasing cloud resources and design a platform for customers to submit their resource purchase demands, including quantity, rental time, and deadline. Based on the information provided by the platform, customers can determine the number of service programs purchased from each provider and achieve the lowest cost. After formulating the proposed issue as a knapsack packing problem, we propose an incentive backtracking algorithm to show the cooperation configuration. Experiments are conducted to show the cost difference between the states of a single resource purchase and collaboration. Our main contributions are as follows.

- We propose a cooperation between cloud customers on resource demands with different deadline constraints, and design an incentive cooperation platform.
- We add a deadline constraint on the basis of the 2D rectangular packing problem [10] and propose a heuristic Deadline-constrained Resource Coalition Allocation (DRCA) algorithm to calculate the near-optimal solution. The farther the customer's deadline, the higher the customer's priority and the lower the price.
- A backtracking algorithm based on the knapsack packing problem is proposed to configure the customer cooperation strategy, determine the number of service programs purchased from each provider, and calculate the resource price and the execution order of each customer to minimize the cost of all cloud customers.
- Experiments are performed to evaluate the proposed algorithm in terms of cost analysis of customers, deadline analysis, and effectiveness evaluation.

The remainder of the paper is organized as follows: In Section 2, we introduce the related work. Section 3 describes the system model and the coalition formation game. In Section 4, we prove the existence of the optimal solution of the coalition game. In Section 5, a heuristic DRCA algorithm is proposed to calculate the near-optimal solution. In Section 6, extensive experimental results indicate the feasibility and effectiveness of our algorithm. We conclude the works of this paper in Section 7.

2. Related work

Cloud computing technology has been rapidly developed and widely used owing to its flexible resources and on-demand payment models, and has attracted academic and industrial attention. We present a review of the related work on cloud-computing resource procurement, coalition games, and 2D rectangle packing.

Resource procurement has been extensively studied in various areas, such as edge computing [23], smart grids [27], and cloud computing [4,13,16,25,31,37]. In [25], Prasad et al. proposed mechanisms to help customers choose the appropriate providers and purchase resources at reasonable prices. In [4], Baranwal et al. proposed a multi-attribute combinatorial reverse auction for cloud resource procurement, which considers both price and non-price attributes. There are abundant studies on cloud computing resource procurement based on customer groups [31,37]. In [31], economic encounters between consumers and cloud providers are modeled as a many-to-many negotiation. In addition, Sim et al. devised a novel interaction protocol and a novel negotiation strategy. In [37], Wang et al. designed an implementation scheme to support group-buying on the cloud market with the method of coalition formation game. Besides, numerous studies focused on the behavior of providers in the face of multi-customer resource procurement [7,29,36]. However, previous efforts did not consider multi-customer resource requirements under deadline constraints. Our work highlights the cooperative resource procurement among customers under deadline constraints to charge the lowest cost for each customer.

Game theory is an important mathematical method for studying conflicts and cooperation between intelligent rational decision-makers [2,19,28,30]. Coalition games play an increasingly important role in computer science [20,21,24,34,35]. In [35], Tram et al. offered an optimal price policy for each provider to maximize final revenue from the perspective of competition and cooperation among providers. [20] proposed a game theory framework (termed Cooper) for task collocation, in which preserving performance was presented to find user collocation preferences and determine stable matches among them. In [21], Marinescu et al. formed dynamic rack-level coalitions of servers and created a package of these coalitions to address resource management issues for big data applications. In [24], Pillai proposed a resource allocation mechanism to minimize wastage and configure services before actual requests. In our work, we heuristically formulate a coalition mechanism based on established coalition rules to minimize the cost of each customer who purchases resources with his deadline constraints.

The 2D rectangle packing problem is a subset of the classical packing problem. Numerous algorithms, especially heuristic algorithms, have been proposed in previous years [1,3,8]. In [3], Baker et al. proposed the Bottom-Left (BL) heuristic algorithm that was improved to create a Bottom-Left-Fill (BLF) algorithm [8]. In [1], the Greedy Randomized Adaptive Search Procedure (GRASP) algorithm was proposed for the constrained 2D nonguillotine cutting problem to maximize the value of the cutting pieces. Huang et al. proposed a heuristic algorithm based on two important concepts: corner-occupying action and caving degree [10]. More algorithms for 2D rectangular packing can be found in [14,38]. Heuristically, we approximately view the coalition system, where a coalition purchases a service resource program is seen as a 2D rectangular packing problem, and the optimal goal is to maximize the resource utilization of the program. The difference between our work and the existing works is that each customer needs to consider his position constraint, i.e., deadline.

Table 2
Configuration of an ECS instance of Alibaba Cloud [9].

District	I/O	CPU	Memory	System disk
NC1	ecs.g5.2xlarge	8cores	32G	ultrahigh.io200GB

Table 3
Resource service programs of the ECS instance.

$SP_{j,k}$	$M_{j,k}$	$T_{j,k}$ (day)	$Price_{j,k}$	Real unit price
$SP^0_{j,k}$	1	1	87.312	87.312
$SP^1_{j,k}$	1	30	1039	34.633
$SP^2_{j,k}$	3	90	8433	31.233
$SP^3_{j,k}$	5	180	26 580	29.533
...	...	...	...	...

3. System model and coalition formation game

3.1. Cloud instance providers

A set of cloud service providers is denoted as $\mathcal{N} = \{1, 2, \ldots, N\}$ and indexed by j. Provider j supports K_j resource types which is indexed by k. The kth type of service resources provided by the provider j is represented as $SP_{j,k}$. There are $H_{j,k}$ service programs of resource $SP_{j,k}$. The sth service program is denoted as $SP^s_{j,k} = (M^s_{j,k}, T^s_{j,k}, P^s_{j,k})$ $(s \in \{1, 2, \ldots, H_{j,k}\})$, where $M^s_{j,k}, T^s_{j,k}, P^s_{j,k}$ are the number of resources $SP_{j,k}$, the available service time, and the price of the service program, respectively. Additionally, we suppose $SP^0_{j,k} = (1, 1, P^0_{j,k})$ as a standard service program with a standard price $P^0_{j,k}$. For example, the configuration of an Elastic Compute Service (ECS) instance of Alibaba Cloud [9] and the corresponding service resource programs are shown in Tables 2 and 3, respectively. The unit prices of $SP^1_{j,k}$ and $SP^2_{j,k}$ are 37.544 and 31.233, respectively, while the on demand standard service price is 87.312. As the number of cloud instances or service time increases, the price concession for service programs increases.

3.2. Cloud customers

A set of cloud service customers is denoted as $\mathcal{M} = \{1, 2, \ldots, M\}$ and indexed by i. These customers can either be a set of individuals or enterprises who have cloud resource demands with various deadline constraints and want to cooperate with others to minimize their costs. We assume that the customers are trustworthy and not malicious. The resource purchase demand u_i of customer i is defined as: $u_i = (u_{i,j,k})$, and $u_{i,j,k} = \langle m_{i,j,k}, t_{i,j,k}, d_{i,j,k}\rangle$, where $m_{i,j,k}$ is the number of resources of $SP_{j,k}$ required by the customer i, $t_{i,j,k}$ is the rental time, and $d_{i,j,k}$ is the corresponding deadline. In addition, we denote the set of customers that require $SP_{j,k}$ as $\mathcal{M}_{j,k}$. The collection of demands of customers in $\mathcal{M}_{j,k}$ is denoted as $u_{\mathcal{M}_{j,k}}$. As an example, if customer 1 needs 5 VMs of resource $SP_{1,1}$ for 6 h and his deadline is 10 h, and the resource configuration of this type is shown as Table 2. Therefore, his resource purchase demand can be represented as $u_1 = (u_{1,1,1}) = \langle m_{1,1,1} = 5, t_{1,1,1} = 6, d_{1,1,1} = 10\rangle$.

The objective of a customer is to purchase resources to meet his demand while minimizing the cost. Each customer can participate in multiple programs at the same time. Therefore, the individual optimal purchasing strategy of customer i can be formulated as an optimization problem:

$$\begin{aligned} \min \quad & \sum_{j,k,s}(x^s_{i,j,k}P^s_{j,k} + x^0_{i,j,k}P^0_{j,k}), \\ & C1 : x^s_{i,j,k} \in \mathbb{N}, x^0_{i,j,k} \in \mathbb{N}, \\ & C2 : \sum_{s,r} m^{s,r}_{i,j,k} = m_{i,j,k} - \frac{x^0_{i,j,k}}{t_{i,j,k}}, \\ & C3 : \text{if } x^s_{i,j,k} > 0, \forall r \in \{1, \ldots, x^s_{i,j,k}\}, \\ & \quad m^{s,r}_{i,j,k} \le M^s_{j,k}, t^{s,r}_{i,j,k} \le \min\{T^s_{j,k}, d_{i,j,k}\}. \end{aligned} \tag{1}$$

$x_i = (x^s_{i,j,k})(j \in \mathcal{N}, k \in K_j, s \in H_{j,k})$ and $x^0_i = (x^0_{i,j,k})$ $(C1)$ are the purchase profiles of customer i and represent the number of service programs $SP^s_{j,k}$ and $SP^0_{j,k}$, respectively. $m^{s,r}_{i,j,k}$ is the number of resources and $t^{s,r}_{i,j,k}$ is the time length that the customer i uses in the rth service program $SP^s_{j,k}$ $(r \in \{1, \ldots, x^s_{i,j,k}\})$. $C2$ shows that the total number of resource purchased by customer i from $SP_{j,k}$ is equal to the resource requirement of customer i. $C3$ guarantees that each service program selected by customer i meets the requirements of customer i. Note $t^{s,r}_{i,j,k} = t_{i,j,k}$.

3.3. An illustrating example for customers' coalition

We give an example to illustrate customers' coalition equilibrium state and their choice of programs. We denote the unit price of a service program as the symbol up. In the provider profile (shown in Table 4(a)), the first column lists the service programs and the second column lists the corresponding service program prices. In particular, the third column lists the corresponding unit price up. We consider three customers $\mathcal{M} = \{1, 2, 3\}$. As listed in Table 4(b), the requirements of customer 1, 2, 3 are $\langle 3, 5, 9\rangle$, $\langle 2, 4, 5\rangle$ and $\langle 4, 3, 8\rangle$, respectively.

All candidate coalitions formed by customers $\mathcal{M} = \{1, 2, 3\}$ are enumerated in the first column of Table 4(c). The second column shows the selected programs that achieve the total cost minimum. The third column shows each customer's cost by calculating the proportion in the corresponding coalition. For example, the coalition $\{2, 3\}$ selects the program 5×8 at the price 32. Then the cost of customer 2 is equal to $\frac{2\cdot 4}{2\cdot 4+4\cdot 3} \cdot 32 = 12.8$. The last column shows the total cost of all customers in the coalition. From Table 4(c), customers 1, 2 and 3 prefer to cooperate together because it minimizes each customer's cost as well as the overall cost of all customers in the coalitions. In the current situation, no customer wants to leave this coalition, and no other customers want to join so that it reaches a equilibrium situation.

As the number of customer increases, it is difficult for each customer to find an appropriate coalition to get a lower price. However, coalition game can solve this problem well, because it can formulate a series of rules to make the overall system reach equilibrium state, i.e., individual stability and group stability. In Sections 3.4 and 3.5, we introduce the cooperative purchase platform and the coalition formation game in detail.

3.4. Cooperative Purchase Platform (CPP)

In real life, it is difficult for individual customers to find many collaborators. A Cooperative Purchase Platform (CPP) is devised to allow customers to submit their resource purchase demands. The main tasks of CPP include classifying customer purchase information and calculating customer cooperation solutions. The CPP holds the following characteristics:

(1) Time-window-based purchase demands collection. Considering that the purchase demands are arriving in a endless

Table 4
An illustrating example for customers' coalition.

(a) provider profile

Program	price	*up*	Program	price	*up*
1 × 1	P^0 : 1	1	1 × 1	P^0 : 1	1
5 × 5	P^1 : 22.5	0.9	5 × 8	P^4 : 32	0.8
8 × 8	P^2 : 43.52	0.68	6 × 10	P^5 : 42	0.7
10 × 10	P^3 : 60	0.6	8 × 9	P^6 : 46.8	0.65

(b) Customers' requirement

Customer	1	2	3
Requirement	⟨3, 5, 9⟩	⟨2, 4, 5⟩	⟨4, 3, 8⟩

(c) Coalitional customer strategies and costs

Formed coalitions	selected program	(c_1, c_2, c_3)	$\sum_{i\in\mathcal{M}} c_i$
{1}, {2}, {3}	P^0, P^0, P^0	15, 8, 12	35
{1, 2}, {3}	P^1, P^0	14.67, 7.83, 12	34.5
{1, 3}, {2}	P^4, P^0	17.78, 8, 14.22	40
{1}, {2, 3}	P^0, P^4	15, 12.8, 19.2	47
{1, 2, 3}	P^4	13.7, 7.3, 11.0	32

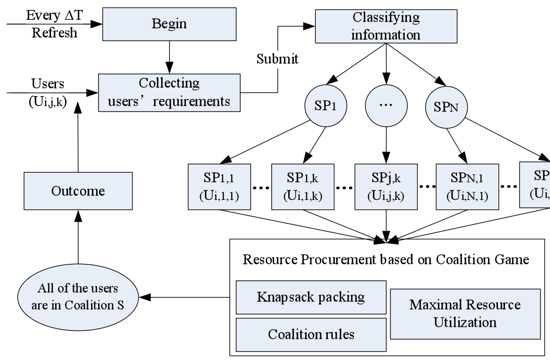


Fig. 1. Framework of cooperative purchase platform.

sequence from different customers, CPP provides a time-window-based purchase demand collection mechanism. Let $\triangle T$ be the length of each time window. Customer purchase demands in the same time window will be involved in a coalition game.

(2) Profit-driven and social-driven operating patterns. CPP supports two operating patterns: benefit-driven pattern and social-driven pattern. Based on the benefit-driven pattern, CPP maximizes the customers' initial fee by providing convenience to customers as a third-party agency. Based on the social-driven pattern, CPP can be seen as being provided by providers and free to customers in order to attract more customers to purchase their resources. We use the social-driven pattern in this paper.

(3) Group purchases in parallel by resource category. The CPP matches the customer's purchase demands with the provider's computing resource categories, such as CPU, memory, and data storage. Customers are divided into multiple groups according to their demands of computing resources, and each group can be operated in parallel.

Fig. 1 is the framework of cooperative purchase platform. The CPP is refreshed every $\triangle T$ time. During this time, customers submit their requirements to CPP, which then sorts and classifies these requirements. Further, the platform will show collaborative interactions between customers by using the method of formulating coalition rules, modifying knapsack packing, and maximizing resource utilization to be introduced in the following section. At last, CPP informs customers of the result of the coalition.

3.5. A cloud-customer coalition formation game

To provide a stable coalition structure and purchasing strategy for a group of selfish customers, cooperative game theory offers a useful set analysis tools and algorithms [2,30]. The cloud resource purchasing problem in our work can be formulated as a non-transferable utility coalition formation game $(\mathcal{M}, u_{\mathcal{M}}, c)$, where $\mathcal{M}$ is the collection of all customers, $u_{\mathcal{M}}$ is the total demand of customers in $\mathcal{M}$, and c is a cost function for each coalition.

We set $\mathcal{S} = \{\mathcal{S}_{j,k}\}$, $\mathcal{S}_{j,k} = \{S_{l,j,k}\}$ and $x_{\mathcal{S}} = \{x^s_{S_{l,j,k}}\}$, where $\mathcal{S}$ is a set of customer coalitions, $x_{\mathcal{S}}$ is the purchase cost profile of $\mathcal{S}$, and $C4 : \bigcup_l S_{l,j,k} = \mathcal{M}_{j,k}$. In addition, a *collection* is any family $\mathcal{S}$ of mutual coalitions, and $\mathcal{S}$ is a *partition* of $\mathcal{M}$. In addition, we also consider the set of customers who purchase standard service as a coalition and $C4 : x^0_{S_{l,j,k}} \in \mathbb{N}$.

We take a set of customers who can jointly purchase a service program $SP^s_{j,k}$ as a coalition. If a customer set collaborates to purchase multiple programs, the set can further be split into two or more coalitions. We can know that a coalition either purchases a program ($C4 : x^s_{S_{l,j,k}} \in \{0, 1\}$ and $\sum_s x^s_{S_{l,j,k}} = 1$) or standard service ($C4 : \sum_s x^s_{S_{l,j,k}} = 0$ and $x^0_{S_{l,j,k}} = \sum_{i\in S_{l,j,k}} m^0_{i,j,k} t_{i,j,k}$).

We establish a M-T plane coordinate system, where the m-axis and t-axis represent the number of services and service time, respectively. Supposing $i \in S_l \subset \mathcal{S}_{j,k}$ and $x^s_{S_l} = 1$, the bottom left and top coordinates of the customer i are $(m^s_{li,S_l}, t_{li,S_l})$ and $(m^s_{ri,S_l}, t_{ri,S_l})$, respectively. We denote the position of customer i in the program $SP^s_{j,k}$ as $\langle m^s_{li,S_l}, t_{li,S_l}, m^s_{ri,S_l}, t_{ri,S_l}\rangle$, where $m^s_{ri,S_l} - m^s_{li,S_l} = m^s_{i,S_l}, t_{ri,S_l} - t_{li,S_l} = t_{i,j,k}$. In other words, the regional position of customer i can be simply denoted as $Pos(u^s_{i,S_l}) = \{m^s_{li,S_l} \le m \le m^s_{ri,S_l}, t_{li,S_l} \le t \le t_{ri,S_l}\}$. $m^s_{i,S_{l,j,k}}$ and $m^0_{i,S_{l,j,k}}$ are the number of resources in program $SP^s_{j,k}$ and $SP^0_{j,k}$ required by the customer i ($i \in S_{l,j,k}$), respectively. For every customer i, the sum of $m^s_{i,S_{l,j,k}}$ and $m^0_{i,S_{l,j,k}}$ should equal to his requirement $m_{i,j,k}$, i.e., $C5 : \sum_{s,S_{l,j,k}} x^s_{S_{l,j,k}} m^s_{i,S_{l,j,k}} = m_{i,j,k} - m^0_{i,j,k}$. In addition, customers in S_l must meet the following conditions of $SP^s_{j,k}$:

- $C6 : \forall\ i \in S_l,\ 0 \le m^s_{li,S_l} < m^s_{ri,S_l} \le M^s_{j,k},\ 0 \le t_{li,S_l} < t_{ri,S_l} \le \min\{T^s_{j,k}, d_{i,j,k}\}$,
- $C7 : \forall\ i_1, i_2 \in S_l,\ Pos(u^s_{i_1,S_l}) \cap Pos(u^s_{i_2,S_l}) = \varnothing$.

Thus, the optimal purchase strategy for all customers can be characterized by the following integer programming

$$
\begin{aligned}
&\min\quad c(\mathcal{S}, x_{\mathcal{S}}) = \sum_{l,j,k,s} (x^s_{S_{l,j,k}} P^s_{j,k} + x^0_{S_{l,j,k}} P^0_{j,k}),\\
&C4 : x^s_{S_{l,j,k}} \in \{0, 1\},\ x^0_{S_{l,j,k}} \in \mathbb{N},\\
&\qquad \sum_s x^s_{S_{l,j,k}} \in \{0, 1\},\ \bigcup_l S_{l,j,k} = \mathcal{M}_{j,k},\\
&C5 : \sum_{s,S_{l,j,k}} x^s_{S_{l,j,k}} m^s_{i,S_{l,j,k}} = m_{i,j,k} - m^0_{i,j,k},\\
&C6 : \text{if } \sum_s x^s_{S_{l,j,k}} = 0,\ x^0_{S_{l,j,k}} = \sum_{i\in S_{l,j,k}} m^0_{i,j,k} t_{i,j,k},\\
&\qquad \text{if } x^s_{S_{l,j,k}} = 1,\ x^0_{S_{l,j,k}} = 0,\ \text{for } \forall\ i \in S_{l,j,k},\\
&\qquad 0 \le m^s_{li,S_{l,j,k}} < m^s_{ri,S_{l,j,k}} \le M^s_{j,k},\\
&\qquad 0 \le t_{li,S_{l,j,k}} < t_{ri,S_{l,j,k}} \le \min\{T^s_{j,k}, d_{i,j,k}\},\\
&C7 : \text{for } \forall\ i_1, i_2 \in S_{l,j,k}, Pos(u^s_{i_1,S_{l,j,k}}) \cap Pos(u^s_{i_2,S_{l,j,k}}) = \varnothing.\\
&\qquad \text{for } \forall\ i \in S_{l_1,j,k} \cap S_{l_2,j,k},\quad t_{li,S_{l_1,j,k}} = t_{li,S_{l_2,j,k}}.
\end{aligned}
\tag{2}
$$

We use **c** to represent the cost set for all customers.

We note that this integer programming problem generates a classic NP-hard problem: a rectangular packing problem with deadline constraints. To address this problem, each customer coalition has to apply certain problem solvers. We assume that the computing power of any coalition is the same, and they adopt the same problem solver: Pseudo Maximal Resource Utilization with Deadline (PMRUD). The output of PMRUD is not required to be perfectly optimal. In Section 5.1.2, we introduce a depth-first method (backtracking) for this solver. The outcome of the coalition game consists of the coalitions formed by all customers, their purchase strategies, and the corresponding costs.

Definition 1. The coalition formation game $(\mathcal{M}, u_{\mathcal{M}}, c)$ is a tuple $(\mathcal{S}, x_{\mathcal{S}}, \mathbf{c})$, where $\mathcal{S}$ is the set of all candidate coalitions of the customers, $x_{\mathcal{S}}$ is the purchase cost profile of $\mathcal{S}$ and $\bigcup_l S_{l,j,k} = \mathcal{M}_{j,k}$, and $\mathbf{c}$ is the cost set for all customers.

4. Optimal solution of the coalition game

In this section, we establish three coalition rules for the customer coalition game and prove that there is a unique optimal solution satisfying both individual stability and group stability. In addition, the optimal solution is proved to be equivalent to the system, where each coalition's selected program can optimize the cost for each customer and achieves maximum resource utilization.

4.1. Definitions

In a cooperative game, there are two conditions for the existence of a coalition game. For each coalition, the overall return of the coalition is higher than the sum of each member's return. Within the coalition, each member obtains more revenue than he does not join the coalition. Besides, the equilibrium of a cooperative game is to make the game reach individual stable and group stable. For the convenience of the proof of the theorem in Section 4.2, we give the following definitions.

Definition 2. An outcome of the set of all candidate coalitions of customers $\mathcal{S}$ is individually rational (IR) if $c_i(\mathcal{S}) \le c_i(\{i\})$ for all $i \in \mathcal{M}$.

IR means that each customer is at least as well as doing it alone.

Definition 3. An outcome of the set of all candidate coalitions of customers $\mathcal{S}$ is Contractually Group Stable (CGS) if there do not exist $S_1, S_2 \in \mathcal{S}_{j,k}$ and $S' \subseteq S_1$ such that $c_i(S_2 \cup S') \le c_i(\mathcal{S})$ for all $i \in S_2 \cup S'$ with at least one inequality holding strictly for $i \in S'$, and $c_i(S_1 \setminus S') \le c_i(\mathcal{S})$ for all $i \in S_1 \setminus S'$.

If S' is a singleton coalition formed by each customer in $\mathcal{M}_{j,k}$, then the outcome $\mathcal{S}$ is Contractually Individually Stable (CIS), i.e., CGS implies CIS.

Definition 4. The outcome $\mathcal{S}$ is $\mathbb{D}_{cp}$-stable if 1) for any collection set $(\bigcup S)$ of coalitions $(S \in \mathcal{S})$, $c_i(\bigcup S) \ge c_i(\mathcal{S})$ for at least a certain coalition S, 2) for any coalition $S \in \mathcal{S}$ and any partition π_S of set S, $c_i(\pi_S) \ge c_i(S)$ for all $i \in S$.

$\mathcal{S}$ is $\mathbb{D}_{cp}$-stable if no group of customers is interested in leaving $\mathcal{S}$.

4.2. Rules of merge, split, and transfer

We first introduce a concept of the comparison relation $\triangleright$. *Comparison relation* $\triangleright$: Given two collections $\mathcal{A}$ and $\mathcal{B}$ that are partitions in the same set K, such that $K = \bigcup \mathcal{A} = \bigcup \mathcal{B}$, $\mathcal{A} \triangleright \mathcal{B}$ indicates that $c_i(\mathcal{A}) \le c_i(\mathcal{B})$ for all customers in K and at least one inequality holds strictly. Then, given two joint coalitions A and B, i.e., $A \cap B \neq \emptyset$, $A \triangleright B$ means that $c_i(A) \le c_i(B)$ for all customers $u_i \in A \cap B$. We introduce three irreversible rules:

(1) Merge rule: If $\{\bigcup_{i=1}^{l} S_i\} \triangleright \{S_1, \ldots, S_l\}$, then $\{S_1, \ldots, S_l\} \to \{\bigcup_{i=1}^{l} S_i\}$.

(2) Split rule:

- If $\{S_1, \ldots, S_l\} \triangleright \{\bigcup_{i=1}^{l} S_i\}$, then $\{\bigcup_{i=1}^{l} S_i\} \to \{S_1, \ldots, S_l\}$;
- If $\{A\} \triangleright \{A \cup B\} \triangleright \{B\}$, then $\{A \cup B\} \to \{A, B\}$.

(3) Transfer rule: Subset S' in the coalition S_1 can join into another disjoint coalition S_2 or become a single coalition:

- If $\{S_1 \setminus S', S_2 \cup S'\} \triangleright \{S_1, S_2\}$, then $\{S_1, S_2\} \to \{S_1 \setminus S', S_2 \cup S'\}$;
- If $\{S_1 \setminus S'\} \triangleright \{S_1, S_2\} \triangleright \{S_2 \cup S'\}$, then $\{S_1, S_2\} \to \{S_1 \setminus S', S_2, S'\}$;
- If $\{S_2 \cup S'\} \triangleright \{S_1, S_2\} \triangleright \{S_1 \setminus S'\}$, then $\{S_1, S_2\} \to \{S_1 \setminus S', S_2 \cup S'\}$.

Note that the customers are selfish in the above rules. According to the merge rule, only when the cost of each customer in the newly synthesized coalition is reduced are two customer sets merged. According to the split and transfer rules, once the customer set S forms a coalition or merges into other coalitions at a lower cost, the set S will leave the original coalition regardless of the cost changes that the remaining customers would face.

Theorem 1. *For a set of customers who initially form singleton coalitions and iteratively apply the merge, split, and transfer rules, the coalition formation game* $(\mathcal{S}, x_{\mathcal{S}}, \mathbf{c})$ *has an optimal solution* $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ *that satisfies IR, CGS and* $\mathbb{D}_{cp}$*-stable simultaneously.*

Proof. The cost for each customer involved in the coalition is lower than the cost if he were to purchase the resources by himself. Otherwise, the customers would prefer to purchase resources alone. Thus, the coalition formation game satisfies *IR*.

A set of customers initially forms singleton coalitions and obtains a coalition collection $(\mathcal{S}_1, \mathcal{S}_2, \ldots)$ by using the merge rule. Then, we apply the split and transfer rules after each step. As each customer is selfish, each operation reduces the cost for at least one customer but there is no guarantee that the cost of other customers will be reduced.

Assuming that there are two identical coalitions $\mathcal{S}_{k_1} = \mathcal{S}_{k_2}$ and $k_1 < k_2$, there must exist a coalition S in $\mathcal{S}_{k_1}$ that performs one of the three rules and reforms the coalition S in $\mathcal{S}_{k_2}$ in the opposite direction. Since these rules are irreversible, it can be proven that the coalitions in sequence $(\mathcal{S}_1, \mathcal{S}_2, \ldots)$ are pairwise different. In addition, the total number of partitions for a finite customer set is finite. Thus, the customer coalition sequence formed by these rules must have a finite length. Therefore, we can say that the coalition game $(\mathcal{S}, x_{\mathcal{S}}, \mathbf{c})$ has an optimal solution $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ by applying the merge, split, and transfer rules.

Given $S_1, S_2 \in \mathcal{S}_{j,k} \subset \mathcal{S}$ and $S' \in S_1$, if there exists a $S' \in S_1$ such that $c_i(S_2 \cup S') \le c_i(\mathcal{S})$ for all $i \in S_2 \cup S'$ and $c_i(S_1 \setminus S') \le c_i(\mathcal{S})$ for all $i \in S_1 \setminus S'$, then $\{S_1, S_2\} \to \{S_1 \setminus S', S_2 \cup S'\}$ based on the transfer rule. Thus, $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ satisfies the definition of CGS.

Finally, no coalitions in $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ satisfy the conditions in the merge, split, and transfer rules, i.e., $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ coincides with the definition of $\mathbb{D}_{cp}$-stable. □

Theorem 2. *The optimal solution* $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ *obtained by applying the merge, split, and transfer rules is equivalent to the system, where the selected program* $SP^s_{j,k}$ $(x^s_{S^*_{l,j,k}} = 1)$ *of each coalition* $S^*_{l,j,k}$ *can optimize the cost for each customer and maximize resource utilization.*

Proof. According to Theorem 1, $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ satisfies CGS and is $\mathbb{D}_{cp}$-stable. Assuming that $S^*_1, S^*_2 \in \mathcal{S}^*_{j,k} \subset \mathcal{S}^*$ and there exists a customer set S in coalition S^*_1 that can be merged with another

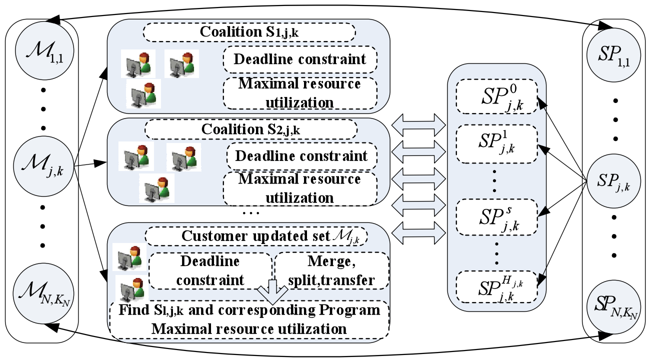


Fig. 2. The framework for multi-customer cooperative in purchasing service.

coalition S_2^* to reduce the cost for each customer in S, $c_i(S \cup S_2^*) \le c_i(S_1^*)$ for all $i \in S$ and $c_i(S \cup S_2^*) \le c_i(S_2^*)$ for all $i \in S_j^*$ according to the merge rule. However, this contradicts the fact that $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ satisfies CGS. Therefore, $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ makes the selected program $SP^s_{j,k}$ ($x^s_{S^*_{l,j,k}} = 1$) of each coalition $S^*_{l,j,k}$ optimize the cost for each customer in their own coalitions.

Assuming that there exists a program $SP^s_{j,k}$ selected by coalition $S \in \mathcal{S}_{j,k}$ that does not reach the maximum resource utilization, at least one customer set S can be added to $SP^s_{j,k}$ according to the merge rule to increase the resource utilization of $SP^s_{j,k}$. However, this contradicts the fact that $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ satisfies CGS and is $\mathbb{D}_{cp}$-stable. Therefore, $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ makes the selected program $SP^s_{j,k}$ ($x^s_{S^*_{l,j,k}} = 1$) for each coalition $S^*_{l,j,k}$ maximize resource utilization.

In turn, a solution found as $(\mathcal{S}^*, x^*_{\mathcal{S}}, \mathbf{c}^*)$ is the optimal solution if the selected program $SP^s_{j,k}$ ($x^s_{S^*_{l,j,k}} = 1$) has the lowest cost for each customer and the maximum resource utilization. □

4.3. Resource utilization

If $x^s_{S_{l,j,k}} = 1$, the resource utilization rate of service program $SP^s_{j,k}$ is the ratio between the sum of the resources actually required by all customers in the set $S_{l,j,k}$ and the resource capacity of the provided service $SP^s_{j,k}$. We denote the resource utilization rate of coalition $S_{l,j,k} \in \mathcal{M}_{j,k}$ in $SP^s_{j,k}$ as $U^s_{S_{l,j,k}}$:

$$U^s_{S_{l,j,k}} = \frac{\sum_{i \in S_{l,j,k}} m^s_{i,S_{l,j,k}} \cdot t_{i,j,k}}{M^s_{j,k} \cdot T^s_{j,k}}. \tag{3}$$

Therefore, the real price of each customer in $S_{l,j,k}$ is defined as:

$$RP^s_{S_{l,j,k}} = \frac{up^s_{j,k}}{U^s_{S_{l,j,k}}}. \tag{4}$$

There are several choices for $SP^s_{j,k}$ to choose appropriate customers from the customer set $\mathcal{M}_{j,k}$. $SP^s_{j,k}$ composed of various customer sets will have different resource utilization rates. Obviously, the larger the resource utilization, the lower the real price for customer set $S_{l,j,k}$. Therefore, it is vital for $SP^s_{j,k}$ to choose an appropriate customer set $S_{l,j,k}$ to maximize the resource utilization $U^s_{S_{l,j,k}}$. In addition, we denote the maximum resource utilization rate of $SP^s_{j,k}$ composed of $S_{l,j,k}$ as $maxU^s_{S_{l,j,k}}$.

Fig. 2 illustrates a framework in which multiple customers purchase various cloud instance service programs provided by multiple providers. Customers are grouped according to the type of resource service required, and the customers in each group cooperative with each other. The center part of Fig. 2 shows the formation of coalitions based on multiple programs provided by providers.

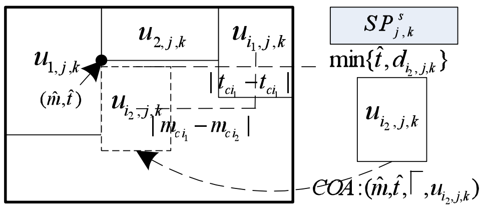


Fig. 3. An example of the corner-occupying action.

5. A deadline-constrained resource coalition allocation method

In this section, we propose the maximal resource utilization model and the Pseudo Maximum Resource Utilization (PMRU) algorithm. Based on the Algorithm PMRU, we propose the heuristic Deadline-constrained Resource Coalition Allocation (DRCA) algorithm to calculate the near-optimal solution.

5.1. Modeling maximal resource utilization

Given a program $SP^s_{j,k}$, we should find coalition S ($S \in \mathcal{M}_{j,k}$) that maximizes the resource utilization of $SP^s_{j,k}$ to satisfy the deadline constraints of customers in coalition S. The modeling for finding $S_{l,j,k}$ to maximize the resource utilization of $SP^s_{j,k}$ is defined as:

$$\begin{aligned} S_{l,j,k} &= \arg\max_S U^s_S = \arg\max_S \sum_{i \in S} \frac{m^s_{i,S} t_{i,S}}{M^s_{j,k} T^s_{j,k}}, \\ C8 &: 0 \le m^s_{li,S} < m^s_{ri,S} \le M^s_{j,k}, \\ &\quad 0 \le t_{li,S} < t_{ri,S} \le \min\{T^s_{j,k}, d_{i,j,k}\}. \\ C9 &: for\ \ \forall\, i_1, i_2 \in S_{l,j,k}, \\ &\quad Pos(u_{i_1,S}) \cap Pos(u_{i_2,S}) = \varnothing. \end{aligned} \tag{5}$$

Here, $C8$ represents that service program $S_{l,j,k}$ meets the requirements of all customers in coalition S. $C9$ guarantees that there are no overlapping positions between any two customers in coalition $S_{l,j,k}$.

5.1.1. Fundamental conceptions and strategies

(1) Corner-Occupying Action (COA): Tuple $(\hat{m}, \hat{t}, Act, u_{i,j,k})$ represents a COA, where $u_{i,j,k}$ is the demand of customer i related to the corresponding COA, Act is one of the corner shapes $\{\ulcorner, \urcorner\}$, and $(\hat{m}, \hat{t})$ is the coordinate of the top-left corner corresponding to shape $\ulcorner$ or the top-right corner corresponding to shape $\urcorner$. Customer to be packed occupies a corner formed by those two previously packed customers or the sides of the service program. In addition, customer i to be packed should satisfy all the constraints in Eq. (5). Note that the top position of $u_{i,j,k}$ is equal to $\min\{\hat{t}, d_{i,j,k}\}$. If Act is $\ulcorner$, $pos(u_{i,j,k}) = (\hat{m}, min\{\hat{t}, d_{i,j,k}\} - t_{i,j,k}, \hat{m} + m_{i,j,k}, min\{\hat{t}, d_{i,j,k}\})$, and if Act is $\urcorner$, $pos(u_{i,j,k}) = (\hat{m} - m_{i,j,k}, min\{\hat{t}, d_{i,j,k}\} - t_{i,j,k}, \hat{m}, min\{\hat{t}, d_{i,j,k}\})$. In addition, $(\hat{m}, \hat{t}, Act, u_{i_1,j,k})$ and $(\hat{m}, \hat{t}, Act, u_{i_2,j,k})$ are regarded as two COAs. An available COA means that at least one customer can be packed into the service program based on the corresponding corner position and Act. We denote a set of all available COAs as SC.

When the customer i is packed into the service program, the corners that are no longer available will be deleted and all new available corners formed by customer i and previous customers will be added.

As shown in Fig. 3, customers 1, 2, and i_1 have been packed in $SP^s_{j,k}$. In the set of remaining customers, i_2 can be placed in Act $\ulcorner$ with coordinates $(\hat{m}, \hat{t})$. COA: $(\hat{m}, \hat{t}, \ulcorner, u_{i,j,k})$ is an available COA.

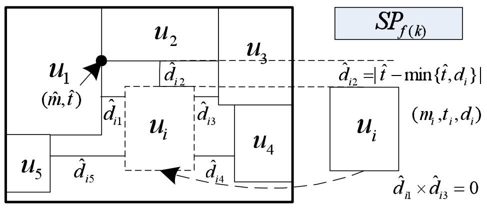


Fig. 4. Distance between one customer and neighbors.

(2) Distance between two customers. For two given customers i_1 and i_2, distance $\hat{d}_{i_1 i_2}$ between them is defined as:

$$\begin{aligned}\hat{d}_{i_1 i_2} =& max\left\{|m_{ci_1}-m_{ci_2}|-\frac{m_{i_1,j,k}+m_{i_2,j,k}}{2},0\right\}\\ &+\ max\left\{|t_{ci_1}-t_{ci_2}|-\frac{t_{i_1,j,k}+t_{i_2,j,k}}{2},0\right\},\end{aligned} \tag{6}$$

where (m_{ci_1},t_{ci_1}) and (m_{ci_2},t_{ci_2}) are the central coordinates of customers i_1 and i_2, respectively. As shown in Fig. 3, $\hat{d}_{i_1 i_2} = |m_{ci_1}-m_{ci_2}| - \frac{m_{i_1,j,k}+m_{i_2,j,k}}{2}$.

The upper and lower sides of a service program $(M \times T)$ can be viewed as customers packed into the program with resource request $(M,0)$, and the left and right sides of $(M \times T)$ can be considered customers with resource request $(0,T)$. For a given customer i and some customers packed in the service program $\{i_1|i_1 \in \{1,2,\ldots\}\}$, the distance between i and other customers is defined as $\hat{d}_i = \min\hat{d}_{il} + \min\hat{d}_{iu} + \min\hat{d}_{ir}$, where $\min\hat{d}_{il}$, $\min\hat{d}_{iu}$, and $\min\hat{d}_{ir}$ are the minimum distance between customer i and the customer set on the left, upper, and right of customer i, respectively. Moreover, at least one of $\min\hat{d}_{il}$ and $\min\hat{d}_{ir}$ is 0.

As shown in Fig. 4, $\min\hat{d}_{il}=\hat{d}_{i1}$, $\min\hat{d}_{iu}=\hat{d}_{i2}$, and $\min\hat{d}_{ir}=\hat{d}_{i3}$. If COA is $(\hat{m},\hat{t},\ulcorner,u_{i,j,k})$, $\hat{d}_{i1}=0$ and $\hat{d}_{i2}=\hat{t}-\min\{\hat{t},d_{i,j,k}\}$.

(3) Caving degree of COA and the first strategy for the maximum caving degree. Customer i is packed into $SP^s_{j,k}$ based on a feasible COA and the distance $\hat{d}_i$. The caving degree C_i of the COA is defined as:

$$C_i = 1-\frac{\hat{d}_i}{\sqrt{m_{i,j,k}t_{i,j,k}}}. \tag{7}$$

According to Eq. (7), $C_i \le 1$ and the closer C_i is 1, the closer customer i is located at his nearest customer or side (excluding the customers and sides that form the corner). During the packing process, we always select the COA with the largest caving degree and pack the corresponding customer into $SP^s_{j,k}$.

(4) Order section. During the packing process, if there are multiple COAs with the same maximum caving degree, we have the following selection order: Select the COA whose *Act* coordinate is the optimal ($Act_1(\hat{m}_1,\hat{t}_1)$ is better than Act_2 $(\hat{m}_2,\hat{t}_2)$ if $\hat{t}_1 > \hat{t}_2$, or $\hat{t}_1=\hat{t}_2$, $\hat{m}_1<\hat{m}_2$). Select the COA with the lowest index of the corresponding customer.

(5) Split of customer demands. The demand of a customer can be split into multiple sub-demands to purchase as long as those service resources collectively meet the customer's demand (i.e., same starting time, sufficient, deadline). Considering the continuity of service time and the complexity of task migration, customer's demand is split according to the number of resources rather than service time. However, if the demand of each customer is split, the amount of calculations of the whole system will be huge. Thus, we split the following two situations.

Case 1: Before the cooperation, if any $SP^s_{j,k}$ cannot meet the demand $u_{i,j,k}$ of customer i, we split the customer's demand. We find out the program with the maximal service time T_{max}:

$$T_{max} = \max_{s\in H_{j,k}}\{T^s_{j,k}\}. \tag{8}$$

Then, we find out the program $SP^s_{j,k}$ with the largest number of resources M_{max} that satisfies $t_{i,j,k} \le T^s_{j,k}$, where

$$M_{max} = \max_{s\in H_{j,k}}\{M^s_{j,k}|t_{i,j,k} \le T^s_{j,k}\}. \tag{9}$$

If $m_{i,j,k} > M_{max}$, $u_{i,j,k}$ is split into multiple demands $(u_{i\times r,j,k})$. The splitting process is described in Algorithm 1.

Algorithm 1 Splitting the customers' demands in case 1.

```
Require: $\mathcal{M}_{j,k}$, $u_{\mathcal{M}_{j,k}}$, $SP_{j,k}$.
Ensure: $u_{\mathcal{M}_{j,k}}$.
 1: for ($i \in \mathcal{M}_{j,k}$) do
 2:   $r \leftarrow 0$;
 3:   $Start.u_{i,j,k} \leftarrow \infty$;
 4:   while ($m_{i,j,k} > M_{max}$) do
 5:     $r \leftarrow r+1$;
 6:     $u_{i\times r,j,k} \leftarrow \langle M_{max}, t_{i,j,k}, d_{i,j,k}\rangle$;
 7:     $Start.u_{i\times r,j,k} \leftarrow Start.u_{i,j,k}$;
 8:     $m_{i,j,k} \leftarrow m_{i,j,k} - M_{max}$;
 9:   $u_{i\times(r+1),j,k} \leftarrow m_{i,j,k}, t_{i,j,k}, d_{i,j,k}\rangle$;
10: return $u_{\mathcal{M}_{j,k}}$.
```

Considering that the splitting of customer's demand does not affect the customer's resource usage, we need to ensure that the service resources from different coalitions will be delivered to the customer i at the same time. Therefore, we add a start time constraint for each customer and the start time is initially set to ∞. Once one of the split demands $u_{i\times r,j,k}$ of customer i is scheduled to a certain position, the start time of the remaining split demands of the customer i coincides with the start time of $u_{i\times r,j,k}$.

Case 2: In the case of cooperative resource procurement, there are still some available spaces in the program $SP^s_{j,k}$ after the current optimal program $SP^s_{j,k}$ and the corresponding coalition $S_{l,j,k}$ are selected. In fact, the resource utilization of $SP^s_{j,k}$ reaches a pseudo optimal. As shown in Fig. 5, we first obtain the pseudo-optimal coalition strategy $\mathcal{S}_{j,k}$ without splitting the demands of customers, and $RP_{S_{1,j,k}} \le RP_{S_{2,j,k}} \le \cdots \le P^0_{j,k}$. The selected programs of the coalition $S_{1,j,k}$ with the lowest real price has some spaces available for other customers to place a part of their resource demands. For each space in $S_{1,j,k}$, the position of the space can be represented as $space = \langle m'_l, m'_r, t'_l, t'_r\rangle$, where (m'_l,t'_l) and (m'_r,t'_r) are the coordinates of the lower-left corner and top-right corner of the space, respectively. If the customer i $(i \notin S_{1,j,k})$ wants to join $S_{1,j,k}$ by splitting his resource demand, $t_{i,j,k}$ and $d_{i,j,k}$ should satisfy $t_{i,j,k} \le t'_r - t'_l$ and $t'_l + t_{i,j,k} \le d_{i,j,k}$. $u_{i\times1,j,k} = \langle m'_r - m'_l, t_{i,j,k}, d_{i,j,k}\rangle$ and $u_{i\times2,j,k} = \langle m_{i,j,k}-(m'_r-m'_l), t_{i,j,k}, d_{i,j,k}\rangle$. If $cost(u_{i\times1,j,k}) + cost(u_{i\times2,j,k}) < cost(u_{i,j,k})$, then the demand $u_{i,j,k}$ is split to $u_{i\times1,j,k}$ and $u_{i\times2,j,k}$. $S_{1,j,k}$ and $u_{S_{1,j,k}}$ are updated, and $S_{1,j,k} \leftarrow S_{1,j,k}\cup\{i\}$ and $u_{S_{1,j,k}} \leftarrow u_{S_{1,j,k}} \cup u_{i\times1,j,k}$. In the splitting process, if there are multiple customers competing for a space, we choose the customer with the largest ratio of $\frac{m_{i\times1,j,k}}{m_{i,j,k}}$. If no customer joins the coalition $S_{1,j,k}$, the resource utilization of the program selected by the coalition $S_{1,j,k}$ reaches the maximum value. Then, $u_{\mathcal{M}_{j,k}}$ and $\mathcal{M}_{j,k}$ are updated by removing the demands $u_{S_{1,j,k}}$. The updated $u_{\mathcal{M}_{j,k}}$ and $\mathcal{M}_{j,k}$ participate in the next round of calculations until $u_{\mathcal{M}_{j,k}} = \emptyset$ and $\mathcal{M}_{j,k} = \emptyset$.

5.1.2. The PMRUD algorithm

First, we consider the situation where customers' demands are not split in the process of multi-customer cooperation. We propose a greedy algorithm based on the above concepts and strategies. Packing the $(q+1)$th demand with deadline (q) is denoted as PCD$(\cdot)$ and q indicates that there are already q demands placed in $SP^s_{j,k}$.

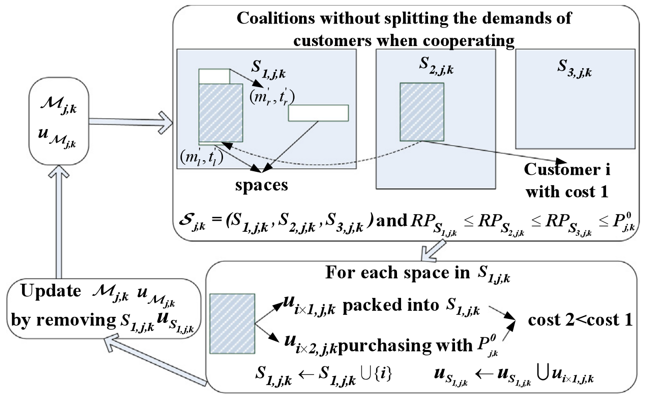

Coalitions without splitting the demands of customers when cooperating
$S_{1,j,k}$
$S_{2,j,k}$
$S_{3,j,k}$
(m_r', t_r')
(m_l, t_l)
spaces
Customer i with cost 1
$\mathcal{M}_{j,k}$
$u_{\mathcal{M}_{j,k}}$
$\mathcal{S}_{j,k} = (S_{1,j,k}, S_{2,j,k}, S_{3,j,k})$ and $RP_{S_{1,j,k}} \le RP_{S_{2,j,k}} \le RP_{S_{3,j,k}} \le P^0_{j,k}$
For each space in $S_{1,j,k}$
$u_{i\times 1,j,k}$ packed into $S_{1,j,k}$
$u_{i\times 2,j,k}$ purchasing with $P^0_{j,k}$
cost 2<cost 1
$S_{1,j,k} \leftarrow S_{1,j,k} \cup \{i\}$
$u_{S_{1,j,k}} \leftarrow u_{S_{1,j,k}} \cup u_{i\times 1,j,k}$
Update $\mathcal{M}_{j,k}$ $u_{\mathcal{M}_{j,k}}$ by removing $S_{1,j,k}$ $u_{S_{1,j,k}}$

Fig. 5. Example of splitting the number of resources requested by customer.

Algorithm 2 PCD(·) algorithm.

```
Require: $\mathcal{M}$, $u_{\mathcal{M}_{j,k}}$, $SP^s_{j,k}$, $q$, Configuration(q), $S_{lq}$.
Ensure: $U_{SP^s_{j,k}}$, $S_{l,j,k}$, Configuration$^s_{S_{l,j,k}}$.
 1: while $SC \neq \emptyset$ and $q < |u_{\mathcal{M}_{j,k}}|$ do
 2:   Minc← ∞; $u'_{\mathcal{M}_{j,k}} \leftarrow u_{\mathcal{M}_{j,k}} \setminus u_{S_{lq}}$;
 3:   for every $(m, t, Act, u_{i\times r,j,k}) \in SC$ $(u_{i\times r,j,k} \in u'_{\mathcal{M}_{j,k}})$ do
 4:     Calculate the caving degree $C_i$ based on Eqs. (6) and (7);
 5:     if (Minc> $C_i$) then
 6:       $SC' \leftarrow \{(m, t, Act, u_{i\times r,j,k})\}$;
 7:     else
 8:       if (Minc= $C_i$) then
 9:         $SC' \leftarrow SC' \cup \{(m, t, Act, u_{i\times r,j,k})\}$;
10:   if $|SC'| > 1$ then
11:     Select a single COA according to the selecting order (1) to
        (2);
12:   if $(m, t, Act, u_{i\times r,j,k})$ is the selected COA then
13:     $S_{lq} \leftarrow S_{lq} \cup \{i\}$;
14:     Configuration$(q+1)$← Configuration(q)∪ Pos($u_{i\times r,j,k}$);
15:     $q \leftarrow q + 1$; Update the available SC;
16: Calculate the resource utilization $U_{SP^s_{j,k}}$;
17: return $U_{SP^s_{j,k}}$, $S_{l,j,k}$, Configuration$^s_{S_{l,j,k}}$.
```

In Algorithm 2, q indicates that all q demands in S_{lq} ($S_{lq} \in \mathcal{M}_{j,k}$) have been packed into the service program $SP^s_{j,k}$, and the current position status is represented by Configuration (q). The COAs with the largest caving degree are recorded in the set SC and the optimal COA is selected according to the selection order in Section 5.1.1. When there are no available COAs or all customers in $\mathcal{M}_{j,k}$ have been positioned into $SP^s_{j,k}$, the algorithm loop stops. Then, the resource utilization of $SP^s_{j,k}$ is calculated.

We know that the resource utilization obtained by Algorithm PCD(·) is not globally optimal. Based on Algorithm PCD(·), we use the depth-first method (backtracking) to propose Algorithm 3-Pseudo Maximal Resource Utilization with a Deadline, which is denoted as PMRUD.

Algorithm PMRUD ensures that in the process of selecting customers at each step, each selected customer obtained by the *while* loop is optimal in the current situation. As shown in Fig. 6, in each *while* loop, every available COA tries to be placed in $SP^s_{j,k}$, and Algorithm PCD(.) is called to calculate resource utilization $U_{SP^s_{j,k}}$. Subsequently, the selected COA is removed of the list. In addition, the algorithm selects the COA set with the highest resource utilization, then selects the optimal COA according to the selection order, and places the corresponding customer into $SP^s_{j,k}$. Finally, configuration$^s_{S_{l,j,k}}$ and COA set SC are updated.

Algorithm 3 PMRUD algorithm.

```
Require: $\mathcal{M}$, $u_{\mathcal{M}_{j,k}}$.
Ensure: $U_{SP^s_{j,k}}$, $S_{l,j,k}$, Configuration$^s_{S_{l,j,k}}$.
 1: $q \leftarrow 0$; Configuration$(q) \leftarrow \emptyset$; $S_{l,j,k} \leftarrow \emptyset$;
 2: while $SC \neq \emptyset$ and $q < |u_{\mathcal{M}_{j,k}}|$ do
 3:   $\hat{S}_{l,j,k} \leftarrow S_{l,j,k}$; $\hat{SC} \leftarrow SC$; $SC'' \leftarrow \emptyset$;
 4:   Configuration← Configuration(q); MaxU← 0;
 5:   for each $(m, t, Act, u_{i\times r,j,k}) \in SC$ do
 6:     $\hat{q} \leftarrow q + 1$; Configuration(q) ← Configuration(q)∪
        Pos($u_{i\times r,j,k}$); Update SC;
 7:     Call PCD($\hat{q}$);
 8:     if $(MaxU < U_{SP^s_{j,k}})$ then
 9:       $SC'' \leftarrow \{(m, t, Act, u_{i\times r,j,k})\}$;
10:     else
11:       if $(MaxU = U_{SP^s_{j,k}})$ then
12:         $SC'' \leftarrow SC'' \cup \{(m, t, Act, u_{i\times r,j,k})\}$;
13:   if $|SC''| > 1$ then
14:     Select a single COA according to the selecting order (1) to
        (2);
15:   if $(m, t, Act, u_{i\times r,j,k})$ is the selected COA then
16:     $S_{l,j,k} \leftarrow \hat{S}_{l,j,k} \cup \{i\}$;
17:     $q \leftarrow q + 1$;
18:     Configuration(q)← Configuration ∪ Pos($u_{i,j,k}$);
19:     Update the available SC;
20: Calculate the resource utilization $U_{SP^s_{j,k}}$;
21: return $U_{SP^s_{j,k}}$, $S_{l,j,k}$, Configuration$^s_{S_{l,j,k}}$.
```

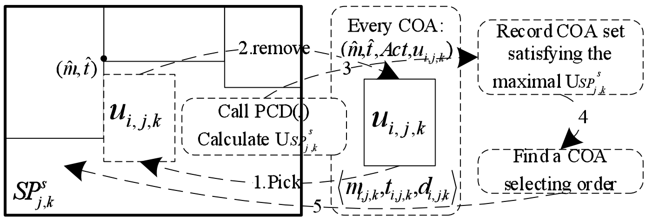

$(\hat{m}, \hat{t})$
$u_{i,j,k}$
$SP^s_{j,k}$
2.remove
3
Every COA: $(\hat{m}, \hat{t}, Act, u_{i,j,k})$
Call PCD()
Calculate $U_{SP^s_{j,k}}$
$u_{i,j,k}$
1.Pick
$(m_{i,j,k}, t_{i,j,k}, d_{i,j,k})$
5
Record COA set satisfying the maximal $U_{SP^s_{j,k}}$
4
Find a COA selecting order

Fig. 6. Illustration of Algorithm PMRUD.

5.2. The DRCA algorithm

In this section, we propose the heuristic Deadline-constrained Resource Coalition Allocation (DRCA) algorithm. The detailed steps of the DRCA algorithm are shown in Algorithm 4.

First, $u_{\mathcal{M}_{j,k}}$ is updated by splitting the customer demands according to Case 1 in Section 5.1.1. The inner *while* loop from line 5 to line 21 obtains the pseudo-optimal coalitions for the current customer set $\mathcal{M}_{j,k}$ and the corresponding demands $u_{\mathcal{M}_{j,k}}$. In each update of the outer *while* loop, $u_{\mathcal{M}_{j,k}}$ is split again according to Case 2 in Section 5.1.1 in order to obtain the lth optimal coalition $S_{l,j,k}$ and the corresponding $SP^{\hat{h}_l}_{j,k}$. If no customer in the current coalition can find another coalition to reduce costs, then this coalition is the best choice for all customers involved in the coalition. Additionally, the resource utilization of the service program is maximized, which means that no suitable customers can join the coalition. The remaining demand $m_{i,j,k}$ of each customer i in $S_{l,j,k}$ is updated, and the set $\ddot{S}$ is denoted as a set of customers satisfying $m_{i,j,k} \neq 0$, from line 24 to 27. As mentioned above, the customers are selfish, so the coalition $S_{l,j,k} \setminus \ddot{S}$ will leave the customer set $\mathcal{M}_{j,k}$, and $\mathcal{M}_{j,k}$ will be updated to $\mathcal{M}_{j,k} = (\mathcal{M}_{j,k} \setminus S_{l,j,k}) \bigcup \ddot{S}$. $u_{\mathcal{M}_{j,k}}$ is also updated to $u_{\mathcal{M}_{j,k}} \setminus u_{S_{l,j,k}}$. Algorithm DRCA repeats until $\mathcal{M}_{j,k}$ and $u_{\mathcal{M}_{j,k}}$ are empty sets.

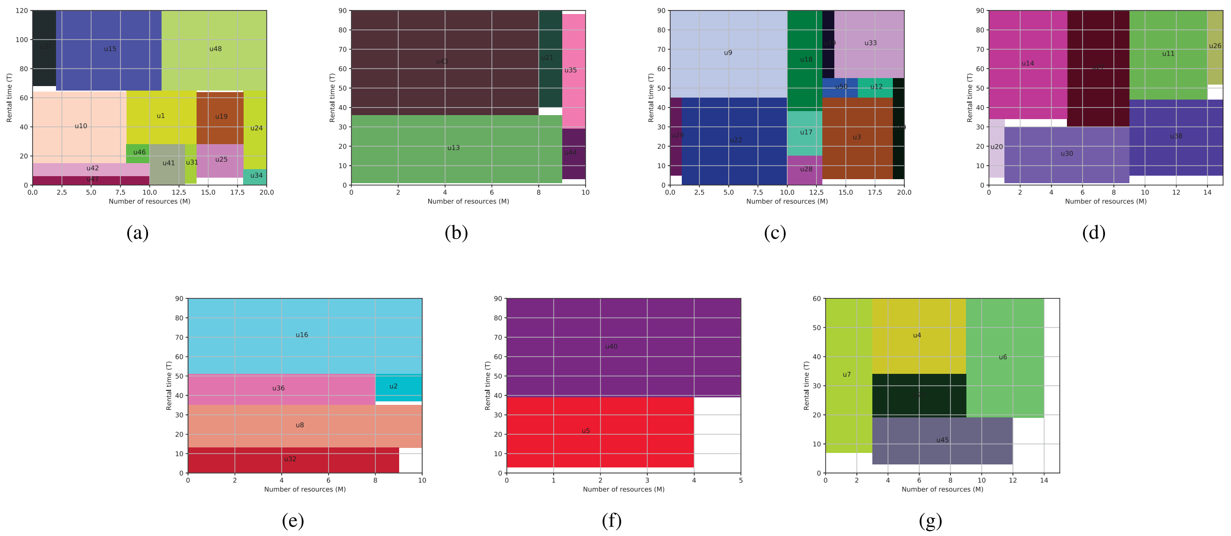


Fig. 7. Optimal customer cooperation configuration for given service programs.

We know that M and N are the number of customers and providers, respectively. We set $K = \max_{j\in\mathcal{N}}\{K_j\}$ and $H = \max_{j\in\mathcal{N},k\in K_j}\{H_{j,k}\}$. In Algorithm 1, we set $m_{max} = \max_{i\in\mathcal{M}_{j,k}} m_i$, then the computational overhead of Algorithm 1 is $O(M\frac{m_{max}}{M_{max}}) = O(M)$. In Algorithm 2, q customers have been scheduled, and there are $|u_{\mathcal{M}_{j,k}}| - q$ customers left to be scheduled. In each process of selecting an appropriate customer, the required update time is $O(b)$. Thus, the computational overheads of Algorithm 2 is $O((|u_{\mathcal{M}_{j,k}}|-q)b)$. Since the out-of-loop consumption is the same in Algorithms 3 and 4, every *while* loop in Algorithm 3 (the PMRUD algorithm) except calling PCD($\hat{q}$) is represented as $O(c)$, and the *for* loop in Algorithm 4 (the DRCA algorithm) except calling PMRUD is $O(e)$. Then, the computation overheads of Algorithm 3 (PMRUD) in the worst case is $O(\frac{M(M+1)}{2}b + Mc) = O(M^2b)$. Furthermore, the time overheads of the inner *while* loop in Algorithm 4 is $O(\frac{M(M+1)(2M+1)}{6}Hb + Me) = O(M^3Hb)$ and the computational overheads of Algorithm 4 is $O(\frac{[M(M+1)]^2}{2}Hb) = O(M^4Hb)$. At last, the overheads of the whole system is $O(M^4HKNb)$.

6. Experiments

6.1. Experiment setup

To validate the feasibility and effectiveness of our DRCA algorithm, we have performed extensive simulations and experiments. Our simulation environment is an Intel(R) core(TM) i5-6200U 2cores CPU@2.3 GHz 2.4 GHz with Windows version 10. The simulation program is written in Python supported by the Pycharm 2017.3.2 IDE. We consider the general-purposed computing service research ECS provided by the Alibaba company [9]. The product configuration of the service programs is listed in Table 2 in Section 3.1. The product has two types of payment prices, annual or monthly and on demand, as listed in Table 5(a). If a customer rents the service on a monthly basis for less than one year, the price of 1 $\sim$ 2, 3 $\sim$ 5, and 6 $\sim$ 9 months are 1039, 937, and 886 per server per month, respectively. If the customer rents on an annual basis, the price is 824.25 for the first year, 661 for two years, and 494 for three years. However, if the customer rents the service for less than one month, the customer's service price on demand is 2619.36 (87.312 per day). To highlight the applicability of our algorithm, we assume that more product service programs are based on Table 5(a). As given in Table 5(b), we have added preferential strategies for renting multiple cloud instance programs. The first column represents the number of cloud instances, and the first row indicates the renting time (unit of month). The number in row i and column j ($i, j \neq 1$) represents the total cost of corresponding program with renting M instances and T months.

Table 5
Resource dataset from Alibaba Cloud used in the experiments.

(a) The price of the service program, unit: per service per day			
	1∼2 months	3∼5 months	6∼9 months
Annual or monthly	1039	937	886
On demand	2619.36	2619.36	2619.36
	1 year	2 years	3 years
Annual or monthly	824.25	661	494
On demand	2619.36	2619.36	2619.36

(b) The prices of programs						
M\T (month)	1	2	3	6	12	...
5	5195	10 390	14 055	26 580	49 455	
10	10 390	20 780	28 110	53 160	98 910	
15	15 585	31 170	42 165	79 740	148 365	
20	20 780	41 560	56 220	106 320	197 820	
...	...	...	...	...	...	...

6.2. Customer cost analysis

We select 50 customers and randomly generate the demands for each customer, where the number of resources is gradually set from 1 to 10, the rental time is gradually set from 1 to 60, and the deadline ranges from the value of the corresponding rental time to 180.

Fig. 7 shows the selected service programs and the configuration of each customer. The blank area in a service program indicates that the server is idle. Table 6 lists the resource utilization of each selected service program and the real price of each customer in the selected service program. In addition, Fig. 8 shows a comparison of the cost of purchasing service resources individually and the cost each customer participates in a coalition to purchase them, the values of which are represented by the orange and blue marks, respectively. The cost of each customer is reduced and the reduction ratio can reach as much as 63.6%, which indicates the feasibility of the proposed mechanism.

Algorithm 4 Heuristic DRCA algorithm.

```
Require: $\mathcal{M}_{j,k}$, $u_{\mathcal{M}_{j,k}}$, $\mathbf{SP}_{j,k}$.
Ensure: $\mathcal{S}_{j,k}$, $x_{\mathcal{S}_{j,k}}$.
1: Update $u_{\mathcal{M}_{j,k}}$ by splitting the demands of customers according
   to case 1 (by calling Algorithm 1 in Section 5.1.1);
2: $l \leftarrow 0$;
3: while $u_{\mathcal{M}_{j,k}} \neq \emptyset$ do
4:   $l \leftarrow l+1$; $\check{S}_l \leftarrow \emptyset$; $\ddot{S} \leftarrow \emptyset$; $w \leftarrow 0$; $\tilde{u}_{\mathcal{M}_{j,k}} \leftarrow u_{\mathcal{M}_{j,k}}$;
     $s \leftarrow |\mathbf{SP}_{j,k}|$; $\tilde{\mathcal{M}}_{j,k} \leftarrow \mathcal{M}_{j,k}$;
5:   while $\tilde{u}_{\mathcal{M}_{j,k}} \neq \emptyset$ do
6:     $w \leftarrow w+1$; $\bar{S} \leftarrow \emptyset$;
7:     for $s$ from $|\mathbf{SP}_{j,k}|$ to 1 do
8:       $minp \leftarrow P^0_{j,k}$; $h_w \leftarrow \infty$, $\tilde{S} \leftarrow \emptyset$;
9:       Find $S_w \subseteq \mathcal{M}_{j,k}$ for service programs $SP^s_{j,k}$ such that
         the resource utilization $U^s_{S_w}$ is maximized (by calling
         PMRUD algorithm in Section 5.1.2);
10:      if ($RP^s_{S_w} < minp$) then
11:        $minp \leftarrow RP^s_S$; $h_w \leftarrow s$, $\tilde{S} \leftarrow S$;
12:    if ($minp < P^0_{j,k}$) then
13:      $S_w \leftarrow \tilde{S}$; $x^{h_w}_{S_w} \leftarrow 1$;
14:    else
15:      if ($minp = P^0_{j,k}$) then
16:        $S_w \leftarrow \tilde{\mathcal{M}}_{j,k}$, $h_w \leftarrow 0$;
17:    for ($i \in S_w$) do
18:      $\tilde{m}_{i,j,k} \leftarrow \tilde{m}_{i,j,k} - m^{h_w}_{i,S_w}$;
19:      if ($\tilde{m}_{i,j,k} \neq 0$) then
20:        $\bar{S} \leftarrow \bar{S} \bigcup \{i\}$;
21:    $\tilde{\mathcal{M}}_{j,k} \leftarrow (\tilde{\mathcal{M}}_{j,k} \backslash S_w) \bigcup \bar{S}$, $\tilde{u}_{\mathcal{M}_{j,k}} \leftarrow \tilde{u}_{\mathcal{M}_{j,k}} \setminus u_{S_w}$;
22:  Split $u_{\mathcal{M}_{j,k}}$ according to case 2 and obtain the split customer
     set $\check{S}_l$ and demand set $u_{\check{S}_l}$ (Section 5.1.1);
23:  $\hat{h}_l \leftarrow h_1$,$S_{l,j,k} \leftarrow S_1 \bigcup \check{S}_l$, $u_{S_{l,j,k}} \leftarrow u_{S_1} \bigcup u_{\check{S}_l}$;
24:  for ($i \in S_{l,j,k}$) do
25:    $m_{i,j,k} \leftarrow m_{i,j,k} - m^{\hat{h}_l}_{i,S_{l,j,k}}$;
26:    if ($m_{i,j,k} \neq 0$) then
27:      $\ddot{S} \leftarrow \ddot{S} \bigcup \{i\}$;
28:  $\mathcal{M}_{j,k} \leftarrow (\mathcal{M}_{j,k} \backslash S_{l,j,k}) \bigcup \ddot{S}$, $u_{\mathcal{M}_{j,k}} \leftarrow u_{\mathcal{M}_{j,k}} \setminus u_{S_{l,j,k}}$;
29: return $\mathcal{S}_{j,k}$, $x_{\mathcal{S}_{j,k}}$.
```

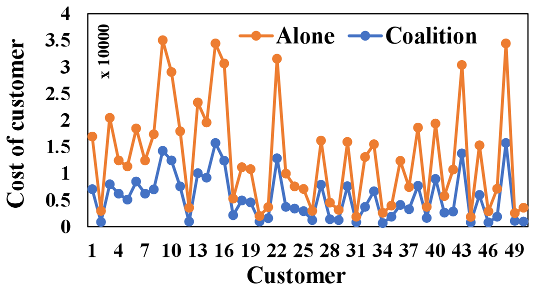


Fig. 8. Comparison of the cost of each customer. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

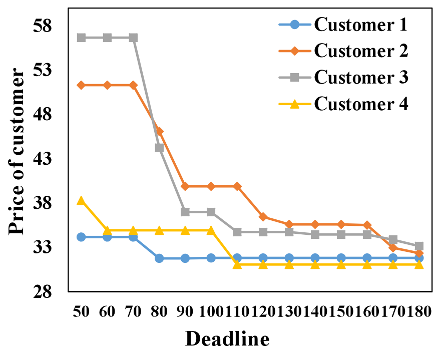
6.3. Impact of deadline on procurement price

We validate the heuristic DRCA algorithm by analyzing the price for each customer affected by the deadline. We select four customers from 100 customers. They are customer 1, 2, 3, and 4. Each customer has a deadline from 50 to 180 with an increment of 10. In addition, when analyzing the impact of the deadline on a customer, the procurement demands and deadlines of other customers remain unchanged.

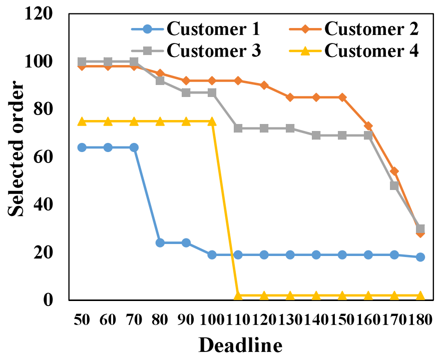
Table 6
Resource utilization of each selected service program.

Program	U	RP	Program	U	RP
(a) 20 × 120	0.98375	31.749	(e) 10 × 90	0.981	31.835
(b) 10 × 120	0.98	31.871	(f) 5 × 90	0.886	35.226
(c) 20 × 90	0.985	31.691	(g) 15 × 60	0.837	41.340
(d) 15 × 90	0.951	32.839	On demand		87.312

Program: $M \times T(day)$, RP: per service per day.



(a)



(b)

Fig. 9. Impact of deadline on customer price.

Fig. 9(a) shows the impact of the deadline constraints on the purchase price, and Fig. 9(b) shows the order of the corresponding customers are selected in the coalition game. As the deadline increases, the real price for each customer shows a downward trend. As shown in Fig. 9(b), the larger the customer's deadline, the easier it is for the customer to associate with others. As a result, we can know that the customers with farther deadline are more likely arranged with high priority and lower price, but their start times of using resource are late, which also provides convenience for the customers with short deadline (urgent customers) to use resources as early as possible.

6.4. Effectiveness and performance evaluation

6.4.1. Effectiveness evaluation

To illustrate how the scale of the customers and service programs affects the entire system, we analyze the time overheads and the percentage of total customer cooperative savings. The number of customers M increases from 100 to 7000 in increments of 100, and the number of programs ranges from 10 to 450 with an increment of 10. The experiment results are presented in Figs. 10 and 11.

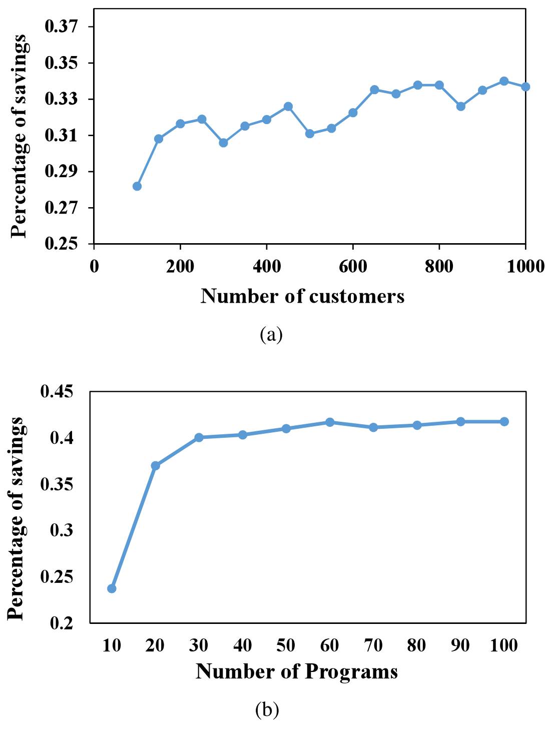


Fig. 10. Effectiveness of the DRCA algorithm.

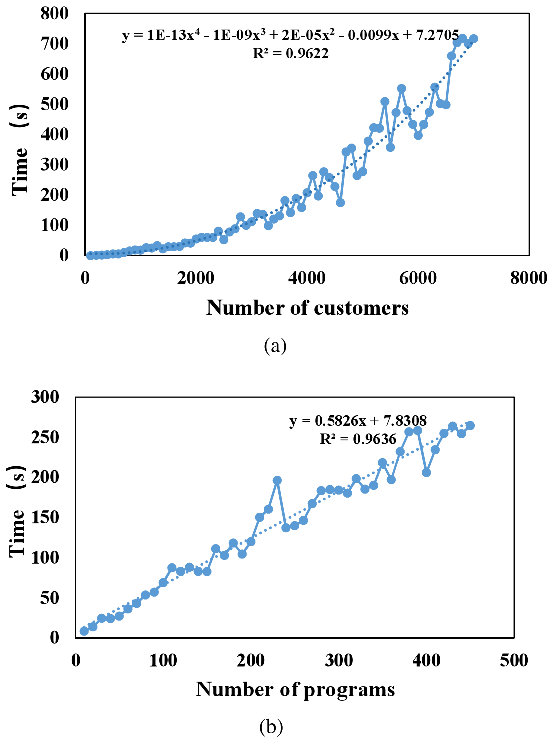


Fig. 11. Time overheads of customers and service programs.

Given 12 service programs, Fig. 10(a) shows the percentage curve of customer cooperative savings as the number of customers increases. Fig. 10(b) shows the effect of the number of programs on the savings for a given 1000 customers. As the number of customers increases, the trend of the line increases and reaches a value of 34%, and the curve in Fig. 10(b) increases to a relatively stable value of 42%. Fig. 10 shows the effectiveness and practicality of our algorithm to help customers reduce their costs. The more customers and programs, the higher the overall percentage of customer cooperative savings.

6.4.2. *Performance evaluation*

In the experiments, we adjust the number of customers and service programs and record the execution time of the entire system. Fig. 11(a) and (b) show the execution time curves of the system as the number of customers and programs increases, respectively.

In Fig. 11(a), as the scale of customers increases, the time curve increases in a polynomial. The blue dashed line is the trend line of the time curve, which is a fourth-order polynomial. The fitting degree of the trend line and the time curve is 0.9622. As shown in Fig. 11(b), the time curve of the entire system increases linearly. The dashed line is the trend line of the curve with a fitness of 0.9636. Fig. 11(a) and (b) verify the time overhead we evaluated in Section 5.2. The system can determine the time interval $\triangle T$ based on the size of customers, providers, and programs.

7. Conclusions

Our study focused on the problem of using multi-customer collaboration strategies to configure appropriate resources to meet the needs of each customer, thereby minimizing customer costs. We established a coalition game system based on three coalition rules and proved that there is a unique and optimal solution in the coalition game that satisfies individual stability and group stability. The optimal solution proved to be equal to the system where the selected service program can reduce the costs of customers in their own coalitions and achieve maximum resource utilization. A heuristic DRCA algorithm was proposed to calculate the optimal solution. Extensive experiments were simulated to validate the feasibility and effectiveness of the proposed algorithm.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The authors are grateful to the anonymous reviewers for their constructive comments. The research was partially funded by the National Key R&D Program of China (Grant No. 2018YFB0203800), the Program of National Natural Science Foundation of China (Grant No. 61876061), the National Outstanding Youth Science Program of National Natural Science Foundation of China (Grant No. 61625202).

References

[1] R. Alvarez-Valdes, J.M. Tamarit, A grasp algorithm for constrained two-dimensional non-guillotine cutting problems, J. Oper. Res. Soc. 56 (4) (2005) 414–425.
[2] K. Apt, A. Witzel, A generic approach to coalition formation, Int. Game Theory Rev. 11 (03) (2009) 347–367.
[3] B.S. Baker, E.G.J. Coffman, R.L. Rivest, Orthogonal packings in two dimensions., SIAM J. Comput. 9 (4) (1980) 846–855.
[4] G. Baranwal, D. Vidyarthi, A truthful and fair multi-attribute combinatorial reverse auction for resource procurement in cloud computing, IEEE TSC PP (99) (2016) 1.
[5] W. Borjigin, K. Ota, M. Dong, In broker we trust: A double-auction approach for resource allocation in nfv markets, IEEE Trans. Netw. Serv. Manag. 15 (4) (2018) 1322–1333.

[6] J. Cao, K. Hwang, K. Li, A.Y. Zomaya, Optimal multiserver configuration for profit maximization in cloud computing, IEEE TPDS 24 (6) (2013) 1087–1096.
[7] J. Chase, D. Niyato, Joint optimization of resource provisioning in cloud computing, IEEE TSC PP (99) (2017) 1.
[8] B. Chazelle, The bottomn-left bin-packing heuristic: An efficient implementation, IEEE TC C-32 (8) (2006) 697–707.
[9] https://tco.aliyun.com/tco/ecs/calculator.
[10] W. Huang, D. Chen, R. Xu, A new heuristic algorithm for rectangle packing, Comput. Oper. Res. 34 (11) (2007) 3270–3280.
[11] J. Joy, L.K. Kumar, Cost and deadline optimization along with resource allocation in cloud computing environment, in: International Conference on Advanced Computing and Communication Systems, 2013, pp. 1–6.
[12] M. Kesavan, I. Ahmad, O. Krieger, R. Soundararajan, A. Gavrilovska, K. Schwan, Practical compute capacity management for virtualized datacenters, IEEE TCC 1 (1) (2013) 1.
[13] C. Lee, P. Wang, D. Niyato, A real-time group auction system for efficient allocation of cloud internet applications, IEEE TSC 8 (2) (2015) 251–268.
[14] S.C.H. Leung, D. Zhang, C. Zhou, T. Wu, A hybrid simulated annealing metaheuristic algorithm for the two-dimensional knapsack packing problem, Comput. Oper. Res. 39 (1) (2012) 64–73.
[15] D. Li, C. Chen, J. Guan, Y. Zhang, J. Zhu, R. Yu, Dcloud: Deadline-aware resource allocation for cloud computing jobs, IEEE Trans. Parallel Distrib. Syst. 27 (8) (2016) 2248–2260.
[16] H. Li, K. Ota, M. Dong, A. Vasilakos, K. Nagano, Multimedia processing pricing strategy in gpu-accelerated cloud computing, IEEE Trans. Cloud Comput. (2017) 1.
[17] C. Liu, K. Li, K. Li, A framework of price bidding configurations for resource usage in cloud computing, IEEE TPDS 27 (8) (2016).
[18] C. Liu, K. Li, K. Li, Minimal cost server configuration for meeting time-varying resource demands in cloud centers, IEEE Trans. Parallel Distrib. Syst. 29 (11) (2018) 2503–2513.
[19] C. Liu, K. Li, C. Xu, K. Li, Strategy configurations of multiple users competition for cloud service reservation, IEEE Trans. Parallel Distrib. Syst. 27 (2) (2016) 508–520.
[20] Q. Llull, S. Fan, S.M. Zahedi, B.C. Lee, Cooper: Task colocation with cooperative games, in: IEEE International Symposium on High PERFORMANCE Computer Architecture, 2017, pp. 421–432.
[21] D.C. Marinescu, A. Paya, J.P. Morrison, A cloud reservation system for big data applications, IEEE TPDS 28 (3) (2017) 606–618.
[22] J. Mei, K. Li, Z. Tong, Q. Li, K. Li, Profit maximization for cloud brokers in cloud computing, IEEE Trans. Parallel Distrib. Syst. 30 (1) (2019) 190–203.
[23] D.T. Nguten, L.B. Le, V. Bhargava, Edge computing resource procurement: An online optimization approach, in: IEEE 4th World Forum on Internet of Things (WF-IoT), 2018, pp. 807–812.
[24] P.S. Pillai, S. Rao, Resource allocation in cloud computing using the uncertainty principle of game theory, IEEE Syst. J. 10 (2) (2016) 637–648.
[25] A.S. Prasad, S. Rao, A mechanism design approach to resource procurement in cloud computing, IEEE TC 63 (1) (2013) 17–30.
[26] M.A. Rodriguez, R. Buyya, Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds, IEEE Trans. Cloud Comput. 2 (2) (2014) 222–235.
[27] A. Safdarian, M. Fotuhi-Firuzabad, M. Lehtonen, F. Aminifar, Optimal electricity procurement in smart grids with autonomous distributed energy resources, IEEE Trans. Smart Grid 6 (6) (2015) 2975–2984.
[28] O. Shehory, S. Kraus, Methods for task allocation via agent coalition formation, Artificial Intelligence 101 (1–2) (1998) 165–200.
[29] W. Shi, L. Zhang, C. Wu, Z. Li, F.C.M. Lau, An online auction framework for dynamic resource provisioning in cloud computing, IEEE/ACM Trans. Netw. 24 (2016) 2060–2073.
[30] Y. Shoham, Computer science and game theory., Commun. ACM 51 (8) (2008) 74–79.
[31] K.M. Sim, Agent-based interactions and economic encounters in an intelligent intercloud, IEEE TCC 3 (3) (2015) 358–371.
[32] J. Simao, L. Veiga, Partial utility-driven scheduling for flexible sla and pricing arbitration in clouds, IEEE TCC 4 (4) (2016) 467–480.
[33] L. Tang, H. Chen, Joint pricing and capacity planning in the iaas cloud market, IEEE TCC 5 (1) (2017) 57–70.
[34] S. Tanzil, O. Gharehshiran, V. Krishnamurthy, A distributed coalition game approach to femto-cloud formation, IEEE TCC PP (99) (2016) 1.
[35] T. Truonghuu, C.K. Tham, A novel model for competition and cooperation among cloud providers, IEEE TCC 2 (3) (2014) 251–265.
[36] X. Wang, X. Wang, H. Che, K. Li, M. Huang, C. Gao, An intelligent economic approach for dynamic resource allocation in cloud services, IEEE TCC 3 (3) (2015) 275–289.
[37] J. Wang, X. Xiao, J. Wang, K. Lu, X. Deng, A.A. Gumaste, When group-buying meets cloud computing, in: INFOCOM'16, 2016, pp. 1–9.
[38] L. Wei, D. Zhang, Q. Chen, A least wasted first heuristic algorithm for the rectangular packing problem, Comput. Oper. Res. 36 (5) (2009) 1608–1614.
[39] S. Zaman, D. Grosu, Combinatorial auction-based allocation of virtual machine instances in clouds, in: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, Indianapolis, 2010, pp. 127–134.
[40] Z. Zhou, M. Dong, K. Ota, G. Wang, L.T. Yang, Energy-efficient resource allocation for d2d communications underlaying cloud-ran-based lte-a networks, IEEE Internet Things J. 3 (3) (2016) 428–438.

Junyan Hu received the B.S. degree in mathematics and applied mathematics from Hunan University, China, in 2014. She is currently working toward the Ph.D. degree in computer science and technology at Hunan University, China. Her research interests are mainly in game theory, grid computing, tensor analysis, cloud and edge computing. She has published 2 papers in journals such as IEEE Transactions on Services Computing, ACM Transactions on Embedded Computing Systems. She won the Best Paper Award in NPC 2019.

Kenli Li received the Ph.D. degree in computer science from Huazhong University of Science and Technology, China, in 2003. He was a visiting scholar with the University of Illinois at Urbana–Champaign from 2004 to 2005. He is currently the dean and a full professor of computer science and technology with Hunan University and deputy director of National Supercomputing Center in Changsha. His major research areas include parallel computing, high performance computing, grid, and cloud computing. He has published more than 320 research papers in international conferences and journals such as the IEEE Transactions on Computers, the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Signal Processing, the Journal of Parallel and Distributed Computing, ICPP, and CCGrid. He serves on the editorial board of the IEEE Transactions on Computers. He is an outstanding member of the CCF. He is a senior member of the IEEE.

Chubo Liu received the B.S. degree and the Ph.D. degree in computer science and technology from Hunan University, China, in 2011 and 2016, respectively. He is currently an associate professor of computer science and technology with Hunan University. His research interests are mainly in game theory, approximation and randomized algorithms, cloud and edge computing. He has published over 20 papers in journals and conferences such as the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Cloud Computing, IEEE Internet of Things Journal, and the Theoretical Computer Science, etc. He won the Best Paper Award in NPC 2019 and the IEEE Technical Committee on Scalable Computing Early Career Researcher (ECR) Award in 2019. He is a member of IEEE and CCF.

Jianguo Chen received the Ph.D. degree from College of Computer Science and Electronic Engineering at Hunan University, China. He was a visiting Ph.D. student at the University of Illinois at Chicago from 2017 to 2018. He is currently a postdoctoral in University of Toronto and Hunan University. His major research areas include parallel computing, cloud computing, machine learning, data mining, bioinformatics and big data.

Keqin Li is a SUNY Distinguished Professor of computer science with the State University of New York. His current research interests include cloud computing, fog computing and mobile edge computing, energy efficient computing and communication, embedded systems and cyberphysical systems, heterogeneous computing systems, big data computing, high-performance computing, CPU–GPU hybrid and cooperative computing, computer architectures and systems, computer networking, machine learning, intelligent and soft computing. He has published over 750 journal articles, book chapters, and refereed conference papers, and has received several best paper awards. He has served on the editorial boards of the IEEE Transactions on Parallel and Distributed Systems, the IEEE Transactions on Computers, the IEEE Transactions on Cloud Computing, the IEEE Transactions on Services Computing, and the IEEE Transactions on Sustainable Computing. He is an IEEE Fellow.