On Bidding Algorithms for a Distributed Combinatorial Auction

Benito Mendoza* and José M. Vidal

Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
mendoza.usc@gmail.com, vidal@sc.edu

Abstract

Combinatorial auctions (CAs) are a great way to solve complex resource allocation and coordination problems. However, CAs require a central auctioneer who receives the bids and solves the winner determination problem, an NP-hard problem. Unfortunately, a centralized auction is not a good fit for real world situations where the participants have proprietary interests that they wish to remain private or when it is difficult to establish a trusted auctioneer. The work presented here is motivated by the vision of distributed CAs; incentive compatible peer-to-peer mechanisms to solve the allocation problem, where bidders carry out the needed computation. For such a system to exist, both a protocol that distributes the computational task amongst the bidders and strategies for bidding behavior are needed. PAUSE is combinatorial auction mechanism that naturally distributes the computational load amongst the bidders, establishing the protocol or rules the participants must follow. However, it does not provide bidders with bidding strategies. This article revisits and reevaluates a set of bidding algorithms that represent different bidding strategies that bidders can use to engage in a PAUSE auction, presenting a study that analyzes them with respect to the number of goods, bids, and bidders. Results show that PAUSE, along with the aforementioned heuristic bidding algorithms, is a viable method for solving combinatorial allocation problems without a centralized auctioneer.

Keywords: Multiagent Systems, Combinatorial Auctions, Resource Allocation.

*Corresponding author
11 Westchester Terrace
Annandale, NJ 08801
Tel. 803-447-6303

1 Introduction

Combinatorial auctions (CAs)—auctions that allow bids for bundles of items—provide a great way of allocating multiple distinguishable items amongst bidders whose perceived valuations for combinations of those items differ [3]. The bundle bidding enabled by these mechanisms allows bidders to benefit from combining the complementarities of the items being auctioned and to better express the value of any synergies.

Companies and governments around the world have left behind administered allocation systems, single item auctions, and other ad hoc mechanisms—traditionally used to sell valuable commodities, solve sourcing problems, or allocate scarce resources—to take advantage of the power of CAs, maximizing revenue and minimizing cost of sales [23]. For example, recently the Federal Communications Commission (FCC) has raised close to $20 billion in its 700-MHz auction using these mechanisms. CAs have been fundamentally changing the way valuable resources are sold, allowing the creation of electronic markets to supplement traditional sales channels like bilateral negotiated contracts, and also, creating totally new markets for scenarios where there was none before.

However, CAs require a central auctioneer who receives the bids from the bidders, carries out the needed computation to solve the winner determination problem (WDP)—finding the best allocation of items to bidders—and in many cases, gets a commission for the service [23]. Unfortunately, a centralized auction is not a good fit for real-world situations where the participants in a resource allocation have proprietary interests that they wish to remain private and secure—sometimes their bids represent valuable information like production cost—or when, simply, it is difficult to establish a trusted auctioneer. In addition, the winner determination problem is NP-hard problem [21], its complexity grows exponentially with the number of bids and items.

The motivation for the work presented here is the vision of **distributed combinatorial auctions**; incentive compatible peer-to-peer mechanisms where the bidders are the ones who solve the winner determination problem; and consequently, there is no need for a central auctioneer. For example, imagine a distributed combinatorial eBay—a Web 2.0 variation on this idea is known as zBay [24]—a distributed electronic market place where sellers can advertise their goods and buyers can place combinatorial bids and distributively find near optimal clearings. A more detailed example is a distributed combinatorial reverse auction in a B2B scenario—an auction used for buying instead of selling where the roles of buyers and sellers are reversed. Computer manufacturers (the buyers) publish their requirement for computer components such as memory chips, hard drives, processors, and mother boards. Computer component manufacturers (the sellers) develop agents which try to sell their particular goods by placing combinatorial bids to satisfy the buyers demands. Imagine that, in a first round, the sellers place bids for the items they want to sell. After that, the initial bids are disclosed to all the sellers. Then, they have the task of finding an allocation or deal (a set of bids) that satisfies all

the requirements of the buyers. Thus, using his own and other seller's bids, each seller agent searches for a set of bids for which they can get some utility, while satisfying buyers' requirements, and propose it as a deal. Notice that if the deals accepted by the buyer do not include a bid from a given agent, this agent gets zero utility. Thus, seller agents have to negotiate (indirectly) by proposing new bids or adjusting the price of their current ones to create deals where they maximize their own utility and at the same time reduce the price the buyer has to pay.

A system using the above described mechanism can also effectively and distributively calculate the solution to complex coordination problems. For such a system to exist, both a protocol that distributes the computational task amongst the bidders and strategies for bidding behavior are needed [17]. The PAUSE (Progressive Adaptive User Selection Environment) mechanism [10, 11] is an increasing price combinatorial auction that naturally distributes the problem of winner determination amongst the bidders in such a way that they have an incentive to perform the calculation. Thus, the task of the auctioneer is reduced to simply making sure that the bidders follow the rules established by PAUSE. It can even be envisioned to completely eliminating the auctioneer and, instead, have every agent perform the task of the auctioneer. A system implementing a PAUSE auction, would achieve much more efficient allocations than would be possible with sequential or simultaneous single item auctions—some of the approaches used to solve combinatorial auctions in a distributed way [7]—with no need to rely in a central auctioneer, with no need for the bidders to reveal their true valuations, and eliminating the exposure problem—the problem of exposing the bidders to the possibility that they will win some, but not all the items they desire [2].

PAUSE, as an auction mechanism, establishes the protocol or rules the participants must follow. However, it is not concerned with how the bidders determine what they should bid, that is, it does not provide a bidding strategy. In addition, little is known about the performance of the PAUSE auction under different type of problems or bids distributions. In [15, 16], a set of bidding algorithms, representing different bidding strategies that bidders can use to engage in a PAUSE auction, where introduced. This article presents a study that analyzes the scalability of those algorithms with respect to the number of goods, bids, and bidders. It also compares the revenue, the allocative efficiency, and the bidders' expected utility of the allocation found by PAUSE with those of the revenue-maximizing solution—as found by CASS [4], a well known centralized winner determination algorithm. The results of this study show that the PAUSE auction along with the developed heuristic bidding algorithms is a viable method for solving combinatorial allocation problems without the use a centralized auctioneer.

2 Related Work

Although the research of various aspects of combinatorial auctions is vast (for a good review, [3] is recommend), the study of distributed winner determination algorithms for combinatorial auctions is still relatively new. One way of distributing the WDP among the bidders is provided by the Virtual Simultaneous Auction (VSA) [4] which is based on market-oriented programming ideas [26]. The VSA assumes the bids themselves perform the computation. It is an iterative algorithm where successive auctions for the items are held and the bidders change their bids based on the last auction's outcome. The auction is guaranteed to find the optimal solution when the bidding terminates. Unfortunately, there is no guarantee that bidding will terminate and experimental results show that in most cases bidding appears to go on forever. Another approach consists of the algorithms for distributing the WDP in combinatorial auctions presented in [17], but these algorithms assume the computational entities are the items being sold and thus end up with a different type of distribution. In [19] the authors present a distributed mechanism for calculating VCG payments in a mechanism design problem. Their mechanism roughly amounts to having each agent calculate the payments for two other agents and give these to a secure central server which then checks to make sure results from all pairs agree, otherwise a re-calculation is ordered. This general idea, which they call the redundancy principle, could also be applied to our problem but it requires the existence of a secure center agent that everyone trusts. Another interesting approach is given in [18] where the bidding agents prioritize their bids, thus reducing the set of bids that the centralized winner determination algorithm must consider, making the problem easier. Finally, in the computation procuring clock auction [1] the agents are given an ever-increasing percentage of the surplus achieved by their proposed solution over the current best. As such, it assumes the agents are impartial computational entities, not the set of possible buyers as assumed by the PAUSE auction.

Approximate algorithms are one way to tackle the complexity of the WDP. One of the simplest methods is the greedy algorithm described in [12]. This greedy algorithm has been used to speed up an iterative combinatorial auction (which is also multistage auction) with noteworthy results [20]. In this work it is also suggested that with slightly less greedy strategy, approximate algorithms will give even further performance improvements to this kind of auctions. [27] presents an approximate algorithm to solve the WDP combining linear programming, a sequence of greedy algorithms, and hill-climbing to generate local improvements in the order of bids. In [8] different heuristics for sorting the bids are considered. Other meta-heuristics have been applied to improve the quality of the solutions, amongst them, a simulated annealing [5] and the stochastic local search technique [9] bringing some improvements in speed and quality of the solution's.

3 The PAUSE Auction

A PAUSE auction for m items has m stages. Stage 1 consists of having simultaneous ascending price open-cry auctions and during this stage the bidders can only place bids on individual items. At the end of this stage it is known what the highest bid for each individual item is and who placed that bid. Each successive stage $k = 2, 3, \ldots, m$ consists of an ascending price auction where the bidders must submit bidsets that cover all items but each one of the bids must be for k items or less. The bidders are allowed to use bids that other agents have placed in previous rounds when building their bidsets, thus allowing them to find better solutions. Also, any new bidset has to have a sum of bid prices which is bigger than that of the currently winning bidset. At the end of each stage k all agents know the best bid for every subset of size k or less. Also, at any point in time after stage 1 has ended there is a standing bidset whose value increases monotonically as new bidsets are submitted. Since in the final round all agents consider all possible bidsets, it is known that the final winning bidset will be one such that no agent can propose a better bidset. Note, however, that this bidset is not guaranteed to be the one that maximizes revenue since this is an ascending price auction so the winning bid for each set will be only slightly bigger than the second highest bid for the particular set of items. That is, the final prices will not be the same as the prices in a traditional combinatorial auction.

The PAUSE auction makes the job of the auctioneer very easy. All it has to do is verify that each new bidset has a revenue bigger than the current winning bidset and that every bid in an agent's bidset that is not its does indeed correspond to some other agents' previous bid. The computational problem shifts from one of winner determination to one of bid generation. Each agent must search over the space of all bidsets which contain at least one of its bids. The search is made easier by the fact that the agent needs to consider only the current best bids and only wants bidsets where its own utility is higher than in the current winning bidset. Each agent also has a clear incentive for performing this computation, namely, its utility only increases with each bidset it proposes (of course, it might decrease with the bidsets that others propose). Finally, the PAUSE auction has been shown to be envy-free in that at the conclusion of the auction no bidder would prefer to exchange his allocation with that of any other bidder [11].

It can even be envisioned to completely eliminating the auctioneer and, instead, have every agent perform the task of the auctioneer. That is, all bids are broadcast and when an agent receives a bid from another agent it updates the set of best bids and determines if the new bid is indeed better than the current winning bid. The agents would have an incentive to perform their computation as it will increase their expected utility. Also, any lies about other agents' bids are easily found out by keeping track of the bids sent out by every agent (the set of best bids). Namely, the only one that can increase an agent's bid value is the agent itself. Anyone claiming a higher value for some other agent is lying.

4 Problem Formulation: Bidding in the PAUSE Auction

In the PAUSE auction the bidders are the ones who carry out the computation for finding the allocation. Their job is to find the set of bids that form a valid solution (bidset) and that maximizes their utility. The agents maintain a set B of the current best bids, one for each set of items of size $\leq k$, where k is the current stage. At any point in the auction, after the first round, there will also be a set $W \subseteq B$ of currently winning bids. One restriction on the PAUSE auction is that the union of the items of all the currently winning bids should be equal to the set that contains all the available items. Thus, W is a set of bids that covers all the items and currently maximizes the revenue, where the revenue (r) of W is given by

$$r(W) = \sum_{b \in W} b^{\text{price}}. \tag{1}$$

Agent i's value function is given by $v_i(S) \in \Re$ where S is a set of items. Given an agent's value function and the current winning bidset W the agent's utility from W is calculated as

$$u_i(W) = \sum_{b \in W \mid b^{\text{agent}} = i} v_i(b^{\text{items}}) - b^{\text{price}}. \tag{2}$$

That is, the agent's utility for a bidset W is the value it receives for the items it wins in W minus the price it must pay for those items. If the agent is not winning any items then its utility is zero.

The goal of the bidding agents in the PAUSE auction is to maximize their utility, subject to the constraint that their next set of bids must have a total revenue that is at least ϵ bigger than the current revenue, where ϵ is the smallest increment allowed in the auction. Formally, given that W is the current winning bidset, agent i must find a g_i^* such that

$$g_i^* = \arg\max_{g \subseteq 2^B} u_i(g_i), \tag{3}$$

where each g_i is a set of bids that covers all items, $r(g_i) \geq r(W) + \epsilon$, and $\forall_{b \in g}\ (b \in B)$ or ($b^{\text{agent}} = i$ and $b^{\text{price}} > B(b^{\text{items}})$ and $|b^{\text{items}}|) \leq k$), and $B(b^{\text{items}})$ is the value of the bid in B for the set b^{items} (if there is no bid for those items it returns zero). That is, each bid b in g must satisfy at least one of the two following conditions:

1. b is already in B,
2. b is a bid of size less than or equal to k in which the agent i bids higher than the price for the bid in B for the same items.

The only thing missing is an algorithm that bidder agents can use to calculate their utility-maximizing bidset, g_i^*, for each agent. The next section presents a set of bidding algorithms for the PAUSE auction.

PAUSEBID(i, k)

1 $my\text{-}bids \leftarrow \emptyset$
2 $their\text{-}bids \leftarrow \emptyset$
3 **for** $b \in B$
4 **do if** $b^{\text{agent}} = i$ **or** $v_i(b^{\text{items}}) > b^{\text{price}}$
5 **then** $my\text{-}bids \leftarrow my\text{-}bids +$**new** Bid$(b^{\text{items}}, i, v_i(b^{\text{items}}))$
6 **else** $their\text{-}bids \leftarrow their\text{-}bids + b$
7 **for** $S \in$ subsets of k or fewer items such that $v_i(S) > 0$ and $\neg\exists_{b \in B} b^{\text{items}} = S$
8 **do** $my\text{-}bids \leftarrow my\text{-}bids +$**new** Bid$(S, i, v_i(S))$
9 $bids \leftarrow my\text{-}bids + their\text{-}bids$
10 $g^* \leftarrow \emptyset$ ▷ Global variable
11 $u^* \leftarrow u_i(W)$ ▷ Global variable
12 PBSEARCH$(bids, \emptyset, k)$
13 $g^* \leftarrow$ DISTRIBUTEPAYMENTS(i, g^*)
14 **return** g^*

Figure 1: The PAUSEBID algorithm which implements a branch and bound search. i is the agent and k is the current stage of the auction, for $k \geq 2$.

5 Bidding Strategies

The first stage of a PAUSE auction consists of several English auctions, where the bidders submit bids on individual items. In this case, an agent's best response is to bid ϵ higher than the current winning bid until it reaches its valuation for that particular item. The algorithms presented here focus on the subsequent stages: $k > 1$. When $k > 1$, the agents have to find g_i^*. This can be done by performing a complete search on B. However, this approach is computationally expensive since it produces a large search tree. These algorithms represent alternative strategies to overcome this expensive search.

5.1 PAUSEBID, a myopic-optimal bidding algorithm

In the PAUSE auction, a myopic-optimal utility-maximizing bidding strategy guarantees to find the bidset that maximizes the agent's utility given the set of outstanding best bids B at any given time (if one exist), without considering possible future bids. The PAUSEBID algorithm implements this strategy [15].

The PAUSEBID algorithm (shown in Figure 1) uses a branch and bound technique to prune the search tree. Given that bidders want to maximize their utility and that at any given point there are likely only a few bids within B which the agent can dominate,

PAUSEBID starts by defining $my\text{-}bids$ to be the list of bids for which the agent's valuation is higher than the current best bid, as given in B. It sets the value of these bids to be the agent's true valuation—but the agents won't necessarily be bidding their true valuation, as explained later. Similarly, the algorithm sets $their\text{-}bids$ to be the rest of the bids from B. Finally, the agent's search list is simply the concatenation of $my\text{-}bids$ and $their\text{-}bids$. Note that the agent's own bids are placed first on the search list as this will enable us to do more pruning (PAUSEBID lines 3 to 9). The agent can now perform a branch and bound search on the branch-on-bids tree produced by these bids, implemented by PBSEARCH (Figure 2).

The bound PAUSEBID uses is the maximum utility that the agent can expect to receive from a given set of bids. It is called u^*. Initially, u^* is set to $u_i(W)$ (PAUSEBID line 11), where W is the current winning bidset, since that is the utility the agent currently receives and any solution he proposes should give him more utility. If PBSEARCH ever comes across a partial solution where the maximum utility the agent can expect to receive is less than u^* then that subtree is pruned (PBSEARCH line 20). The maximum utility can be determined only after the algorithm has searched over all of the agent's own bids (which are first on the list) because after that it is known that the solution will not include any more bids where the agent is the winner thus the agent's utility will no longer increase. The calculation of the minimum payment is shown in line 18 for the partial solution case and line 8 for the case where having a complete solution in PBSEARCH. Note that in order to calculate the $min\text{-}payment$ for the partial solution case it is needed an upper bound on the payments that the agents must make for each item. This upper bound is provided by

$$h(S) = \sum_{s \in S} \max_{\{b \in B \mid s \in b^{\text{items}}\}} \frac{b^{\text{price}}}{\left|b^{\text{items}}\right|}. \tag{4}$$

This function produces a bound identical to the one used by the Bidtree algorithm [22]—it merely assigns to each individual item in S a value equal to the maximum bid in B divided by the number of items in that bid.

To prune the branches that cannot lead to a solution with revenue greater than the current W, the algorithm considers both the values of the bids in B and the valuations of the agent. Similarly to (4)

$$h_i(S,k) = \sum_{s \in S} \max_{\{S' \mid s \in S' \wedge v_i(S') > 0 \wedge |S'| \leq k\}} \frac{v_i(S')}{\left|S'\right|} \tag{5}$$

is defined. Which assigns to each individual item $s \in S$ the maximum value produced by the valuation of S' divided by the size of S', where S' is a set of items for which the agent has a valuation greater than zero, contains s, and its size is less or equal than k. The algorithm uses the heuristics h and h_i (lines 14 and 18 of PBSEARCH), to prune the

```
PBSEARCH(bids, g, k)
 1  if bids = ∅ then return
 2  b ← first(bids)
 3  g ← g + b
 4  Ī_g ← items not in g
 5  if g does not contain a bid from i
 6     then return
 7  if g includes all items
 8     then min-payment ← max(0, r(W) + ε − (r(g) − r_i(g)), Σ_{b∈g | b^agent=i} B(b^items))
 9          max-utility ← v_i(g) − min-payment
10          if r(g) > r(W) and max-utility ≥ u*
11            then g* ← g
12                 u* ← max-utility
13          PBSEARCH(bids, g − b) ▷ b is Out
14     else max-revenue ← r(g) + max(h(Ī_g), h_i(Ī_g, k))
15          if max-revenue ≤ r(W)
16            then PBSEARCH(bids, g − b) ▷ b is Out
17          elseif b^agent ≠ i
18            then min-payment ← (r(W) + ε) − (r(g) − r_i(g)) − h(Ī_g)
19                 max-utility ← v_i(g) − min-payment
20                 if max-utility > u*
21                   then PBSEARCH({x ∈ bids | x^items ∩ b^items = ∅}, g) ▷ b is In
22                 PBSEARCH(bids, g − b) ▷ b is Out
23            else
24                 PBSEARCH({x ∈ bids | x^items ∩ b^items = ∅}, g) ▷ b is In
25                 PBSEARCH(bids, g − b) ▷ b is Out
26  return
```

Figure 2: The PBSEARCH recursive procedure where $bids$ is the set of available bids and g is the current partial solution.

just mentioned branches. A final pruning technique implemented by the algorithm is ignoring any branches where the agent has no bids in the current answer g and no more of the agent's bids are in the list (PBSEARCH lines 5 and 6).

The resulting g^* found by PBSEARCH is thus the set of bids that has revenue bigger than $r(W)$ and maximizes agent i's utility. However, agent i's bids in g^* are still set to its own valuation and not to the lowest possible price. If the agent has only one bid b in g^* then it is simply a matter of reducing the price of that bid (b^{price}) by u^* from the

DISTRIBUTEPAYMENTS(i, g)

1 $surplus \leftarrow \sum_{b \in g \mid b^{\text{agent}}=i} b^{\text{price}} - B(b^{\text{items}})$
2 **if** $surplus > 0$
3 **then** $min\text{-}payment \leftarrow \max(0, r(W) + \epsilon - (r(g) - r_i(g)), \sum_{b \in g \mid b^{\text{agent}}=i} B(b^{\text{items}}))$
4 **for** $b \in g \mid b^{\text{agent}} = i$
5 **do if** $min\text{-}payment \leq 0$
6 **then** $b^{\text{price}} \leftarrow B(b^{\text{items}})$
7 **else** $b^{\text{price}} \leftarrow B(b^{\text{items}}) + min\text{-}payment \cdot \frac{b^{\text{price}} - B(b^{\text{items}})}{surplus}$
8 **return** g

Figure 3: The DISTRIBUTEPAYMENTS function distributes the payments of the bids agent i, included in g, proportionally to the agent's true valuation for each set of items.

agent's true valuation ($b^{\text{price}} = v_i(b^{\text{items}}) - u^*$). However, if the agent has more than one bid then it faces the problem of how to distribute its payments among these bids. Although, there might be many ways of distributing the payments, there does not appear to be a dominant strategy for performing this distribution. Thus, it has been chosen to distribute the payments in proportion to the agent's true valuation for each set of items.

The function DISTRIBUTEPAYMENTS, shown in Figure 3, is responsible for setting the agent's payments so that it can achieve its maximum utility u^*, by following the above mentioned approach. It first calculates the agent's surplus from the bids in g (line 1). If the surplus is zero, there is nothing to do, since the price of the bids cannot be reduced. However, if the bidder has a surplus, the minimum amount the agent has to pay ($min\text{-}payment$) in order to satisfy the revenue restriction (the minimum increment for $r(W)$ has to be ϵ) is calculated (line 3). The $min\text{-}payment$ is the maximum value among zero, the difference between the revenue restriction ($r(W) + \epsilon$) and the amount of the revenue paid from other agents ($r(g) - r_i(g)$), and the sum of the current prices of the agent's bids ($\sum_{b \in g \mid b^{\text{agent}}=i} B(b^{\text{items}})$). Finally, if the $min\text{-}payment$ is less or equal than zero then all the bids are set to $B(b^{\text{items}})$, the current highest price (line 6), otherwise the price of each bid is set proportional to the bid's surplus (line 7).

Notice that distributing the agent's payments proportionately to the agent's valuation for that set of items has the unfortunate effect of revealing, to some extent, the agent's true relative valuation for the items. For example, if an agent increases his bids for two sets of items, but his increase for the first set is much greater than for the second set then it can be deduced that the agent valuates the first set much higher than the second. This knowledge could then, perhaps, be used by a strategic agent to place his own bids. However, based on previous work on agent modeling [25], it is believed, that such strategic thinking will incur in large computational costs and will deliver small

utility gains. But, even if this belief proves wrong, it is a simple matter of changing the surplus distribution method to include some randomness. Of course, even with random distributions, the fact that an agent increases the his bid for certain subsets of items is still a clear signal that its valuation of those subsets is higher than the current price (perhaps, a lot higher?). An opponent might be able to use this knowledge to make better decisions about which sets of items he should bid on.

Because of these strategic issues it cannot be claimed that the PAUSEBID strategy is a dominant strategy: the best strategy to use regardless of the other agents' strategies. However, we it can be claimed that at each time it is called it returns the bid that maximizes the agent's utility while still having a revenue greater than the current solution and increasing the agent's utility over the one it is currently receiving.

5.2 GREEDYPAUSEBID, an approximate bidding algorithm

Approximate algorithms forgo optimality in favor of heuristics and simple local searches which deliver a solution very quickly. The approximate bidding algorithm presented here is based in strategies used by a well-known approximate algorithm for solving the WDP for centralized CAs, the greedy algorithm described in[12]. The greedy algorithm is a very simple linear time algorithm and can be summarized into two steps.

1. The bids are sorted by $b^{\text{price}}/|b^{\text{items}}|^c$ for some number c, $0 \leq c \leq 1$. The authors showed that $c = 0.5$ is the approximate best value, it guarantees an approximation ratio of at least $\sqrt{m}$, where m is the number of goods.

2. Proceed down the sorted list of bids accepting bids if the goods in demand are still unallocated and not conflicted, where bids b_j and b_k conflict if $b_j^{\text{items}} \cap b_k^{\text{items}} \neq \emptyset$.

The GREEDYPAUSEBID algorithm (Figure 4) implements the idea discussed above maximizing the bidder's utility, instead of the seller's revenue, under the condition that the resulting revenue has to be greater or equal than $r(W) + \epsilon$. It starts by defining $my\text{-}bids$ to be the list of bids for which the agent's valuation is higher than the current best bid, as given in B. As in PAUSEBID, it sets the value of these bids to be the agent's true valuation. Similarly, it sets $their\text{-}bids$ to be the rest of the bids from B. If $my\text{-}bids$ is empty, there is no bid that the agent can dominate at this time and the algorithm ends. The function SORTFORGREEDY (called in lines 12 and 16) sorts the list of bids received as first parameter by $b^{\text{price}}/|b^{\text{items}}|^c$. After $my\text{-}bids$ is sorted, it takes the first bid and add it to the initial bidset g, to make sure that the solution includes the bid from $my\text{-}bids$ with the highest rank. Finally, to complete the allocation containing all the items, the agent's search list is simply the concatenation of $their\text{-}bids$ and the rest of $my\text{-}bids$ sorted again by the same criteria (lines 16 and 15 respectively). After it finishes walking down the $bids$ list, it has an allocation g. However, agent i's bids in g are still set to his own valuation and not to the lowest possible price. If $r(g) \leq r(W) + \epsilon$, then the algorithm

```
GREEDYPAUSEBID(i, k, c)
 1  my-bids ← ∅
 2  their-bids ← ∅
 3  for b ∈ B
 4       do if b^agent = i or v_i(b^items) > b^price
 5            then my-bids ← my-bids +new Bid(b^items, i, v_i(b^items))
 6            else  their-bids ← their-bids +b
 7  for S ∈ subsets of k or fewer items such that v_i(S) > 0 and ¬∃_{b∈B} b^items = S
 8       do my-bids ← my-bids +new Bid(S, i, v_i(S))
 9  g ← ∅
10  if my-bids = ∅
11     then return g
12  my-bids ← SORTFORGREEDY(my-bids, c)
13  b ← first(my-bids)
14  g ← g + b
15  bids ← their-bids +rest(my-bids)
16  bids ← SORTFORGREEDY(bids, c)
17  while bids ≠ ∅
18       do b ← first(bids)
19          bids ← rest(bids)
20          I_g ← items in g
21          if b^items ∩ I_g = ∅
22            then g ← g + b
23                 bids ← {x ∈ bids | x^items ∩ b^items = ∅}
24  if r(g) > r(W) + ε
25     then g ← DISTRIBUTEPAYMENTS(g)
26          if u_i(g) ≤ u_i(W)
27            then g ← ∅
28     else  g ← ∅
29  return g
```

Figure 4: The GREEDYPAUSEBID algorithm returns an empty bidset if the solution it finds does not improve the utility of bidder i. c is the bids sorting factor and k is the current stage of the auction, for $k \geq 2$.

ends. Otherwise (when $r(g) > r(W)+\epsilon$), it calls the procedure DISTRIBUTEPAYMENTS with g as parameter (line 25), the same method used by PAUSEBID. After distributing the payments of g the algorithm ends by returning g if the utility that the agent receives from g is greater than that it gets from W, otherwise it returns an empty bidset.

5.3 The GREEDYPAUSEBID+HILL algorithm

A simple extension to the greedy approach consists of using a local search algorithm that continuously updates the initial allocation found by the greedy algorithm [6], thus locally searching in the remaining bids to improve the solution. This idea is implemented in the GREEDYPAUSEBID+HILL algorithm. This algorithm starts with the solution provided by GREEDYPAUSEBID and then explores the neighborhood of that solution, using a simple *hill climbing*, looking for allocations that generate a higher utility for the bidder. It consist of two main steps:

1. Call GREEDYPAUSEPID algorithm with appropriate input and $c = 0.5$.
2. If the solution g returned by GREEDYPAUSEBID is not empty then call HILL-CLIMBING with $bids$ = my-$bids$ + $their$-$bids$ sorted by c.

After having an algorithm that can find g_i^*, it would be of great interest to analyze the solutions that this algorithm and the PAUSE auction generate. It would be also interesting determining how long it would take for populations of agents using this algorithm to arrive at a solution, as well as quantifying the bidders expected utility of this solution. Also it would be interesting to know if the agents in a PAUSE auction arrive at the revenue-maximizing solution. The following sections present a set of metrics and the results of an experimental analysis of all this interesting points.

6 Experimental Setup

The goal of the experimental analysis presented here is to observe the computational scalability and the efficiency of the PAUSE auction in different economic scenarios or problems, when bidders use the bidding algorithms presented in previous sections. Thus, it aims to identify the type of problems where the PAUSE auction is more suitable. In general, the time required to solve the WDP is tied to the number of bids and goods in the auction. In the PAUSE auction there is one more factor involved, the number of bidders; since bidders are the ones that actually solve the problem. Using several combinations of these factors, the experiment tests the three bidding algorithms under four different problems obtained from CATS, as shown in Section 6.3. CATS, described in Section 6.1, is a generator of combinatorial auction instances widely used as testbed

for winner determination algorithms [13]. Finally, using the metrics explained in Section 6.2, the solutions obtained by this experiment are analyzed from different points of view—the seller's, the buyer's, and the designer's point of view.

6.1 Economic distributions

The Combinatorial Auction Test Suite (CATS) [13] features five bid distributions that represent instances of problems from realistic, economically motivated domains, as well as a collection of artificial distributions that have been widely used in the literature. In [14] an empirical experiment shows that solving the WDP with some bid distributions is harder than with others; in their experiment they use a centralized method based on a linear programing solver. The same job presents a classification of ten of the CATS' distributions by gross hardness, which corresponds to the running time required to solve the WDP. Four of the ten CATS distributions were carefully selected to conduct this experiment, making sure to cover the whole spectrum of hardness.

- **Scheduling (*Temporal Adjacency*)**. The motivation of this distribution is the problem of *distributed job-shop scheduling with one resource*. This represents problems where the bidders want to use the resource for a given number of time units. They have one or more deadlines with different values for them. It is assumed that all jobs are eligible to start in the first time-slot and each job is allocated continuous time on the resource. This bid distribution is the second easiest amongst all the CATS distributions and allows to place XOR bids.

- **L2 (*weighted random*)**. An artificial distribution with no economic motivation. The level of gross hardness of this distribution is in the middle of the whole gross hardness spectrum.

- **Arbitrary (*arbitrary relationships*)**. The motivation of this distribution is the type of problems where the goods do not give rise to a notion of adjacency, but regularity in complementarity relationships can still exist, for example, physical objects like collectables, semiconductors, etc. This distribution has the third place in gross hardness amongst all the CATS distributions and allows XOR bids.

- **L3 (*uniform*)**. This is another artificial distribution, no economic motivation, and is the hardest to solve amongst all the CATS distributions.

6.1.1 Using CATS files to feed the decentralized combinatorial auction

CATS was created to test centralized algorithms that find the winning bidset in combinatorial auctions. These algorithms ignore the identity of the bidders, focusing instead on the bids. Their goal is to find the allocation of items to bidders that maximizes revenue

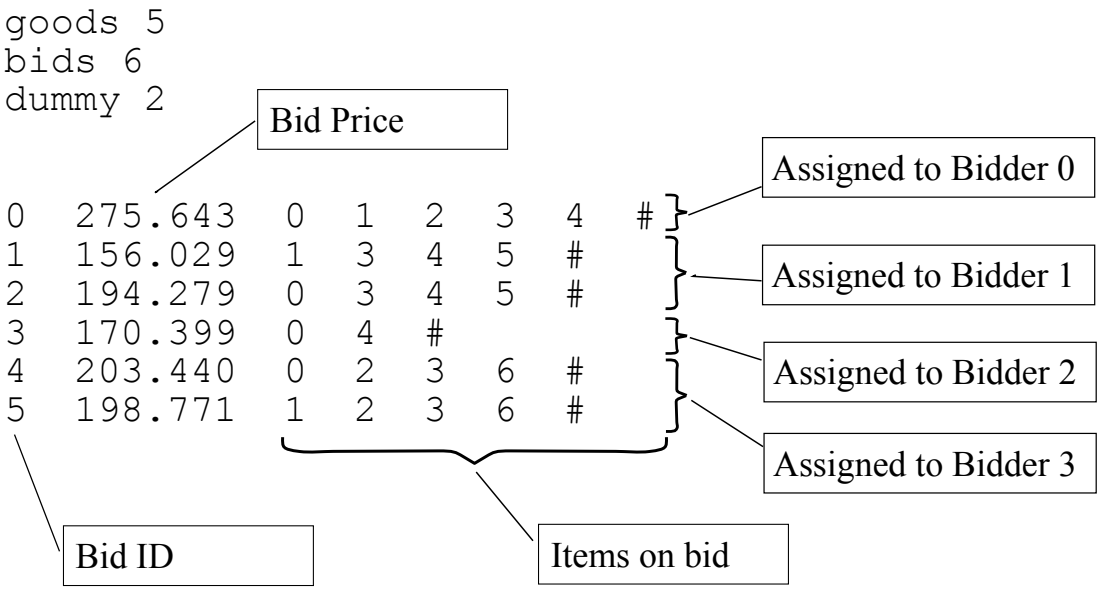


Figure 5: An example of the content of a CATS file and how it has been used. It contains six bids enumerated from 0 to 5; five items, from 0 to 4; and two dummies, 5 and 6.

regardless of who placed those bids. For this reason, CATS does not identify the bidder that placed the bid, nor does provides with the bidder's utility function, it only includes a bid *id*, the price of the bid, and the set of items in the bid. To implement a PAUSE auction it is needed to know every bidder's valuations. In order to use a CATS file in the PAUSE auction testbed without changing the economic distribution, it is necessary to assign each bid from the file bid to a different bidder, except when the bids are XOR bids. The XOR bids in the CATS file include an easily identifiable dummy item. All bids with the same dummy item are assigned to one bidder. In addition, CATS does not offer a direct way to control the number of bidders. The only way to do this is by generating a file and then check how many bidders can be obtained. For instance, if 100 runs with 10 bidders are needed, a number much bigger than 100 problem instances have to be generated; throwing out all those instances having a number of bidders different than 10. Figure 5 shows an example of a CATS file and how it was used to create a data set for the experiments presented here.

6.2 Metrics

In order to determine the scalability of the PAUSE auction using the bidding algorithms previously introduced, the *running time* required to clear an auction is analyzed. Know-

ing how the PAUSE auction scales based on the bidding strategy is of interest not only to the seller and the buyer, but also the auction designer.

In an auction the seller wants to maximize the *revenue*. Thus, the revenue generated by the PAUSE auction is compared to the maximum possible revenue for that auction—assuming truthful bidding, the way in which CATS is usually used to evaluate algorithms to solve the WDP. This is done by calculating a **revenue ratio**.

$$revenueRatio(W_s) = \frac{r(W_s)}{r*}, \tag{6}$$

where r^* is the optimal revenue, which was obtained by using the Combinatorial Auctions Structured Search (CASS) algorithm [4], and $r(W_s) = \sum_{b \in W_s} b^{\text{price}}$ is the revenue generated by the allocation W_s found by the PAUSE auction using bidding strategy s. It is known beforehand that the prices paid in PAUSE are lower than those paid in a centralized first price sealed-bid combinatorial auction when assuming truthful bidding, because PAUSE is an English auction. Thus, the prices paid are roughly the second highest price plus some ϵ—the minimum bid increment allowed in the auction. In the experiments ϵ is set to be 1.

A designer would be interested also in the *(allocative) efficiency* of the solutions found by the PAUSE auction. The revenue-maximizing solution is efficient since the bidders with highest valuation (paying higher) are the ones that win the items. In order to compare the efficiency, the **efficiency ratio** is defined as the ratio of the sum of the valuations of the winning bidders for the bids they win to the maximum revenue.

$$efficiencyRatio(W_s) = \frac{\sum_{b \in W_s} v_{b^{\text{agent}}}(b^{\text{items}})}{r*}, \tag{7}$$

where $v_{b^{\text{agent}}}(b^{\text{items}}) \in \Re$ is the valuation that the bidder b^{agent}, the winner of bid b, has for items b^{items}. It gives the efficiency of the allocation W_s found by PAUSE using the bidding algorithm s as compared to $r*$, the optimal revenue. A ratio of 1 means that the PAUSE solution has allocated the items to the same buyers that they are allocated to in the revenue-maximizing allocation—or at least to a set of buyers with the same sum of valuations.

When assuming truthful bidding in a centralized first price sealed-bid auction the winners pay their true valuation, thus their utility is zero. In the PAUSE auction, the bidders' utility can be greater than zero, since as mentioned before, the winners end up paying less, given that PAUSE is an increasing price auction. When considering whether or not to participate in a PAUSE auction, bidders would be more interested in knowing the *expected utility* from switching to the PAUSE auction or, more precisely, the expected utility of choosing amongst the different bidding algorithms. Thus, the **bidders' expected utility ratio** is calculated by dividing the sum of the bidders utility by $r*$.

$$expectedUtilityRatio(W_s) = \frac{\sum_{b \in W_s} u_{b^{\text{agent}}}(b)}{r*}, \tag{8}$$

	Bidders	Bids	Goods
for Bidders$*$	(4 to 10)	20	10
for Bids	5	(12-20)	10
for Goods	5	20	(10 to 15)

Table 1: Combination of parameters used to create the data set for each one of the CATS distributions described in the previous section. $*$The number of bidder cannot be manipulated in the L2 and L3 distributions, in these cases the number of bidders is equal to the number of bids.

where $u_{b^{\text{agent}}}(b) = v_{b^{\text{agent}}}(b^{\text{items}}) - b^{\text{price}}$ is the utility obtained by bidder b^{agent}, who is wining bid b; and as before, W_s is the allocation found by PAUSE using the bidding algorithm s and $r*$ is the optimal revenue.

6.3 Experimental settings

The experiments presented here consist of combinations of the parameters mentioned at the beginning of this section. A dataset for each one of the CATS distribution mentioned above was created. Each data set contains 100 bid files or auctions for some combinations of number of bidders, bids, and goods. Table 1 shows the combinations used in the experiments. To analyze the effect of one variable the value of the other two was fixed.

For each bidding strategy, a PAUSE auction over each distribution was carried out. It is important to remember that in each auction all the bidders use the same bidding algorithm. The experiments where carried on in an SGI Altix 4700 with 128 Itanium Cores @ 1.6 GHz/8MB Cache and 256 GB of RAM (shared-memory system).

7 Results

In this section, the findings of the experiments are shown. The average value of each one of the metrics mentioned before is calculated and plotted for its analysis.

7.1 Time

To exactly determine how the running time of each bidding strategy increases, a regression analysis was carried out. The curve of the corresponding running time is calculated as,

$$y = b \times m^{x_i}, \tag{9}$$

where the dependent variable y (average time in this case) is a function of the independent variable x under analysis (the number of bidders, bids, or goods), b is the slope

Distribution	Bidding Algorithm	m	b
Arbitrary	PAUSEBID	1.115	1848.56
	GREEDYPAUSEBID	1.094	2.682
	GREEDYPAUSEBID+HILL	1.087	24.738
Scheduling	PAUSEBID	2.718	6.474
	GREEDYPAUSEBID	1.381	0.208
	GREEDYPAUSEBID+HILL	1.374	2.413

Table 2: The slope coefficient b and the intercept coefficient m, obtained by logarithmic regression, that describe the curve of *average time* for each bidding strategy as a function of the number of $bidders$, under different CATS distribution.

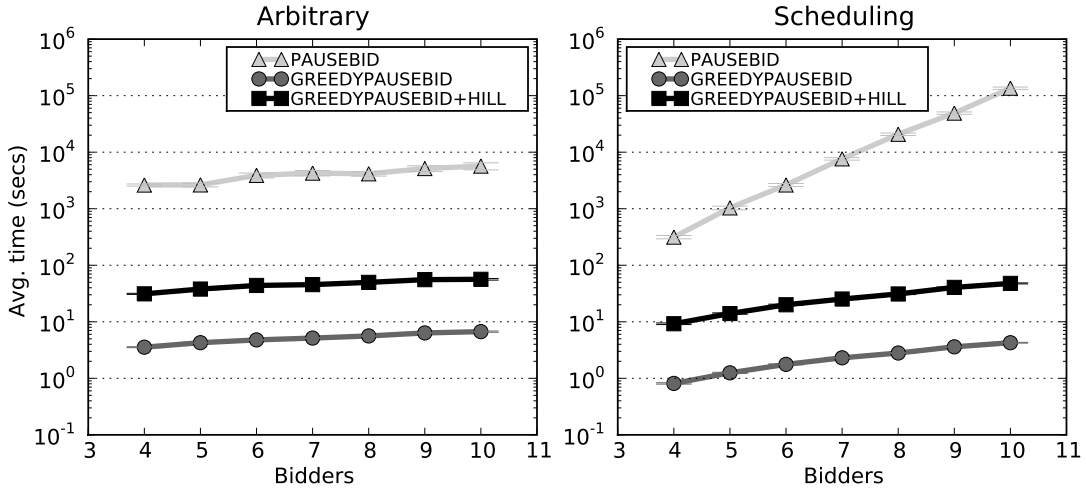
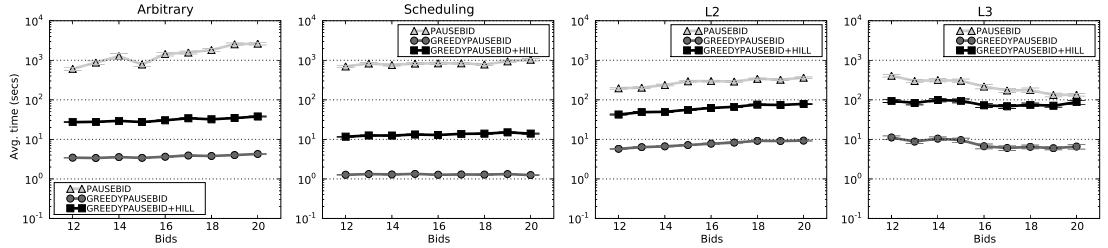


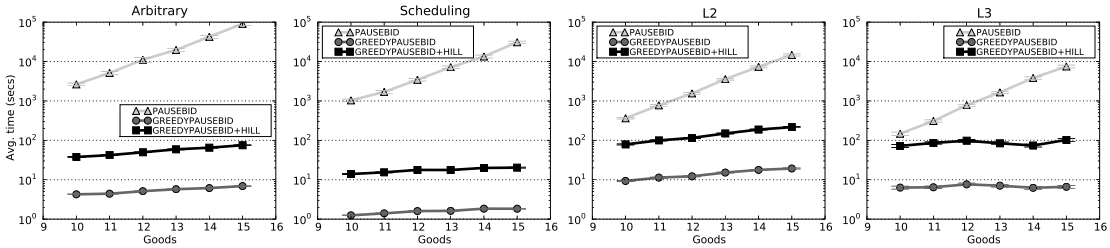
Figure 6: *Average Time* as a function of the number of *bidders* in the auction. For the *arbitrary* and *scheduling* distributions. The number of goods and bids is fixed to 10 and 20 respectively.

coefficient and m is the intercept coefficient. The m value is a base corresponding to the exponent x, and b is a constant value. When m is 1, the corresponding curve grows up linearly on steps of b. When m is less than 1 the curve will decrease. Thus, a curve having an m much bigger than 1 indicates an exponential increment.

The first test looks at the running time needed to clear an auction as a function of the number of bidders, as shown in Figure 6. In the arbitrary distribution the number of bidders does not significantly affect the running time. The time remains linear for all the algorithms. On the other hand, in the scheduling distribution, the number of bidders has a stronger effect over the running time, with PAUSEBID being the most affected since its



(a) *Average Time* as a function of the number of *bids* (the number of goods and bidders is fixed to 10 and 5 respectively)



(b) *Average Time* as a function of the number of *goods* (the number of bids and bidders is fixed to 20 and 5 respectively)

Figure 7: *Average Time* for each distribution as a function of the *bids* and *goods* in the auction. In the L2 and L3 distributions the number of bidders is the same as the number of bids.

curve grows exponentially. As shown in Table 2 its corresponding intercept coefficient m is greater than 2. However, the time remains linear for the greedy algorithms. For these algorithms, it is easier to solve the problem in the scheduling distribution than in the arbitrary distribution. For PAUSEBID it is the opposite—the arbitrary distribution is easier than the scheduling distribution. The slope of its corresponding curve in the scheduling distribution is much bigger, and although with few bidders (3-7) the time is low, it soon becomes higher.

Figure 7 shows the average running time as function of the number of bids and goods. In general, the number of bids does not affect the running time of the greedy algorithms, whose curve remains linear. As shown in Table 3, the corresponding intercept coefficient m is close to 1 and in some cases less than that. The number of bids affects the PAUSEBID running time in the arbitrary distribution, although this effect is not as big as the one provoked by the number of goods. It is interesting that the running time of PAUSEBID in L3 drops as the number of bids increases, this phenomenon is discussed later. The number of goods has an exponential effect over the running time of PAUSEBID in all the distributions. Table 4 shows that its corresponding intercept coefficient m is around 2—which indicates an exponential growth.

Distribution	Bidding Algorithm	m	b
Arbitrary	PAUSEBID	1.191	82.713
	GREEDYPAUSEBID	1.028	2.393
	GREEDYPAUSEBID+HILL	1.040	16.445
Scheduling	PAUSEBID	1.034	486.593
	GREEDYPAUSEBID	0.998	1.338
	GREEDYPAUSEBID+HILL	1.026	8.832
L2	PAUSEBID	1.079	82.0315
	GREEDYPAUSEBID	1.065	2.792
	GREEDYPAUSEBID+HILL	1.082	17.0171
L3	PAUSEBID	0.887	1743.058
	GREEDYPAUSEBID	0.931	26.187
	GREEDYPAUSEBID+HILL	0.968	138.530

Table 3: The slope coefficient b and the intercept coefficient m, obtained by logarithmic regression, that describe the curve of *average time* for each bidding strategy as a function of the number of $bids$, under different CATS distribution.

Distribution	Bidding Algorithm	m	b
Arbitrary	PAUSEBID	2.043	2.0
	GREEDYPAUSEBID	1.113	1.419
	GREEDYPAUSEBID+HILL	1.160	8.364
Scheduling	PAUSEBID	1.995	0.917
	GREEDYPAUSEBID	1.077	0.625
	GREEDYPAUSEBID+HILL	1.076	6.907
L2	PAUSEBID	2.114	0.203
	GREEDYPAUSEBID	1.149	2.412
	GREEDYPAUSEBID+HILL	1.214	11.704
L3	PAUSEBID	2.161	0.074
	GREEDYPAUSEBID	0.982	8.225
	GREEDYPAUSEBID+HILL	1.030	58.299

Table 4: The slope coefficient b and the intercept coefficient m, obtained by logarithmic regression, that describe the curve of *average time* for each bidding strategy as a function of the number of $bids$, under different CATS distribution.

In general, the running time of GREEDYPAUSEBID and GREEDYPAUSEBID+HILL remains linear in all the bid distributions, independently of the number of bidders, bids, and goods—the same pattern found in a previous experimental simulation [16]. GREEDYPAUSEBID is 10 times faster than GREEDYPAUSEBID+HILL. For these algorithms, the scheduling distribution is the easiest to solve, followed by the arbitrary, L3,

and L2. GREEDYPAUSEBID is 99 times faster than PAUSEBID, or more, depending on the number of goods being auctioned, or the number of bidders in the scheduling distribution. For PAUSEBID the easiest bid distribution is L3 followed by L2, scheduling, and arbitrary.

Something important, shown by the experiments, is that when using the PAUSE auction, where the allocation problem is solved by the bidders, the hardness of these distributions changes, depending in the bidding strategy. For example, L3 is one of the easiest for all the bidding strategies; however, according to the study presented in [14], it is the hardest CATS distribution for centralized solutions. In the L3 all the bids are for exactly 3 items. Thus, for this distribution, the PAUSE auction is basically developed in two stages, stage 1 (very fast, since there are no singleton bids) and stage 3. Because of the way CATS files are used here, each bidder corresponds to exactly one bid (the same case for L2, although the bids in that distribution can be for any number of items). That is, bidders have preferences for only one set of three items. The space search for each bidder is very small, since few of the bids from other bidders do not conflict with its bid. In other words, in this distribution, there are many bidders with common interest in at least one item. So, when forming their search tree the bids of theirs competitors are left out, reducing the search space. On the other hand, in the centralized approach, the search tree produced by this type of bid distributions has many leafs and nodes and very large branches, given the way these three-item bids can be combined.

7.2 Revenue

In general, for all the bidding strategies, the average revenue ratio increases as function of the number of bidders in the auction, as shown in Figure 8. It seems that as the competence increases, the prices that the winning bidders pay also increase. Figure 8 also shows that the revenue ratio is higher in the arbitrary distribution than in the scheduling distribution. In both distributions, PAUSEBID generates higher revenue; although the approximate bidding strategies get very close to it as the number of bids increases. In the arbitrary distribution, the average revenue ratio ranges between 0.75 and 0.91, while in the scheduling distribution it ranges between 0.60 and 0.87, in both cases it depends on the number of bidders and the bidding strategy.

Figure 9 shows the average revenue ratio as function of the number of bids and goods. In the arbitrary and scheduling distributions the revenue ratio decreases non-monotonically as a function of both the number of bids and goods, for all the bidding strategies. However, in these two distributions, the PAUSEBID generates higher revenue than the other two algorithms. In the arbitrary distribution, the difference is about 2% higher than GREEDYPAUSEBID+HILL and 3% higher than GREEDYPAUSEBID. In the scheduling distribution, the difference is about 4% higher than GREEDYPAUSEBID+HILL and 7% higher than GREEDYPAUSEBID. Also, in these two distributions, GREEDYPAUSEBID+HILL has a higher revenue ratio than GREEDYPAUSEBID, although

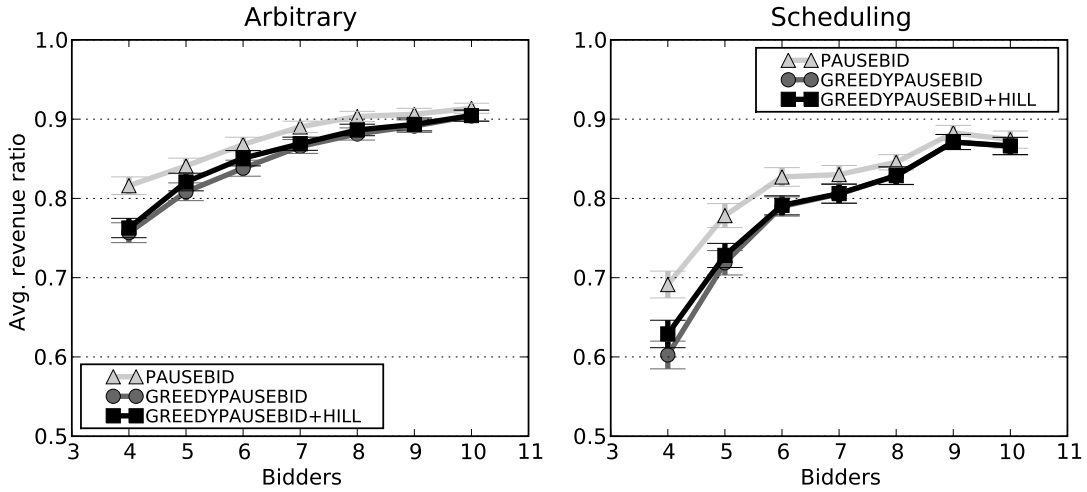
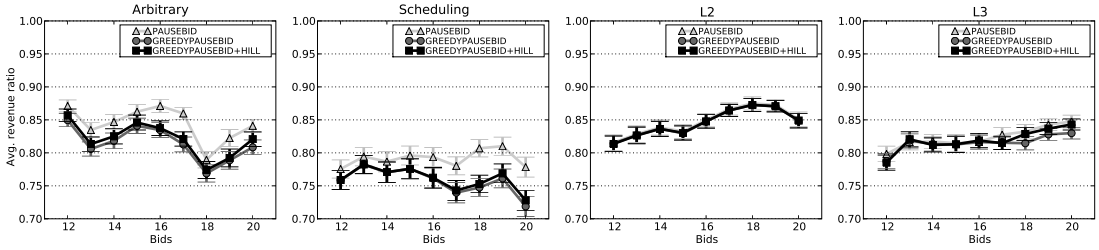
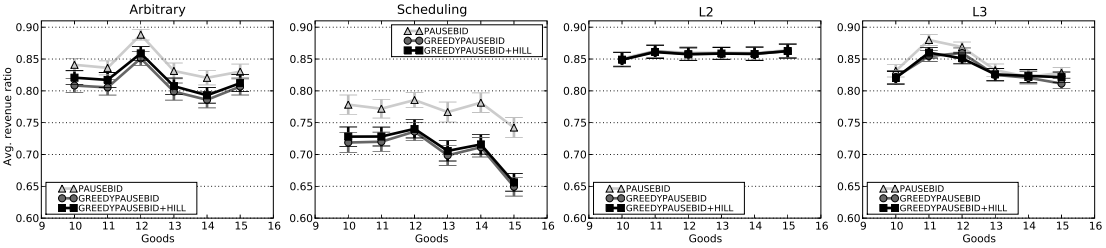


Figure 8: The *Average Revenue Ratio* as a function of the number of *bidders* in the auction. For the *arbitrary* and *scheduling* distributions. The number of goods and bids is fixed to 10 and 20 respectively.



(a) *Average Revenue Ratio* as a function of the number of *bids* in the auction (the number of goods and bidders is fixed to 10 and 5 respectively)



(b) *Average Revenue Ratio* as a function of the number of *goods* in the auction (the number of bids and bidders is fixed to 20 and 5 respectively)

Figure 9: *Average Revenue Ratio* for each distribution as a function of the *bids* and *goods* in the auction. In the L2 and L3 distributions the number of bidders is the same as the number of bids.

the difference is very small, between 1% and 2%.

Figure 9 also shows that, in the L2 distribution, all the bidding strategies have the same revenue ratio; which increases non-monotonically as a function of the number of bids but remains linear as a function of the number of goods. Similarly, in the L3 distribution, the revenue ratio of all the bidding strategies increases non-monotonically as a function of the number of bids, and decreases non-monotonically as a function of the number of goods. However, in the L3, there is a small difference (less than 1%) of revenue ratio in favor of the PAUSEBID algorithm.

The arbitrary distribution is the one where the PAUSE auction generates higher revenue, followed by the L2, and the L3. On the other hand, the scheduling distribution is the one where the lowest revenue is obtained.

The allocations found by PAUSE and these algorithms do not always have the same distribution of items to bidders of that of the revenue-maximizing solution (as shown in all the experiments). The cases where the algorithms fail to arrive at the distribution of the revenue-maximizing solution are those where there is a large gap between the first and second valuation for a set (or sets) of items. If the revenue-maximizing solution contains the bid (or bids) using these higher valuation then it is impossible for the PAUSE auction to find this solution because that bid (those bids) is never placed. For example, if agent i has $v_i(1) = 1000$ and the second highest valuation for (1) is only 10 then i only needs to place a bid of 11 in order to win that item. If the revenue-maximizing solution requires that 1 be sold for 1000 then that solution will never be found because that bid will never be placed.

7.3 Allocative efficiency

As expected the allocations found by PAUSEBID, since it is a myopic optimal strategy, have higher efficiency ratio than those found by the greedy strategies. As shown in Figures 10 and 11, in this experiment the allocations found by PAUSEBID have an efficiency ratio grater than 0.97 independently of the distribution and the number of bidders, bids, and goods. The allocations found by the greedy strategies have lower efficiency ratio than those found by PAUSEBID. However, their corresponding efficiency ratio is pretty high too, grater than 0.90 for most of the cases.

As shown in Figure 10, the efficiency ratio increases as a function of the number of bidders in both, the arbitrary and scheduling distributions. The fact that the efficiency ratio is not 1 (except for PAUSEBID in the scheduling distribution with more than 5 bidders) indicates that the allocation of goods to bidders obtained by PAUSE differs, sometimes, from that obtained by a centralized winner determination algorithm. Although, the efficiency of the solutions found by all the algorithms is very high. The average efficiency ratio of all the algorithms is very close to 1 when the number of bidders goes above 8. In average, the allocations obtained in the scheduling distribution are more efficient than those obtained in the arbitrary solution.

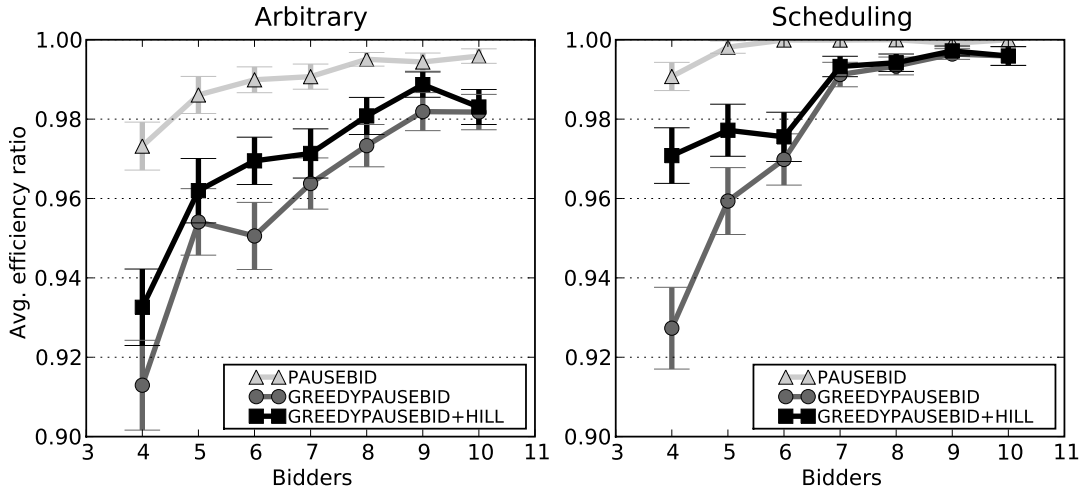
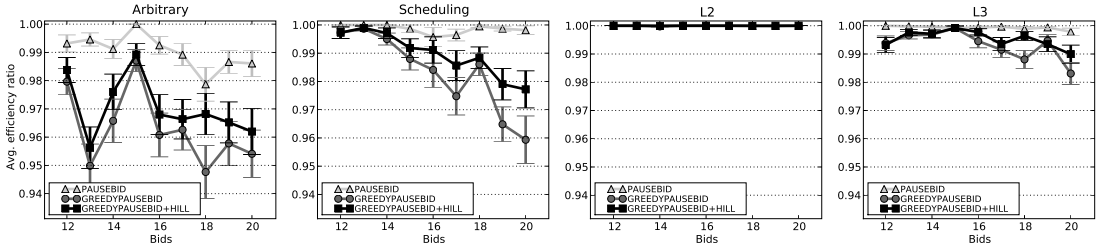
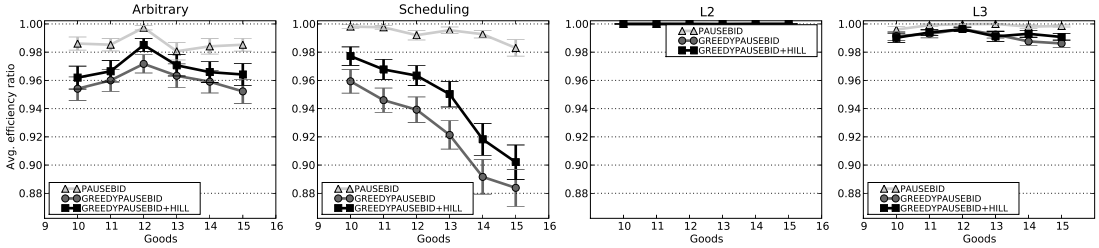


Figure 10: The *Average Efficiency Ratio* as a function of the number of *bidders* in the auction. For the *arbitrary* and *scheduling* distributions. The number of goods and bids is fixed to 10 and 20 respectively.



(a) *Average Efficiency Ratio* as a function of the number of *bids* (the number of goods and bidders is fixed to 10 and 5 respectively)



(b) *Average Efficiency Ratio* as a function of the number of *goods* (the number of bids and bidders is fixed to 20 and 5 respectively)

Figure 11: *Average Efficiency Ratio* for each distribution as a function of the *bids* and *goods* in the auction. In the L2 and L3 distributions the number of bidders is the same as the number of bids.

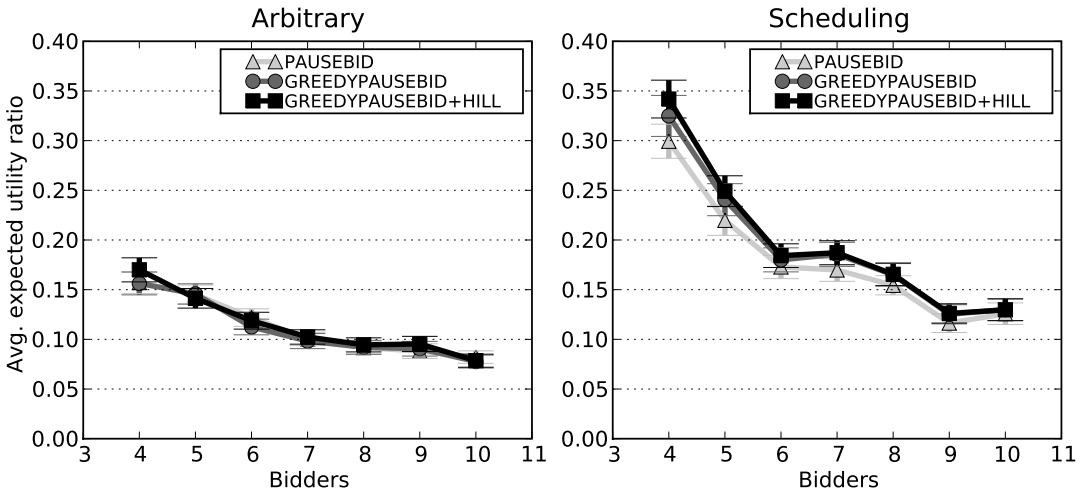


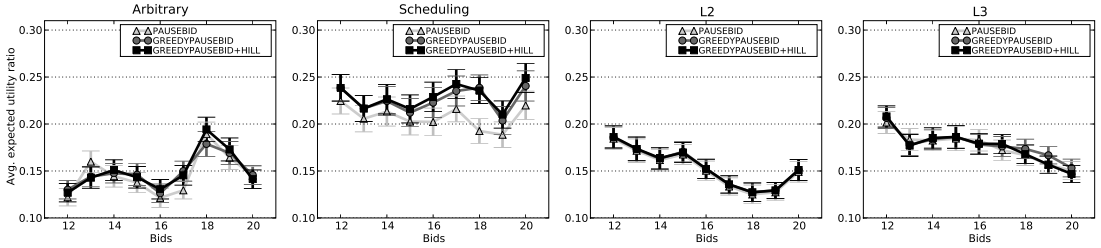
Figure 12: The *Average expected Utility Ratio* as a function of the number of *bidders* in the auction. For the *arbitrary* (a) and *scheduling* (s) distributions. The number of goods and bids is fixed to 10 and 20 respectively.

In general, the number of bidders has a positive effect on the revenue and the efficiency of the final allocations. It seems that as the competition increases, the bidders tend to increase the prices of their bids, thus, the resulting allocations provide more revenue for the seller and reflect more precisely the nature of the bidders' private valuations

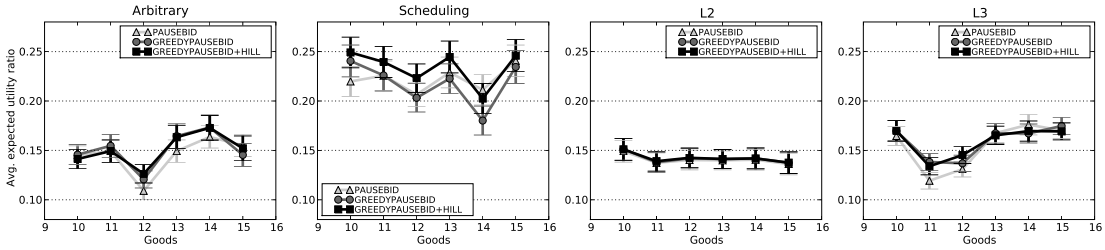
Figure 11 shows how the number of bids and goods affect the efficiency. In the L2 distribution, all the biding strategies reach the highest efficiency ratio. In other words, in this distribution, the average efficiency ratio is not affected at all for any of these parameters. The efficiency ratio is also very high in the L3 distribution (more than 0.99), for all the bidding strategies. On the other hand, in the arbitrary and scheduling distributions, the efficiency ratio decreases non-monotonically as a function of both the number of bids and goods. Although in general, all the bidding strategies are very close to each other in terms of efficiency, the efficiency of the approximate strategies is lower. The efficiency of the solution obtained by the approximate algorithms is more affected by the number of bids and goods in the scheduling distribution. In this distribution, the slope of the curves of GREEDYPAUSEBID and GREEDYPAUSEBID+HILL is bigger than in other distributions, specially the one corresponding to the number of goods.

7.4 Bidders' expected utility

The average bidders' expected utility ratio (Figures 12 and 13), produces curves with values roughly inverse to that of the average revenue ratio (Figures 8 and 9). This



(a) *Average expected Utility Ratio* as a function of the number of *bids* (the number of goods and bidders is fixed to 10 and 5 respectively)



(b) *Average expected Utility Ratio* as a function of the number of *goods* (the number of bids and bidders is fixed to 20 and 5 respectively)

Figure 13: *Average expected Utility Ratio* for each distribution as a function of the *bids* and *goods* in the auction. In the L2 and L3 distributions the number of bidders is the same as the number of bids.

relation was expected, since the higher the payments (the revenue) the lower the possible expected utility. That means that in distributions where the revenue of all the bidding strategies is the same, L2 and most of the case of L3, the bidding strategy does not have any effect over the bidders' expected utility (Figure 13). The only incentive to choose one over the other would be speed (the lowest running time). Since in general GREEDYPAUSEBID is the fastest, that would be the best choice.

In the case where the revenue ratio is different amongst the bidding strategies, the approximate strategies offer higher bidders' expected utility (another incentive in addition to speed). However, GREEDYPAUSEBID, the one that in general provides lower revenue, does not offer the highest expected utility; unexpectedly, GREEDYPAUSEBID+HILL does provide the highest utility—although the error bars in the plots overlap most of the time. The reason for this is that, as shown before, the allocations obtained by GREEDYPAUSEBID+HILL are, in general, more efficient. This means that in the allocations obtained by GREEDYPAUSEBID+HILL, the winners are bidders that have higher valuations than those in the allocations obtained by GREEDYPAUSEBID, and yet they do not pay as much as the bidders in the allocations obtained by PAUSEBID. This shows a relationship between efficiency and bidders utility.

8 Conclusions

In general, the PAUSE auction with either myopic-optimal or approximate bidders, produces highly efficient allocations. Bidders have an incentive to join a PAUSE auction instead of a centralized auction, since they have the opportunity to obtain the goods with prices 25% to 10% lower than the prices they would pay when participating in a centralized auction. However, as expected, this incentive has an impact on the sellers' revenue. The revenue produced by the allocations obtained by PAUSE falls between 75% and 90% of the optimal revenue, depending on the combination of different factors (goods, bids, bidders, bidding strategy, and bid distribution). That is, a small part of the revenue is taken from the sellers and given to the buyers.

The running time of the PAUSEBID algorithm grows exponentially as a function of the bidders and goods, and varies depending in the bid distribution. The running time of the heuristic-approximate bidding strategies remains linear independently of the number of bidders, bids, goods, and bid distribution.

The approximate algorithms offer higher utility to the bidders than PAUSEBID, which would make them the best strategy for the bidders. GREEDYPAUSEBID+HILL provides higher bidders' utility than GREEDYPAUSEBID, about 1% higher. However it is 10 times slower. Bidders should consider the trade off between utility and time.

In summary, the experiments presented here have shown that, over a representative set of problems, the PAUSE auction along with the heuristic bidding strategies is a realistic method for solving combinatorial allocation problems without the use a centralized auctioneer. The PAUSE auction is both efficient in terms of the solution it finds as well as the time it takes to find it. Although the revenue generated by a PAUSE auction is lower than the optimal, it has been found that it increases as the number of bidders in the auction increases, as consequence of the increased competition. Thus, the incentives provided by PAUSE will attract more bidders, which would result in more benefits for the seller, in addition to the savings from eliminating the cost of having a central auctioneer.

Future work will consist of looking at how to further increase the scalability of the PAUSE auction by allowing agents to place incomplete bid sets and only communicate their bids to a subset of agents. The goal is to develop new auctions which can scale to any number of bidders while distributing the computational cost evenly among them.

Acknowledgments

Acknowledgment is made to the University of South Carolina's High Performance Computing Group for the computing time used in this research.

References

[1] P. J. Brewer. Decentralized computation procurement and computational robustness in a smart market. *Economic Theory*, 13(1):41–92, January 1999.

[2] P. Cramton. Simultaneous ascending auctions. In Cramton et al. [3], chapter 4, pages 99–114.

[3] P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[4] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence*, pages 548–553. Morgan Kaufmann Publishers Inc., 1999.

[5] N. Fukuta and T. Ito. Towards better approximation of winner determination for combinatorial auctions with large number of bids. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 618–621, 2006.

[6] N. Fukuta and T. Ito. Short-time approximation on combinatorial auctions: a comparison on approximated winner determination algorithms. In *DEECS '07: Proceedings of the 3rd international workshop on Data enginering issues in E-commerce and services*, pages 26–33, New York, NY, USA, 2007. ACM.

[7] P. Gradwell and J. Padget. Markets vs auctions: Approaches to distributed combinatorial resource scheduling. *Multiagent and Grid Systems*, 1(4):251 – 262, 2005.

[8] R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In *AI '01: Proceedings of the 14th Biennial Conference of the Canadian Society on Computational Studies of Intelligence*, pages 57–66, London, UK, 2001. Springer-Verlag.

[9] H. H. Hoos and C. Boutilier. Solving combinatorial auctions using stochastic local search. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 22–29. AAAI Press / The MIT Press, 2000.

[10] F. Kelly and R. Steinberg. A combinatorial auction with multiple winners for universal service. *Management Science*, 46(4):586–596, 2000.

[11] A. Land, S. Powell, and R. Steinberg. PAUSE: A computationally tractable combinatorial auction. In Cramton et al. [3], chapter 6, pages 139–157.

[12] D. Lehmann, L. I. Oćallaghan, and Y. Shoham. Truth revelation in approximately efficient combinatorial auctions. *Journal of the ACM*, 49(5):577–602, 2002.

[13] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM conference on Electronic commerce*, pages 66–76. ACM Press, 2000.

[14] K. Leyton-Brown and Y. Shoham. A test suite for combinatorial auctions. In Cramton et al. [3], chapter 18, pages 451–478.

[15] B. Mendoza and J. M. Vidal. Bidding algorithms for a distributed combinatorial auction. In *Proceedings of the Autonomous Agents and Multi-Agent Systems Conference*, 2007.

[16] B. Mendoza and J. M. Vidal. Approximate bidding algorithms for a distributed combinatorial auction (short paper). In Padgham, Parkes, Müller, and Parsons, editors, *Proceedings of the 7th International Conference on Autonomous Agents and Multiagent Systems*, Estoril, Portugal, May 2008.

[17] M. V. Narumanchi and J. M. Vidal. Algorithms for distributed winner determination in combinatorial auctions. In *LNAI volume of AMEC/TADA*. Springer, 2006.

[18] S. Park and M. H. Rothkopf. Auctions with endogenously determined allowable combinations. Technical report, Rutgets Center for Operations Research, January 2001. RRR 3-2001.

[19] D. C. Parkes and J. Shneidman. Distributed implementations of vickrey-clarke-groves auctions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and MultiAgent Systems*, pages 261–268. ACM, 2004.

[20] D. C. Parkes and L. H. Ungar. Iterative combinatorial auctions: Theory and practice. In *Procedings of the 17th National Conference on Artificial Intelligence (AAAI-00)*, pages 74–81, 2000.

[21] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[22] T. Sandholm. An algorithm for winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, February 2002.

[23] T. Sandholm. Expressive commerce and its application to sourcing: How we conducted $35 billion of generalized combinatorial auctions. *AI Magazine*, 28(3):45–58, 2007.

[24] J. M. Tenenbaum. AI meets web 2.0: Building the web of tomorrow, today. *AI Magazine*, 27(4), 2006.

[25] J. M. Vidal and E. H. Durfee. Learning nested models in an information economy. *Journal of Experimental and Theoretical Artificial Intelligence*, 10(3):291–308, 1998.

[26] G. Weiss, editor. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 1999.

[27] E. Zurel and N. Nisan. An efficient approximate allocation algorithm for combinatorial auctions. In *Proceedings of the ACM Conference on Electronic Commerce*, 2001.

Authors' Biographical Notes

Benito Mendoza is a postdoctoral research engineer at ExxonMobil Research and Engineering Company. His research interests are in the areas of multiagent systems, situation awareness, and distributed information fusion. He has a PhD. in Computer Science and Engineering from the University of South Carolina and a MSc in Artificial Intelligence and a BSc in Computer Science from the University of Veracruz, Mexico.

José M. Vidal is an associate professor at the University of South Carolina. His research interests are in the area of multiagent systems. He has a PhD. from the University of Michigan and a BSE from the Massachusetts Institute of Technology, both in Computer Science and Engineering.