An Improved Search Technique for Optimal Winner Determination in Combinatorial Auctions

Merlin Nandy
Doctoral Student, MIS
Indian Institute of Management Calcutta
merlin@email.iimcal.ac.in

Ambuj Mahanti
Management Information Systems Group
Indian Institute of Management Calcutta
am@iimcal.ac.in

Abstract

Combinatorial auctions allow bidders to bid their synergistic values. Because of complementarities between different assets, bidders give their preferences not just for particular items but also for sets or bundles of items. This form of auction shows great potential under some given circumstances but is still in its infancy. The difficulty lies in finding the optimal price that is the price of a revenue maximizing set of winning bids. Determining the revenue maximizing set of winning bids is NP-complete. In this paper we review some of the existing approaches as detailed by Sandholm in one of his recent studies. We propose a heuristic, which is a lower bound on the maximal revenue, or an inadmissible heuristic. We then present an algorithm called, "Iterative Threshold Search (ITS) – Hybrid". We show using this algorithm that, although inadmissible, such a heuristic near the optimal can be a better choice in practice than a surely admissible or upper bound heuristic. We establish through experiments that this new heuristic coupled with the proposed search algorithm improves the performance result significantly over the one presented by Sandholm.

1. Introduction

Auctions: An auction is a sale in which a seller presents his product on a public platform. The selling price in an auction is determined by the bids made by interested buyers. The price they bid is based on their own valuation of, and need for, the product. The product is sold to the highest bidder.

Electronic Commerce: Electronic commerce refers to the buying and selling of products and services, and the transfer of funds, through public or private digital networks. Negotiation [1] is the process in which two or more parties multilaterally bargain goods or information for mutual intended gain. An online auction can be the first step towards fully automated electronic negotiations, and furthermore how the new electronic commerce technology provides additional degrees of freedom when running auctions. As a key component of business-to-business Internet marketplaces, auctions pose computational and engineering challenges both in the design of the auction servers and in the construction of software to support the decision tasks of auction participants. They play a major role in today's Electronic Commerce. Important features of Internet auctions [2,9] are larger participation, from geographically dispersed participants and allowing the seller more degrees of freedom. But they have their drawbacks also, like accommodation of anonymous bidders and optimization problems.

Combinatorial Auctions: Combinatorial auction allows bidders to bid for sets of items due to complementarities between different assets, the advantage being that the bidders bid their synergistic values. Example of complementarity is an auction of used electronic equipment, in which a bidder values a particular TV at x and a particular VCR at y but values the pair at $z > x + y$. This study considers such bids only with single unit items. The difficulty in this form of auction lies in finding the revenue maximizing set of winning bids in such an auction.

Combinatorial optimization techniques play an important role in solving these problems. Research in this area of winner determination in combinatorial auctions has been uncovering many important aspects of the field. Sandholm's Branch-on-Items (BOI), Branch-on-Bids (BOB), and Combinatorial Auction BOB (CABOB) are algorithms to solve the winner determination problem and have gained significant popularity. Joni L. Jones, and Gary J. Koehler whose paper involves a mixture of conventional programming heuristics with Constraint Programming [17] have also carried out studies in this area. Kevin Leyton-Brown, Shoham and Tennenholtz worked on multi-unit combinatorial auctions and introduced CAMUS

(Combinatorial Auction Multi-Unit Search) [18] an algorithm for determining the optimal set of winning bids in general multi-unit combinatorial auctions. Gonen and Lehman's contribution [21] also has been in the area of Multi-Unit Combinatorial Auctions (MUCAs). They proposed a method based on the branch and bound technique. Holger H. Hoos, Craig Boutilier [19] developed a framework for studying the solution of combinatorial auctions using stochastic local search (SLS) techniques. The paper by Mito and Fujita proposes three heuristic bid-ordering schemes for solving winner determination problem [20]. In this paper we propose a search algorithm which combines features of IDA* (Iterative Deepening A*) and DFBB (Depth First Branch and Bound) such that even with inadmissible heuristic, the algorithm is guaranteed to find an optimal solution. This paper brings out the fact that admissibility is not always an advantage. It may be possible to design inadmissible heuristics very near to the optimal ones, which could give better performance results. Sandholm [11] talks about an admissible heuristic and a technique of threshold reduction that he uses in his main search running IDA*. We suggest that a natural efficient lower bound heuristic is better than the technique of threshold reduction. We establish the validity of our proposition through detailed experiments. Our algorithm ITS-Hybrid is an integrated algorithm. It is capable of handling both upper bound (admissible) and lower bound (inadmissible) heuristics. It mimics the characteristics of IDA* for an upper bound heuristic and that of DFBB for a lower bound heuristic. Combinatorial optimization problems like winner determination in combinatorial auction are found to have close variants in application domains such as Supply Chain Formation (SCF) and Multi-Dimensional 0-1 Knapsack Problem (MDKP). We have given a brief introduction to SCF and MDKP in later sections.

The paper begins with a discussion on applications work in Section 2. The related work is described in Section 3, followed by a discussion on our proposed study in Section 4. Section 5 introduces the new heuristic. We describe the proposed algorithm in Section 6. Section 7 talks about experimental setup, and empirical findings are presented in Section 8. Section 9 is on summary and conclusion.

Notation and Definitions: In this paper we have followed the standard notation and definitions of [5], which are as follows:

n: a node (or state) in the search tree (or the state space)

s: start node of the search tree

h* (n) : maximal revenue or optimal solution below node n

h* (s) : maximal revenue from the auction or the optimal price

h (n): heuristic value at node n

g (n): sum of the prices of the bids on the path from s to n

f (n): g (n) + h (n)

$h(n) \geq h^*(n)$, for all nodes n in the search tree, implies h is an upper bound on the optimal price or is admissible

$h(n) < h^*(n)$, for atleast one node n in the search tree, implies h is inadmissible. If $h(n) \leq h^*(n)$, for all nodes n in the search tree, then h is said to be a lower bound on the optimal price

2. Applications

Combinatorial problems, such as the one that has been discussed in the paper, i.e., combinatorial auction, have close links with the classical Knapsack problem, Multi-Dimensional Knapsack Problem (MDKP) and problem of Supply Chain Formation (SCF). We discuss these applications below.

2.1 Multi-Dimensional Knapsack Problem (MDKP)

Knapsack problems are an important class of problems that have applications in any area where tasks must be scheduled or budgeted -- in fields such as management, business and defense. The objective of the original knapsack problem is to optimize resource allocation by distributing a fixed amount of resource among several actions in order to get a maximum payoff. A multidimensional knapsack problem (MDKP) is precisely a multi-unit combinatorial auction, where, each item is a dimension, and inserting a bid in the solution bid-set corresponds to putting an object into the knapsack.

2.2 Supply Chain Formation (SCF)

Supply chain management is one of the critical success factors of an organization. But in this rapidly changing economy, companies must adapt to quickly changing demands, prices, and resource availability. The current scenario focuses on this aspect of supply chain that is supply chain formation. *S*upply Chain Formation is the problem of dynamically forming sequences of production in harmony by matching suppliers, producers, and consumers. Production technologies contain strong complementarities and to solve such a problem one of the protocols is the combinatorial

auction protocol, where the bidders bid on combination of items. Hence, supply chain formation can be described as the problem of assembling a network of agents that utilize local knowledge and communication to transform basic goods to composite goods of value. It is formed in a bottom up manner. Goods refer to resource or task. These agents aim to maximize their value in a supply chain. Suppliers, producers, and consumers constitute the network. An allocation is a sub-graph containing nodes as agents that acquire or provide goods, all goods that are acquired or provided, and all edges as the acquiring/providing relationship between the goods and agents in the allocation. An optimal allocation is the one that maximizes the difference between sum of consumer value and sum of supplier cost. Hence it attempts finding the best combination of nodes and edges that would maximize the gain. This problem can be formulated as a combinatorial auction problem [16].

3. Related Work

The field of combinatorial auction is a well-researched area. Currently, the research community is focusing on approximate and heuristic algorithms to solve a combinatorial auction (CA) problem. Many heuristic algorithms have been proposed to solve the winner determination problem with primary contribution from researcher Tuomas Sandholm. Winner determination is NP–complete [10]. He developed search-based algorithms that determine winners optimally [11] with hundreds of items while prior approaches only handled tens. His papers started an active research area within AI and further study was carried out for multi unit CA and exchanges with multiple buyers and sellers. Our discussion in this paper would be within the boundary of single unit CA. His algorithm, BOB [12], solved the winner determination problem and its generalizations even faster by structural improvements, faster data structures, and by identifying and solving tractable special cases during search. He also uncovered a more general polynomially solvable class of combinatorial auctions. Later he developed a yet faster algorithm, CABOB [12], for optimal anytime winner determination. It is currently the fastest algorithm for this purpose. Our work basically proposes modifications to his first paper [11].

Problem Formulation

M: Any set of items to be auctioned
S: Combination of a set of items,
S: $\{S \subseteq M\}$
A: A set of partitions
W: A partition in *A,* which is a set of subsets of items so that each item is included in at most one of the subsets in it.
Highest bid price for a combination of auctioned items

$$b(s) = \max_{i \in \text{bidders}} \{ b_i(S) \}$$

If $b_i(S) = 0$, agent i has not submitted any bid on S
If $b(S) = 0$, no bidder has submitted a bid on combination of S
Objective function:

$$\max_{W \in A} \{ \sum_{S \in W} \underline{b}(S) \}$$

The objective is to maximize the auctioneer's revenue given that each winner pays the price of her winning bid. Determining the winning set of bids is NP-complete and inapproximable, but existing literature shows that optimal search algorithms do very well on the average.

BOI (Branch on Items)

This paper [11] tackled the winner determination problem and proposed a highly optimized algorithm where IDA* was used in the main search.
The strategy adopted for the search algorithm was as follows:

i. Allow bidding on all combinations.
ii. Find the optimal solution i.e. maximization of revenue that the seller can obtain from the bids.
iii. Completely avoid loops and redundant generation of vertices
iv. Capitalize heavily on sparseness of bids i.e. work with relevant partitions, which is a small subset of all partitions.

Sandholm [11] used tree search to achieve these goals. The input is a list of bids, $B_1 \ldots\ldots\ldots B_n$, one for each $S \in S$:
$\{B_1 \ldots\ldots\ldots B_n\} = \{(B_1.S, B_1.\underline{b}), \ldots\ldots (B_n.S, B_n.\underline{b})\}$
where $B_j.S$ is the set of items in bid *j,* and $B_j.\underline{b}$ is the price in bid *j.*
Each path in the search tree consists of a sequence of disjoint bids, that is, bids that do not share items with each other. A path in the tree corresponds to a partition.
Let *U* be the set of items that are already used on the path.

$$U = \bigcup_{j \mid B_j \text{ is on the path.}} \{ B_j.S \}$$

And let F be the set of unallocated items:
$F = M - U$

A path ends when no bid can be added to the path. This occurs when for every bid, some of its items have already been used on the path. As the search proceeds down a path, a tally *g* is kept of the sum of the prices of the bids on the path:

$g = \sum \{ B_j.\underline{b} \}$
$j \mid B_j$ is on the path

At every search node, the revenue from the path, given by the g-value, is compared to the best g-value found so far in the search tree to determine whether the current solution path is the best one or not. If so, it is stored as the best solution found so far. When the search is over, the stored solution is the optimal solution.

Sandholm [11] has introduced two heuristics, which are as follows:

Heuristic 1

Take the sum over unallocated items of the item's maximal contribution. An item's maximal contribution is the price of the bid divided by the number of items in that bid, maximized over the bids to which the item belongs. We call this heuristic as h_{san}

Heuristic 2

Identical to Heuristic 1, except that recomputing an item's maximal contribution every time a bid is appended to the path since some other bids may now be excluded. A bid is excluded if some of its items are already used on the current path in search or if it constitutes a noncompetitive pair with some bid on the current path.

Remark: *Sandholm [11] felt that both these heuristics were an upper bound on the optimal revenue. And in all his experiments he had used Heuristic 1. Our new heuristic presented later is quite similar to this Heuristic 2, which we prove that it is actually a lower bound.*

4. Proposed Study

Sandholm [11] proposed and experimented with an upper bound heuristic, which we refer to as h_{san}. He conducted two sets of experiments. Firstly, he ran IDA* using h_{san} and, secondly he adopted a measure of threshold reduction which reduces the number of iterations of algorithm IDA*. This technique of reducing the number of thresholds and thereby reducing the number of iterations in IDA* search was originally proposed by Chakrabarti et al [7]. It was shown in [7] that the successive thresholds could be modified in IDA* search, so as to guarantee the expansion of a sufficient number of new nodes in successive iterations resulting in total expansion of O (N) nodes, as opposed to O (N^2) in IDA*. The problem in this case would be that though the number of iterations is reducing, we still don't know the exact number of iterations that would actually take place. As Sandholm [11] argues, if the reduction in threshold is not very high then the number of iterations does not reduce significantly, whereas, if the reduction in threshold is very high then the threshold may fall too much below the optimal. When the threshold goes below the optimal, the search becomes like DFBB [11]. But if it falls much below the optimal then it results in too many node expansions and generations.

In this study we propose an algorithm, which is a modified version of Iterative Threshold Search (ITS) [8]. Here it combines features of both IDA* and DFBB algorithms. We also propose a heuristic, which is a lower bound on the optimal and lies very close to it. ITS-Hybrid with this heuristic returns an optimal solution in one iteration. It runs quite efficiently because of the accuracy of the heuristic [3,4]. ITS-Hybrid with an admissible heuristic works in a way identical to IDA*. Thus, when it runs with the upper bound heuristic proposed in [11], it gives identical performance results.

5. New Heuristic

In this section we present a new heuristic for solving the winner determination problem.

Heuristic h_1:

Categorize bids item-wise, i.e., list all bids with item 1, with item 2 and so on. Find the maximum average item price for all items. In the first iteration, the item whose maximum average price is highest among the given set of items, is selected. Pick the bid with the maximum average item price for the selected item. Now drop all the bids that contain items that are present in the selected bid. Add the bid price for the selected bid to TotalPrice and update its value. We then consider remaining bids only and select a bid for the next item whose maximum average price is highest among the remaining items. Resolve ties arbitrarily. Finally, heuristic h_1 = TotalPrice

We now explain the computation of h_1 through an example, and present a procedure for it later in this section. We also give an example of the complete search tree below.

Example 1: In Table-1 we illustrate the computation of h_1. Initially all the bids are present in iteration 1. For each bid the total and per item average prices are shown in column 1 and 2 respectively. At the end of iteration 1, bid (3,5) is selected on the basis of the maximal average item price 5.5 occurring for items 3 and 5. Then all bids containing either 3 or 5 or both

are dropped. A horizontal line marks a dropped bid. The price of the selected bid is 11. The variable TotalPrice is also set to 11. In the second iteration item 4 gives the maximal average item price for bid (4). Next this bid is selected having price 4. The TotalPrice is updated to 15. Similarly, in the third and last iteration bid (1,2) for item 1 is selected having price 4, and the TotalPrice is updated to 19.

Thus, h_1 = TotalPrice = 11 + 4 + 4 = 19.

For this example, h_{San} = Sum of the maximal average contribution of each and every item = 4.33+4+5.5+4+5.5 = 23.33. Now for the same example h* = Optimal Price = Maximal revenue from

Table 1: Showing the iteration-wise computation of h_1

Total bid price	Avg. Price	Bids		
		Iteration1	Iteration 2	Iteration 3
1	1	1	1	1
2	2	2	2	2
3	3	3	~~3~~	~~3~~
4	4	4	**4**	~~4~~
5	5	5	~~5~~	~~5~~
4	2	1,2	1,2	**1,2**
13	4.33	1,3,5	~~1,3,5~~	~~1,3,5~~
8	4	1,4	1,4	~~1,4~~
8	4	2,5	~~2,5~~	~~2,5~~
11	5.5	**3,5**	~~3,5~~	~~3,5~~
SELECTION		Item: 3 Bid: 3,5 Price: 11 TP=11	Item: 4 Bid: 4 Price: 4 TP=15	Item: 1 Bid: 1,2 Price: 4 TP=19

TP : TotalPrice

a set of disjoint bids covering all items = 11(from the bid 3,5) + 8 (from the bid 1,4) + 2 (from the bid 2) = 21.

Example 2:

We now illustrate the complete search tree in Figure 1 for the above example. Here the start node represents a null node, which is shown as ϕ. Every other node is represented by a bid. We show the items in each bid within a rectangular box. The bid prices are not shown here. The first level of the tree considers bids that include item 1. The children of a node are those bids: a) that include the item with the smallest index among the items that have not been used on the path yet, and b) that do not include items that have already been used on the path. Bottom most nodes of the tree are the leaf nodes and each path represents a partition. The total revenue of a path is the sum of the prices of the participating bids in it. A path with maximal revenue will represent an optimal path.

A heuristic search algorithm would normally explore a small portion of a complete search tree. The size of the portion explored will vary with the quality of the heuristic and the efficiency of the search algorithm chosen.

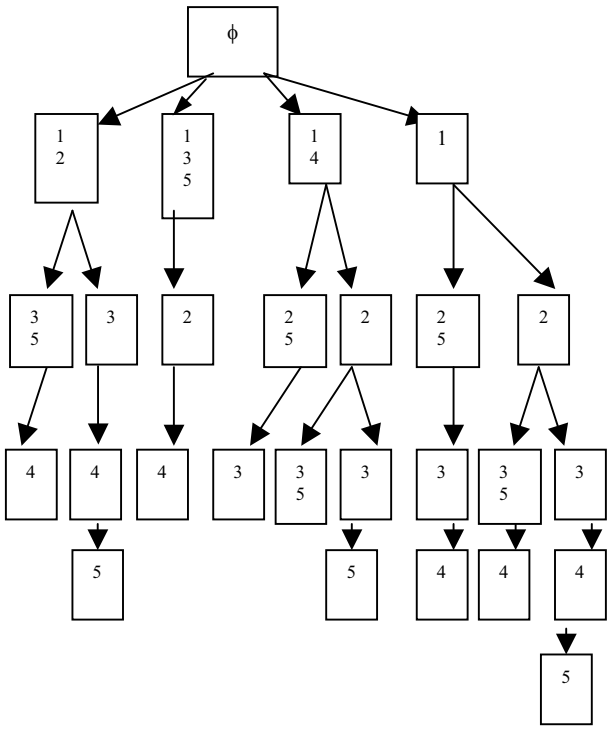

Figure 1: A complete search tree

5.1 Procedure for the new heuristic

Heuristic h_1

1. *Create two arrays, one storing all the distinct items in each bid and the other storing the average prices of the bids. Set a marked / unmarked bit for each item and each bid to 0, showing them as initially unmarked. Set* $h_1 \leftarrow 0$.
2. *Repeat the following steps until there is no unmarked item.*
   a. *Select a bid having the maximum average item price among all unmarked bids.*
   b. *For the bid selected (in step a), count the number of items present in it. Copy the item indices of these items to a list called droppedItems. Mark those items by setting their marked / unmarked bit to 1.*
   c. *Calculate the bid price as product of the average item price and number of items found in step (b).*
   d. $h_1 \leftarrow h_1$ + *bid price*

e.Now for every bid, the marked / unmarked bit is set to 1 (marked), if it contains any of the droppedItems.□

6. Algorithm – ITS Hybrid

Under any best first search algorithm an admissible heuristic [5,14] can improve the search efficiency significantly, if it estimates the optimal accurately. Such a heuristic helps in confining the search to a small portion of the complete search space. But the design of an accurate admissible heuristic is a difficult task. Moreover, an admissible heuristic, which estimates the optimal with high percentage of error, would be of little use. Thus, often, it is more beneficial to work with a heuristic that may not be admissible but generally estimates the optimal closely. In the current context of combinatorial auctions, an admissible heuristic would act as an upper bound on the maximal revenue. If the quality of this heuristic estimate is poor then the search algorithm IDA* would make too many iterations [11] before finding an optimal solution.

On the other hand, if we have a lower bound heuristic, then by using this heuristic algorithm DFBB can find an optimal solution in the first iteration itself. If the error in the heuristic estimation is small then DFBB performs efficiently. Thus keeping these two search aspects in mind here we integrate the features of IDA* and DFBB into one algorithm, such that it should run equally efficiently like IDA* and DFBB while running with upper bound and lower bound heuristics respectively. Here we present such an algorithm and call it as ITS-Hybrid.

Algorithm – ITS Hybrid

1. *Initialize current_node = start node:*
   Set current_threshold = f-value of the start node; goal_found = 0; number_of_iteration=0;
   // f-value (node) = g-value (node) + h-value (node)
2. *Set next_threshold = 0 and current_best_solution_cost = 0;*
3. *Do while (goal_found = 0 and f-value (current_node) >= current_threshold)*
   3.1 Increment number_of_iteration by 1
   3.2 Do while (1)
   Call function to expand current_node and generate its children
   a.1 ***If*** *(number of available children of the current_node >=1)*
   b.1 ***If*** *(f-value (current_node) >= current_threshold)*
   // SEARCH PROCEEDS IN DEPTH
   Generate available leftmost child node and Set current_node = child_node;
   Reduce number of available children of the current_node by 1; Continue;
   b.2 ***Else*** *// NEXT THRESHOLD*
   Set next_threshold = max (next_threshold, f-value (current_node))
   c.1 ***If*** *the current_node is not the root node*
   Backtrack by setting current_node = parent node Continue;
   c.2 ***Else*** *break;*
   a.2 ***Else***
   b.1 ***If*** *at an intermediate node // NO VALID CHILDREN*
   *c.1****If*** *the current_node is not the root node*
   Backtrack by setting current_node = parent node; continue;
   *c.2****Else*** *break;*
   b.2 ***Else*** *// IT IS AT A LEAF NODE*
   c.1 ***If****(number_of_iteration > 1)*
   Current_best_solution_cost = f-value (current_node)
   Set goal_found =1; break;
   *c.2****Else*** *FIRST ITERATION*
   Current_best_solution_cost = max (Current_best_solution_cost, f-value (current_node))
   Set goal_found =1;
   *d.1****If*** *the current_node is not the root node*
   Backtrack by setting current_node = parent node;Continue;
   *d.2****Else*** *break;*
4. *Output current_best_solution_cost;* □

7. Experimental Setup

To evaluate the performance of the algorithm with two different heuristics (namely, h_{san} and h_1), we conducted experiments on a 1 GHz Pentium III machine with 512 MB RAM. Three of CATS (Combinatorial Auction Test Suite) distributions [13] were considered for experiments. CATS is a suite of distributions for modeling realistic bidding behavior. This suite is built on data from specific applications of combinatorial auctions. The distributions are parameterized by number of goods and bids, facilitating the study of algorithm performance. Our algorithm was tested on distributions: **regions**, **scheduling**, and **arbitrary.** For each one of them we used the default parameters in the

CATS instance generators, and varied the number of items and bids. For heuristic comparison graphs we took 50 problem instances of a single size having 20 items and 200 bids from each CATS distribution. Each of the heuristic graphs represents one of the distributions from CATS. For performance evaluation of the algorithms, for each CATS distribution, six different sets of data were generated with varying problem sizes, i.e., different combination of (items, bids). The six problem sizes considered are (10,100), (15,130), (20,160), (40,200), (50,250), (65,330). For each problem size 75 random problem instances were taken. Sandholm [11, 15] also used these distributions.

8. Experimental Results

In the following three subsections we discuss the estimation accuracies of heuristics h_{san} and h_1, execution time speedups, and number of nodes expanded and generated by the ITS-Hybrid algorithm under different test scenarios.

8.1. Estimation accuracies of h_{san} and h_1

The distance that lies between a heuristic estimate and the optimal value measures the error in the heuristic estimation. In all the three graphs below (Figures 2a, 3a, 4a), generated from the CATS distribution mentioned above, for 50 problem instances each, the distance between h_1 and h* graphs is found to be much less than the distance between h_{san} and h* graphs. This establishes that h_1 estimates h* more accurately than $h_{san.}$ The Table-2 presents mean percentage of error, and the standard deviation of error calculated from the 50 random problem instances of size 20 items and 200 bids under three different distributions of CATS. As can be found from the table the mean percentage error for h_1 varies from 5.8006% to 9.3498% in comparison to that of h_{san} widely varying from 41.1641% to 60.6184%, signifying that h_1 is many times more accurate than h_{san}. The values of standard deviation also support h_1 favorably showing that the dispersion of values is much less compared to $h_{san.}$

8.2 Time Speedups

It should be noted that our experimental results and Sandholm [11] are not directly comparable, as our work does not use the preprocessors that were used in [11]. These preprocessors prune the search space drastically and thereby reduce the required search effort.

Table 2: Error analysis for all three CATS distributions

	Arbitrary Av. h* = 137.5612		Regions Av. h* = 136.7796		Scheduling Av. h* = 135.9932	
	h_1	h_{san}	h_1	h_{san}	h_1	h_{san}
Mean % error	8.1	58.1	9.3	60.6	5.8	41.1
Standard deviation	7.2	10.4	10.1	13.1	8.6	44.6

Since the basic objective of this study is to compare the power of two heuristics, we have avoided these preprocessors. However, had these preprocessors been also used, then problems of much larger size could be solved using the same execution time.

Figures 2b, 3b, 4b show the execution times of ITS-Hybrid using our proposed heuristic h_1 compared to h_{san}, and algorithm with threshold-modification at each iteration. We run ITS-Hybrid in all three cases. When using h_1 it runs like DFBB. With h_{san} it runs like IDA*. Under the technique of threshold-modification, it initially runs like IDA* and in the last iteration it runs like DFBB.

Table-3 gives the percentage of excess execution time taken for h_{san}, and threshold-modification (as threshold = 0.95 * threshold)[♣] with h_{san}. Note that per node computation time of h_{san} is almost negligible compared to the computation time of h_1. This explains why the execution time plots for algorithm with h_1 and algorithm with threshold -modification lie so close in the graphs[♣]. Summarily, algorithm ITS-Hybrid with heuristic h_1 performs exceedingly well in terms of execution time compared to its running with h_{san}.

8.3 Node Expansions and Generations

The gaps in terms of number of nodes generated and expanded in the three cases make the difference in execution times. We present the node expansion and generation data in Tables 4, 5, and 6. The ratio of nodes generated by the algorithm with h_{san} to the algorithm with h_1 varies from 7.6 to 36.6 times for CATS regions, from 13.17 to 62.41 times for CATS arbitrary and from

[♣] Sandholm [11] also used 0.95 - threshold-modification factor

[♣] The graphs 2b, 3b and 4b have been plotted with varying scales to fit them in the pages.

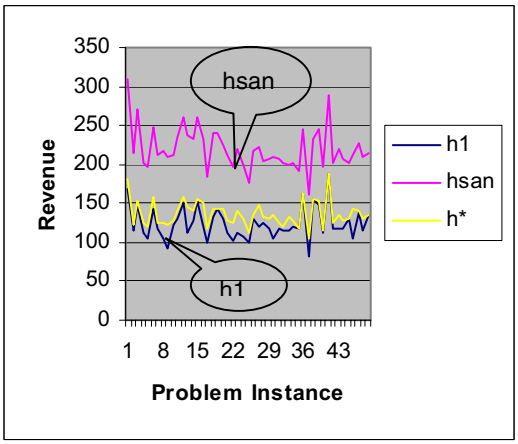

Figure 2a: Heuristic comparison of 50 problem instances of size 20 items and 200 bids on CATS regions

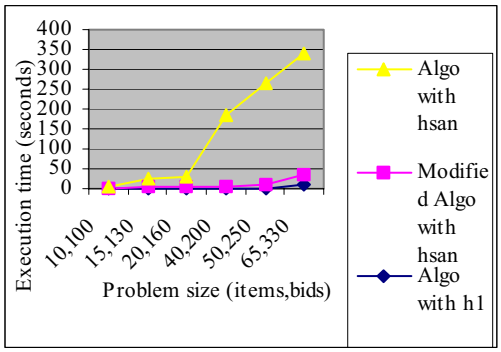

Figure 2b: Av. execution time under h_{san}, h_1, and h_{san} with threshold modification by 5% decrease for CATS regions

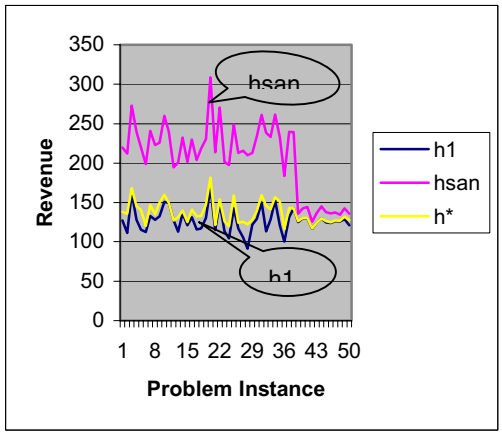

Figure 3a: Heuristic comparison of 50 problem instances of size 20 items and 200 bids on CATS scheduling

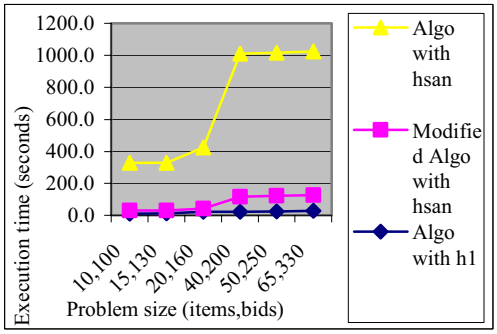

Figure 3b: Av. execution time under h_{san}, h_1, and h_{san} with threshold modification by 5% decrease for CATS scheduling

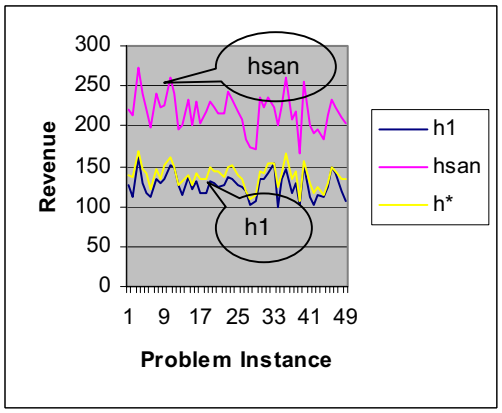

Figure 4a: Heuristic comparison of 50 problem instances of size 20 items and 200 bids on CATS arbitrary

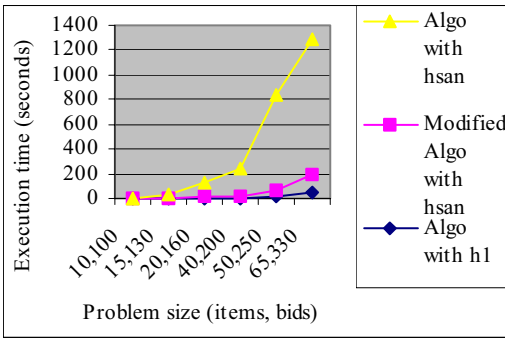

Figure 4b: Av. execution time under h_{san}, h_1, and h_{san} with threshold modification by 5% decrease for CATS arbitrary

Table 3: % Excess execution time taken by (IDA*, h_{san}) and (IDA* with threshold=0.95*threshold, h_{san}) for 6 different problem sizes of 75 random problem instances each on three distributions of CATS

Scheduling		Arbitrary		Regions	
% Excess time for h_{san}	% Excess time for Modified Algo with h_{san}	% Excess time for h_{san}	% Excess time for Modified Algo with h_{san}	% Excess time for h_{san}	% Excess time for Modified Algo with h_{san}
2636.3	77.5	2739.1	738.6	2149.1	454.2
2560.4	74.9	9090.3	1285.3	3756.7	327.6
1692.5	4.6	10298.9	815.6	4251.6	262.4
3905.6	319.7	5963.8	320.6	9504.7	78.0
3515.7	287.8	5492.5	304.0	14287.4	353.1
3196.6	261.7	2596.7	259.7	2779.5	149.1

Legend:

P: Problem sizes (items,bids) - 1: (10,100), 2: (15,130), 3: (20,160), 4: (40,200), 5: (50,250), 6: (65,330)

I: No. of Iterations

NE: No. of Nodes Expanded

NG: No. of Nodes Generated

Modified Algo with h_{san} (Modified Threshold in IDA* = 0.95*Threshold)

Table 4: Average performance data (rounded to nearest integer) in terms of number of iterations, number of nodes expanded and number of nodes generated for 6 different problem sizes of 75 random problem instances each on CATS regions

	Algo with h_1			Algo with h_{San}			Modified Algo with h_{San}		
P	I	NE	NG	I	NE	NG (h_{san}/ h_1)	I	NE	NG (h_{san}/ h_1)
1	1	1174	347	10	1321	2642 (7.61)	4	368	736 (2.1)
2	1	449	896	20	4329	8659 (9.66)	4	549	1097 (1.2)
3	1	555	1107	22	5618	11239 (10.2)	4	557	1112 (1.1)
4	1	1096	2190	55	40150	80332 (36.6)	5	1346	2693 (1.3)
5	1	1107	2211	61	45006	90051 (40.7)	4	710	1419 (0.7)
6	1	3289	6574	83	66101	132253 (20.1)	7	3359	6724 (1.1)

1.17 to 52.01 times for CATS scheduling. On the other hand, with the threshold-modification, the node generation factor varies only from 0.7 to 2.1 times for CATS regions, from 1.1 to 6.1 times for CATS arbitrary and from 0.1 to 1.2 times for CATS scheduling. The variations in node expansions are almost identical to node generations.

Table 5: Average performance data (rounded to nearest integer) in terms of number of iterations, number of nodes expanded and number of nodes generated for 6 different problem sizes of 75 random problem instances each on CATS arbitrary

	Algo with h_1			Algo with h_{San}			Modified Algo with h_{San}		
P	I	NE	NG	I	NE	NG (h_{san}/ h_1)	I	NE	NG (h_{san}/ h_1)
1	1	144	283	13	1864	3727 (13.17)	5	594	1190 (4.2)
2	1	185	366	27	6869	13743 (37.55)	6	1106	2217 (6.1)
3	1	961	1917	53	29027	58081 (30.29)	7	2091	4187 (2.2)
4	1	1449	2895	65	35053	70142 (24.23)	7	2556	5119 (1.8)
5	1	5400	10798	161	220142	440408 (40.79)	9	5633	11274 (1.1)
6	1	138-00	27598	360	861009	1722338 (62.41)	10	132-22	26453 (0.9)

Table 6: Average performance data (rounded to nearest integer) in terms of number of iterations, number of nodes expanded and number of nodes generated for 6 different problem sizes of 75 random problem instances each on CATS scheduling

	Algo with h_1			Algo with h_{San}			Modified Algo with h_{San}		
P	I	NE	NG	I	NE	NG (h_{san}/ h_1)	I	NE	NG (h_{san}/ h_1)
1	1	8811	17619	40	26159	52344 (2.97)	5	1508	3029 (0.17)
2	1	8936	17623	40	26164	52349 (2.97)	5	1516	3037 (0.17)
3	1	13333	26659	27	15586	31186 (1.17)	5	1986	3978 (0.15)
4	1	6755	13510	77	122402	244860 (18.12)	7	7838	15684 (1.16)
5	1	3522	7045	79	14317	28647 (4.07)	10	5564	11134 (1.58)
6	1	4187	8375	82	217734	435575 (52.01)	12	9379	18767 (2.24)

9. Conclusion

Combinatorial auction is a thrust area of research in Electronic Commerce. Sandholm [11,12,15] has presented some important results in this domain. He developed an upper-bound heuristic called, h_{san} , and used search algorithms IDA* and DFBB for determining the revenue maximizing set of winning bids. He also presented some improved performance results in terms

of the execution time and total number of nodes generated using the threshold-modified IDA*. This was done by multiplying the threshold value of an iteration by a suitable factor. Sandholm used a threshold modification of 5% by setting threshold = 0.95*threshold. This technique reduces the number of iterations in IDA* and results in less number of node expansions and generations. This idea was originally proposed in [7]. Under this threshold modification technique the last iteration of the algorithm runs like DFBB. In this study we propose an algorithm called, "Iterative Threshold Search (ITS)– Hybrid". This algorithm combines the features of IDA* and DFBB. It runs like IDA* if the heuristic is an upper bound (admissible), or like DFBB if the heuristic is a lower bound (inadmissible). We also propose a lower-bound heuristic called, h_1. We show that this new heuristic is much more accurate with respect to its mean distance from the optimal and the corresponding standard deviation than the upper-bound heuristic used by Sandholm. We have tested the dispersion and central tendency on three sets of samples of size fifty each taken from three different problem spaces chosen from CATS. Finally, we observe that ITS-Hybrid with h_1 performs significantly better than both IDA* with h_{san}, and the threshold-modified IDA* with h_{san}. Besides this graphical presentation of the experimental data, detailed statistical analysis to show that the results are statistically different could be one interesting issue of future research. We believe this study would enhance the results proposed and pioneered by Sandholm. This would encourage better use of AI heuristic search techniques in Combinatorial Auctions.

10. References

[1] Manoj Kumar, Stuart I. Feldman, "Business Negotiations on the Internet", IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598

[2] Cassady, Ralph Jr., "Auctions and Auctioneering", University of California Press, Berkeley, CA, 1967.

[3] Harris, L.R., 1974. The Heuristic Search Under Conditions of Error. Artificial Intelligence 5 (3): 217-234

[4] Bagchi, A., Mahanti, A., 1983. Search Algorithms Under Different Kinds of Heuristics – A Comparative Study. JACM 30(1): 1-21

[5] Judea Pearl, "Heuristics", Addison-Wesley Publication Company, 1984

[6] P.P. Chakrabarti, S. Ghose, A. Pandey, S.C. De Sarkar, Increasing Search Efficiency Using Multiple Heuristics, Information Processing Letters 30 (1989) 33-36

[7] P.P. Chakrabarti, S. Ghose, U.K. Sarkar, S.C. De Sarkar, Reducing Reexpansions in Iterative-Deepening Search by Controlling Cutoff Bounds, Artificial Intelligence 50 (1991) 207-221

[8] Subrata Ghosh, Ambuj Mahanti, Dana S. Nau: ITS: An Efficient Limited-Memory Heuristic Tree Search Algorithm. AAAI 1994: 1353-1358

[9] Carrie Beam, Arie Segev, and J. George Shanthikumar, "Electronic Negotiation through Internet-based Auctions", CITM Working Paper 96-WP-1019, December 1996

[10] M.H. Rothkopf, A. Peke¡ c, R.M. Harstad, Computationally Manageable Combinatorial Auctions, Management Sci. 44 (8) (1998) 1131–1147.

[11] Tuomas Sandholm. "An Algorithm for Optimal Winner Determination in Combinatorial Auctions", In *IJCA*I, pages 542–547, 1999., First appeared as Washington Univ., Dept. of Computer Science, WUCS-99- 01, Jan. 28th.

[12] Tuomas Sandholm and Subhash Suri, "Improved Algorithms for Optimal Winner Determination in Combinatorial Auctions and Generalizations", In *AAA*I, pages 90–97.

[13] Leyton-Brown, Mark Pearson, Yoav Shoham, Towards a Universal Test Suite for Combinatorial Auction Algorithms, 2000

[14] Richard E. Korf, Recent Progress in the Design and Analysis of Admissible Heuristic Functions, AAAI, 2000

[15] Tuomas Sandholm, S. Suri, A. Gilpin, D. Levine, "CABOB: A Fast Optimal Algorithm for Combinatorial Auctions", in: Proc. IJCAI-01, Seattle, WA, 2001, pp. 1102–1108.

[16] WE Walsh, MP Wellman, and F Ygge Combinatorial Auctions for Supply Chain Formation, Proc. IJCAI-01, Seattle, WA, 2001, pp. 1102–1108.

[17] Joni L. Jones, Gary J. Koehler, An Allocation Heuristic for Combinatorial Auctions, 2001

[18] Kevin Leyton-Brown and Yoav Shoham and Moshe Tennenholtz, An Algorithm for Multi-Unit Combinatorial Auctions, AAAI, 2000

[19] Holger H. Hoos, Craig Boutilier, Solving Combinatorial Auctions using Stochastic Local Search, AAAI, 2000

[20] Masaya Mito, Satoshi Fujita, On Heuristics for Solving Winner Determination Problem in Combinatorial Auctions, IPSJ SIGNotes Algorithms

[21] Rica Gonen, Daniel Lehmann, Optimal Solutions for Multi-Unit Combinatorial Auctions:Branch and Bound Heuristics, ACM, 2000