Approximate Bidding Algorithms for a Distributed Combinatorial Auction

(Short Paper)

Benito Mendoza and José M. Vidal
Computer Science and Engineering
University of South Carolina
Columbia, SC 29208
mendoza2@engr.sc.edu, vidal@sc.edu

ABSTRACT

Distributed allocation and multiagent coordination problems can be solved through combinatorial auctions (CAs). However, most of the existing winner determination algorithms (WDAs) for CAs are centralized. The PAUSE auction is one of a few efforts to release the auctioneer from having to do all the work. The PAUSEBID bidding algorithm[6] generates myopically-optimal bids for agents in a PAUSE auction but its running time is exponential on the number of bids. We present new approximate bidding algorithms that not only run in linear time but also increase the utility of the bidders as result of small decrement in revenue.

Categories and Subject Descriptors

I.2.11 [**Computing Methodologies**]: Distributed Artificial Intelligence—*Intelligent Agents, Multiagent Systems.*

Keywords

Combinatorial auctions, coordination, task and resource allocation.

1. INTRODUCTION

CAs have generated significant interest as automated mechanisms for buying and selling bundles of goods and for solving problems of task and resource allocation and multiagent coordination[8]. In CAs bidders can place bids on combinations of items rather than just individual items which introduces computational and economic challenges. For example, it makes it hard to find the allocation of items to bidders which maximizes the auctioneer's revenue. This computational problem, known as the winner determination problem (WDP), is NP-Hard[7]. Notwithstanding the complexity of the WDP, several algorithms that have a satisfactory performance for problem sizes and structures occurring in practice have been developed [1, 9] however most of them are centralized– they require all agents to send their bids to a centralized auctioneer who then runs a WDA. We believe that distributed solutions to the winner determination problem should be studied as they offer a better fit for some applications as when, for example, agents do not want to reveal their valuations to the auctioneer, the auctioneer does not want to perform all the computation, or when it is difficult to establish a trusted auctioneer.

The PAUSE (Progressive Adaptive User Selection Environment) auction[3, 4] is one of a few efforts to distribute the WDP amongst the bidders. PAUSE establishes the rules the participants have to adhere to so that the work is distributed amongst them, making 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 his does indeed correspond to some other agents' previous bid (we envision completely eliminating the auctioneer by having every agent perform this task). The computational problem shifts from one of winner determination to one of bid generation where bidders have a clear incentive for performing this computation. However, PAUSE is not concerned with how the bidders determine what they should bid (strategy). PAUSEBID and CACHEDPAUSEBID are bidding algorithms which generate myopically-optimal bids[6] for PAUSE. Test results show that on over 95% of the trials both algorithms find the same allocation as given by the revenue-maximizing solution. Still, their running time remains exponential as the problem is NP-Hard and test results showed that the time to calculate new bids does increase exponentially.

In many real life applications the time spent in finding a solution is more critical than the optimality of the solution found. A number of approximate WDA for the centralized WDP have been developed which generate solutions very close to the revenue maximizing solution, they get even closer as the number of bids increases, and have excellent performances in a very restricted time period. In this paper we present new approximate bidding algorithms for PAUSE which we created borrowing some ideas from existing centralized approximate WDAs. Our new algorithms greatly reduced the time required to clear a PAUSE auction and increase the agents' expected utility gains, which we found counterintuitive since we would expect that not-optimal agents would do worse than optimal agents. However the sum of expected utility increases up to 20% when switching from using PAUSEBID to one of our approximate bidding algorithms. Our test results show that these new bidding systems generate in average a little less revenue, between 1.5% and 5% below, than a system using PAUSEBID.

Cite as: Title (Short Paper), Author(s), *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

2. APPROXIMATE BIDDING ALGORITHMS

The PAUSEBID algorithm and its variants implement a complete search over the set of current best bids B in order to find the new bidset which is myopically-optimal[1]. Approximate bidding algorithms forgo optimality in favor of heuristics and simple local searches which deliver a solution very quickly. The approximate algorithms we have developed are based in strategies used by i) a well-known approximate algorithm for solving the WDP for centralized CAs, the greedy algorithm described in[5], and ii) the well-known hill climbing algorithm. The greedy algorithm is a very simple linear algorithm and can be summarized into two steps.

1. The bids are sorted by $b^{\text{value}}/|b^{\text{items}}|^c$ for some number c, $0 \leq c \leq 1$. The authors showed that $c = 0.5$ was the approximate best value.

2. Proceed down the sorted list of bids accepting bids if the goods in demand are still unallocated and not conflicted, where bids b and $b1$ conflict if $b^{\text{items}} \cap b_1^{\text{items}} \neq \emptyset$.

2.1 The GREEDYPAUSEBID algorithm

The GREEDYPAUSEBID algorithm (figure 1) 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 $r(W) + \epsilon$ (W is the current winning bidset, ϵ is the minimum bid increment, and r the revenue function). We start by defining *my-bids* to be the list of bids for which the agent's valuation is higher than the current best bid, as given in B. We set the value of these bids to be the agent's true valuation (but we won't necessarily be bidding true valuation, as we explain later). Similarly, we set *their-bids* to be the rest of the bids from B. If *my-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 $b^{\text{value}}/|b^{\text{items}}|^c$. After *my-bids* is sorted, we take the first bid and add it to the initial solution g, to make sure that the solution includes the bid from *my-bids* with the highest rank. Finally, to complete the solution (a bidset that contain all the items), the agent's search list is simply the concatenation of *their-bids* and the rest of *my-bids* ordered again by the same criteria (lines 16 and 15 respectively). After we finish walking down the *bids* list, we have a solution g. However, agent i's bids in g are still set to his own valuation and not to the lowest possible price. If g has revenue less than or equal to $r(W) + \epsilon$, then the algorithm ends. When $r(g) > r(W) + \epsilon$, we call the procedure DISTRIBUTEPAYMENTS with g as parameter (line 25). This function sets the agent's payments so that it can achieve its maximum utility (while satisfying the revenue constraint). We have chosen to distribute the payments in proportion to the agent's true valuation for each set of items (the way PAUSEBID does it). 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 what it gets from W, otherwise it returns an empty bidset.

2.2 The GREEDYPAUSEBID+HILL algorithm

A simple extension to the greedy WDA consists of using a local search algorithm that continuously updates the allocation found by the greedy algorithm[2], searching in the remaining bids to improve the solution. We implement this idea in the GREEDYPAUSEBID+HILL algorithm. This algorithm starts with the solution provided by GREEDYPAUSEBID and then explores the neighborhood of that solution looking for allocations that generate a higher utility for the bidder. It consist of two main steps:

1. Call GREEDYPAUSEBID algorithm with appropriate input and $c = 0.5$.

2. If the solution g returned by GREEDYPAUSEBID is not empty then call HILLCLIMBING with *bids* = *my-bids* + *their-bids* sorted by c.

[1]We recommend [6] to get a good understanding of the problem of bidding in the PAUSE auction.

```
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^value
 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 1: Our 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 $k \geq 2$.

3. TEST AND COMPARISON

Our approximate algorithms allow an agent to place good, but not necessarily myopically-optimal, bids in a PAUSE auction. But the use of these new strategies raises several questions. Does a system of approximate bidders arrive at a different solution than a optimal biding agents system? Do bidders lose utility by switching to these algorithms? Is there any difference in the revenue generated? Are they

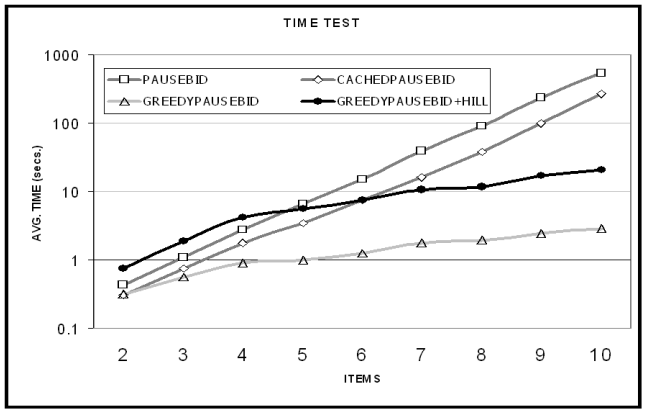

Figure 2: Average time in seconds that it takes to finish an auction as a function of the number of items in the auction.

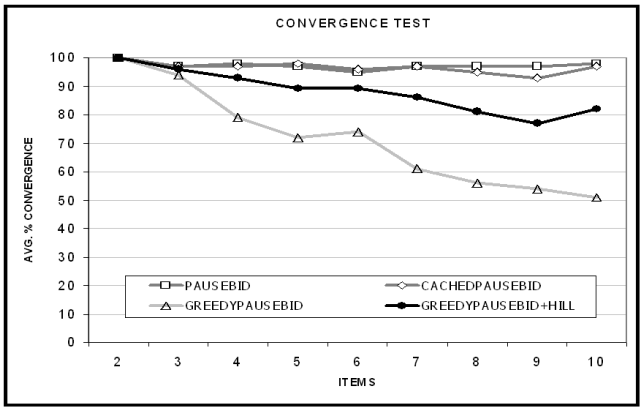

Figure 3: Average percentage of convergence, which is the percentage of times that the algorithms reach the revenue-maximizing solution.

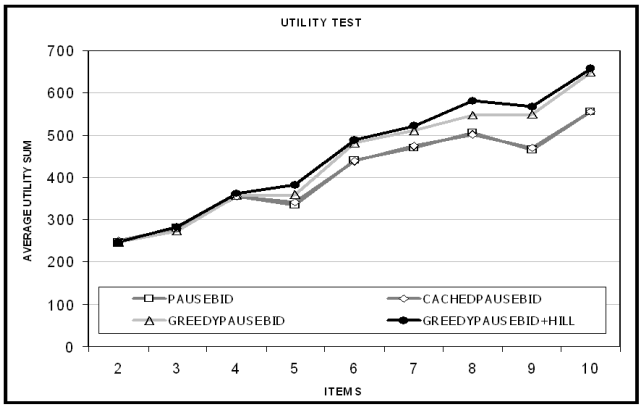

Figure 4: Average of the utility sum from 100 auctions as a function of the number of items.

faster? To answer these questions we have performed several experiments in which we create bidder agents that have randomly generated value functions as generated by the algorithm in [6] (bidders with super-additive preferences). We fixed the number of agents to be 5 and we experimented with different number of items, namely from 2 to 10. We ran 100 auctions for each number of items, and the same auction was run with all the algorithms.

As expected, the approximate algorithms are faster than the optimal algorithms (figure 2). The fastest one is GREEDYPAUSEBID which is up to 99.5% and 99% faster than PAUSEBID and CACHEDPAUSEBID respectively. The running time of these algorithms is linear in the number of bids since, at worst, they will check all the bids once. Of course, in practice they only check a small subset of all the bids. The difference in execution speed between the myopic-optimal (exponential time) algorithms and the approximate (linear time) algorithms is even more clear as the number of items increases.

We then compared the solutions found by both optimal and approximate algorithms to the revenue-maximizing solution as found by CASS[2] [1] when given a set of bids that corresponds to the agents' true valuation. Note, however, that the revenue from PAUSE on all the auctions is always smaller than the revenue of the revenue-maximizing solution when the agents bid their true valuations since PAUSE uses English auctions. The optimal algorithms converge to the same distribution as the revenue-maximizing solution more than 90% of the time[6]. As shown in figure 3, the percentage of convergence to that solution is lower for the approximate algorithms, dropping as a function of the number of items from 100% with 2 items to 70% with 10 items, for GREEDYPAUSEBID. GREEDYPAUSEBID+HILL does better, it remains closer to the optimal algorithms, around 10% below. This proves that a system with approximate bidders arrives to different solutions than a system with myopic-optimal bidders.

We then calculated the sum of the utilities obtained by each agent in each auction. We can see (in figure 4) that the sum of the agents' utility increases as a function of the number of items and that the approximate algorithms generate higher utility than the optimal ones. Being GREEDYPAUSEBID+HILL the algorithm that generates the highest utility. In auctions with few items – 2, 3, and 4 – the utility generated by GREEDYPAUSEBID+HILL is about the same that the one generated by PAUSEBID, but with greater number of items this difference in utility is about 20%. Thus we can see that, on average, the agents always get higher utility by using the approximate algorithms instead of the myopic-optimal ones.

From an agent's individual point of view, the sum of the utilities is not as important as the individual utility. A bidder would be more interested on the expected individual utility of switching from one system to another. We compare the actual utility that an agent gained in two different systems (agents bid using PAUSEBID in one and GREEDYPAUSEBID+HILL in the other). Figure 5 shows that the agent got the same utility (the difference in individual utility is zero) in both systems 54% of the 900 auctions, that only in 17% of the auctions the difference in individual utility was in favor of the PAUSEBID (lower than zero) and that in 28% favored GREEDYPAUSEBID+HILL. Notice that even in the cases with no difference in the utility the approximate system would be

[2]CASS implements a centralized WDA to find the solution that maximizes revenue.

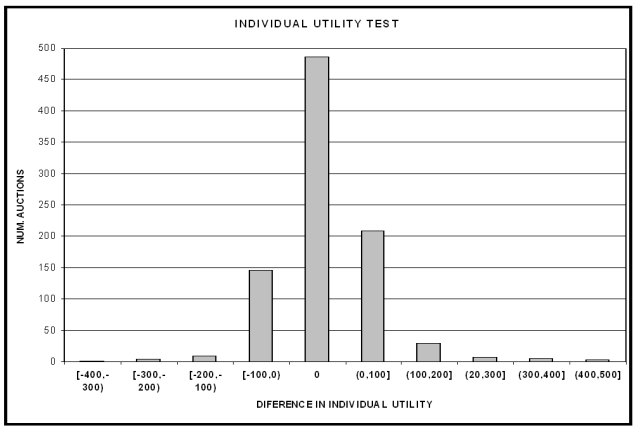

Figure 5: Distribution of the difference in individual utility (the utility that an agent gets in one system minus the utility it gains in the other). The negative side is in favor of PAUSEBID and the positive side is in favor of GREEDYPAUSEBID+HILL.

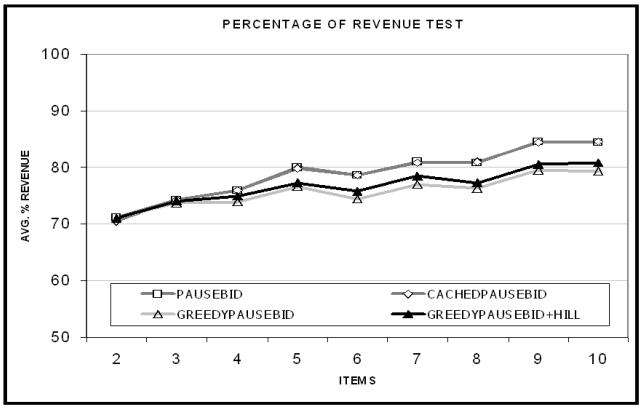

Figure 6: Average percentage of revenue from our algorithms relative to the revenue-maximizing solution as a function of the number of items.

more attractive, since bidders do not need to do hard computation and they do not need to wait too long for the auction to finish. This result is counter-intuitive (receiving a higher utility when using an approximate algorithm than when using an algorithm that guarantees a bid that gives them the highest possible utility). However, we must consider that the agents are engaged in a PAUSE auction and that their final utility depends, not on the intermediate bids placed by the agents, but only on the final result of the action. It seems that by having all agents place less than optimal bids, the final result is that they all, on average, get higher utility.

We calculated a percentage representing the proportion of the revenue given by each of the algorithms relative to the revenue-maximizing solution (figure 6). This percentage generally increases as the number of items increases and is lower for the approximate bidders than for the myopic-optimal. In absolute terms, our tests show that the average revenue for the four algorithms was approximately 274.29 times the number of items in the auction, with the greedy algorithm resulting is sligthly lower revenue, thus smaller slopes.

We also note that the social welfare (sum of buyers' utility and seller's revenue) remains constant for all algorithms. Thus, the greedy algorithms end up increasing the buyer's utility at the cost of reducing the seller's revenue. In effect, by having all buyers be less sophisticated in their bidding we end up with a a lower selling price, as might be expected.

4. CONCLUSIONS

We have introduced two new approximate algorithms which tackle the complexity of generating good bids in a PAUSE auction. Based on the results of our experiments, we believe that these approximate bidding algorithms are a realistic tool for the development of large-scale distributed CAs. Furthermore, bidders have double incentive to use them: they are faster and they give the bidders higher utility. Of course, they do reduce total revenue and thus, the seller's utility. But this loss is small compared to the gain in utility and speed. We envision a future where complex sourcing problems are solved by millions of automated agents bidding and negotiating in distributed CAs and even more complex negotiation networks. The algorithms we present here show us that linear-time approximation methods are viable and, thus, we should be able to scale up to very large numbers.

5. REFERENCES

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

[2] 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.

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

[4] A. Land, S. Powell, and R. Steinberg. PAUSE: A computationally tractable combinatorial auction. In *Combinatorial Auctions*, pages 139–157. 2006.

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

[6] B. Mendoza and J. M. Vidal. Bidding algorithms for a distributed combinatorial auction. In *Proceedings of the AAMAS Conference*, 2007.

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

[8] 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.

[9] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. CABOB: a fast optimal algorithm for winner determination in combinatorial auctions. *Management Science*, 51(3):374–391, 2005.