Auction has been used to allocate resources or tasks to processes, machines or other autonomous agents in distributed systems. Among various types of auctions, combinatorial auction (CA) allocates a bundle of items to each agent at once. Finding an optimal auction result for CA that maximizes total winning bid is NP-hard. Many time-efficient approximations to this problem work with a bid ranking function (BRF). However, existing approximations are mostly for single-unit resource and demand an auctioneer. This paper proposes the first auctioneerless open-bid multi-unit CA (MUCA) scheme. It includes a BRF-based winner determination scheme that enables every agent to locally compute a critical bid value for it to win the MUCA and accordingly take its best response to other agent’s bid and win declarations. It also allows each winner to locally compute its payment for a critical-value-based pricing scheme. We analyze stabilization, correctness, and consistency properties of the proposed approach. Simulation results confirm that the proposed approach identifies exactly the same set of winners as the centralized counterpart regardless of initial bid setting, but at the cost of lower total winning bid and payment.

I. INTRODUCTION

Auction is a trading process that allows seller to identify potential buyers and the prices the buyers are willing to pay. We may use auctions as resource and task allocation schemes. Unlike conventional approaches that assume zero or fixed price of resource or task, auction-based approaches can allocate resource/task to requesters in a way that reflects actual demand and supply conditions. For this reason, auctions have been used to allocate different types of resources or tasks to a fleet of autonomous, self-interest agents. Existing examples include but not limited to the allocations of wireless spectrum [1], [2], cloud resource [3], [4], [5], [6], servers in mobile edge computing [7], [8], tasks of robots [9], [10], [11], and computation or sensing tasks of mobile devices [12], [13].

Depending on how many types and how many units of the items are to sell, auctions can be classified into four categories as shown in Table I. Single-unit single-item (SUSI) is the simplest form of auction, where only item is to sell and there is only one winner. Some studies considered a single item of multiple supplying units (multi-unit single-item or MUSI) [14], [4], [6], where more than one bidders can be winners. We consider the allocation of multiple types of items via auctions. A typical example is in cloud environment, where we have computation, memory, storage, network, and other types of resources. If there is only one supplying unit for each type of item, the auction is of single-unit multi-item (SUMI), where two or more agents (i.e., bidders) could be winners at the same time provided that no two winners have conflicting interests (i.e., they place bids on a common item). If there can be multiple supplying units for some type of item, the auction is of multi-unit multi-item (MUMI), where two or more agents having conflicting interests can be winners at the same time provided that the set of winning bids is feasible. A set of winning bids is feasible if for each type of item, the total amount of units demanded by all winning bids does not exceed the supply.

A. Combinatorial Auction

Identifying a feasible set of winning bids with the highest total bid is winner determination problem (WDP). WDP is NP-hard even for SUMI. Some approaches take sequential single-item auction [21], [20], [11], where bidders bid for one item at a time. This type of auction can be executed in polynomial time. However, it typically applies to SUMI and is not suitable when agents are single-minded [18] (i.e., bidders are only interested in and thus places a bid on a particular bundle of items).

For single-minded agents, combinatorial auction (CA) [19], [26] allocates a bundle of items to bidders at once. We refer to CAs for SUMI and MUMI as SUCA and MUCA, respectively. As WDP for CA is NP-hard [19], most existing approaches take approximations. We particularly consider approximations that are based on a bid ranking function (BRF), which defines a total order on bid requests. BRF-based approximations are time efficient and suited to single-minded agents, but do not guarantee optimality.

Another point of auction-based resource allocation is to identify bidders’ valuations on resources. In general, different bidders have different valuations on the same type of resource. For example, computation resource is more valuable to computation-intensive tasks than to communication-intensive tasks. An auction mechanism is economically efficient if it...
maximizes social welfare, i.e., the aggregated valuation from all winning bidders. Economically efficient auction is desirable when it is that the total utility of resource requesters rather than the revenue of the seller that is of concern.

If every bid represents the bidder’s true valuation on the bid items, the auction has a nice property called truthful bidding. Truthful bidding ensures that the total winning bid is exactly the social welfare that we want to maximize. Truthful bidding is challenging, however, because bidder’s valuations on bid items are considered local and private (i.e., not revealed to other bidders and the auctioneer who conducts the auction). Whether bidders are willing to bid truthfully primarily depends on pricing scheme that decides the payment of each winner in auction. From the perspective of game theory, a pricing scheme is incentive compatible if every bidder’s dominant strategy is to bid truthfully.

In a first-price pricing scheme, winner pays exactly her bid. Consequently, rational bidders will not place any bids higher than their valuations on bid items because doing so only incurs negative payoffs. On the other hand, bidders tend to lower their bids so as to increase payoffs by paying less when they turn out to be winners. Therefore, the first-price pricing scheme is not incentive compatible and the auction result is not economically efficient generally. As a remedy, Vickrey proposed second-price scheme [27] for SISU auction, where the winner pays the second highest bid. This payment rule is incentive compatible.

Incentive-compatible pricing scheme together with an optimal winner determination can ensure economical efficiency. The most well-known incentive-compatible pricing scheme for CA is the Vickrey-Clarke-Groves (VCG) mechanism [27], [28], [29]. VCG relies on the optimal solution to the WDP, so it is not computationally feasible. We consider a computationally-efficient heuristic called critical-value-based payment that depends on the definition of the BRF for the associated WDP. It has been proved that if the associated BRF is monotone, a critical-value-based payment is incentive compatible [18]. However, VCG and most BRF-based approaches [18], [30] are for SUCA.

B. Related Work and Motivation

Apart from the types and the number of units of the bid items, auctions can also be classified by how the auction is conducted. Many auctions implicitly assume a single entity, i.e., an auctioneer, to conduct auctions. An auctioneer is needed in a sealed-bid auction, where all participants send their bids to the auctioneer without knowledge of other bids. It is the auctioneer that declares winners and associated payments. Many previous CA approaches fall into the category of sealed-bid auctions and thus all need an auctioneer [31], [18], [30], [23], [32].

The auctioneer is a single point of failure and can be a performance bottleneck. There have been some approaches that attempt to duplicate or partition the load of auctioneer to a set of brokers [33]. Kutanoglu and Wu [34] decomposed the WDP for CA into subproblems each solved by a local agent. However, an auctioneer is still needed to collect and update bidding information for coordinating an iterative auction.

When there are multiple units or types of items to sell, there could be multiple auctioneers, for which decentralized approaches also have been proposed. Lewis et al. [17] proposed a decentralized adaptive pricing scheme for sellers to determine the best selling prices in a posted-price model. In [13], buyers send their bids to multiple sellers, which then locally determine the set of winning bid candidates. Each buyer then chooses its final seller. A reverse auction is proposed for the allocation of sensing tasks to smart devices in [12], where buyers (task owners) individually work as auctioneers of their own task auctions. Smart devices as sellers submit their asking prices to buyers, which then announce the auction results. An additional step is needed to handle the case when a seller becomes winners of multiple auctions. In all these decentralized approaches, sellers sell the same type of item with different quantities, which renders these approaches decentralized MUSI. Multiple auctioneers have also been proposed for decentralized SUMI auctions [35].

Our study focuses on decentralized auctioneerless auction, where bidders themselves coordinate the set of winners and payments. This type of auction is preferable in many cases, especially when supplies of and requests for resources exhibit locality property. Some examples are listed below.

- Resource is only accessible to “local” users. An example is wireless spectrum resource.
- Users only have interest in locally-accessible resource.
- An example is virtualized resource provided by edge servers in mobile edge computing environment.

In these cases, potential competitors contending for the same type of resource tend to cluster together. Therefore, a decentralized auction for bidders to coordinate the set of winners is more robust and scalable than an auctioneer-based approach. Furthermore, in applications like robot task allocations, it is more desirable to let robots themselves coordinate their tasks because, compared with a central coordination approach that decides a global task allocation, the decentralized approach has a shorter response time.

For SUSI, Esteva and Padget [15] proposed an auctioneerless approach to WDP based on leader election protocol running on a ring overlay network. Their approach targets at single-item auction so only one winner is possible. For SUMI, sequential single-item auctions are also decentralized and auctioneerless. Each agent independently and incrementally constructs its own bundle of items. In each round of the auction, a single item is to sell to one agent via (possibly reverse) single-item auction. Each agent places its bid on the item based on the reward it might receive from adding the item to its bundle. Sequential single-item auctions have been proposed for the assignment of different tasks to a fleet of robotic agents [21], [9], [20], [11].

In this paper, we propose a decentralized auctioneerless approach to MUCA where bidders autonomously decide whether they themselves are winners and how much they should pay.
The contributions of this work are summarized as follows.

- We propose a winner determination protocol for MUCA which works with any given BRF that is monotone. This protocol is deadlock free because it allows bidders to revise and update their bid requests whenever they want to react to other bidder’s updates (as their best responses). This protocol guarantees stabilization in the face of dynamic bidder interactions. It meets resource constraint and conforms to the BRF-based winner determination rule. For a specific BRF, the proposed decentralized approach can yield the same set of winners as the centralized counterpart, despite that the ranks of bid requests and payments may be different in the two approaches.

- We consider two pricing schemes, first-price payment and critical-value-based payment, and analyze the impact of pricing scheme on agent’s bidding strategies in the framework of game theory. We show that, with critical-value-based payment, truthful bidding is every agent’s weakly dominant strategy. We also propose a method for each winner to locally determine how much it should pay if critical-value-based payment is in effect.

To the best knowledge of the authors, this is the first decentralized CA approach that possesses these properties. We have conducted extensive simulations to investigate the performance of the proposed approach.

The rest of this paper is organized as follows. Sec. II covers the background and Sec. III presents the game model for our problem. We present the proposed scheme in details in Sec. IV and analyze its properties in Sec. V. Section VI contains the simulation results that confirm the advantage of our scheme. The last section concludes this paper.
II. PROBLEM DEFINITION

We consider a set of \( n \) bidding agents (bidders) \( A = \{a_1, a_2, \ldots, a_n\} \) and \( m \) different types of resources \( R = \{r_1, r_2, \ldots, r_m\} \). Let \( q_i = (q_{i1}, q_{i2}, \ldots, q_{im}) \) be a supply vector such that \( q_i \geq 1 \) is the total number of units (or identical instances) of resource type \( r_j \). For SUCA, \( q_i = 1 \) for all \( i \). Agent \( a_i \) submits a request vector \( s_i = (s_{i1}^1, s_{i2}^2, \ldots, s_{im}^m) \), where \( s_{ij} \leq q_j \) is the number of units of resource type \( r_j \) requested by \( a_i \). For SUCA, \( s_i \) reduces to a set (named bundle) \( S_i \subseteq R \). Theoretically speaking, \( a_i \) requests may deviate from what \( a_i \) desires if such a deviation could bring \( a_i \) a higher expected payoff. For now we assume that no agent cheats at the request vector. Later we will prove that indeed no agent has the incentive to cheat.

The bid \( a_i \) places on \( s_i \) is denoted by \( b_i(s_i) \) or simply \( b_i \), which together with \( s_i \) forms \( a_i \)'s bid request \( (s_i, b_i) \). We assume that an agent only submits one bid request. If an agent may submit multiple requests (i.e., OR bids [36]), we can treat \( A \) as a set of requests rather than agents. If an agent is allowed to submit but not to win multiple requests (i.e., XOR bids [36]), we may manually add mutual-exclusive relation between each pair of requests submitted by the same agent.\(^1\)

A. Winner Determination Problem

Given a set of bid requests \( B = \{(s_i, b_i(s_i))\}_{i=1}^n \), the winner determination problem (WDP) is to find a setting of \( X = (x_1, x_2, \ldots, x_n) \), where \( x_i \in \{0, 1\} \) for all \( i \), that maximizes the total winning bid

\[
\sum_{x_i=1} b_i(s_i) \tag{1}
\]

subject to the resource capacity constraint defined as

\[
\sum_{i=1}^n (x_i \cdot s_{ik}) \leq q_k \text{ for all } k = 1, \ldots, m. \tag{2}
\]

The WDP for SUCA is an instance of the maximum weight set packing problem, which is known to be NP-hard [19]. Some approaches guarantee optimality but may be time-efficient for some problem instances [36, 37]. Some approaches are time-efficient and achieve optimality by restricting the form or size of bid requests [38]. Some approaches use heuristic or approximation techniques for time efficiency but not optimality. Hoos and Boutilier [39] used stochastic local search algorithm as an approximation to WDP. Zurel and Nisan [40] also proposed an approximation which runs the linear-programming relaxation of the packing problem and then refines the solution by local improvements in the order of bids (hill-climbing). The hill-climbing concept was also adopted by Fukuta and Ito [41] to improve the performance of a simple greedy approach [18]. They also considered the use of simulated annealing technique. Other approximation approaches include dynamic programming [3] and genetic algorithm [42].

\(^1\)One possible way of doing this is through the creation of dummy goods [22]. Also note that all OR bids can be converted into equivalent XOR bids [36].

In this paper, we mainly consider approximations that use a BRF to define a total order \( \prec \) on \( \{(s_i, b_i)\}_{i=1}^n \) such that \( (s_j, b_j) \prec (s_i, b_i) \) if \( (s_j, b_j) \) ranks higher than \( (s_i, b_i) \). Algorithm 1 shows the general framework for greedy allocations which examines all bid requests in the order defined by \( \prec \) to determine whether each request can be granted.

**Algorithm 1** BRF-based Greedy Allocation

\[
\begin{align*}
1: & \quad B \leftarrow \{(s_i, b_i(s_i))\}_{i=1}^n \\
2: & \quad x_i \leftarrow 0 \text{ for all } i \\
3: & \quad \text{while } B \neq \emptyset \text{ do} \\
4: & \quad \text{let } (s_k, b_k) \text{ be the request that ranks first in } B \\
5: & \quad \text{if } q - s_k \geq 0 \text{ then} \\
6: & \quad x_k \leftarrow q - s_k \\
7: & \quad x_k \leftarrow 1 \\
8: & \quad \text{end if} \\
9: & \quad B \leftarrow B \setminus \{(s_k, b_k)\} \\
10: & \quad \text{end while}
\end{align*}
\]

There have been many BRFs proposed for SUCA. The BRF proposed by Lehmann et al. [18] favors a request that maximizes normalized bid value defined as

\[
w_x(S_i, b_i) = \frac{b_i}{|S_i|^\alpha}, \tag{3}
\]

where \( \alpha \) is a configurable parameter. Mito and Fujita [30] considered several possible BRFs inspired by the heuristics for the maximum weighted independent set (MWIS) problem [43]. Let \( N_i \) be the set of all conflicting requests for request \((S_i, b_i)\). One such BRF sets a priority defined as

\[
w_n(S_i, b_i) = \frac{b_i}{(|N_i| + 1)^\beta}, \tag{4}
\]

where \( \beta \) is a configurable parameter. Another BRF considered by them is

\[
w_\phi(S_i, b_i) = \frac{\phi(S_i, b_i)}{(\sum_{(S_j, b_j) \in N_i} b_j + 1)^\beta}, \tag{5}
\]

where

\[
\phi(S_i, b_i) = \frac{b_i}{(\sum_{(S_j, b_j) \in N_i} |S_i \cap S_j| + 1)^\alpha}. \tag{6}
\]

Function \( \phi(\cdot) \) alone could also be a BRF.

Not too many approaches have been proposed for MUCA. Leyton-Brown et al. [22] proposed an optimal WDP algorithm. This algorithm uses techniques like branch-and-bound and dynamic programming, which makes it difficult to be decentralized. As an approach to allocating fine-grained spectrum resources, Jia et al. [23] generalized the BRF \( w_x(\cdot) \) defined in (3) to MUCA. The proposed BRF is

\[
w_m(S_i, b_i) = \frac{b_i}{(\sum_{k=1}^m s_{ik}^k)^\alpha}. \tag{7}
\]

The same BRF has also been used for the allocation of virtual machine instances in clouds [24, 25]. The work in [44] generalized the BRF to consider scarcity of resources with
α = 0.5. Mashayekhy et al. [5] considered the following BRF for a bid request in a unit of time.

\[ w_d(s_i, b_i) = \frac{b_i}{\prod_{s_j \neq 0} s_j^k}. \]  

(8)

Some BRFs for SUCA like (4) do not consider the number of resource instances. When being used in MUCA, these BRFs may perform poorly. BRFs like (5) and (6) have not yet been extended to handle multi-unit resources. A possible extension is to replace |\(S_i \cap S_j\)| in (6) with some matching term like \(s_i \cdot s_j\).

B. Pricing Scheme

The VCG mechanism generalizes the second-price scheme to ensure truthful bidding in CAs. VCG demands that each winner \(a_i\) in VCG has to pay the social opportunity cost (i.e., the reduction of the total winning bid excluding \(a_i\)'s) due to the presence of \(a_i\)'s request. Suppose that we have a set of request pairs \(B = \{(S_i, b_i)\}_{i=1}^n\). Let \(B_{-i}\) denote \(B \setminus \{(S_i, b_i)\}\). Let \(W\) and \(W_{-i}\) are the sets of winning requests with the highest total bid given \(B\) and \(B_{-i}\), respectively. Each winning request \((S_i, b_i) \in W\) has to pay \(p_i = \sum_{(S_j, b_j) \in W_{-i}}(b_j - \sum_{(S_j, b_j) \in W \setminus \{(S_i, b_i)\}} b_j)\). VCG payment has been used in [45].

VCG is economically efficient but computationally infeasible because determining \(W\) is NP-hard.

For BRF-based winner determination designed for SUCA, Lehmann et al. [18] defined monotonicity property for BRF, which states that the BRF gives \((S_j, b_j)\) a rank equal to or higher than that of \((S_i, b_i)\) if \(S_i \subseteq S_j \land b_i \geq b_j\). BRF \(w_{\phi}(\cdot)\) has the monotonicity property. BRFs \(w_n(\cdot), w_n(\cdot)\), and \(\phi(\cdot)\) do not ensure monotonicity because it is possible that \(S'_i \subseteq S_i\) but \((S'_i, b_i) \not\equiv (S_i, b_i)\) as long as the set \(N_i\) remains unchanged for both \(S_i\) and \(S'_i\).

An allocation of resource to bidders is exact if each bidder \(a_i\) is allocated either its request \(s_i\) (if \(a_i\) wins) or nothing (otherwise) [18]. For a BRF-based winner determination for SUCA with both the exactness and monotonicity properties, it is proved [18] that there is a critical value \(c_i\) for each \(b_i\) such that \(a_i\) gets \(S_i\) if \(b_i \geq c_i\) and \(a_i\) gets nothing if \(b_i < c_i\).

For example, assume that \((S_i, b_i)\) is a winning request and \((S_j, b_j)\) is the request that has the highest rank in the set of requests that do not win because of the presence of \((S_i, b_i)\). For the BRF defined in (3), \((S_i, b_i)\) is a winning request because \(w_n(S_i, b_i) > w_n(S_j, b_j)\), which implies that

\[ b_i > b_j \frac{|S_i|^\alpha}{|S_j|^\alpha}. \]  

(9)

On the other hand, \((S_i, b_i)\) would not be a winning request if \(w_n(S_i, b_i) < w_n(S_j, b_j)\) or, equivalently, if

\[ b_i < b_j \frac{|S_i|^\alpha}{|S_j|^\alpha}. \]  

(10)

Therefore, \(c_i = b_j \times |S_i|^\alpha / |S_j|^\alpha\) is the critical value for \(b_i\).\(^2\)

\(^2\)Though not explicitly stated, critical values should also exist for other monotone BRFs with exact allocations [30].

If a BRF-based winner determination is used for which the criticality property holds, then the following pricing scheme ensures truthful bidding in a sealed-bid CA [18].

\[ p_i = \begin{cases} c_i, & \text{if } x_i = 1, \\ 0, & \text{otherwise.} \end{cases} \]  

(11)

Intuitively, \(c_i\) does not depend on \(b_i\), so \(a_i\) cannot decrease its payment by unilaterally manipulating \(b_i\). This implies that criticality-based pricing schemes are strategy-proof.

It is not difficult to see that a critical value also exists for each bidder in a MUCA with the same setting. In this paper, we shall extend critical-value-based payment for MUCA.

III. DYNAMIC MUCA GAME

In the proposed framework, bidder independently sets up bid request and then notifies all competitors of that setting. The setting may cause the competitors to make their own moves. Because notifications take arbitrary time and there is no synchronization scheme to coordinate bidder’s moves, bidders make moves one after another in a non-deterministic manner. We thus model MUCA as a dynamic game. In contrast, bidders in sealed-bid CAs place their bids without bidding information of any others, rendering it a static (one-shot) game.

We assume a BRF \(bf\) which maps any bid request to a positive real number. It defines a total order \(\preceq\) on \(B = \{(s_i, b_i)\}_{i=1}^n\) as follows.

**Definition 1 (Ranks on Bid Requests):** Given two bid requests \((s_i, b_i)\) and \((s_j, b_j)\), we have \((s_i, b_i) \preceq (s_j, b_j)\) if \(bf(s_i, b_i) \geq bf(s_j, b_j)\), and \((s_i, b_i) \prec (s_j, b_j)\) if \(bf(s_i, b_i) > bf(s_j, b_j)\).

We assume that \(bf\) is monotone. The monotonicity of BRF defined in [18] is for SUCA. We now generalize the monotonicity property to MUCA as follows.

**Definition 2 (Monotonicity for Multi-unit BRF):** Let \(s_i = (s_i^1, s_i^2, \ldots, s_i^n)\) be \(a_i\)'s request vector. Let \(S = \{s_1, s_2, \ldots, s_m\}\). Define binary relation \(\preceq\) on \(S\) as \(s_i \preceq s_j\) if \(s_i^k \leq s_j^k\) for all \(k \in \{1, \ldots, m\}\). Define binary relation \(\prec\) on \(S\) as \(s_i \prec s_j\) if \(s_i \preceq s_j\) and \(s_i \not\equiv s_j\). A BRF is monotone if \((s_j, b_j) \preceq (s_i, b_i)\) whenever \(s_j \leq s_i\) and \(b_j \geq b_i\).

By this definition, both \(w_m(\cdot)\) and \(w_d(\cdot)\) defined in Sec. II-A are monotone and can be used in the MUCA game.

To simplify our design and analysis, we assume that for any two bid requests \((s_i, b_i)\) and \((s_j, b_j)\), either \((s_i, b_i) \prec (s_j, b_j)\) or \((s_j, b_j) \prec (s_i, b_i)\). That is, no two bid requests have the same rank. Although BRF like \(w_m(\cdot)\) and \(w_d(\cdot)\) does not have this property, we can easily make it by introducing some tie-breaking rule for ranks like unique bidder identifiers.

Each agent \(a_i\) has a valuation on \(s_i\) denoted by \(\nu_i(s_i)\), which is private. We do not allow for externalities, which means that \(\nu_i(\cdot)\) does not depend on any \(\nu_j(\cdot)\) with \(j \neq i\). Possibly different from \(s_i\), each agent \(a_i\) has a need vector \(d_i = (d_i^1, d_i^2, \ldots, d_i^n)\), where \(d_i^k \leq q_j\) is the units of resource type \(r_j\) really needed by \(a_i\). We assume exact allocation, so each bidder \(a_i\) is allocated either its request \(s_i\) (if \(a_i\) wins)
or nothing (otherwise). Moreover, the assumption of single-minded agents indicates that every agent \(a_i\) is interested in \(d_i\) only. Formally,

\[
\nu_i(s_i) = \begin{cases} 
\nu_i(d_i), & \text{if } d_i \leq s_i, \\
0, & \text{otherwise.} 
\end{cases}
\]  

(12)

Therefore, winning \(s_i\) such that \(s_i < d_i\) gives no value to \(a_i\). On the other hand, the monotonicity property implies that \((d_i, x_i) < (s_i, b_i)\) for all \(d_i < s_i\) but \(\nu_i(s_i) = \nu_i(d_i)\). Therefore, submitting \(s_i\) such that \(d_i < s_i\) only lowers the probability of winning the auction (due to the monotonicity property) without increasing the value of the win. In other words, agent \(a_i\) has no incentive to manipulate \(s_i\) and \(s_i\) is not part of \(a_i\)'s strategy in the game. Thus it is reasonable to assume that \(a_i\) sets up \(s_i\) initially and does not change \(s_i\) during the game.

Each agent’s primary strategy is its bid \(b_i\). In open ascending-price auctions and other decentralized auctions [33], [31], [16], agents can only raise their bids. We take the same assumption.

Besides \(b_i\), every \(a_i\) also needs to declare whether it wins or not currently with \(b_i\). We use \(x_i\) to denote \(a_i\)'s declaration, where \(x_i = 1\) if \(a_i\) declares a win and \(x_i = 0\) otherwise. We use \(N_i\) to denote agent \(a_i\)'s neighboring nodes in the conflict graph, i.e., the set of \(a_i\)'s competitors. Every agent \(a_i\) needs to notify all agents in \(N_i\) of \(x_i\). Similarly, \(a_i\) needs win declaration \(x_j\) of every agent \(a_j \in N_i\) for its own win declaration. Win declaration information is needed because of the locality property that we want to exploit in designing decentralized auction protocol. For example, suppose that \(a_2\) in Fig. 1 is a winner only if \(a_4\) is not, which in turn depends on whether \(a_5\) wins. Due to locality, \(a_2\) does not have knowledge of \(a_5\)'s win and thus cannot locally deduce \(a_5\)'s win. Therefore, agent \(a_4\) should notify \(a_2\) of its win declaration.

Including \(x_i\) in \(a_i\)'s strategy adds another dimension to agent's strategy space. The value of \(x_i\) should be interpreted as \(a_i\)'s willingness to win and pay. This interpretation allows bidders to withdraw their current bids. In contrast, bidders in any other auction have no freedom to configure \(x_i\)'s because bidders are implicitly assumed to be always willing to win with their current bids.

It is theoretically possible that \(a_i\) declares a win (i.e., \(x_i = 1\)) or loss (\(x_i = 0\)) without a matching bid \(b_i\). The correctness of \(x_i\) depends on the relationship between \(b_i\) and \(a_i\)'s critical value \(c_i\). As proved by Lehmann et al. [18], if a BRF with both the exactness and monotonicity properties is used for winner determination, there is a critical value \(c_i\) for every \(a_i\) such that \(a_i\) wins if \(b_i > c_i\) and does not if \(b_i < c_i\). To define the correctness of win declaration in MUCA, we extend the definition of critical value for SUCA in [18] to MUCA as follows.

Definition 3 (Critical Value for MUCA): Given \(B_{-i} = B \setminus \{s_i, b_i\}\) and \(X_{-i} = \{x_j | j \neq i\}\), \(a_i\)'s critical value \(c_i\) is the minimal value that \(a_i\) can win by placing a bid \(b_i > c_i\) (which is also the maximal value that \(a_i\) will definitely lose by placing \(b_i < c_i\), if \(c_i > 0\)) with respect to \(B_{-i}\) and \(X_{-i}\). Formally,

\[
\sum_{(s_j, b_j) < (s_i, b_i)} (x_j \cdot s_j) \leq q_k - s_i^k \quad \text{for all } s_i^k \neq 0
\]  

(13)

if \(b_i > c_i\). If (13) holds when \(b_i \geq 0\), we define \(c_i = 0\). Otherwise, we also have

\[
\sum_{(s_j, b_j) < (s_i, b_i)} (x_j \cdot s_j^k) > q_k - s_i^k \quad \text{for some } s_i^k \neq 0
\]  

(14)

when \(b_i < c_i\).

Because \(a_i\) wins only if \(b_i \geq c_i\) and does not only if \(b_i \leq c_i\), we have the following definition.

Definition 4 (Correctness of Win Declaration): For a pair \((b_i, x_i)\) declared by any agent \(a_i\), \(x_i = 1\) if \(b_i = 1\) and \(b_i > c_i\) or \(x_i = 0\) and \(b_i < c_i\). When \(b_i = c_i\), which implies that the BRF value of \((s_i, b_i)\) is the same as that of another bid request \((s_j, b_j)\), whether \(x_i\) is correct depends on the tie-breaking rule used to determine the rank order between \((s_i, b_i)\) and \((s_j, b_j)\).

The setting of \(x_i\) directly affects \(a_i\)'s utility. Let \(p_i\) be the price that \(a_i\) has to pay at the end of the auction. We consider both first-price payment and critical-value-based payment. In the first-price payment, each winner \(a_i\) pays its winning bid, i.e., \(p_i = b_i\). In the critical-value-based payment, \(p_i = c_i\) for each winner \(a_i\). The utility of \(a_i\) is defined to be \(a_i\)'s payoff in the auction, i.e., \(a_i\)'s valuation on \(s_i\) minus \(p_i\) if \(a_i\) declares a win, and zero otherwise. Formally, given \(B_{-i} = B \setminus \{s_i, b_i\}\) and \(X = \{x_j\}_{j=1}^n\),

\[
u_i(B_{-i}, X) = x_i (\nu_i(s_i) - p_i).
\]  

(15)

The problem with (15) is that agent’s utility has nothing to do with the correctness of win declaration. When \(\nu_i(s_i) > p_i\), \(a_i\) can get a positive utility by declaring \(x_i = 1\) regardless of whether \(b_i \geq c_i\). On the other hand, when \(\nu_i(s_i) < p_i\), \(a_i\) can get a zero (instead of negative) utility by declaring \(x_i = 0\) even if \(b_i > c_i\).

To ensure correct win declarations, we propose the following rules for agents that falsify win declarations.

R1 If \(x_i = 0\) but \(b_i > c_i\), \(a_i\) gets nothing and pays \(p_i\).

R2 If \(x_i = 1\) but \(b_i < c_i\), \(a_i\) gets \(s_i\) and pay \(p_i + \rho\), where \(\rho > 0\) is a penalty.

With this treatment, the following theorem shows that falsifying win declaration is not beneficial if any false declaration can always be detected.

Theorem 1: If false win declarations are always detected, no agent has the incentive to falsify win declaration.

Proof: Consider any agent \(a_i\) and let \(v_i = \nu_i(s_i)\). One type of false win declaration is \(x_i = 0\) but \(b_i > c_i\). If \(v_i \geq p_i\), then declaring \(x_i = 1\) will give \(a_i\) a non-negative utility \(v_i - p_i\) (instead of \(-p_i\) by R1) so \(a_i\) has no incentive to declare \(x_i = 0\). Therefore, it must be the case that \(v_i < p_i\). When \(x_i = 0\) is detected false at the end of the auction, \(a_i\) has to pay \(p_i\) by R1, which is higher than the loss \(p_i - v_i\) if \(a_i\) declares a win instead. So \(a_i\) would rather set \(x_i\) to 1. Now consider the other case that \(x_i = 1\) but \(b_i < c_i\). If \(v_i < p_i\), declaring
By (15), any agent $a_i$ has no incentive to do so. Therefore, it must be the case that $v_i \geq p_i$. In that case, raising $b_i$ to some value between $c_i$ and $v_i$ (or $c_i$ if $v_i = c_i$) would give $a_i$ a utility $v_i - p_i - \rho$ that $a_i$ has to pay by R2. So $a_i$ has no incentive to declare $x_i = 1$ with $b_i < c_i$.

However, we cannot guarantee the detection of false win declarations if some agents collude with one another. If we preclude the possibility of collusion and always detect any single false declaration, then no agent has the incentive to make a false win declaration. Theorem 2 shows the feasibility of detecting any single false win declaration.

**Theorem 2:** After the MUCA game ends, any single false win declaration can be detected.

**Proof:** Without loss of generality, let $a_i$ be the only agent with false $x_i$. Let $A_k = \{a_j | s_j^k \neq 0\}$ be the set of all agents that request $r^k$. All these agents are competitors so any of them has knowledge of all other’s bid requests and win declarations. Consider two possible cases of false declarations:

- $x_i = 1$ is false. If $x_i = 1$ is correct, $b_i$ should be larger than $c_i$. By Definition 3, $b_i > c_i$ implies that $\sum_{(s_i, b_i) < (s_i, b_i)} (x_i \cdot s_i^k) \leq q_k - s_i^k$ for all $s_k^i \neq 0$. All agents in $\cup_{k=0}^{b_i} A_k$ can collaboratively verify whether the above condition holds. If it does not hold, then the declaration $x_i = 1$ is false.

- $x_i = 0$ is false. If $x_i = 0$ is correct, $b_i$ should be less than $c_i$. By (14), $b_i < c_i$ implies the existence of some $s_i^k \neq 0$ such that $\sum_{(s_i, b_i) < (s_i, b_i)} (x_i \cdot s_i^k) > q_k - s_i^k$. Therefore, any agent in $A_k$ can be able to verify whether the above condition holds. If it does not hold, then the declaration $x_i = 0$ is false.

Note that we do not need to perform false win declaration during the auction. It suffices to perform the detection once when the auction ends.

**IV. MUCA PROTOCOL**

Designing a MUCA protocol faces two primary challenges. One is how each bidder locally determines whether the bidder itself is a winner according to the given BRF. It is not trivial because bidders competing for a common resource may be winners at the same time. The other is to make each bidder independently figure out how much it should pay for the auction. This is not trivial for critical-value-based payments. This section addresses these two issues and also discusses agent’s bidding strategies with respect to different pricing schemes.

**A. Decentralized BRF-based Winner Determination**

We now describe the details of the proposed decentralized winner determination scheme. Because pricing does not affect the result of winner determination, the proposed scheme works for both first-price auctions and auctions with critical-value-based payment.

Each agent $a_i$ in the scheme is free to set up $b_i$ and $x_i$. By (15), any agent $a_i$’s best response (the setting of $b_i$ and $x_i$) depends on the relationship between $v_i = v_i(s_i)$ and $p_i$.

If $p_i > v_i$, then declaring a win (i.e., $x_i = 1$) will give $a_i$ a utility $u_i = v_i - p_i < 0$. Therefore, $a_i$ would rather declare $x_i = 0$. On the other hand, if $p_i < v_i$, then declaring $x_i = 1$ and setting $b_i$ to a value higher than $c_i$ will yield a utility $u_i = v_i - p_i > 0$. This is higher than what $a_i$ can get by declaring $x_i = 0$. Therefore, $a_i$’s best response is

$$BR_i = \begin{cases} x_i = 0, & \text{if } p_i > v_i, \\ x_i = 1 \text{ and } b_i \in (c_i, v_i), & \text{if } p_i < v_i. \end{cases}$$

In (16), $a_i$ does not set $b_i$ a value higher than $v_i$. The reason is obvious in case of first-price auctions. For critical-value-based payment, this can be justified by the following lemma.

**Lemma 1:** In case of critical-value-based payment, if an agent $a_i$ can win and get positive utility by setting $b_i$ to some value higher than $c_i$, the value of $b_i$ should not exceed $v_i$ if $a_i$ does not know when the auction will end.

**Proof:** The premise indicates that $p_i = c_i < v_i$. If this is the last bid of the auction, setting $b_i$ to some value higher than $c_i$ (say, $b'_i$) and winning the bid gives $a_i$ a utility $u_i = v_i - c_i > 0$. This holds regardless of whether $b'_i > v_i$. That is, picking up $b'_i > v_i$ does not give $a_i$ extra benefit compared with another selection $c_i < b'_i < v_i$. On the other hand, if this is not the last bid of the auction, other agents may raise their bids and thus collectively increase $b_i$’s critical value in the future. It is therefore possible that if $a_i$ picks up $b'_i > v_i$, it may face a critical value $c'_i$ in the future that is greater than $v_i$ and less than $b'_i$. At that time, declaring $x_i = 0$ is false and will incur a negative utility by R1 while declaring $x_i = 1$ will give $a_i$ a negative utility $u_i = v_i - c_i'$. Therefore, $a_i$ should never set $b_i$ a value higher than $v_i$.

By Lemma 1, the maximal bid that $a_i$ may place is $v_i$. Different agents may have different ideas about how to place their initial bids, so we assume that $b_i$ is an arbitrary value in $[a, v_i]$ initially, where $a$ is the starting bid set by the system. We also assume a minimum bid increment $\epsilon \geq 1$: whenever $a_i$ wants to raise $b_i$ to declare a win, $b_i$ should be at least $c_i + \epsilon$. The initial value of $x_i$ is not important, so it could be either 0 or 1. We assume that each agent $a_i$ broadcasts $(s_i, b_i, x_i)$ to all other agents in the beginning of the scheme so each agent $a_i$ has knowledge of $N_i$, $B$, and $(x_j)_{j=1}^i$ initially.

We assume that the logical channel between every agent and any of its neighbor delivers messages without loss and in a first-in-first-out (FIFO) manner. Each agent $a_i$ keeps a local copy of $(b_j, x_j)$ for each $a_j \in N_i$. When $a_i$ receives a new update of $(b_j, x_j)$ from another agent $a_j$, it executes Algorithm 2 as a response. $a_i$ first updates its knowledge about $b_j$ and $x_j$ (Line 2), and checks whether $a_i$ can win with its current bid by identifying $a_i$’s key predecessor.

**Definition 5 (Key Predecessor):** If $a_i$ can win with its current bid, $a_i$’s key predecessor is $a_i$ itself. Otherwise, $a_i$’s key predecessor is $a_k$ if $(s_k, b_k)$ is the request that ranks the lowest among all winning requests whose absence alone would make $(s_i, b_i)$ granted. Intuitively, $a_i$ can make $a_k$ “absent” in
Algorithm 2 Best Response of Agent $a_i$

1: **On receiving update($b'_i$, $x'_i$) from $a_j \in N_i$**
2: \[(b'_i, x'_i) \leftarrow (b_i, x_i)
3: \]
4: \[C \leftarrow \{(s_j, b_j) | a_j \in N_i \land x_j = 1 \land (s_j, b_j) \prec (s_i, b_i)\}
5: \]
6: \[k \leftarrow \text{key предшественник}(i, C)
7: \]
8: \[\text{if } k = i \text{ then}
9: \]
10: \[(b'_i, x'_i) \leftarrow (b_i, 1)
11: \]
12: \[\text{else}
13: \]
14: \[c_i \leftarrow \min\{c_i, b_i \prec (s_i, b_i)\}
15: \]
16: \[p_i \leftarrow c_i + \epsilon \text{ or } \exists j \ni \text{ get payment; } \epsilon: \text{ minimum allowable increment}
17: \]
18: \[\text{if } p_i < v_i \text{ then}
19: \]
20: \[b'_i \leftarrow b \in [c_i + \epsilon, v_i]; x'_i \leftarrow 1
21: \]
22: \[\text{else}
23: \]
24: \[\exists k \ni b_i \leq c_i \text{ and } \sigma \geq v_i
25: \]
26: \[(b'_i, x'_i) \leftarrow (b_i, 0)
27: \]
28: \[\text{end if}
29: \]
30: \[\text{end if}
31: \]
32: \[\text{end if}
33: \]
34: \[\text{end if}
35: \]
36: \[\text{end for}
37: \]
38: \[\text{end while}
39: \]

Therefore, $a_i$'s critical value $c_3$ is 93.33. The key predecessor of $a_5$ is $a_1$ because $a_5$'s request can be granted only if $a_5$ outbids $a_1$. Neither $a_2$ nor $a_4$ is $a_5$'s key predecessor, even though both outbid $a_5$.

Algorithm 3 Procedure key предшественник($i, C$)

1: \[\text{total}_i[k] \leftarrow 0 \text{ for all } k \in \{1, \ldots, m\}
2: \]
3: \[\text{while } C \neq 0 \text{ do}
4: \]
5: \[\text{Let } (s_j, b_j) \text{ be the request that has the highest rank in } C
6: \]
7: \[\text{for all } k \in \{1, \ldots, m\} \text{ do}
8: \]
9: \[\text{total}_i[k] \leftarrow \text{total}_i[k] + s^k_j
10: \]
11: \[\text{if } s^k_j > 0 \land s^k_i + \text{total}_i[k] > q_k \text{ then}
12: \]
13: \[\text{return } j
14: \]
15: \[\text{end if}
16: \]
17: \[\text{end for}
18: \]
19: \[\text{end while}
20: \]
21: \[\text{return } i
22: \]

Agent $a_i$ can outbid its key predecessor and thus win its request by setting $b_i$ to a value equal to or greater than $c_i + \epsilon$. However, whether it is worthy for $a_i$ to win depends on the relationship between $p_i$ and $v_i$. If $p_i < v_i$, $a_i$ can win and get positive utility by changing $b_i$ to some value in the range $[c_i + \epsilon, v_i]$ (to be discussed shortly). Otherwise, $a_i$ has no incentive to change $b_i$ because winning the request incurs a negative payoff (as $u_i = v_i - p_i < 0$). If $a_i$ ever changes $b_i$ or $x_i$, $a_i$ notifies all $a_i$'s competitors of the update.

Table IV shows a running example of the proposed approach, where three agents contend for two types of resources. Two agents turn out to be winners at the end of the auction.

### Table IV

<table>
<thead>
<tr>
<th>Bidder</th>
<th>Request vector ($s_k$)</th>
<th>Valuation ($t(v)$)</th>
<th>Initial ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>(1, 0)</td>
<td>9</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>$a_2$</td>
<td>(1, 1)</td>
<td>13</td>
<td>(8, 1)</td>
</tr>
<tr>
<td>$a_3$</td>
<td>(0, 1)</td>
<td>10</td>
<td>(1, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Agent</th>
<th>Old ($b_i, x_i$)</th>
<th>$c_i$</th>
<th>New ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_3$</td>
<td>(1, 0)</td>
<td>4</td>
<td>(5, 1)</td>
</tr>
<tr>
<td>2</td>
<td>$a_1$</td>
<td>(2, 1)</td>
<td>4</td>
<td>(5, 1)</td>
</tr>
<tr>
<td>3</td>
<td>$a_2$</td>
<td>(5, 1)</td>
<td>10</td>
<td>(11, 1)</td>
</tr>
<tr>
<td>4</td>
<td>$a_1$</td>
<td>(5, 1)</td>
<td>5.5</td>
<td>(7, 1)</td>
</tr>
<tr>
<td>5</td>
<td>$a_2$</td>
<td>(11, 1)</td>
<td>14</td>
<td>(11, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Bidder</th>
<th>Request vector ($s_k$)</th>
<th>Valuation ($t(v)$)</th>
<th>Initial ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>(1, 0)</td>
<td>9</td>
<td>(2, 1)</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>(1, 1)</td>
<td>13</td>
<td>(8, 1)</td>
<td></td>
</tr>
<tr>
<td>$a_3$</td>
<td>(0, 1)</td>
<td>10</td>
<td>(1, 0)</td>
<td></td>
</tr>
</tbody>
</table>

Therefore, $a_i$'s critical value $c_3$ is 93.33. The key predecessor of $a_5$ is $a_1$ because $a_5$'s request can be granted only if $a_5$ outbids $a_1$. Neither $a_2$ nor $a_4$ is $a_5$'s key predecessor, even though both outbid $a_5$.

### Table IV

<table>
<thead>
<tr>
<th>Bidder</th>
<th>Request vector ($s_k$)</th>
<th>Valuation ($t(v)$)</th>
<th>Initial ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>(1, 0)</td>
<td>9</td>
<td>(2, 1)</td>
</tr>
<tr>
<td>$a_2$</td>
<td>(1, 1)</td>
<td>13</td>
<td>(8, 1)</td>
</tr>
<tr>
<td>$a_3$</td>
<td>(0, 1)</td>
<td>10</td>
<td>(1, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Step</th>
<th>Agent</th>
<th>Old ($b_i, x_i$)</th>
<th>$c_i$</th>
<th>New ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$a_3$</td>
<td>(1, 0)</td>
<td>4</td>
<td>(5, 1)</td>
</tr>
<tr>
<td>2</td>
<td>$a_1$</td>
<td>(2, 1)</td>
<td>4</td>
<td>(5, 1)</td>
</tr>
<tr>
<td>3</td>
<td>$a_2$</td>
<td>(5, 1)</td>
<td>10</td>
<td>(11, 1)</td>
</tr>
<tr>
<td>4</td>
<td>$a_1$</td>
<td>(5, 1)</td>
<td>5.5</td>
<td>(7, 1)</td>
</tr>
<tr>
<td>5</td>
<td>$a_2$</td>
<td>(11, 1)</td>
<td>14</td>
<td>(11, 0)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Bidder</th>
<th>Request vector ($s_k$)</th>
<th>Valuation ($t(v)$)</th>
<th>Initial ($b_i, x_i$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_1$</td>
<td>(1, 0)</td>
<td>9</td>
<td>(2, 1)</td>
<td></td>
</tr>
<tr>
<td>$a_2$</td>
<td>(1, 1)</td>
<td>13</td>
<td>(8, 1)</td>
<td></td>
</tr>
<tr>
<td>$a_3$</td>
<td>(0, 1)</td>
<td>10</td>
<td>(1, 0)</td>
<td></td>
</tr>
</tbody>
</table>

**Bidding Strategy**

Because we design the MUCA protocol to be independent of the pricing scheme, $b_i$ is generally set to be in the range $[c_i + \epsilon, v_i]$ in Line 11 of Algorithm 2. If we set up a particular pricing scheme, agents may have different bidding strategies here.

In the first-price payment, each winner $a_i$ pays its winning bid, i.e., $p_i = b_i$. For this reason, the best response of each agent in the protocol is either to declare $x_i = 0$ or to minimize $b_i = c_i + \epsilon$. We call this bidding strategy minimal bid increment.
TABLE III
Key Predecessor and Critical Value Example with \( q = (3, 2, 2, 2) \)

<table>
<thead>
<tr>
<th>Bidder ( (a_i) )</th>
<th>Demand vector ( (s_i) )</th>
<th>Bid ( (b_i) )</th>
<th>BRF ( (b_i / \sum_{k=1}^{m} s_k^i) )</th>
<th>( x_i )</th>
<th>Key predecessor</th>
<th>Critical value ( (c_i) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( a_1 )</td>
<td>((1, 0, 1, 0, 0))</td>
<td>50</td>
<td>25.00</td>
<td>1</td>
<td>( a_1 )</td>
<td>-</td>
</tr>
<tr>
<td>( a_2 )</td>
<td>((0, 0, 0, 2, 1))</td>
<td>70</td>
<td>23.33</td>
<td>1</td>
<td>( a_2 )</td>
<td>-</td>
</tr>
<tr>
<td>( a_3 )</td>
<td>((0, 1, 0, 1, 2))</td>
<td>93</td>
<td>23.25</td>
<td>0</td>
<td>( a_2 )</td>
<td>93.33</td>
</tr>
<tr>
<td>( a_4 )</td>
<td>((2, 1, 1, 0, 0))</td>
<td>90</td>
<td>22.50</td>
<td>1</td>
<td>( a_4 )</td>
<td>100</td>
</tr>
<tr>
<td>( a_5 )</td>
<td>((1, 0, 2, 1, 0))</td>
<td>63</td>
<td>15.75</td>
<td>0</td>
<td>( a_2 )</td>
<td>-</td>
</tr>
</tbody>
</table>

In the critical-value-based payment, setting \( b_i \) to any value not less than \( c_i + \epsilon \) gives \( a_i \) the same payment \( p_i = c_i \) if \( a_i \) wins. To minimize its critical value \( c_i \), \( a_i \) needs to minimize the bids of non-winning competitors. Consider two possible bidding strategies: minimal bid increment (placing bid \( b_i = c_i + \epsilon \) as a response) and truthful bidding (setting \( b_i = v_i \) as \( a_i \)'s initial bid). The former usually causes incremental increases of competitor's bids. In contrast, announcing an agent's highest possible bid (i.e., its valuation) can make all non-winning competitors quit bidding at the earliest possible time in an ascending-price auction. As a result, the bids of non-winning competitors are minimized. Therefore, truthful bidding is every agent's weakly dominant strategy. Refer to the payoff matrix of winner\(^4\) shown in Table V.

When every agent bids truthfully, the outcome of Algorithm 2 will be equivalent to that of Algorithm 1. However, winner's payoff will also be low. Later we shall show that agent's bidding strategies do not affect the auction outcome (i.e., winners are always winners irrespective of their bidding strategies). Therefore, we do not assume any particular bidding strategy in our protocol design.

C. Payment Determination

After winner determination ends, each winner \( a_i \) should independently figure out how much it should pay for the auction. The task is trivial for first-price auctions (\( p_i = b_i \)). For the critical-value-based payment, the task is to identify \( a_i \)'s key successor.

Definition 6 (Key Successor): Let \( a_i \) be a winner. Bidder \( a_k \neq a_i \) is \( a_i \)'s key successor if \((s_k, b_k)\) is the request that ranks the highest among all non-winning requests that would be granted if \((s_i, b_i)\) were not present. If there is no such request, \( a_i \)'s key successor is defined to be \( a_i \) itself.

If \( a_i \)'s key successor is \( a_k \neq a_i \), \( a_i \)'s payment \( p_i \) is the minimal value of \( b_i \) that makes the rank of \((s_i, b_i)\) equal to or higher than that of \((s_k, b_k)\). If \( a_i \)'s key successor is \( a_i \) itself, then \( p_i = 0 \). This payment is exactly \( a_i \)'s critical value.

Let us revisit the example shown in Table III. Table VI shows the key successor and payment, respectively, for each winner. The key successor for \( a_2 \) is \( a_3 \) because \((s_3, b_3)\) would be granted if \((s_2, b_2)\) were not present. For \((s_2, b_2)\) to outrank \((s_3, b_3)\), \( b_2 \) should be greater than

\[
c_2 = \frac{b_3 (\frac{\sum_{k=1}^{m} s_k^i}{\sum_{k=1}^{m} s_k^j})^\alpha}{\sum_{k=1}^{m} s_k^j} = \frac{93 \times 3}{4} = 69.75.
\]

Therefore, \( a_2 \)'s payment \( p_2 \) is 69.75. The key successor for \( a_1 \) is \( a_2 \) itself because neither \( a_3 \)'s nor \( a_5 \)'s request would be granted if \( a_1 \)'s request were not present. Therefore, \( a_1 \)'s payment is 0.

The relationship between key predecessor and key successor is not symmetric. For example, \( a_1 \) is \( a_3 \)'s key predecessor in Table III but \( a_5 \) is not \( a_1 \)'s key successor here.

Key predecessor and key successor identifications need different knowledge of bid requests. In the former case, an agent \( a_i \) only needs bid requests from all agents in \( N_i \) (\( C \) in Algorithm 3). In contrast, \( a_i \) in the latter case should have knowledge of all other agent’s bid requests (\( C \cup D \) in Algorithm 4). The reason is that for \( a_j \) to determine whether \( a_j \in N_i \) is \( a_i \)'s key predecessor, \( a_i \) needs to know whether \( a_j \) does not win simply because of \( a_i \), or there is another winner that also prevents \( a_j \) from winning the bid. In the latter case, \( a_j \) is not \( a_i \)'s key successor.

Algorithm 4 Procedure key_successor\((i, B, x)\)

1: \( C \leftarrow \{(s_j, b_j) | (s_j, b_j) \prec (s_i, b_i)\} \)
2: \( \text{total}\_unit[k] \leftarrow 0 \) for all \( k \in \{1, \ldots, m\} \)
3: for all \((s_j, b_j) \in C\) such that \( x_j = 1 \) do
4: for all \( k \in \{1, \ldots, m\} \) do
5: \( \text{total}\_unit[k] \leftarrow \text{total}\_unit[k] + s_k^j \)
6: end for
7: end for
8: \( D \leftarrow \{(s_j, b_j) | (s_i, b_i) \prec (s_j, b_j)\} \)
9: while \( D \neq \emptyset \) do
10: Let \((s_j, b_j)\) be the request that has the highest rank in \( D \)
11: if \( x_j = 1 \) then
12: \( \text{total}\_unit[k] = \text{total}\_unit[k] + s_k^j \) for all \( k \)
13: else
14: \( \text{total}\_unit[k] + s_k^j \leq q_k \) for all \( k \), \( s_k^j > 0 \) then
15: return \( j \)
16: end if
17: end if
18: \( D \leftarrow D \setminus \{(s_j, b_j)\} \)
19: end while
20: return \( i \)
Algorithm 4 details how to identify the key successor for \( a_i \). The loop from Lines 3 to 7 first counts in all resource units that are allocated to winners who outbid \( a_i \). We skip resource allocation to \( a_i \) to mimic the absence of \( a_i \) in the auction. Then, the loop from Lines 9 to 19 checks all potential key successors \( a_j \) in the order of their bid-request ranks. If \( a_j \) is a winner, it cannot be a key predecessor so we just count in the amount of resource units allocated to it (Line 12). If \( a_j \) is not a winner, \( a_j \) is \( a_i \)'s key successor if its request can be granted with the residual capacity (Line 14).

It is a concern whether a winner \( a_i \) is possible to claim a lower payment \( c_j^f < c_j \). If this happens, \( a_i \)'s key successor, say, \( a_j \), will find out that \((s_j, b_j) \prec (s_i, c_i^f)\) and become a winner. Therefore, such a false claim is detectable.

### V. Protocol Analyses

In this section, we analyze whether the proposed protocol stabilizes (i.e., eventually stops), and, if it does, whether the outcome is correct (i.e., meeting the capacity constraint and conforming to the BRF-based winner determination rule) and consistent (with Algorithms 1 in terms of the set of winners). We also show the individual rationality property of the proposed protocol.

#### A. Stabilization

The protocol may potentially not stabilize because every time \( a_i \) changes \( b_i \) or \( x_i \), the rank of its bid request and thus the key predecessors of other agents may change. That may cause another agent’s reaction and change the set of (declared) winners. For convergence, we consider first how each agent’s knowledge about bids and win declarations evolves with time. Let \( b_i^t = (b_i^t, b_{i+1}^t, \ldots, b_n^t) \) and \( x_i^t = (x_i^t, x_{i+1}^t, \ldots, x_n^t) \) denote \( a_i \)'s knowledge of all \( b_j \)'s and \( x_j \)'s respectively, after the \( t \)-th execution of Line 17 of Algorithm 2 by \( a_i \). Let \( b_i^0 \) and \( x_i^0 \) be \( a_i \)'s initial knowledge. We assume \( b_i^0 = 0 \) and \( x_i^0 = 0 \), where \( t \geq 0 \), for all \( a_j \not\in N_i \). Because the execution of Line 17 of Algorithm 2 is triggered by a new update on some \( (b_j, x_j) \) and changes \( b_i \) or \( x_i \), we have \( b_i^t \neq b_i^{t+1} \) and \( x_i^t \neq x_i^{t+1} \) for all \( t \geq 0 \). A sequence \( \xi_i^t = (b_i^0, x_i^0), (b_i^1, x_i^1), (b_i^2, x_i^2), \ldots \) is a transition path of agent \( a_i \) that is of length \( t \).

**Theorem 3:** Any transition path of any agent is finite.

**Proof:** Consider any agent \( a_i \) and one of its transition paths \( \xi_i = (b_i^0, x_i^0), (b_i^1, x_i^1), (b_i^2, x_i^2), \ldots \). Since agents can only raise their bids and no agent places a bid higher than its valuation, there exists some integer \( t_i \) such that \( b_i^t \) no longer changes when \( t \geq t_i \). Although the values of \( t_i \) may be different for different \( a_i \)'s, eventually \( b_i^t \) will stabilize for all \( a_i \)'s. After that, the rank of bid requests is finalized. Without loss of generality, assume that \( (s_j, b_j) \prec (s_i, b_i^t+1) \) for all \( 1 \leq j < n \). We already know that no agent \( a_j \) has the incentive to falsify \( x_j \). It follows that the value of \( x_i \) will eventually stabilize. Because of this, the value of \( x_2 \) will eventually stabilize, and so on. Therefore, \( \xi_i \) must be finite.

#### B. Correctness

Because all update messages sent by \( a_i \) are delivered in the same order by all agents in \( N_i \), all these agents have the same knowledge of \((b_i, x_i)\) when the auction ends. Let \( (\hat{b}_i, \hat{x}_i) \) be the last value of \((b_i, x_i)\) sent by \( a_i \). When the protocol stabilizes, the outcome (i.e., the collective settings of \( b_j \)'s and \( x_j \)'s) is denoted by \( O = (\hat{b}, \hat{x}) \), where \( \hat{b} = (\hat{b}_1, \hat{b}_2, \ldots, \hat{b}_n) \) and \( \hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n) \). We consider \( O \) correct if it meets the capacity constraint and conforms to the BRF-based winner-determination rule defined below.

**Definition 7 (BRF-based Winner Determination Rule):** Let \( \prec \) be a total order defined by a BRF on bid requests \( B = \{s_i, b_i\}_{i=1}^n \). An outcome of the CA \( O = (\hat{b}, \hat{x}) \) conforms to the BRF-based winner determination rule if for every \( \hat{x}_i \), where \( 1 \leq i \leq n \), \( \hat{x}_i = 1 \) only if

\[
\sum_{j=1}^{n} (\hat{x}_j \cdot s_j^k) \leq q_k \quad \text{for all } s_j^k \neq 0. \tag{21}
\]

It is possible that two or more outcomes conform to the BRF-based winner determination rule. Table VII shows an example with two conforming outcomes. We have \((s_1, b_1) \prec (s_1, b_2) \) in the first outcome and \((s_1, b_1) \prec (s_2, b_2) \) in the second one. Both outcomes conform to the rule, though the bids are different.

With the following two theorems, we show that the proposed protocol always ends up with a correct outcome.

** Lemma 3:** Let \( O = (\hat{b}, \hat{x}) \) be an outcome of Algorithm 2. \( O \) conforms to the BRF-based winner determination rule, i.e., for every \( \hat{x}_i \), where \( 1 \leq i \leq n \), \( \hat{x}_i = 1 \) only if \( (\hat{b}, \hat{x}) \) holds.

**Proof:** Every execution of Algorithm 2 calls procedure *key_predecessor*. For every agent \( a_j \) such that \( \hat{x}_i = 1 \), the
call to key_predecessor in $a_i$’s last execution of Algorithm 2 returns either $j \neq i$ or $i$.

In the former case, $a_j$ is $a_i$’s key predecessor by Lemma 2. The result $\hat{x}_j = 1$ can only be set in Line 11 of Algorithm 2. In the same line, $a_i$ increases $b_i$ to some value $b_i$ such that $(s_i, b_i)$ outranks $(s_i, b_j)$. Because $(s_i, b_i) \prec (s_i, b_j)$, we have
\[
\sum_{(s_i, b_i) \prec (s_i, b_j)} (x_l \cdot s_l^i) \leq \sum_{(s_i, b_i) \prec (s_i, b_j)} (x_l \cdot s_l^j) \quad (22)
\]
for all $s_l^k \neq 0$. By (18), we have
\[
\sum_{(s_i, b_i) \prec (s_i, b_j)} (x_l \cdot s_l^i) + s_l^k \leq q_k \quad (23)
\]
In the latter case, $a_i$ must pass the while loop in Algorithm 3 without finding any $s_l^i \neq 0$ such that
\[
\sum_{(s_j, b_j) \prec (s_i, b_i)} (x_l \cdot s_l^j) + s_l^k > q_k, \quad (24)
\]
which is equivalent to (23).

In either case, any agent $a_i \neq a_i$ may later change $b_l$ or $x_l$. If any of those changes ever affected $a_i$’s best response, $(b_i, \hat{x}_i)$ would not be $a_i$’s final decision. Therefore, (23) still holds when the protocol ends. That is
\[
\sum_{(s_j, b_j) \prec (s_i, b_i)} (\hat{x}_l \cdot s_l^j) + s_l^k \leq q_k \quad \text{for all } s_l^k \neq 0. \quad (25)
\]

Lemma 4: Let $\mathcal{O} = (\bar{B}, \bar{x})$ be an outcome of Algorithm 2. $\mathcal{O}$ meets the resource capacity constraint specified in (2), i.e.,
\[
\sum_{i=1}^{n} (\hat{x}_l \cdot s_l^k) \leq q_k \quad \text{for all } k = 1, \ldots, m.
\]

Proof: For each $k, 1 \leq k \leq m$, let $(s_i, b_i)$ be the bid request that is of the lowest rank in $B$ such that $\hat{x}_i = 1$ and $s_i^k \neq 0$. This implies for all request $(s_j, b_j) \in B$ that ranks lower than $(s_i, b_i)$, we have either $s_j^k = 0$ or $\hat{x}_j = 0$. Therefore,
\[
\sum_{j=1}^{n} (\hat{x}_j \cdot s_l^k) = s_l^k + \sum_{(s_j, b_j) \prec (s_i, b_i)} (\hat{x}_j \cdot s_l^j).
\]
Because $\hat{x}_i = 1$, we know that (21) holds by Lemma 3. We thus have the proof.

Theorem 4: The outcome of Algorithm 2 is correct.

Proof: It directly follows from Lemmas 3 and 4.

C. Consistency

Even though Algorithm 2 converges and the outcome is correct, the outcome may deviate from that obtained by Algorithm 1 with the same BRF. Let $\bar{B} = (b_1, b_2, \ldots, b_n)$ and $\bar{B}' = (b'_1, b'_2, \ldots, b'_n)$ be two vectors that represent the final bids found by Algorithms 2 and 1, respectively. Because $0 \leq b_l \leq v_l$ for all $b_l \in \bar{B}$ and $b'_l = v_l$ for all $b'_l \in \bar{B}'$, we have $\bar{B} \leq \bar{B}'$. Consequently, the ranks of bid requests in these two algorithms can be different. In fact, Outcomes 1 and 2 in Table VII are exactly the results found by Algorithms 1 and 2, respectively. The ranks of bid requests are obviously different.

Despite the difference in ranks, we shall prove that Algorithm 2 is consistent in the sense that it identifies the same set of winners as Algorithm 1 using the same BRF.

Theorem 5: Let $\bar{B} = (b_1, b_2, \ldots, b_n)$ and $\bar{B}' = (b'_1, b'_2, \ldots, b'_n)$ be two vectors that represent the final bids found by Algorithms 2 and 1, respectively. Let $\hat{x} = (\hat{x}_1, \hat{x}_2, \ldots, \hat{x}_n)$ and $\hat{x}' = (\hat{x}'_1, \hat{x}'_2, \ldots, \hat{x}'_n)$ be two vectors that represent the final win declarations of Algorithms 2 and 1, respectively. If these two algorithms use the same BRF, then $\hat{x} = \hat{x}'$.

Proof: Without loss of generality, we assume that $(s_i, b'_i = v_i)$ is ranked $i$-th in Algorithm 1. However, the corresponding bid request $(s_i, b_i)$ is not necessarily ranked $i$-th in Algorithm 2. Let $c = (c_1, c_2, \ldots, c_n)$ and $c' = (c'_1, c'_2, \ldots, c'_n)$ be two vectors that represent the critical values for $(\bar{B}, \hat{x})$ and $(\bar{B}', \hat{x}')$, respectively.

By way of contradiction, assume that $\hat{x} \neq \hat{x}'$. Let $k$ be the smallest number such that $\hat{x}_k \neq \hat{x}'_k$. There are two possible cases.

- $(\hat{x}_k, \hat{x}'_k) = (0, 1)$. A necessary condition for $\hat{x}'_k = 1$ is
  \[
  \sum_{j=1}^{k} (\hat{x}_j \cdot s_l^j) \leq q_l \quad \text{for all } l \text{ such that } s_l^k \neq 0. \quad (26)
  \]
  Another necessary condition for $\hat{x}'_k = 1$ is $c'_k < v_k$. Because $\bar{B} \leq \bar{B}'$, we have $c_k \leq c'_k$. Together with the condition that $c'_k < v_k$, Algorithms 2 is free to set $\hat{x}_k$ to some value $b \in (c_k, v_k]$ and thus wins the auction. Therefore, it is impossible that $\hat{x}_k = 0$.

- $(\hat{x}_k, \hat{x}'_k) = (1, 0)$. The result $\hat{x}'_k = 0$ implies that
  \[
  s_k^l + \sum_{j=1}^{k-1} (\hat{x}_j \cdot s_l^j) > q_l \quad \text{for some } l \text{ such that } s_l^k \neq 0. \quad (27)
  \]
  Let $l$ be one such $l$. That is,
  \[
  \hat{x}_k + \sum_{j=1}^{k-1} (\hat{x}_j \cdot s_l^j) > q_l. \quad (28)
  \]
  If $k = 1$, which implies $s_1^l > q_l$, then $\hat{x}_1$ must be 0 as well by Lemma 4. Therefore, $k$ must be greater than 1. Let $A'_k = \{a_j | j < k, s_j^l \neq 0, \hat{x}_j = 1\}$ be the set of agents that also request $r_l$ with bids requests outranking $a_j$’s and getting granted by Algorithm 1. Let $B'_k = \{(s_j, b_j) | a_j \in A'_k\}$ be the set of bid requests submitted by all the agents in $A'_k$ when running Algorithm 2. If $(s_k, b_k)$ does not outrank any $(s_j, b_j) \in B_k$ in the end of Algorithm 2, then
  \[
  \hat{x}_k + \sum_{(s_j, b_j) \prec (s_k, b_k)} (\hat{x}_j \cdot s_l^j) \geq s_k^l + \sum_{j=1}^{k-1} (\hat{x}_j \cdot s_l^j) \quad (29)
  \]
  because $\hat{x}_j = \hat{x}_j'$ for all $j < k$. By (28) and (29), we have
  \[
  s_k^l + \sum_{(s_j, b_j) \prec (s_k, b_k)} (\hat{x}_j \cdot s_l^j) > q_l. \quad (30)
  \]
which implies that \( \hat{x}_k \) cannot be 1. Therefore, \((s_k, \hat{b}_k)\) must outrank some \((s_j, \hat{b}_j)\) in \(B_k\) to declare \(x_k = 1\). Let \((s_p, \hat{b}_p)\) be the bid request that ranks the lowest in \(B_k\). It has two properties. First, \(\hat{x}_p = 1\) because \(a_p \in A_k^p\) and \(\hat{x}_j = \hat{x}'_j\) for all \(j, 1 \leq j < k\). By Lemma 3, we then have

\[
\nu_p(s_i) = \sum_{j=1}^{m} (s_j^i \times v_{i,j}).
\]  

As mentioned, the proposed approach does not demand particular initial value of each \(x_i\). We thus tested three possible settings for the initial values of \(x_i\): all 1’s, all 0’s, and randomly selected 1’s and 0’s.

We assumed critical-value-based payments but did not assume any particular bidding strategy. More specifically, the new bid in Line 11 of Algorithm 2 was randomly selected from the range \([c_i + \epsilon, v_i]\). The value of \(\epsilon\) was set to a small number (0.001) to maximize the range of the new bid selection.

For each possible setting, we generated 100 test data and performed 10 trials for each data. Each result is an average over these 1000 trials.

A factor that affects the performance metrics is competition intensity (CI), the ratio of the total number of edges in the conflict graph to the maximum (i.e., \(n(n-1)/2\)). We fixed \(m, n\), and \(q_{\text{max}}\), and varied the value of \(p_s\) from 0.01 to 0.14 to adjust CI. Fig. 2a shows how CI changes with increasing \(p_s\).

When CI increased, the numbers of winners identified with both BRFs decreased, as Fig. 2b shows. Here \(w_{\text{m}}\) slightly outperformed \(w_n\) due to the consideration of the number of instances in its function definition. For a specific BRF, the centralized greedy approach and the proposed decentralized approach yielded exactly the same set of winners.

Figure 3 shows how total winning bid changes with respect to \(p_s\). For both BRFs, the centralized approach outperformed the decentralized counterpart, and the performance of the decentralized approach was not affected by the initial setting of \(x_i\)'s. Although there were more winners with \(p_s = 0.01\) than with \(p_s \geq 0.02\) in both BRFs, the total winning bid with \(p_s = 0.01\) was not always higher than that with \(p_s \geq 0.02\). The reason is that although there were more winners with \(p_s = 0.01\) than with \(p_s \geq 0.02\), each winner with \(p_s = 0.01\) generally requested fewer resource instances than that with \(p_s \geq 0.02\). Because each agent’s bid was roughly in proportion to the number of requested instances, we obtained the highest total winning bid with \(p_s = 0.02\) or \(p_s = 0.03\). When \(p_s\) increased further, the total winning bid decreased because the number of winners decreased significantly as Fig. 2b indicates.

Figure 4 shows how the total payment changes with increasing CI. Here all approaches exhibit behaviors similar to the result of total winning bid. However, the total payment is always lower than the total winning bid under any circumstance. The performance gap between the centralized and the proposed decentralized approach becomes smaller when CI is larger. This trend is also similar to that exhibited in Fig. 3.
We studied the convergence time of the decentralized approach by measuring the total number of moves taken by all bidders before reaching the final result. We did not count the initial setting and broadcast of bid requests; only changes of bid requests (sending of updates) count. Fig. 5 shows the results, which clearly depend on the initial settings of $x_i$'s. Generally speaking, the all-1s initial setting demanded fewer moves than the all-0s initial setting when there were more winners (i.e., when $p_a$ is small). On the other hand, the all-0s initial setting outperformed the all-0s initial setting when there were few winners (i.e., when $p_a$ is large). The random setting generally lies between these two extremes, and would be the best choice if we do not know the CI value beforehand.

Regardless of the initial setting of $x_i$'s, on average each agent took fewer than two moves before stability.

VII. CONCLUSIONS

We have proposed decentralized winner and payment determination protocols based on a given BRF for MUCA. It allows bidders to locally determine their bid and willingness to win by identifying their key predecessors. By exchanging that information with other competitors, other bidders can take moves so as to reach a consensus. For critical-value-based payment, winners determine their payments by identifying their respective key successors. We have proved that the proposed approach eventually stabilizes and is correct in the sense that the result meets the capacity constraint and conforms to the BRF-based winner-determination rule. We also proved that the proposed approach is consistent with
the centralized counterpart using the same BRF in the sense that both approaches identify the same set of winners. The proposed approach also ensures truthful bidding and individual rationality. Simulation results confirms the correctness and consistency of the proposed approach at the cost of lower total winning bid and payment.

REFERENCES


