Cost Performance Driven Multi-request Allocation in D2D Service Provision Systems

Dandan Li[1], Hongyue Wu[1], Shizhan Chen[1](✉), Lei Dong[1], Zhuofeng Zhao[2], and Zhiyong Feng[1]

[1] College of Intelligence and Computing, Tianjin University, Tianjin, China
{lddan,hongyue.wu,shizhan,2118218002,zyfeng}@tju.edu.cn

[2] Beijing Key Laboratory on Integration and Analysis of Large-Scale Stream Data, North China University of Technology, Beijing, China
edzhao@ncut.edu.cn

Abstract. Device-to-Device (D2D) communication has emerged as a promising technique to cope with the increasing heavy traffic in mobile networks. A critical problem in D2D service is request allocation, which aims to find the best provider for each of the proposed service requests. Most of the existing work focuses on optimizing the communication resource allocation, such as interference management, spectrum allocation, etc. In this paper, we originally address the request allocation problem with the object of maximizing the cost performance of requests. Moreover, we especially consider the impact of multi-service interactions on the service quality in a feasible plan for the provider. To solve this problem, we propose a combinatorial auction-based request allocation model. Furthermore, and develop a pruning-based request allocation algorithm called *RABP* to maximize the overall cost performance of requests. Extensive simulation results demonstrate that *RABP* performs well in improving the cost performance and is conducive to enhancing the load balancing among mobile devices.

Keywords: D2D communication · Request allocation · Cost performance · Combinatorial auction · Multi-service inter-impact

1 Introduction

Over the last few years, the explosive growth of mobile computing applications as well as the ever-improving requirements of users have posed tremendous challenges to current cellular network architecture [1]. According to Ericsson Mobility Report [2], the total mobile network traffic is forecast to exceed 300EB per month by 2026. 5G networks will carry more than half of the world's smartphone traffic. Although edge computing and fog computing have been proposed to share the core network's burden at the edge of the network, in the 5G era, the exponential growth of data traffic in mobile networks has made the spectrum resources of the mobile network go short. As a SPECTRUM-AND-ENERGY efficiency technology, device-to-device (D2D) communication has been widely applied in mobile computing. Under a certain level of interference, D2D enables two devices to reuse the spectrum of cellular networks and connects proximity devices directly with each other [3], which brings low translation delay and better quality of service. Thus, D2D is expected to be a key enabling technology supported by the next-generation cellular networks.

According to the Cisco Visual Networking Index projection [4], video services are expected to account for 79% of the total internet traffic by 2022. Thus a representative example of D2D service is video stream service in edge computing. When a video goes viral over the internet, a large number of mobile users make requests for it. This creates immense pressure on the edge. To improve the response speed of the edge, D2D service is utilized to provide higher QoS for demanders, particularly those who are experiencing poor cellular conditions.

Existing surveys on D2D communication largely focus on interference management and power control, such as [5–7]. Another research hotspot is D2D-enabled data traffic offloading. Most of them discuss how to assign tasks to minimize communication delay or energy consumption, such as [8–11]. Part of the research focuses on the D2D content sharing [12–14], aiming to achieve optimal or stable matching between D2D requesters and providers based on the preferences of both sides. Furthermore, some researchers proposed effective incentive mechanisms to guarantee the provider's revenue in the process of data transaction [15–17].

However, two issues have not been addressed in the existing literature. One is that current surveys on D2D mainly focus on the optimization of communication latency or energy while neglecting differences in service quality and price among different providers. However, device performance plays an important role in the D2D service quality. Consider such a scenario, the requester issues a computationally intensive task, as not much data needs to be transferred, the requester would prefer to choose a provider with strong computing power over one with faster communication speed but restricted computing resources if both of them can establish a stable D2D link. Furthermore, the service price is also a vital metric for matching between requesters and providers. The other one is that the impact of multi-service interactions on service quality is not considered when the provider handles many requests at once. As providers are typically resource-constrained mobile users, performing more requests will dramatically increase competition for system resources, thus increasing system load and resulting in service quality degradation. For ease of expression, we name the impact of multi-service interactions on service quality as multi-service inter-impact.

To address the above problems, we originally concentrate on optimizing the quality as well as the price of service on all D2D requests. In detail, we refer to the ratio of service quality to the price of a request as cost performance. We describe multi-service inter-impact in a feasible plan from the existing literature on request scheduling and pricing. To fully express the cost performance of different feasible plans for a provider, we adopt the combinatorial auction based on improved XOR language. We devise a request allocation algorithm *RABP* to obtain the optimal result. The algorithm consists of two steps, firstly searching for all feasible bid sets, then pruning out the candidate strategies that are concluded not to be the optimal solutions. The simulation results demonstrate that the proposed method significantly outperforms baseline methods. The main contributions of this paper are summarized as follows:

1) Taking into account the multi-service inter-impact, we formulate the request allocation problem and propose a combinatorial auction model based on improved XOR language.
2) We design an algorithm $RABP$ to maximize the overall cost performance of all requests, which can obtain the optimal result.
3) Simulation results demonstrate the proposed algorithm performs better than baseline methods in different scenarios. Additionally, the $RABP$ is conducive to enhancing the load balancing among devices with the increase of the service deployment rate.

The rest of this paper is organized as follows: Sect. 2 reviews the related work. The system model and problem formulation are presented in Sect. 3. The request allocation algorithm is elaborated in Sect. 4. Section 5 and Sect. 6 give the simulation results and conclusions, respectively.

2 Related Work

As one of the key technologies to expand the network capacity, D2D communication has attracted a lot of attention. A number of papers concentrate on the optimization of interference coordination and system throughput. Based on this, different request allocation schemes have been proposed.

Some researchers adopted graph theory to complete the request allocation problem. For example, [3] aimed to maximize the accessed D2D links while minimizing the total power consumption, then modeled the D2D pairing problem as a min-cost max-flow problem and solved it by the Ford-Fulkerson algorithm. [18] studied the problem of maximizing cellular traffic offloading in D2D communication. The author formulated the maximal matching problem in a bipartite graph, proposing a distributed matching algorithm. [19] originally proposed a joint user-relay selection with load balancing schemes, adopting the Kuhn-Munkres algorithm to maximize the overall matching utility.

Some authors addressed stable matching of D2D requesters and providers by using algorithms from matching theory. For example, to encourage data forwarding among cooperative users, [14] modeled the relay nodes selection problem as a stable matching problem and solved it by modified deferred acceptance algorithm. Similarly, [12] proposed a two-sided physical-social-aware provider-demander scheme for matching requesters and providers, and developed a distributed algorithm based on the Dinkelbach iteration and deferred acceptance approach. In [20], the energy-efficient resource allocation between cellular users and D2D users was constructed as a one-to-one matching problem, then the GS algorithm was applied to the energy efficiency optimization scheme.

Many auction models have emerged to solve the incentive problem for D2D providers. [16] designed Rado, a randomized auction mechanism, to provide incentives for users to participate in D2D content sharing. [21] presented a truthful double auction-based model (TAD) to reward the D2D sellers. The auction model that was applied to allocate requests in D2D mainly aimed to improve the spectrum efficiency or control the power consumption. For example, [22] presented an iterative combinatorial auction mechanism to allocate spectrum resources to optimize the system sum rate over the resource sharing of both D2D and cellular modes. [23] proposed a multi-round combinatorial double auction (MCDA) algorithm to optimize the energy efficiency over the resource allocation in D2D.

As most of the research on request allocation aimed to optimize the total utility of communication resources, we focus on the assignment between requests and providers according to the service cost performance. We originally formulate the request allocation problem as an overall cost performance maximization problem and develop an optimal algorithm based on an improved XOR-language combinatorial auction scheme. Beyond that, we take full consideration of the resource constraints of mobile devices.

In cloud computing, some researchers considered the service correlations, which means the service quality and price is not just dependent on itself but also on other services being provided by the cloud. For example, [24] presented a new cooperative coevolutionary approach for dynamic service selection with inter-service correlations. [25] proposed an extended service model which considered the correlation in the service composition process as well as the service matching process. The author designed a reservation algorithm to reserve services with correlations in the matching stage. To systematically model quality correlations and enable efficient queries of quality correlations for service compositions, [26] proposed a novel approach named Q^2C. [27] focused on QoS-aware service composition and took QoS correlations between services into account. Then proposed the service selection method $CASP4NAT$. However, most of them considered the correlations between services by offering discounts to bundle services. This makes sense for large service providers such as the cloud, but not for resource-constrained mobile devices. Furthermore, some research like [27] generates the service correlations at random and only considers correlations between two services. In contrast, we get the impact of multi-service interactions on service quality via the scheduling and pricing model. We consider the degradation in service quality caused by competition for resources by requests in a feasible plan.

3 System Model

3.1 System Architecture

As illustrated in Fig. 1, we consider a **D2D S**ervice **P**rovision (**D2DSP**) system which consists of one BS equipped with an edge cloud and multiple mobile devices (MDs). The MDs are divided into two categories: supplying MDs and demanding MDs. The supplying MDs are D2D Providers who send the resource supplying information to BS. The demanding MDs are D2D requesters who send resource demand information (request) to BS. The base station acts as the broker to match the supplying and demanding MDs.

In our paper, the MDs are assumed to have no prior knowledge of others, thus all of them directly send their information to the base station. As shown in Fig. 1, After the BS receives all requests in a period, it broadcasts the requests to the providers who satisfy the following two conditions:

1) Communication conditions, which means the provider and requester must be within the D2D communication range of each other, only in this way, can they establish a stable D2D link.

2) Service conditions, based on the above, the provider must deploy the service required by the request.

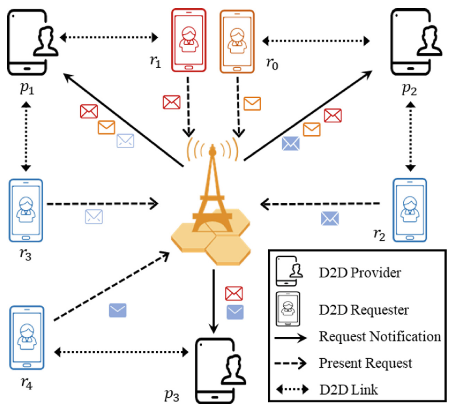
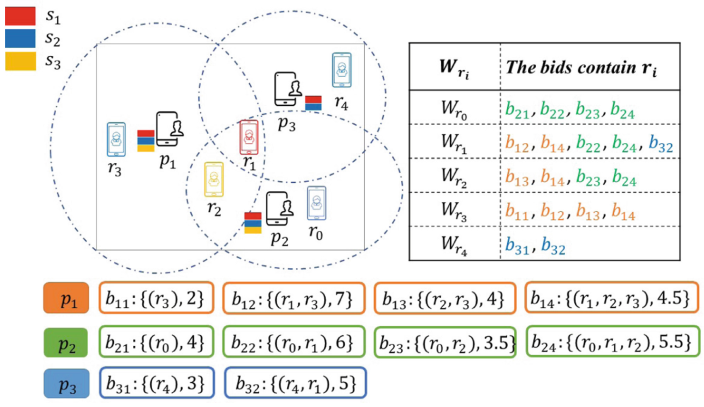


Fig. 1. D2DSP architecture

Figure 1 and Fig. 2 present an example. The distribution of MDs is presented in Fig. 2. Within the 2D area circled by the box, each provider can cover the requests within a dotted ellipse, the rectangle next to the provider represents the service deployed by the provider, the color of the requests illustrates the service required to complete them. So p_1 covers r_1, r_2, r_3 and deploys the services required by them, in this case, p_1 is able to complete r_1, r_2 and r_3. Similarly, p_2 can complete r_0, r_1 and r_2, while p_3 can complete r_1 and r_4. As a result, upon receiving all requests the base station presents r_1 to p_1, p_2, p_3; r_2 to p_1, p_2; r_3 to p_1; r_4 to p_3 and r_0 to p_2, as shown in Fig. 1.

In our paper, a request corresponds to one service, we use the terms request and service interchangeably in the paper. We allocate multiple requests in the chronological order in which they arrive. The definitions of request and service are as follows.


s_1
s_2
s_3
r_3
p_1
p_3
r_4
r_1
r_2
p_2
r_0
p_1
$b_{11}:\{(r_3), 2\}$
$b_{12}:\{(r_1, r_3), 7\}$
$b_{13}:\{(r_2, r_3), 4\}$
$b_{14}:\{(r_1, r_2, r_3), 4.5\}$
p_2
$b_{21}:\{(r_0), 4\}$
$b_{22}:\{(r_0, r_1), 6\}$
$b_{23}:\{(r_0, r_2), 3.5\}$
$b_{24}:\{(r_0, r_1, r_2), 5.5\}$
p_3
$b_{31}:\{(r_4), 3\}$
$b_{32}:\{(r_4, r_1), 5\}$

W_{r_i}	*The bids contain* r_i
W_{r_0}	$b_{21}, b_{22}, b_{23}, b_{24}$
W_{r_1}	$b_{12}, b_{14}, b_{22}, b_{24}, b_{32}$
W_{r_2}	$b_{13}, b_{14}, b_{23}, b_{24}$
W_{r_3}	$b_{11}, b_{12}, b_{13}, b_{14}$
W_{r_4}	b_{31}, b_{32}

Fig. 2. Auction example

Definition 1 (Request). *A request is represented as a 5-tuple* (d, s, L, t_a, t_e) *, where:*

- d *is the index of the service requester who submitted the request;*
- s *is the service required to complete the request;*
- L *is the location of the requester;*
- t_a *is the arrival time of the request;*
- t_e *is the execution time of the request;*

Definition 2 (Service). *A service is represented as a 3-tuple* $(d, \mathcal{G}, F)$ *, where:*

- d *is the index of the provider where the service is deployed;*
- $\mathcal{G}$ *describes the resources required by the service, which can be denoted as* $\mathcal{G} = \{g_{z_1}, g_{z_2}, \ldots, g_{z_m}\}$*, where* m *is the number of the resource type,* g_{z_i} *is the amount of the* z_i *kind of resource.*
- F *is the functional description of the service;*

3.2 Scheduling and Pricing Model

As providers receive request information from the base station, they perform scheduling and pricing model to determine the price and quality of service in different feasible plans (or bids). Specifically, we use the scheduling algorithm *RESP* presented in [28] to arrange the requests. Then run the dynamic pricing method in [29] to calculate the service price. The impact of multi-service interactions on service quality will be determined after the two processes. Then the providers can obtain the cost performance of different feasible plans and submit bids to compete for revenue. To elaborate on this, we present the definition of a provider at first.

Definition 3 (Provider). *A provider* p *is represented as a 5-tuple* $(i, L, \mathcal{S}, \mathcal{A}, \mathcal{V})$, *where:*

- *i is the unique index of the provider in our system;*
- *L is the location of the provider;*
- *$\mathcal{S}=\{s_1, s_2, \dots s_n\}$ is the set of services that the provider deployed;*
- *$\mathcal{A}$ is the function used to describe the available resources of p. Given a time t, the available resources can be denoted as $\mathcal{A}_t = \{a_{z_1,t}, a_{z_2,t}, \dots, a_{z_m,t}\}$, where z_i is the type of resource, m is the total number of resource types, $a_{z_i,t}$ is the amount of z_i at time t.*
- *$\mathcal{V} = \{v_{z_1,t}, v_{z_2,t}, \dots v_{z_m,t}\}$ describes the provider's valuation on its own resources. For example, $v_{z_i,t}$ denotes the price of resource z_i at time t.*

The available resources vector $\mathcal{A}$ is the free resources of the provider that can be used to complete requests. The dynamic resource pricing model used by the provider is based on the inventory theory [29–31]. Due to space constraints, we directly give the function between resource price and the available resource amount n and time t, which was derived from Eqs. 9 and 10 in [29]:

$$v_{z_i,t} = \frac{1}{\alpha}\left[\ln\left(1 + \frac{(\frac{k}{e}t)^n \frac{1}{n!}}{\sum_{i=0}^{n-1}(\frac{k}{e}t)^i \frac{1}{i!}}\right) + 1\right] \tag{1}$$

where n is the available resources of z_i at time t, or called $a_{z_i,t}$. k and α are user-definable constants. According to Eq. (1), the resource price $v_{z_i,t}$ is inversely proportional to the available resource amount $a_{z_i,t}$. The request price can be calculated as a linear summation of the resource type, price and amount, as described in Eq. (2).

$$\rho_r = \rho_s = \sum_{t=r.start}^{r.end} (\mathcal{V}_t \cdot \mathcal{G}_s) \tag{2}$$

where *r.start* and *r.end* are the start and end execution time of the request respectively.

Take an example in Fig. 3, the provider can complete request r_1, r_2 and r_3. But the cost performance of r_3 is different in different feasible plans. Specifically, the scheduling algorithm determines the time interval in which the request is to be executed. Meanwhile, the request price is obtained by the resource prices in the time interval. For different feasible plans, the execution order of requests is different and the resource price varies dynamically with the available resource amount. Therefore the service quality may be completely different for a request. Note that we regard the response time as service quality. As the figure shows, the response time of s_3 is 2 s in the first plan, however, it turns 3 s in the second plan and 5 s in the third plan. Moreover, because the third plan consumes more system resources, the resource price at t_1, t_2, t_3, t_4 and t_5 will be higher than it is in the first two plans. Thus the cost performance of the same request in different plans will be quite different, which is the multi-service inter-impact.

3.3 Auction Model

After obtaining the scheduling and pricing information, the provider submits the feasible plans and corresponding cost performance to the base station in the form

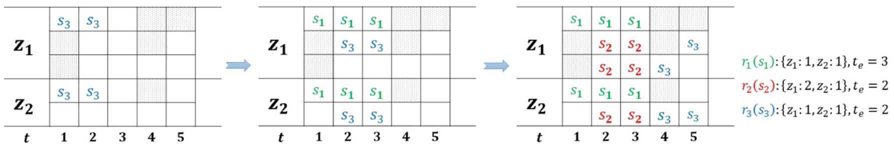

z_1
z_2
t
1 2 3 4 5
s_1
s_2
s_3
$r_1(s_1)$: $\{z_1: 1, z_2: 1\}, t_e = 3$
$r_2(s_2)$: $\{z_1: 2, z_2: 1\}, t_e = 2$
$r_3(s_3)$: $\{z_1: 1, z_2: 1\}, t_e = 2$

Fig. 3. Scheduling & Pricing example

of bids. In order that each provider can express their will sufficiently, we adopt the combinatorial auction model based on improved XOR bidding language, which means every provider p_i can submit multiple bids $B_i = \{b_{i1}, b_{i2}, \dots, b_{ik}\}$, and each bid b_{ik} can contain multiple requests. For example in Fig. 2, as the provider p_3 can complete requests r_1 as well as r_4, p_3 presents bids including $b_{31} = \{(r_4), \phi(b_{31})\}$, $b_{32} = \{(r_1, r_4), \phi(b_{32})\}$. In general XOR bidding language, the bidder will presents b_{31}, b_{32} as well as $b_{33} = \{r_1, \phi(b_{33})\}$, but in our scenario, r_4 can only be completed by p_3, so the b_{33} is meaningless and will definitely not appear in the final result as we are aiming to distribute all requests. Thus we call the bidding language as improved XOR bidding language. The definition of a bid is described in Definition 4.

Definition 4 (Bid). *A bid b is represented as a 6-tuple $(i, c, R, P, \mathcal{Q}, \phi)$, where:*

- *i is the index of provider who presents the bid;*
- *c is the number of requests contained in the bid;*
- *$R = \{r_1, r_2, \dots r_c\}$ is the set of requests in the bid;*
- *$P = \{\rho_{r_1}, \rho_{r_2}, \dots \rho_{r_c}\}$ is the price set of requests set R;*
- *$\mathcal{Q} = \{q_{r_1}, q_{r_2}, \dots q_{r_c}\}$ is the service quality set of requests set R;*
- *$\phi(b)$ is the cost performance of b, which can be calculated by:*

$$\phi(b) = \phi(r_1) + \phi(r_2) + \cdots + \phi(r_c) \tag{3}$$

where $\phi(r_i)$ is the cost performance of r_i, which is defined by Eq. (4):

$$\phi(r_i) = q_{r_i} / \rho_{r_i} \tag{4}$$

Because of the multi-service inter-impact on the requests' cost performance, it is a challenging problem to distribute all requests to obtain the best overall cost performance.

3.4 Problem Formulation

After receiving all bids, the base station acts as an auctioneer to dispatch requests aiming to achieve the best overall cost performance. Since in XOR bidding language, a bidder can win at most one bid even if it submits multiple bids, leading to the first constraint for request allocation:

$$\sum_{j=1}^{k} x_{ij} \le 1, \quad \forall p_i \in P \tag{5}$$

where

$$x_{ij} = \{0, 1\} \tag{6}$$

Among which $x_{ij} = 0$ means the bid b_{ij} is not selected, otherwise b_{ij} is selected. k is the total number of bids submitted by p_i. In addition, a request can only be allocated to one provider, leading to the second constraint:

$$\sum_{b_{ij} \in W_r} x_{ij} \leq 1, \quad \forall r \in R \tag{7}$$

W_r is the bid set including request r, Eq. (7) means only one bid can be selected in W_r. Then the overall cost performance maximization problem can now be formulated:

$$\begin{aligned} &Max \quad \sum_{i=1}^{m} \sum_{j=1}^{k} \phi(b_{ij}) \cdot x_{ij} \\ &s.t. \sum_{j=1}^{k} x_{ij} \leq 1, \quad \forall p_i \in P \\ &\sum_{b_{ij} \in W_r} x_{ij} \leq 1, \quad \forall r \in R \\ &x_{ij} = \{0, 1\} \end{aligned} \tag{8}$$

where $\phi(b_{ij})$ is the cost performance of b_{ij}, m is the number of providers. Other mathematical symbols used in this article are presented in Table 1.

Table 1. Mathematical notations.

Symbol	Description
R	Demanding MD set
B	Bids set
P	Supplying MD set
t_{ij}	The jth feasible bid set (candidate strategy) of request r_0 to r_i
T_i	The candidate strategy set of request r_0 to r_i
W_{r_i}	The bid set including request r_i
b_{ik}	The kth bid from the ith provider
l_{R/T_i}	The length of set R or T_i
$R_{t_{ij}/b}$	The requests set of t_{ij} or bid b
$P_{t_{ij}/b}$	The providers set of t_{ij} or bid b
$\phi(t_{ij}/b)$	The cost performance of t_{ij} or bid b

4 The Proposed RABP Algorithm

In this section, we introduce our dispatching algorithm in detail. To better understand the algorithm, we definite the feasible bid set (or called candidate strategy) and candidate strategy set as follows.

Definition 5 (Feasible Bid Set or Candidate Strategy). *A feasible bid set or called candidate strategy t_i can be represented by a 5-tuple $(d, B_{t_i}, R_{t_i}, P_{t_i}, \phi(t_i))$, where:*

- *d is the index of the candidate strategy;*
- *$B_{t_i} = \{b_1, b_2, \dots b_k\}$ is the set of bids in t_i;*
- *$R_{t_i} = \{r_0, r_1, \dots r_k\}$ is the set of requests covering by t_i;*
- *$P_{t_i} = \{p_1, p_2, \dots p_k\}$ is the set of providers of bid in t_i;*
- *$\phi(t_i)$ is the cost performance of t_i, which can be obtained by Eq. (9):*

$$\phi(t_i) = \phi(b_1) + \phi(b_2) + \cdots + \phi(b_k) \tag{9}$$

The feasible bid set or called candidate strategy $t_i = \{b_0, b_1, \dots, b_k\}$ is the bid set that must cover the requests from r_0 to r_i and may cover other requests in R, which we call them *Following Requests*. Feasible means these bids are from different providers and will not overlap a request.

Definition 6 (Candidate Strategy Set). *A candidate strategy set $T_i = \{t_{i1}, t_{i2}, \dots, t_{in}\}$ is the set of candidate strategy t_i that not be pruned.*

Algorithm 1. *RABP* Algorithm

Input: The requests set R, provider set P, bid set B, the bid set of candidates for each request W_r.
Output: Winning bid set $\hat{t}$.
Initialize: $T_0 = \varnothing$, $\hat{t} = \varnothing$, $i = 0$.

```
1: for every $b \in W_{r_0}$ do
2:    $t_{new} \leftarrow b$ ; $R_{t_{new}} \leftarrow R_b$ ; $P_{t_{new}} \leftarrow P_b$ ; $\phi(t_{new}) \leftarrow \phi(b)$ ;
3:    Pruning($t_{new}$, $T_i$);
4: end for
5: for $i = 1$ to $l_R$ do
6:    for every $t \in T_{i-1}$ do
7:       if $r_i \in R_t$ then
8:          Pruning($t$, $T_i$) ;
9:       else
10:         for every $b \in W_{r_i}$ do
11:            if $P_t \cap P_b = \varnothing$ and $R_t \cap R_b = \varnothing$ then
12:               $t_{new} \leftarrow t \cup b$ ; $R_{t_{new}} \leftarrow R_t \cup R_b$ ; $P_{t_{new}} \leftarrow P_t \cup P_b$ ;
                  $\phi(t_{new}) \leftarrow \phi(t) + \phi(b)$ ;
13:               Pruning($t_{new}$, $T_i$);
14:            end if
15:         end for
16:      end if
17:   end for
18: end for
19: return $\hat{t}$
```

Upon the base station receives bid set B from all service providers, base station acts as an auctioneer to dispatch requests aiming to maximize the overall cost performance, as depicted in Sect. 3.

To solve the problem optimally, we propose a pruning-based request allocation algorithm $RABP$ as shown in Algorithm 1. The algorithm consists of two steps: enumeration and pruning. To begin, we initialize T_0 with bids in W_{r_0} (lines 1–5) since every bid in W_{r_0} is a feasible bid for r_0, then we call Pruning algorithm to determine whether the candidate strategy t_{new} should be pruned (line 6). The pruning rules will be described later.

Algorithm 2. Pruning Algorithm

Input: The current feasible bid set t_{new} for request r_0 to r_i and all feasible bid sets T_i until now for request r_0 to r_i.

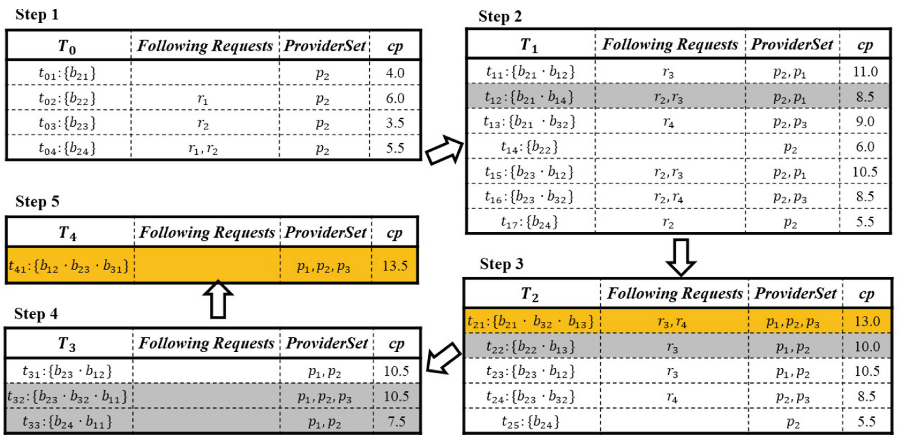
Output: Candidate bid sets T_i.

```
 1: if $R_{t_{new}} = R$ then
 2:    if $\phi(t_{new}) > \phi(\hat{t})$ then
 3:       $\hat{t} \leftarrow t_{new}$;
 4:    end if
 5:    return
 6: else
 7:    $j = 0$
 8:    while $j < l_{T_i}$ do
 9:       if $R_{t_{new}} = R_{t_{ij}}$ then
10:          if $P_{t_{ij}} \subseteq P_{t_{new}}$ and $\phi(t_{ij}) \geq \phi(t_{new})$ then
11:             return
12:          end if
13:          if $P_{t_{new}} \subseteq P_{t_{ij}}$ and $\phi(t_{new}) \geq \phi(t_{ij})$ then
14:             remove $t_{ij}$ from $T_i$
15:          else
16:             $j = j + 1$
17:          end if
18:       else
19:          $j = j + 1$
20:       end if
21:    end while
22:    add $t_{new}$ to $T_i$
23:    return
24: end if
```

After initializing T_0, we try to find the feasible bid set for r_0 to r_i where $i = 1, 2, \ldots, l_{R_c}$ (lines 7–18). We firstly traverse the candidate strategy t in set T_{i-1}, checking whether t has already covered the request r_i, if so, t is already a feasible bid set for r_0 to r_i, then we call Pruning algorithm to determine whether it should be pruned (lines 9–10). Otherwise, traversing the bid b in W_{r_i} (which is the bid set covering r_i), if the bids in t and the bid b are all from different providers and they will not overlap a request (line 13), then we find a new candidate strategy t_{new} for r_0 to r_i, which is the union of t and b. So we update the request set, provider set and the cost performance of t_{new} (lines 14–17) and running Pruning algorithm to check whether the solution has the possibility of becoming the optimal solution. If not, prune it off.

The pruning rules are presented in Algorithm 2. For the candidate strategy t_{new} for r_0 to r_i, if it contains all requests in R (line 1), we compare the cost performance of t_{new} with $\hat{t}$, if the former is greater, which means t_{new} can obtain a better solution than current optimal solution $\hat{t}$, then we update $\hat{t}$ (lines 2–4) and return. If t_{new} only contains part of the requests in R, compare it to all candidate strategies in the current T_i (lines 7–8), if t_{new} and one of the candidate strategy t_{ij} in T_i have the same requests set (line 9), and the provider set of t_{ij} is the subset of t_{new} and the cost performance of t_{ij} is greater than t_{new}, which means t_{ij} can cover the same requests with less providers while achieving a better cost performance than t_{new}, in this case, t_{new} definitely will not be the optimal solution, the algorithm will not add t_{new} to T_i and return directly (lines 10–11). Otherwise, if the provider set of t_{new} is the subset of t_{ij} and the cost performance of t_{new} is greater than t_{ij}, we remove t_{ij} from T_i since it certainly can not obtain a better solution than t_{new} (lines 12–13). Then add t_{new} to T_i (line 18).

The time complexity of $RABP$ is $O(nhl^2)$, where n denotes the length of R, h denotes the average length of candidate bids W_{r_i} for each request r_i. l denotes the average length of the candidate strategy set (T_i). l has a greater impact on the execution time. It is mainly affected by the bid amount submitted by different providers. Through the pruning operation of algorithm 2, the length of T_i can be effectively reduced, thus lowering the time cost of the algorithm.


Step 1

T_0	Following Requests	ProviderSet	cp
$t_{01}:\{b_{21}\}$		p_2	4.0
$t_{02}:\{b_{22}\}$	r_1	p_2	6.0
$t_{03}:\{b_{23}\}$	r_2	p_2	3.5
$t_{04}:\{b_{24}\}$	r_1,r_2	p_2	5.5

Step 2

T_1	Following Requests	ProviderSet	cp
$t_{11}:\{b_{21}\cdot b_{12}\}$	r_3	p_2,p_1	11.0
$t_{12}:\{b_{21}\cdot b_{14}\}$	r_2,r_3	p_2,p_1	8.5
$t_{13}:\{b_{21}\cdot b_{32}\}$	r_4	p_2,p_3	9.0
$t_{14}:\{b_{22}\}$		p_2	6.0
$t_{15}:\{b_{23}\cdot b_{12}\}$	r_2,r_3	p_2,p_1	10.5
$t_{16}:\{b_{23}\cdot b_{32}\}$	r_2,r_4	p_2,p_3	8.5
$t_{17}:\{b_{24}\}$	r_2	p_2	5.5

Step 3

T_2	Following Requests	ProviderSet	cp
$t_{21}:\{b_{21}\cdot b_{32}\cdot b_{13}\}$	r_3,r_4	p_1,p_2,p_3	13.0
$t_{22}:\{b_{22}\cdot b_{13}\}$	r_3	p_1,p_2	10.0
$t_{23}:\{b_{23}\cdot b_{12}\}$	r_3	p_1,p_2	10.5
$t_{24}:\{b_{23}\cdot b_{32}\}$	r_4	p_2,p_3	8.5
$t_{25}:\{b_{24}\}$		p_2	5.5

Step 4

T_3	Following Requests	ProviderSet	cp
$t_{31}:\{b_{23}\cdot b_{12}\}$		p_1,p_2	10.5
$t_{32}:\{b_{23}\cdot b_{32}\cdot b_{11}\}$		p_1,p_2,p_3	10.5
$t_{33}:\{b_{24}\cdot b_{11}\}$		p_1,p_2	7.5

Step 5

T_4	Following Requests	ProviderSet	cp
$t_{41}:\{b_{12}\cdot b_{23}\cdot b_{31}\}$		p_1,p_2,p_3	13.5

Fig. 4. Example for RABP

Fig. 4 shows the process of using the algorithm to solve the example in Fig. 2. Firstly, we get the T_0 initialized by W_{r_0}, as shown in Step 1. Specially, *Following*

Requests means the requests in t_{ij} except r_0 to r_i. Then we traverse t_{0j} in T_0 to get T_1. For example, t_{01}, we check for whether it contains r_1, as it does not contain, we traverse W_{r_1}, if t_{01} can combine with a bid b in W_{r_1} and does not violate the constraints of the Eq. (8), that is, the intersection of the provider set of b and t_{01} is empty, and the intersection of the request set of the two is empty, such as $t_{01} \cdot b_2$, $t_{01} \cdot b_4$ and $t_{01} \cdot b_{10}$, then we obtain a new solution for r_0 to r_1, t_{11}, t_{12} and t_{13} respectively. If the candidate strategy t_{0j} contains r_1, such as t_{02}, we obtain the new solution directly. Every time we get a new solution for r_1, we run Algorithm 2 to determine whether prune it or not. For example, when we obtain t_{15} after combining t_{03} and b_2, firstly judging whether it contains all requests, if not, we compare it with candidate strategies in current T_1, that is t_{11}, t_{12}, t_{13} and t_{14}, we find t_{15} and t_{12} is in the case of the same request set as well as the provider set, but the overall cost performance realized by t_{15} is greater, thus we remove t_{12} from T_1. Otherwise, if in the same case and the overall cost performance realized by t_{12} is greater than t_{15}, we will not add t_{15} to T_1. For example, in step 4, the new solution t_{33} is in the same case of t_{31}, but the overall cost performance of t_{31} is greater, so we prune t_{33} off. If we find a solution contains all requests, such as t_{21} in Step 3, we compare it with current optimal solution $\hat{t}$, if the overall cost performance of new solution (t_{21}) is better than $\hat{t}$, we update $\hat{t}$, if the new solution is worse than $\hat{t}$, for example t_{32} in Step 4, we prune it off directly. Following this pruning rule, we will get the optimal solution stored in $\hat{t}$ after getting T_5.

5 Simulation

To evaluate the effectiveness of the algorithm proposed in this paper, we carry out two sets of experiments. The first one examines the effectiveness of *RABP*, the second one evaluates the load balancing effect of *RABP*.

We implemented the *RABP* algorithm in Python. Our experiments were conducted on a Windows machine equipped with an Intel Core i7-9700 processor and 16 GB RAM. As there is no standard experimental platform, we automatically generate the parameters and use them as the experimental data sets.

5.1 Effectiveness Evaluation

In this section, we evaluate the effectiveness of *RABP*. We choose the reciprocal of the response time as the service quality. Each device is equipped with available resources ranging from 5 to 10. Requests are randomly generated with execution time ranging from 3 to 5, the required number of each resource ranging from 2 to 5. The service number is 10. The function of resource price and the available resource of a provider follows Eq. (1). We set $k = 10$ and $\alpha = 1$. Particularly worth mentioning is the location of the providers and requesters, we generate them in a 2D area and ensure that all provider communication ranges cover the entire region, as shown in Fig. 2. Each request can be completed by at least one of the providers covering it. To evaluate the effectiveness of *RABP*, we compare it with the following four methods:

1) Primal-Dual Approximation Algorithm ($PDAA$): The method presented in [32], which adopts a greedy primal-dual algorithm to obtain the approximate solution of problem 8.
2) Auction Efficiency Maximization Algorithm ($AEMA$): The method proposed in [33], which developed a double auction-based scheme to solve the request allocation problem between cloud computing providers and cloud users. We migrate this method to our environment and modify the fitness function to be feasible strategy's (t_i) cost performance.
3) Greedy Algorithm (GA): Greedy by bid density is widely used to solve the winner determination problem in combinatorial auctions [34–36]. Similar to them, our bid density w is calculated as Eq. (10).

$$w_b = \phi(b)/c \tag{10}$$

4) CALP Algorithm ($CALP$): The method presented in [36], was designed to solve the winner determination and payment problem of combinatorial auction.

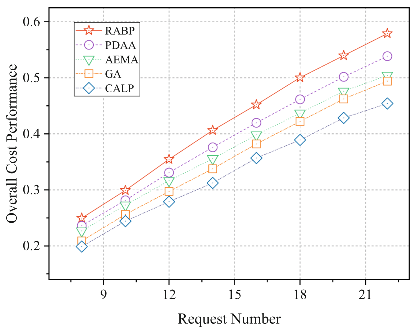
As analyzed in Sect. 4, the scale of the problem is mainly related to the request number, the provider number, and the service deployment rate (sdr). To examine the impact of these three parameters on the result of our method, we consider three sets of parameters, as shown in Table 2. In each set, one of the three parameters is varied while the others remain fixed. All experiments are repeated 500 times, and we use the average values as the result.

Table 2. Variable settings for effectiveness evaluation.

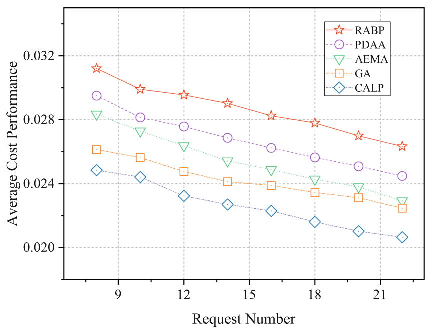
Set	Request number	Provider number	Service deployment rate
1	8–22	15	0.6
2	15	6–20	0.6
3	15	10	0.1–1.0

To explore the impact of request number on the overall cost performance of requests, we set the parameters according to Set 1 in Table 2. The results, shown in Fig. 5(a), demonstrate that the overall cost performance improves with the increasing number of requests. As the allocated request number increases, the overall cost performance will definitely go up. But from Fig. 5(b), we can see that the average cost performance of requests decreases consistently. This is because when the provider number is fixed, the total available resources of all providers remain the same, as the request number rises, more resources are occupied and the request response time increases, causing the rise in price and the drop in service quality. Therefore, the average cost performance of requests continuously decreases.

Obviously, our algorithm significantly outperforms the other four methods as shown in Fig. 5. The $PDAA$ algorithm can obtain a second good result to



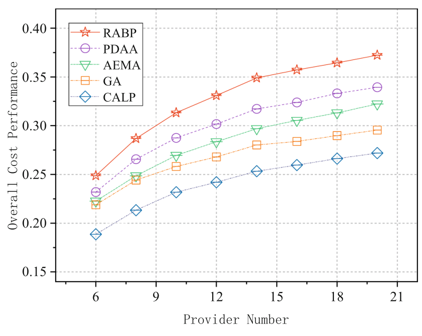
(a) Overall Cost Performance



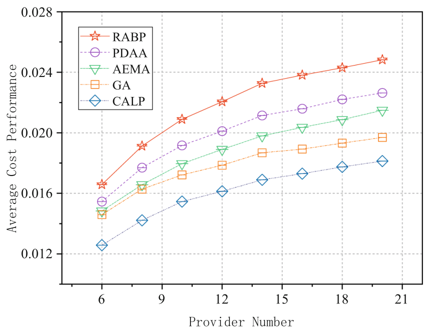
(b) Average Cost Performance

Fig. 5. Impact of the request number

RABP, because it adopts a well-designed primal-dual framework to improve the approximation ratio. The *AEMA* algorithm employs a double auction model and allocates each request with a greedy selection of the most cost-effective bid, so it cannot get a globally optimal solution. Similarly, the *GA* algorithm greedily selects bids with the largest bid density. As *AEMA* and *GA* not allocate requests from a global perspective, thus the optimality of the results is not guaranteed. *CALP* uses a linear programming relaxation method to solve the problem, and it can be seen that the approximate ratio is much worse than *PDAA*.



(a) Overall Cost Performance



(b) Average Cost Performance

Fig. 6. Impact of the provider number

Next, we examine the impact of provider number on the result of the overall cost performance. To this end, we set the experimental parameters in accordance with Set 2 in Table 2. The result is shown in Fig. 6. From Fig. 6(a) we can see that the overall cost performance gradually increases as the provider number grows. As more providers will provide more available resources which is conducive to decreasing the response time and request price, thus improving the cost performance of each request. So, as shown in Fig. 6(b), the average cost performance of requests goes up with the increasing provider number, which is consistent with the overall cost performance result. Meanwhile, we can see that the growth trend is more dramatic between 6 and 16, and then slows down. This is because as the resources required for the requests gradually become sufficient, the improvement in the cost performance of all requests becomes less effective. The comparison of five methods is consistent with the preceding experiment: $RABP$ performs best and the second one is the $PDAA$ algorithm. The $AEMA$ and GA are a little worse than $PDAA$ and $CALP$ is the worst.

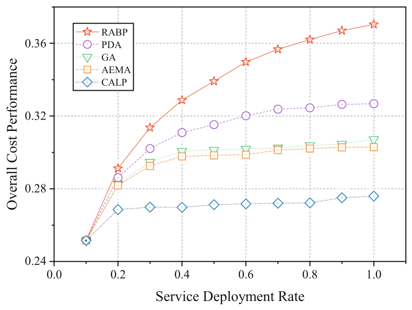


Fig. 7. Impact of the sdr

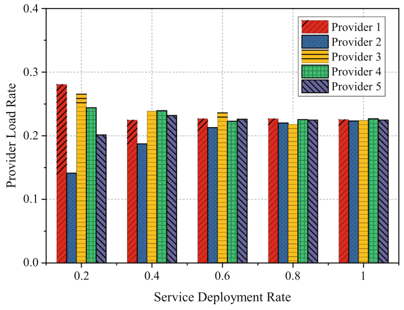


Fig. 8. Load balancing performance with sdr

Finally, we investigate the impact of service deployment rate on the result of overall cost performance. We set the parameters according to Set 3 in Table 2. The results are presented in Fig. 7. From which we can see that our algorithm and $PDAA$ can improve the overall cost performance consistently with the increase of the service deployment rate. $AEMA$ and GA improve the overall cost performance ranging sdr from 0.1 to 0.5 and then slow down. $CALP$ has the worst performance which raises the cost performance between 0.1 and 0.3 and then remains stable. As the service deployment rate increases, the number of requests that a provider can complete gradually increases, thus the average number of requests contained in a bid increases. Other methods greedily select bids that perform best in particular aspects, and on the basis to select other bids. Subjecting to the limitations imposed by already selected bids (constraint limitations in Eq. (8), they can only select among a fraction of the remaining bids, that is, they can only select bids from different providers and which not contain the request covered by selected bids. Thus limiting the optimality of the solution. In contrast, our approach retains all feasible bid sets that are possible to be the optimal solution until the final step. As a result, the superiority of $RABP$ becomes increasingly apparent as the sdr increases.

In summary, whether varying the number of requests, the number of providers, or the service deployment rate, the overall cost performance obtained by *RABP* is significantly better than other methods.

5.2 Load Balancing Performance

In this section, we examine the load balancing performance of *RABP*. To this end, we simulate two scenarios as described below.

Table 3. Variable settings for effectiveness evaluation.

Set	Request number	Provider number	Service deployment rate
1	15	5	0.2–1.0
2	6–20	5	0.6

In the first scenario, we set the parameters according to Set 1 in Table 3 while other parameter settings are the same in the previous section. The result is presented in Fig. 8, which denotes the load balancing effect gradually gets better as the service deployment rate increases. This is because our allocation object is to achieve the best overall cost performance, which means we pursue the lower price as well as request response time. At the same time, a provider equipped with more available resources will bid lower and respond quickly. Thus the probability of winning the bid is greater. Otherwise, the provider with less available resources will bid higher and be less likely to be a winner. So the provider with a lower load rate and more available resources will obtain more requests, promoting load balancing among providers. With the growth of service deployment rate, each request has a greater possibility to be allocated to the provider with sufficient resources, thus improving the load balancing effect.

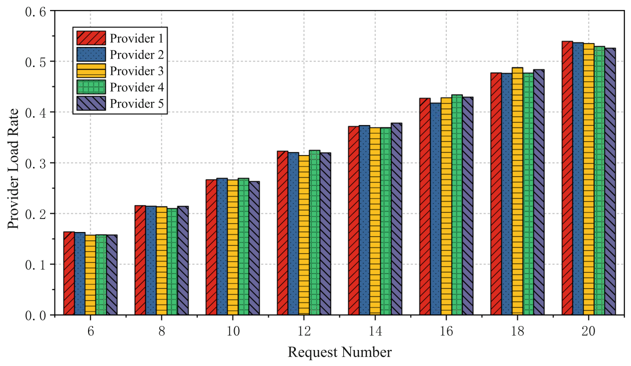


Fig. 9. Load balancing performance with request number

In the second scenario, we set the parameters according to Set 2 in Table 3. We obtain the load rate of providers with the request number continuous increases when the service deployment rate is 0.6. The result is presented in Fig. 9, which demonstrates that the load balancing effect of $RABP$ is excellent when the service deployment rate is 0.6.

6 Conclusion

In this paper, we investigate the request dispatching problem in the D2D environment. We originally focused on the optimization of service cost performance while taking into account the multi-service inter-impact on service quality in a bid. We design a combinatorial auction-based request allocation model, using XOR bid language to motivate providers fully express their matching intention. To solve the request allocation problem optimally, we design the algorithm $RABP$. Extensive simulation results demonstrate that $RABP$ is superior to the $PDAA$, $AEMA$, GA, and $CALP$ in terms of cost performance. Meanwhile, the $RABP$ performs well in load balancing.

As our algorithm is significantly affected by the bid amount, so we will try to design a less complex approach in the future. Furthermore, more complex scenarios with requirements on service quality will be taken into consideration.

Acknowledgement. This work is supported by the National Natural Science Foundation of China grant No. 62102281, and the National Natural Science Key Foundation of China grant No. 62032016 and No. 61832014.

References

1. Jameel, F., Hamid, Z., Jabeen, F., Zeadally, S., Javed, M.A.: A survey of device-to-device communications: Research issues and challenges. IEEE Commun. Surv. Tutor. **20**(3), 2133–2168 (2018)
2. Ericsson: Ericsson mobility report (2021). [Online]. https://www.ericsson.com/49cd40/assets/local/mobility-report/documents/2021/june-2021-ericsson-mobility-report.pdf
3. Zhai, D., Zhang, R., Wang, Y., Sun, H., Cai, L., Ding, Z.: Joint user pairing, mode selection, and power control for D2D-capable cellular networks enhanced by nonorthogonal multiple access. IEEE Internet Things J. **6**(5), 8919–8932 (2019)
4. Cisco: Cisco visual networking index: global mobile data traffic forecast update (2019). [Online]. https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white-paper-c11-738429.html
5. Noura, M., Nordin, R.: A survey on interference management for device-to-device (D2D) communication and its challenges in 5G networks. J. Netw. Comput. Appl. **71**(C), 130–150 (2016)
6. Doumiati, S., Assaad, M., Artail, H.A.: Topological interference management framework for device-to-device communication. IEEE Wirel. Commun. Lett. **7**(4), 602–605 (2018)

7. Wang, Q., Wang, W., Jin, S., Zhu, H.B., Zhang, N.T.: Game-theoretic source selection and power control for quality-optimized wireless multimedia device-to-device communications. In: IEEE Global Communications Conference, pp. 4568–4573. IEEE, New York (2014)
8. Tong, M., Wang, X., Wang, Y., Lan, Y.: Computation offloading scheme with D2D for MEC-enabled cellular networks. In: IEEE/CIC International Conference on Communications in China (ICCC Workshops), pp. 111–116 (2020)
9. Tang, J., Tang, H.B., Zhao, N., Cumanan, K., Zhang, S.Q., Zhou, Y.J.: A reinforcement learning approach for D2D-assisted cache-enabled HetNets. In: IEEE Global Communications Conference (2019)
10. He, Y., Ren, J., Yu, G., Cai, Y.: Joint computation offloading and resource allocation in D2D enabled MEC networks. In: IEEE International Conference on Communications (ICC), pp. 1–6 (2019)
11. Jiang, C.L., Cao, T.F., Guan, J.F.: Intelligent task offloading and collaborative computation over D2D communication. China Commun. **18**(3), 251–263 (2021)
12. Wu, D., Zhou, L., Cai, Y.M., Chao, H.C., Qian, Y.: Physical-social-aware D2D content sharing networks: a provider-demander matching game. IEEE Trans. Veh. Technol. **67**(8), 7538–7549 (2018)
13. Song, W., Zhao, Y., Zhuang, W.: Stable device pairing for collaborative data dissemination with device-to-device communications. IEEE Internet Things J. **5**(2), 1251–1264 (2018)
14. Zhao, Y., Song, W., Han, Z.: Social-aware data dissemination via device-to-device communications: fusing social and mobile networks with incentive constraints. IEEE Trans. Serv. Comput. **12**(3), 489–502 (2019)
15. Li, P., Guo, S., Stojmenovic, I.: A truthful double auction for device-to-device communications in cellular networks. IEEE J. Sel. Areas Commun. **34**(1), 71–81 (2016)
16. Zhu, Y., Jiang, J., Li, B., Li, B.: Rado: a randomized auction approach for data offloading via D2D communication. In: Proceedings of the 2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems, pp. 1–9 (2015)
17. Zhang, Y., Song, L., Saad, W., Dawy, Z., Han, Z.: Contract-based incentive mechanisms for device-to-device communications in cellular networks. IEEE J. Sel. Areas Commun. **33**(10), 2144–2155 (2015)
18. Jiang, J., Zhang, S., Li, B., Li, B.: Maximized cellular traffic offloading via device-to-device content sharing. IEEE J. Sel. Areas Commun. **34**(1), 82–91 (2016)
19. Omran, A., Sboui, L., Rong, B., Rutagemwa, H., Kadoch, M.: Joint relay selection and load balancing using D2D communications for 5G HetNet MEC. In: IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–5 (2019)
20. Zhou, Z., Ota, K., Dong, M., Xu, C.: Energy-efficient matching for resource allocation in D2D enabled cellular networks. IEEE Trans. Veh. Technol. **66**(6), 5256–5268 (2017)
21. Zhao, Y., Song, W.: Truthful mechanisms for message dissemination via device-to-device communications. IEEE Trans. Veh. Technol. **66**(11), 10:307–10:321 (2017)
22. Xu, C., et al.: Efficiency resource allocation for device-to-device underlay communication systems: a reverse iterative combinatorial auction based approach. IEEE J. Sel. Areas Commun. **31**(9), 348–358 (2013)
23. Xue, J., Ma, Q., Shao, H.: Efficient resource allocation for d2d communication underlaying cellular networks: a multi-round combinatorial double auction. In: Proceedings of the 2018 International Conference on Electronics and Electrical Engineering Technology. EEET 2018, New York, NY, USA, pp. 172–176 (2018)

24. Liang, H., Du, Y.: Dynamic service selection with QoS constraints and inter-service correlations using cooperative coevolution. Futur. Gener. Comput. Syst. **76**, 119–135 (2017)
25. Li, H.-F., Zhao, L., Zhang, B.-H., Li, J.-Q.: Service matching and composition considering correlations among cloud services. In: 2015 IEEE International Conference on Systems, Man, and Cybernetics, pp. 509–514 (2015)
26. Zhang, Y., Cui, G., Deng, S., Chen, F., Wang, Y., He, Q.: Efficient query of quality correlation for service composition. IEEE Trans. Serv. Comput. **14**(3), 695–709 (2021)
27. Deng, S., Wu, H., Hu, D., Zhao, J.L.: Service selection for composition with QoS correlations. IEEE Trans. Serv. Comput. **9**(2), 291–303 (2016)
28. Wu, H., et al.: Revenue-driven service provisioning for resource sharing in mobile cloud computing. In: Maximilien, M., Vallecillo, A., Wang, J., Oriol, M. (eds.) ICSOC 2017. LNCS, vol. 10601, pp. 625–640. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-69035-3_46
29. Gallego, G., van Ryzin, G.: Optimal dynamic pricing of inventories with stochastic demand over finite horizons. Manag. Sci. **40**(8), 999–1020 (1994)
30. Li, Y., Hou, Y.: Joint pricing and inventory replenishment decisions with returns and expediting under reference price effects. Math. Probl. Eng. **2019**, 1–17 (2019)
31. Federgruen, A., Heching, A.: Combined pricing and inventory control under uncertainty. Oper. Res. **47**(3), 454–475 (1999)
32. Le, T.H.T., et al.: Auction mechanism for dynamic bandwidth allocation in multi-tenant edge computing. IEEE Trans. Veh. Technol. **69**(12), 15:162–15:176 (2020)
33. Xu, L., Wang, J., Nallanathan, A., Li, Y.: Resource allocation based on double auction for cloud computing system. In: 18th IEEE International Conference on High Performance Computing and Communications (HPCC), Conference Proceedings, pp. 1538–1543 (2016)
34. Jain, V., Kumar, B.: Auction based cost-efficient resource allocation by utilizing blockchain in fog computing. Trans. Emerg. Telecommun. Technol. e4469 (2022)
35. Zhai, Y., Huang, L., Chen, L., Xiao, N., Geng, Y.: COUSTIC: combinatorial double auction for crowd sensing task assignment in device-to-device clouds. In: Vaidya, J., Li, J. (eds.) ICA3PP 2018. LNCS, vol. 11334, pp. 636–651. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-05051-1_44
36. Parida, S., Pati, B., Nayak, S.C., Panigrahi, C.R.: Offer based auction mechanism for virtual machine allocation in cloud environment. In: Pati, B., Panigrahi, C.R., Buyya, R., Li, K.-C. (eds.) Advanced Computing and Intelligent Engineering. AISC, vol. 1089, pp. 339–351. Springer, Singapore (2020). https://doi.org/10.1007/978-981-15-1483-8_29