M a t h e m a t i c a
B a l k a n i c a

New Series Vol. 19, 2005, Fasc. 1-2

A New Search Algorithm for Optimal Winner Determination in Combinatorial Auction

Gallina M. Illieva

Combinatorial auctions are auctions where bidders can bid on combinations of items. Compared to other auction mechanisms in traditional multi-item auctions, they increase the efficiency of the auction, while keeping risks for bidders low. However, the determination of an optimal winner combination achieving maximum revenue is a complex computational problem. This is often the case, for example, in allocating goods, resources, services etc.

In this paper we: 1) describe recent algorithms for winner combination determination and compare them to traditional algorithms; 2) introduce a new integer programming algorithm with three different approaches to cut-off; 3) present and benchmark the new search mechanism to the problem, which enables very general auctions to be treated efficiently and 4) discuss and estimate the impact of the probability distributions chosen for benchmarking.

AMS Subj. Classification: 91B26, 65K05, 90C57

Key Words: Auction; Combinatorial auction; Multi-item auction; Winner determination; Multi-agent system

1. Introduction

Auctions are popular, distributed and autonomy-preserving ways of allocating items. They are extensively used by human bidders and recently by software agents of Internet auction servers. Some auction servers have built-in mobile agent support and enable them to participate in the auctions.

In a traditional auction format where the items are auctioned separately (sequentially or in parallel), to decide what to bid on an item, an agent needs to estimate which other items it will receive in the other auctions. This leads to inefficient allocations where bidders do not get the combinations they want and get combinations that they do not want. Several researchers showed that *Combinatorial Auctions* can be used to overcome these deficiencies [2,3]. Combinatorial auction allows bidders to express complementarity and also substitutability [1,3].

This expressiveness instead of having to speculate into an item's evaluation can lead to more economical allocations of the items.

The determination of an optimal winner combination in combinatorial auctions is an NP-complete and inapproximable problem [2], which has recently attracted some research, e.g. [1,3].

In this paper we look further into the topic and our contributions are:

• Comparison of the recent algorithms by Fujishima [1] and Sandholm [4] to the traditional algorithms for the computationally identical problem of set packing. From this overview, we conclude that many of the main features of recently presented algorithms are rediscoveries of traditional methods in the operations research community.

• A new search algorithm for winner determination in combinatorial auctions with three approaches to pruning is present. The appropriate test to benchmark it is also given in the paper.

• Several experiments to study the significance of the probability distributions of the test sets used for evaluating different algorithms have been described in the paper. Moreover, we discuss and exemplify some of the distributions used for benchmarking in recent literature [1,3].

2. Recent winner determination algorithms and traditional algorithms for corresponding problems

In the basic case a bid states a bundle of commodities, which it values at a given price. For example let us assume that an auctioneer has a set of items (commodities) $M = \{1, 2, \ldots, m\}$, to sell and the buyers have submitted a set of bids, $B = \{B_1, B_2, \ldots, B_n\}$. A bid is a tuple $B_j =< S_j, p_j >$, where $S_j \subseteq M$ is a set of commodities and $p_j > 0$ is a price. Given a collection of such bids, the surplus maximizing combination is the solution to the linear integer programming problem:

$$(2.1) \qquad max \sum_{j=1}^{n} p_j x_j \; s.t. \; max \sum_{j|i \subset S_j}^{n} x_j \leq 1, i = 1, 2, \ldots, m \; and \; x_j \in \{0, 1\}$$

where m is the number of commodities, n is the number of bids and x_j is a binary variable representing whether bid j is selected or not. Mathematically viewed, the problem can be expressed in a general form: find such values for the array of decision variables $\mathbf{x}$ so that the linear function is maximized and the constraints are satisfied. This is a challenge for the integer programming.

There are three different approaches to solve the problem: one approach is to solve the problem approximately; the second approach is to restrict the allowable bids, but that can lead to the economic inefficiency that prevail in

noncombinatorial auctions because bidders may not be able to bid on the combinations they prefer and third approach is to solve the problem using search tree [4].

We focus this presentation around third approach, which features set-partitioning algorithm introduced by Garfinkel and Nemhauser.

Since the problem of Equation (2.1) is equivalent to the definition of set packing and the problems of set packing and set partitioning can be transformed into each other, the Garfinkel-Nemhauser algorithm can be used for winner determination in combinatorial auctions.

One of currently best performing winner determination algorithms, the CASS algorithm [1], is apparently in major parts a rediscovery of the Garfinkel-Nemhauser algorithm. The main principles of both algorithms are to 1) put the bids in lists corresponding to the different commodities (called *bins* by Fujishima); 2) sort the bids in the list in some cost (valuation) related order; 3) do pruning whenever the current combination cannot be better than the best one found so far, and 4) do standard backtracking. There are essentially two significant differences between CASS and the Garfinkel-Nemhauser algorithm: 1) caching of partial search results, and 2) improved pruning.

Compared to the simple pruning described in the Garfinkel-Nemhauser algorithm, Sandholm's search algorithm [3] and the CASS algorithm [1], use more sophisticated technique, which essentially is the ceiling test.

3. A new search algorithm for combinatorial auction winner determination CANS (**C**ombinatorial **A**lgorithm with **N**ode **S**earch)

In this section we present a new algorithm for optimal winner determination. The basic solution strategy is a search with improvements over pruning at each search node in the tree of decisions.

The foundation of our algorithm is a depth-first branch-and-bound tree search that branches on bids. The set of bids that are labelled winning on the path to the current search node is called *CurrentSolution*. *Blocked* is the set of bids, which are in conflict with bids in *CurrentSolution*, e.g. for each bid B_q from *Blocked*, there is the bid B_j in *CurrentSolution*, so that $S_q \cap S_j \neq \phi$. The search is invoked by calling of function $NodeSearch(k)$.

The bids are sorted in advance descending in the list in cost (valuation) related order. This sorting allows better revenue entries to be reached earlier and therefore enables more effective pruning with first heuristic.

Improving the pruning when the current solution's revenue does not exceed maximal revenue so far f^* can increase search speed. Our first heuristic (line #8) evaluates the perspective of the current solution for success through

the use of the next equation:

$$h = \sum_{j=k+1; j \notin Blocked}^{n} p_j \tag{3.1}$$

This way we acquire less accurate upper bound, but much faster. A compromise is made between research expenses and the expenditure for determination of the upper evaluation of each step of the algorithm (line #8).

The CANS Algorithm:

```
Function NodeSearch(k);
1)  Result ← False;
2)  If k <= n, then
      Begin
3)    CardinalityOfBlocked← CardinalityOf(Blocked);
4)    For all B_p such that p <= n and p > k and B_p∩Blocked=∅,
        Begin
5)      CurrentSolution ← CurrentSolution∪B_p;
6)      UpdateOfBlocked(B_p);
7)      If CardinalityOf(CurrentSolution) < m, then
8)        If IsThisSolutionPerspective(CurrentSolution,Blocked), then      /*Heuristic1 & Heuristic2*/
            Begin
9)          Result ← True;
10)         MadeNewInclusion ← NodeSearch(k+1);
            End;
11)     If (CardinalityOf(CurrentSolution) = m) or (not (MadeNewInclusion)), then
                                                  /*A new CurrentSolution is found*/
12)       If Price(CurrentSolution) > Max, then
13)         Max ← Price(CurrentSolution);
14)     If CardinalityOfBlocked≠ CardinalityOf(Blocked), then                /*Heuristic3*/
15)       FindingOfNextClaimant(B_p,Blocked)
        Else
16)       p ← n+1;                          /*such claimants B_p do not exist*/
17)     RestoreOfCurrentSolution;
18)     RestoreOfBlocked;
        End;
      End;
```

The solution can include no more than m bids. This fact allows our second heuristic (it is implicitly included in line #8) to use a modification of Equation (3.1) so that in sum are included only the first $m - t$ bids, where t is the cardinality of set $CurrentSolution$.

We also do the pruning using the state of $Blocked$ before including B_p (e.g. $CardinalityOfBlocked$) and compare it with $Blocked$ after then. In third heuristic (line #14) if the sets are equal, we stop traversing of the claimants' bids after B_p (line #16). If this cut-off is not performed, we would obtain solutions that differ from the previous ones by the absence of bid B_p, ergo they are less lucrative than them.

We visualize algorithm with the next test example containing 4 items and 5 bids with their prices (see Table 1). In Table 2 are showed the same bids after sorting descending by their bid price.

Table 1

Item No. \ Bid No.	1	2	3	4	5
1		x	x		
2	x			x	
3		x		x	x
4			x		x
Bid Price:	5	2	1	6	3

Table 2

Item No. \ Bid No.	4	1	5	2	3
1				x	x
2	x	x			
3	x		x	x	
4			x		x
Bid Price:	6	5	3	2	1

Table 3

Current Solution:	Blocked:	f	f*
4	1, 2, 5	6	0
4, 3	1, 2, 5	7	7
1	-	5	7
1, 5	2, 3	8	8
5	2, 3	3	8
2	3	2	8
3	-	1	8

Table 3 shows the content in sets *CurrentSolution* and *Blocked*, as well as consecutive values of f and f^*.

In a branch-and-bound domain we use specific knowledge to choose the next variable to branch on the search algorithm. Based on the domain we might know that certain variables are "more" important than others, or that solutions are more likely to lie on one side or the other of the branch. For predicting upper bound for more aggressive pruning of the tree we used three domain specific heuristics with special purpose technique. By knowing an upper bound a-priori and based on their costs only many branches can be eliminated early on even before any integer solution is found.

4. Empirical benchmarking

As we have not yet determined algorithm's formal complexity characteristics we conducted empirical tests. In this section we give some empirical data in order to compare the new search algorithm and standard MIP.

Hardware setup of the experiments has been one standard uniprocessor 1,53GHz PC with 256MB of RAM memory. The algorithm was implemented in a program, written in Delphi with bit sets manipulations. The commercial software used in tests is CPLEX Student Version 8.0 with $n_{\max} = m_{\max} = 299$.

Figure 1 to Figure 3 show the results of the respective tests. For each data distribution, 5 instances have been tested, which is sufficient for obtaining a basic illustration. The instance sizes have been selected to match the sizes tested in the literature and/or to give reasonable computation time.

In the figures, the curves denote the time of the optimal solution. For each data distributions are plotted the results of new algorithm CANS and

CPLEX. The running time is very much dependent on the type of problem and we have tested the following three distributions [3].

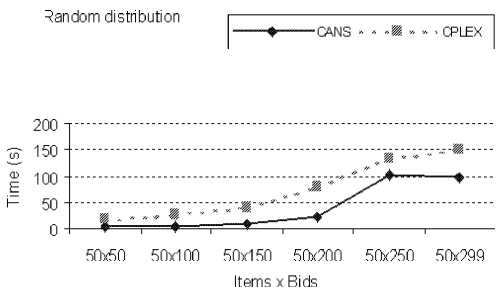

Figure 1 The *Random* distribution for 50 commodities

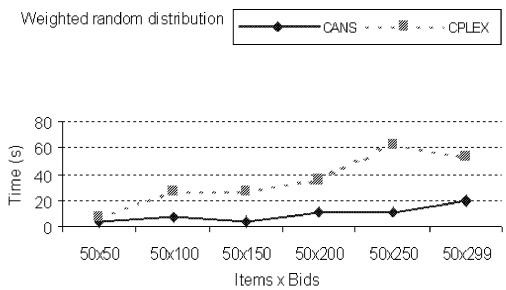

Figure 2 The *Weighted Random* distribution for 50 commodities

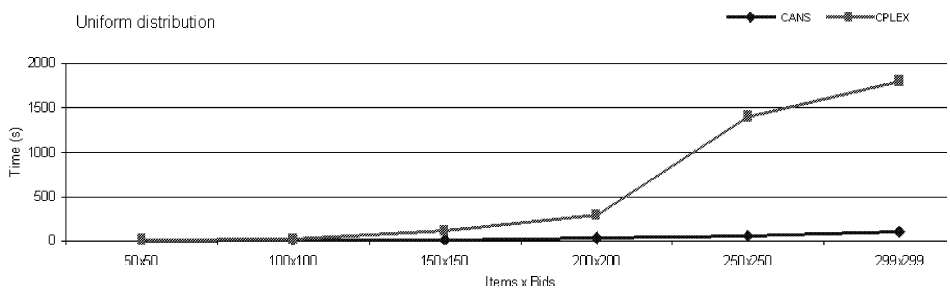

Figure 3. The Uniform distribution for different number of commodities, where each bidder bids for 80% of commodities.

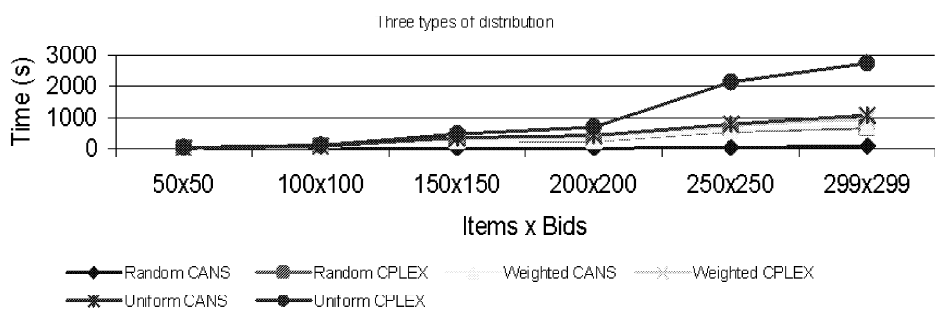

Figure 4. The Random, Weighted random and Uniform distributions for different combinations items x bids.

Random distribution is characterized by that for each bid the number of commodities is requested randomly from 1 to the number of commodities in the market. The actual commodities are requested without replacement. A random

integer valuation belongs to the interval between 1 and 1000. For this and all other distributions we have used integer valuations for simplicity of parsing. New algorithm determines the optimal winner efficiently (see Figure 1) and the times are superior to those obtained by CPLEX.

The characteristics of the **weighted random** distribution are similar in terms of computation time.

Uniform distribution draws the same number of randomly chosen items for each bid. Bid's valuation is an integer value from $[500, 1500]$, multiplied by the number of commodities. New algorithm performs well compared to CPLEX's implementation.

As we can see from experimental results suggests new algorithm work well under random, weighted and uniform distributions. All three distributions have high density of requests in bids k, where $k = \min(\text{Number of items in bid}) \geq m/2$. By conducted tests with "sparse" distribution algorithm is not well, because there is very decreasing cardinality of *Blocked* when a new bid is included to the current solution. Between 50 and 299 bids time complexity of algorithm is at least two times smaller than the same of CPLEX.

In summary, the new algorithm performs very well for the three tested distributions – random, weighted and uniform. For the "dense" distributions the algorithm is faster than CPLEX's algorithm. As CPLEX has been reported to outperform Sandholm's algorithm by around five orders of magnitude for the harder distributions, there is an indication that new algorithm also is faster than Sandholm's algorithm.

From our experiments with two different families of algorithms it is clear suggests that if the probability distribution is known to the auctioneer, it is possible to construct algorithms that capitalize significantly on this knowledge.

The three examples of the *Random* distribution, the *Weighted random* distribution and the *Uniform* distribution with many commodities are very illustrating examples of distributions that at a first glance may seem "hard" but turn out to be rather "easy". As seen above, it is important that we construct efficient algorithm for these special cases. The algorithm proposed above could be used as a part of a more complicated one, which works only if definite condition - "dense" matrix items-bids is available. The construction of realistic probability distributions based on some of our main application areas together with some reasonable agent strategies in certain attractive combinatorial auction models, is important future work. Yet again this underlines the importance of gathering of real-world data for benchmarking on heavily specialized algorithms.

5. Conclusions

In this paper we discussed important computational aspects of optimal

winner determination in combinatorial auctions. We have introduced a new algorithm and compared it with recent approaches to this problem based on traditional method of set partitioning, which can be used for optimal winner determination. The main conclusion of this comparison is that the features of our new algorithm allow many of the "harder" data test distributions to be better manipulated in comparison with the older well-known methods.

It can be proposed for electronic commerce to: 1) investigate state of the art of operations research and combinatorial optimisation algorithms for different settings; 2) develop special purpose algorithms where needed, and 3) gather real world distributions instead benchmarking on arbitrary distributions.

For further research we propose: 1) to adapt the algorithm in order to achieve good performance by "sparse" distributions; 2) to experiment with larger size tests; 3) to benchmark the algorithm on real world data distributions from realistic agent preferences, agent strategies and market mechanisms and 4) to create a combinations of algorithms with sensitive choice of the appropriate ones based on the density of bids distributions evaluation in advance.

If despite new good heuristics and highly efficient algorithms, the problem of finding a good solution still is computationally intractable, then it is very challenging to look ahead and construct a simple and computationally efficient auction with significant economic efficiency.

References

[1] Y. Fujishima, K. Leyton-Brown, Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches, In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI'99, 548–553.

[2] M. H. Rothkopf, A. Pekec, R. M. Harstad. Computationally manageable combinatorial auctions, *Management Science*, **44**(8), 1995, 1131–1147.

[3] T. W. Sandholm. An algorithm for optimal winner determination in combinatorial auctions, *In Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence*, IJCAI'99, 542–547.

[4] T. W. Sandholm, S. Suri. BOB: Improved winner determination in combinatorial auctions and generalizations, *Artificial Intelligence*, **145**, 2003, 33–58.

Department Economics and Social Sciences
University of Plovdiv "Paisii Hilendarski"
Plovdiv 4000, BULGARIA
e-mail: galili@uni-plovdiv.bg

Received 30.09.2003