Optimal Solutions for Multi-Unit Combinatorial Auctions: Branch and Bound Heuristics

Rica Gonen
School of Computer Science and Engineering
Hebrew University, Jerusalem 91904, Israel
rgonen@cs.huji.ac.il

Daniel Lehmann
School of Computer Science and Engineering
Hebrew University, Jerusalem 91904, Israel
lehmann@cs.huji.ac.il

ABSTRACT

Finding optimal solutions for multi-unit combinatorial auctions is a hard problem and finding approximations to the optimal solution is also hard. We investigate the use of Branch-and-Bound techniques: they require both a way to bound from above the value of the best allocation and a good criterion to decide which bids are to be tried first. Different methods for efficiently bounding from above the value of the best allocation are considered. Theoretical original results characterize the best approximation ratio and the ordering criterion that provides it. We suggest to use this criterion.

Keywords

Combinatorial Auctions, Branch and Bound

1. MULTI-UNIT COMBINATORIAL AUCTIONS (MUCAS)

Auctions have been used from times immemorial, but the renewed modern interest in auctions stems from:

- their increased use for selling off government property after WWII and later in extensive denationalizations, and
- the theoretical breakthroughs started by [14].

A very recent surge of interest in auctions stems from their surprising success on the internet. Many foresee internet auctions in which:

- the number of buyers will be large
- the number of items for sale will be large
- the mechanism used to determine the allocation and the payment will be complex both from the game-theoretic strategic and from the computational points of view.

EC'00, October 17-20, 2000, Minneapolis, Minnesota.

This work deals with computational aspects of winner determination in multi-unit combinatorial auctions. Combinatorial auctions are mechanisms in which a number of items are up for sale and bidders bid for subsets of those items. In combinatorial auctions one typically assumes that all items for sale are different. Computational aspects of combinatorial auctions have been considered in [10, 11, 1, 6, 8, 12]. Multi-unit combinatorial auctions are those combinatorial auctions in which certain of the items for sale are identical. We assume n different commodities and k_j identical items of commodity j for $j = 1, \ldots n$. The problem we are trying to solve is that of determining the optimal allocation of the items. A set of bids is given: each bid requests a number of units (possibly zero) from each commodity and offers a price p for the whole set. A subset of the set of all bids is conflict-free if, for each commodity, the sum of the units requested does not surpass the number of units for sale. The problem is to find a conflict-free subset that maximizes the sum of the prices proposed. Multi-unit combinatorial auctions have been considered in [7].

2. BRANCH-AND-BOUND SEARCH

The problem of finding the optimal solution of a multi-unit combinatorial auction is a hard problem, i.e., requires extensive resources, as will be shown formally in Section 5. It is therefore essential that one designs carefully the algorithm and uses the available resources smartly. The method we propose to use is based on the branch-and-bound search technique developed by operations researchers [5]. The use of some branch-and-bound techniques for finding optimal solutions for combinatorial auctions has been considered in [1] and more thoroughly in [8]. The case of multi-unit combinatorial auctions is considered in [7] where a wealth of heuristics are proposed. This paper concentrates on the fundamentals and characterizes the best ordering heuristics.

The general description of the branch-and-bound technique below is well-known. The resource in shortest supply is space (i.e. memory) and therefore we propose to use a depth-first search, with backtracking to cover the whole search space. At any time we essentially keep in memory only a partial solution, i.e., a set of non-conflicting bids. We look for a bid that can be added to the partial solution without creating a conflict and if there is none, know we have attained a possible solution and backtrack. The additional information of which we need to keep track consists only of the best solution found so far and backtracking information. On the whole the space requirements are linear in the size of the problem. Note that, at any time, we have allocated part of the items and are left with a smaller problem of the same type: finding the optimal solution of another, smaller, MUCA.

An exhaustive search such as sketched above will require exponential time for all possible inputs. One may hope to find an algorithm that will run faster on the *easy* inputs. Even though the theoretical considerations of Section 5 seem to preclude that there be a majority of *easy* cases, one may hope that many of the problems that will have to be solved in practice will turn out to be easy. We are hoping for an algorithm that will run fast on those easy problems. The powerful *branch-and-bound* idea is that, in many situations, we may be able to conclude on the spot that adding a specific bid to our partial solution is hopeless and that we can immediately backtrack, saving us the exploration of a whole sub-tree, i.e., enabling us to *prune* a whole sub-tree. All we need for that is a good upper-bound for the value of the optimal solution of the smaller MUCA concerning the items still unallocated. If we have such an upper bound and if the sum of this upper bound and the values of the bids of our partial solution is not larger than the value of the best solution found so far, we may backtrack immediately: no extension of our partial solution may have a value larger than that of the solution we already know. In Section 3, we shall propose a number of ways to bound from above the value that can be obtained from auctioning the still unallocated goods. There is obviously no need to commit to a single method: one may use many such methods, obtain a number of different upper bounds and use the smallest of all.

The pruning described above is most effective if the best solution found so far, that is used to decide whether to prune or not, is in fact a good solution. If we find the optimal (or a very good) solution early on, even though we shall not know that it is the optimal solution (or how good it is), we shall be able to prune more sub-trees than if we had a best solution so far that is not as good. It is therefore essential that we find the best solutions early on. We should, therefore, carefully choose the order in which we try to enter the bids into partial solutions and try the most promising bids first. To summarize, we need a good heuristic to pick the most promising bid in a set of bids, and this is the third component of branch-and-bound method. We shall use such a heuristic to find the most promising bid in the smaller MUCA concerning the items still unallocated iteratively to find the most promising of those bids that does not conflict with the partial solution at hand. We shall then use the upper-bound described above to decide whether to prune the whole sub-tree or not. Notice that we do not propose to rank the bids once and for all: the most promising bid may depend on the partial solution at hand. For example, the normalized criteria described below in Equations 5 and 7 imply such a dependency. In [7] the heuristic proposed to choose the next bid takes into account the whole set of bids. We restrict our attention to heuristics that consider each bid separately. In Section 4 we shall consider different criteria for this task and in Section 5 we shall characterize the best one. It turns out that this optimal criterion does not depend on the partial allocation, or, equivalently, on the set of items still unallocated, and therefore one can simply order the bids according to this criterion once and for all at the start of the algorithm.

In this paper, we shall assume that any subset of bids such that, for any commodity, the sum of the units requested does not exceed the total number of units available is a legal solution. We shall not consider explicitly the case in which certain bids are exclusive, i.e., no legal solution can contain certain pairs of bids. The generalization of our (and essentially any) branch and bound algorithm to the case of possibly exclusive bids is quite straightforward, though: when considering whether to add a specific bid to a partial allocation, check not only whether there are enough units left to satisfy it and whether the sub-tree can be pruned, check also whether the bid considered is not excluded by some bid of the partial solution.

3. BOUNDING FROM ABOVE

We noticed that, at any point, we have a partial allocation, i.e. a set of non-conflicting bids, and are trying to extend it. The best possible extension is, again, the optimal solution to the MUCA of the remaining units to the remaining bids. We shall therefore look for ways to bound from above the value of any MUCA. We also noticed that one may use a number of different methods, obtain numerous upper-bounds, and take the smallest one. We propose three different types of methods.

3.1 Linear Programming

The first method we propose is Linear Programming. Our problem is an integer-programming problem: find $x_i \in \{0, 1\}$ for each bid i that maximizes $\sum_i x_i \, p(i)$ while satisfying the linear constraints: $\sum_i x_i \, q(i, j) \leq k_j$. The variable x_i indicates whether bid i is in the optimal solution ($x_i = 1$) or out of it ($x_i = 0$). The relaxed linear programming problem in which we allow $x_i \in [0, 1]$ will provide a value $\sum_i x_i \, p(i)$ that is an upper-bound for the value of the optimal solution to the original MUCA. Notice that the solution of the relaxation allows for fractional allocation of bids: $x_i = 1/2$ may be interpreted as allocating half of the quantities requested by i at half the price proposed. In the worst case, for theoretical reasons, it seems that the upper bound provided by LP cannot be good, but there are reasons to think that in practice, and especially for large k_j's the bound could be pretty good quite often. An extended discussion of the relation between the solutions to the original and the relaxed problems may be found in [8].

Notice also that, if the optimal solution to the relaxed problem is integral, then we know it is the optimal solution to the original problem. The integrality of the solution to the LP problem is the signal for backtracking.

3.2 Projections

Another, quite different, idea is to consider only one commodity. By projecting bid i on commodity j, to $i' = \langle 0, \ldots, q(i, j), \ldots, 0, p(i) \rangle$, we transform our original MUCA into a knapsack problem. The optimal solution to the knapsack problem is obviously an upper bound for the optimal solution of the original MUCA. The knapsack being polynomially-approximable at any precision, one may easily obtain an upper bound in this way, in fact n such upper bounds, one for each commodity.

3.3 Average price consideration

The bounds described above are not difficult to compute and quite different in spirit (and probably results). Our next bound is related to the Linear Programming one of Section 3.1: it always provide an upper bound that is larger than or equal to the one provided by LP, but it is extremely easy to compute. Consider the average price per unit for bid i: $a(i) = p(i) \,/\, \sum_{j=1}^{n} q(i,j)$. Suppose a is the largest of the $a(i)$'s. Then clearly the value of the optimal solution to the MUCA is bounded from above by $a \sum_{j=1}^{n} k_j$. The value of the optimal solution to the relaxed LP problem is also bounded by this quantity.

4. CHOOSING THE MOST PROMISING BID

Given positive integers $k_1, \ldots, k_n$, how should one compare the attractiveness, at first sight, of two bids $a = \langle q_1, q_2, \ldots, q_n, p\rangle$ and $a' = \langle q'_1, q'_2, \ldots, q'_n, p'\rangle$? We may as well say we are looking for a criterion r, i.e. a real-valued function of the bid a, possibly depending on the parameters k_j's such that $r(a) \geq r(a')$ iff a is more promising than a'.

Some obvious considerations may be made immediately. The corresponding heuristics have been incorporated in [7]. If the quantities q_j and q'_j are equal, but p is larger than p', one should obviously prefer a to a' and in fact, since the two bids conflict, one can remove a' from consideration. Similarly, if the prices p and p' are equal and $q_j \leq q'_j$ for every j, $1 \leq j \leq n$, then one should prefer a to a' and remove a' from consideration. In other terms, the function r should be monotonic in p and anti-monotonic in the q_j's. Notice that any such function will have the effect of removing from consideration all the dominated bids as above.

Many functions come to mind. The simplest one is probably:

$$r(a) = p\,, \tag{1}$$

i.e., order the bids by the price they propose. The most natural function that comes to mind is probably:

$$r(a) = \frac{p}{\sum_{j=1}^{n} q_j}\,, \tag{2}$$

that orders the bids by average price per unit. But one may prefer to normalize the elements of the sum by the number of units of each commodity that are available and use:

$$r(a) = \frac{p}{\sum_{j=1}^{n} q_j / k_j}\,. \tag{3}$$

If one has a geometrical bind one may prefer the Euclidean:

$$r(a) = \frac{p}{\sqrt{\sum_{j=1}^{n} q_j^2}}\,, \tag{4}$$

or the normalized:

$$r(a) = \frac{p}{\sqrt{\sum_{j=1}^{n} (q_j / k_j)^2}}\,. \tag{5}$$

A previous result on combinatorial auctions [6] suggests the consideration of:

$$r(a) = \frac{p}{\sqrt{\sum_{j=1}^{n} q_j}}\,, \tag{6}$$

or of:

$$r(a) = \frac{p}{\sqrt{\sum_{j=1}^{n} q_j / k_j}}\,. \tag{7}$$

Many other criteria could be considered, of the form:

$$r(a) = \frac{p}{(\sum_{j=1}^{n} q_j^l)^m}$$

or

$$r(a) = \frac{p}{(\sum_{j=1}^{n} (q_j / k_j)^l)^m}$$

It is extremely difficult to guess what is the best criterion to use in a branch-and-bound algorithm. Experimental results would be interesting and we are in the process of obtaining such results but the lack of real-life data casts a doubt on the applicability of the conclusions that can be drawn from synthesized data.

Based on each of those criteria, one can devise a (different) greedy algorithm to find a solution to MUCAs: pick the most promising bid in the partial solution, then find the most promising remaining bid that does not conflict with the partial solution, and so on until no bid can be added. We are looking for the criterion that gives the *best* greedy algorithm. In what sense? We choose to compare greedy algorithms by the approximation ratio they provide in the worst-case. A greedy algorithm that provides, on any MUCA, a solution that is at least one-tenth of the optimal solution will be preferred to any algorithm that provides only one-hundredth of the optimal solution on some MUCA. Notice that we could assume a probability distribution on the inputs and compare the expected values of the solutions found (or their ratio to the optimal one), but this is not what we propose to do. Our choice is consistent with that of the theoretical CS community. The next section is devoted to theoretical considerations leading to the characterization of the best criterion.

5. THEORETICAL CONSIDERATIONS: THE WEIGHTED MULTI-SET PACKING (WMSP) PROBLEM

The problem of finding an optimal allocation in a multi-unit combinatorial auction is best described as a generalization of the weighted set packing problem [2, 13]: the weighted multi-set packing problem. It seems the same problem has been studied in [9].

We consider n commodities and assume k_j units of each of n commodities: $j = 1, \ldots, n$ are available. The k_j's are positive natural numbers. We shall denote by k the quantity $\sum_{j=1}^{n} k_j$. On the whole k items are to (may) be allocated. A bid is written: $\langle q_1, q_2, \ldots, q_n, p\rangle$, where the q's are natural numbers with $q_j \leq k_j$ and p is a natural number (or a rational nonnegative number). A bid is an offer to acquire q_j units of each commodity j for a total sum of p. Given a set of bids we want to find a subset that maximizes the sum of the p's such that for every j, $1 \leq j \leq n$, the sum of the q_j's is less than or equal to k_j. A number of special cases are worth noticing:

- if all k_j's are equal to one, the problem is the weighted set packing (WSP) problem,
- if there is only one commodity ($n = 1$), the problem is the knapsack problem.

Our first result, a trivial generalization from [11, 6], shows that the WMSP problem is not only NP-hard but also hard to approximate.

THEOREM 1. *Unless NP=ZPP*[1], *the WMSP problem cannot be approximated within $n^{1/2-\epsilon}$ in polynomial time, for any $\epsilon > 0$.*

PROOF. A graph with v vertices and e edges defines a WMSP problem in the following way. Consider e commodities ($n = e$) and assume k_j units of commodity j are for sale. The k_j's are arbitrary. There are v bids. Bid i offers a price of 1 for k_j units of each of the commodities (i.e. edges) j that vertex i is adjacent to. Any feasible allocation defines a set of independent vertices and its value is the size of the independent set. The solution of the WMSP problem is therefore equivalent to finding a maximal independent set. An $f(n)$-approximation of the WMSP problem provides an $f(e)$-approximation of the Maximal Independent Set problem. By [3], no efficient $k^{1-\epsilon}$-approximation exists for the maximal independent set problem, therefore no $e^{1/2-\epsilon}$-approximation, unless NP=ZPP. □

In [6] it is shown that the WSP problem admits a polynomial-time $n^{1/2}$-approximation and that there is a greedy algorithm that achieves this (optimal) approximation ratio. Notice that finding a polynomial-time k^{-1}-approximation of the Maximal Independent Set is trivial, but finding such an $n^{1/2}$-approximation for WSP is not. In the more general case of WMSP problems, we shall characterize the approximation quality of greedy algorithms as $\sqrt{k} = \sqrt{\sum_{j=1}^{n} k_j}$ in Theorems 2 and 3, but we do not know how to match the lower bound of Theorem 1. For reasons explained above, we are interested in solving (approximately) WMSP problems by a greedy algorithm. For such algorithms, we may prove a result that is stronger than Theorem 1.

THEOREM 2. *No (polynomial-time) greedy algorithm can guarantee an approximation of better than $\sqrt{k}$.*

PROOF. Define a *unit* bid as a bid offering a sum of 1 for a single unit of a single commodity: $\langle 0, 0, \ldots, 1, \ldots, 0, 1\rangle$. There are n different unit bids, but the same unit bid may appear a number of times in the list of bids. Given a greedy algorithm, let u be the unit bid that is ranked first among all unit bids by the algorithm.

Two problems, i.e., sets of bids, will be described. A greedy algorithm will perform badly, i.e. propose a solution that is only a $\sqrt{k}$-approximation, on one of those two problems. In the first problem, there are only two bids:

1. a bid for all the units available: $A = \langle k_1, k_2, \ldots, k_n, \sqrt{k}\rangle$, and
2. a bid u.

In the second situation, there are $k + 1$ bids:

1. a bid A for all the units available as above, and
2. for every j, $j = 1, \ldots, n$, k_j unit bids for commodity j: $\langle 0, 0, \ldots, 1, \ldots, 0, 1\rangle$.

[1]A language L belongs to ZPP if and only if there is some constant c such that there is a probabilistic Turing machine M that on input x runs in expected time $O(|\, x \,|^c)$ and outputs 1 if and only if $x \in L$

If bid A is ranked before u, then, in the second situation, the greedy method obtains $\sqrt{k}$ instead of the optimal k, a $\sqrt{k}$-approximation. If bid u is ranked before bid A, then, in the first situation the greedy method obtains 1 instead of the optimal $\sqrt{k}$, again a $\sqrt{k}$-approximation. For any greedy method, one of the two situations will give a solution that is a factor of $\sqrt{k}$ less than optimal. □

The following positive result provides an upper-bound that matches the lower bound of Theorem 2.

THEOREM 3. *A greedy algorithm (to be described) provides a polynomial-time $\sqrt{k}$-approximation for the WMSP problem.*

PROOF. We shall use the index i to range over the bids. The bid i is: $i = \langle q(i,1), q(i,2), \ldots, q(i,n), p(i)\rangle$. We define $w(i) = \sqrt{\sum_{j=1}^{n} q(i,j)}$. The algorithm we propose is greedy allocation based on the following criterion r to rank the bids, in descending order:

$$r(i) = \frac{p(i)}{w(i)} \tag{8}$$

Let OP be the optimal solution, i.e., the set of bids contained in the optimal solution. The value of the optimal solution is $\alpha = \sum_{i \in OP} p(i)$. Let GR be the solution obtained by the greedy allocation and β its value: $\beta = \sum_{i \in GR} p(i)$. We want to show that:

$$\alpha \leq \beta \sqrt{k}. \tag{9}$$

Notice, first, that we may, without loss of generality, assume that the sets OP and GR have no bid in common. Indeed, if they have, one considers the problem in which the common bids and all the units they request have been removed. The greedy and optimal solutions of the new problem are similar to the old ones and the inequality for the new smaller problem implies the same for the original problem.

Let us consider β. By elementary algebraic considerations:

$$\beta = \sum_{i \in GR} p(i) \geq \sqrt{\sum_{i \in GR} p(i)^2} = \sqrt{\sum_{i \in GR} r(i)^2\, w(i)^2} =$$

$$\sqrt{\sum_{i \in GR} r(i)^2 \sum_{j=1}^{n} q(i,j)} = \sqrt{\sum_{j=1}^{n} \sum_{i \in GR} r(i)^2\, q(i,j)}.$$

Consider α. By the Cauchy-Schwarz inequality:

$$\alpha = \sum_{i \in OP} r(i)\, w(i) \leq \sqrt{\sum_{i \in OP} r(i)^2}\, \sqrt{\sum_{i \in OP} w(i)^2}.$$

But:

$$\sum_{i \in OP} w(i)^2 = \sum_{i \in OP} \sum_{j=1}^{n} q(i,j) = \sum_{j=1}^{n} \sum_{i \in OP} q(i,j).$$

The expression $\sum_{i \in OP} q(i,j)$ represents the total number of units of commodity j allocated in the optimal allocation OP and is therefore bounded from above by k_j, the number of units available. We conclude that:

$$\alpha \leq \sqrt{\sum_{i \in OP} r(i)^2}\, \sqrt{k}.$$

To prove (9), it will be enough, then, to prove that:

$$\sum_{i \in OP} r(i)^2 \leq \sum_{j=1}^{n} \sum_{i \in GR} r(i)^2 \, q(i,j).$$

Consider the optimal solution OP. By assumption, the bids of OP did not enter the greedy solution GR. This means that, at the time such a bid i is considered during the execution of the greedy algorithm, it cannot be entered in the partial allocation already built. This implies that there is a commodity j, not enough units of which are still unallocated to satisfy the quantity $q(i,j)$ requested by bid i. In other terms the sum of $q(i,j)$ and the $q(*,j)$ of all the bids of the greedy solution already considered was larger than k_j:

$$q(i,j) + \sum_{l \in GR, r(l) \geq r(i)} q(l,j) \; > \; k_j.$$

In particular it follows that $q(i,j) > 0$ and therefore $q(i,j) \geq 1$. We attach to every bid i of OP such a commodity $c(i)$. If more than one such suitable commodity exists, we choose one arbitrarily. Let OP_j be the subset of OP that contains those bids i for which $c(i) = j$. The OP_j provide a partition of OP and:

$$\sum_{i \in OP} r(i)^2 = \sum_{j=1}^{n} \sum_{i \in OP_j} r(i)^2.$$

Let us denote by m_j the size of the set OP_j. We shall conclude the proof by showing that, for every j, $1 \leq j \leq n$:

$$\sum_{i \in OP_j} r(i)^2 \leq \sum_{i \in GR} r(i)^2 \, q(i,j).$$

Let l_j be a bid of OP_j whose r is maximal (among bids of OP_j) and let $r_j = r(l_j)$. We know that:

$$\sum_{i \in OP_j} r(i)^2 \leq {r_j}^2 \, m_j.$$

We want to bound m_j from above. We remarked above that:

$$q(l_j, j) \; + \sum_{l \in GR, r(l) \geq r_j} q(l,j) > k_j.$$

and that $q(l,j) \geq 1$ for every $l \in OPT_j$. Therefore:

$$k_j \; \geq \; \sum_{l \in OP_j} q(l,j) \; \geq \; q(l_j,j) + m_j - 1 \; >$$

$$k_j - \sum_{l \in GR, r(l) \geq r_j} q(l,j) \; + m_j - 1$$

and therefore $m_j \leq \sum_{l \in GR, r(l) \geq r_j} q(l,j)$. Clearly, then:

$${r_j}^2 \, m_j \; \leq \; {r_j}^2 \sum_{l \in GR, r(l) \geq r_j} q(l,j) \; \leq$$

$$\sum_{l \in GR, r(l) \geq r_j} r(l)^2 \, q(l,j) \; \leq \; \sum_{l \in GR} r(l)^2 \, q(l,j)$$

□

Theorems 2 and 3 show that the criterion of Equation 6 is optimal, in a worst-case sense.

COROLLARY 1. *The criterion proposed in Equation 6 for choosing the most promising bid guarantees, in the worst-case, the best possible approximation ratio.*

None of the other criteria examined in Section 4 achieves the same approximation ratio. Let us show this explicitly for the normalized criterion of Equation 7, and characterize exactly the approximation ratio achieved by this criterion.

Along the lines of the proof of Theorem 3, one can show that using the normalized ranking criterion:

$$r(i) = p(i) / \sqrt{\sum_{j=1}^{n} q(i,j) \; / \; k_j}$$

achieves a $\sqrt{Mn}$-approximation where M is the maximum of the k_j's. The following example shows that this upper bound is exact. Therefore the normalized criterion is not optimal.

Consider two commodities, and assume there are k units of the first one ($k_1 = k$) and one unit of the second one ($k_2 = 1$). There are two bids. The first one is $\langle k, 1, \sqrt{2} \rangle$ and the second one is $\langle 1, 0, 1/\sqrt{k} \rangle$. Both bids have the same normalized ranking criterion (1). Assume the greedy method places the second bid first. It obtains $1/\sqrt{k}$ instead of the optimal $\sqrt{2}$: an $\sqrt{2\,k}$-approximation, not as good as the $\sqrt{k+1}$-approximation provided by the unnormalized criterion. In Section 2, we indicated that we were willing to consider dynamic, and not only static, ordering criteria for the bids. Our conclusion is that the optimal criterion is a static one, quite a surprising conclusion.

6. PRACTICAL CONSIDERATIONS FOR CHOOSING THE MOST PROMISING BID

As noted in Section 3 the choice of a method for bounding the possible value of an auction from above is quite unproblematic in practice: one may use a wealth of methods and take the smallest upper-bound obtained. In practice, one will, on the basis of the record of past runs, easily find out which of the methods are useless and stop using them. The choice of the most promising bid studied in Section 4 and 5 is much more of a problem. The criterion described in (6) is shown to be theoretically optimal in Corollary 1, but is it the best in practice? It is the criterion that guarantees the best approximation ratio in the worst case, does it provide the best solution in practice? The truth of the matter is that only experience with real MUCAs can tell and, at this moment, no such data exists. We can only point out at two considerations. First, the examples of Section 5 present quite tellingly why the criterion defined in (6) strikes a balance between a criterion favoring large bids and a global view such as the one defined in (1) or one that favors small bids and fine tuning such as defined in (2) or in (4). The proof of Theorem 3 also shows why the unnormalized criterion of (6) should be preferred to the normalized criterion of (7). Secondly, we have confidence in the applicability of theoretical results: techniques that can be proved to be optimal in theory tend to work well.

In [7], the authors use a complex method for choosing the most promising bid: their choice depends on the other bids and also on the results of the bounding-from-above procedure. Since the ordering of the commodities does not depend on the price offered by the bids, but may determine the solution that will be considered first, it seems that this first candidate solution may be arbitrarily worse than the optimal solution. The basic criterion used inside bins is similar to that of (2) and therefore not optimal in theory.

7. EXPERIMENTAL RESULTS

Experimental results are crucial in the assessment of the ideas developed above. Three basic questions should be answered.

1. Does the criterion for choosing the most promising bid that has been shown to be optimal in Corollary 1 perform well in practice, i.e., does a search algorithm that uses this criterion find rapidly the optimal solution?

2. Do the different methods for bounding from above the value of the optimal solution lead to the pruning of many large subtrees? Which of those methods are most useful?

3. Are those bounding methods fast enough to be usable in practice or does their use imply that the algorithm spends an unbearable amount of time in computing upper bounds? A trade-off between the amount of effort spent in pruning and in examining new nodes must be struck.

The best would obviously be to examine those questions on data obtained from real-life combinatorial auctions. Such data is very difficult to find and we could not put our hands on such data. The next best thing that can be done is to use auctions artificially generated. In [7], the authors define a probability distribution over combinatorial auctions and test the average behavior of their algorithm on this distribution. We chose to use the same distribution, with the same values for the parameters.

Let us address the second question first. The answer is emphatically positive. The methods proposed above provide upper bounds that allow for an extremely thorough pruning of the search tree. Linear Programming provides a very tight upper bound that leads to a very short search path over the tree: only a very small fraction of the nodes have to be expanded. For example, considering 40 bids, we found that only 1.6×10^{-9} of the nodes were visited.

The projections bounds are not as good, and provide, for the distribution we used, bounds that are not better than the bound provided by average price considerations. This is probably due to the specific distribution chosen, in which it is rarely the case that a single good is *dominant*, such as, e.g., in the case all bids request all the units of a certain specific good. One of the characteristics of the distribution suggested in [7] is that bids tend to request only a small number of units per good. This kind of distribution leads to an auction with no dominant good for which the projection bounds are quite loose. R. Holte [4] found that, on another distribution, the projection bound is very good. Nevertheless, projection bounds allow us to visit only 1.6×10^{-8} of the nodes.

The average price upper bound is not as good as the Linear Programming one, but is, on average, as good as the projection bound. It allows for a very thorough pruning of the search tree. Considering 40 bids, we found that only 1.6×10^{-8} of the nodes had to be visited. This number is the same as the one for the projection bound and ten times larger than the one for Linear Programming.

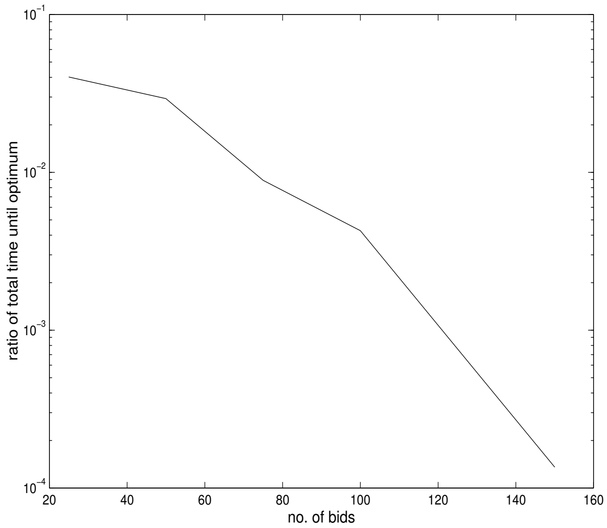

Figure 1: This graph shows the part of the execution time spent until the optimal solution is found as a function of the number of bids for auctions of fourteen goods. Auctions are distributed as proposed by Leyton-Brown, Shoham and Tennenholtz. Times are averaged over ten trials.

Let us now address the third question. We found Linear Programming to be extremely costly (in time) to use, so costly as to render its use infeasible for large combinatorial auctions. Since we have only just begun experimenting, we hope that we shall learn in the future how to use Linear Programming effectively. The average price upper bound is, on the contrary, computed very fast. The projection bounds are not computed as fast. Since we explained above that the projection bounds were, on our distribution, not better, *the results to be presented below have all been obtained by using the average price upper bound exclusively.*

Let us discuss the first question now. The best test of the quality of our ordering heuristic is probably to examine how fast the optimal solution is obtained. We found that, using the ordering criterion described in Equation 6, the optimal solution is found extremely rapidly and the algorithm spends an overwhelming part of its time in showing that this solution is indeed optimal. In Figure 1, we plot the time spent until the optimal solution is found divided by the total time spent in the search for auctions of different sizes. All auctions include 14 goods, the number of bids is found on the x axis. For small auctions this number is of the order of 4%; for larger auctions this number decreases about linearly on a logarithmic scale and for auctions of 150 bids, the optimal solution is found after only 0.014% of the total execution time. It seems that this percentage decreases at least exponentially. Those results seem to improve significantly on those of [7]. The *Percentage Optimality* graph there seems to indicate than more than 10% of the time is spent before the optimal solution is found. Those results indicate that the criterion of Equation 6 that we proved theoretically optimal is extremely good in practice.

Figure 2 describes the execution times of our branch and bound algorithm using the average price upper bound and the ordering heuristic of Equation 6: on the x-axis the num-

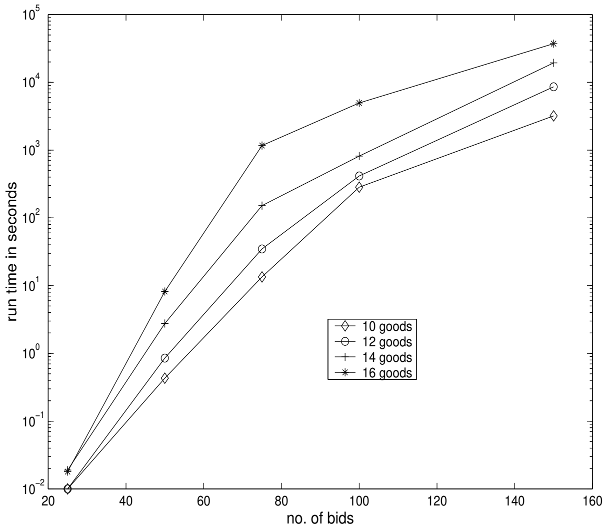

Figure 2: This graph shows the execution time for auctions containing different numbers of bids. Auctions are distributed as proposed by Leyton-Brown, Shoham and Tennenholtz. Times are averaged over ten trials. Each of the four curves corresponds to a different number of goods.

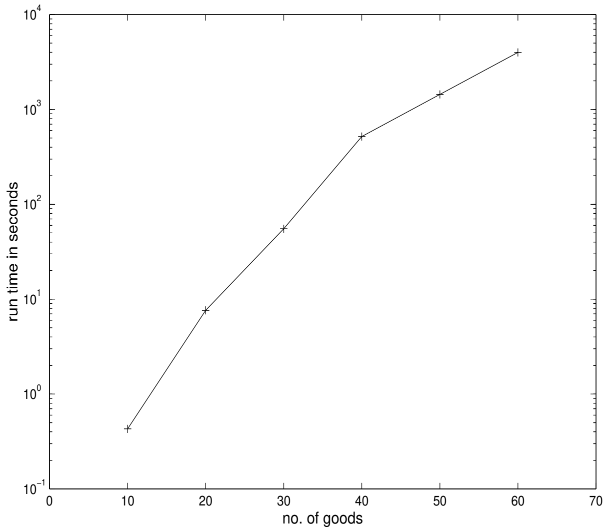

Figure 3: This graph shows the execution time for 50 bids auctions with different numbers of goods. Auctions are distributed as proposed by Leyton-Brown, Shoham and Tennenholtz. Times are averaged over ten trials.

ber of bids, on the y-axis the execution time. Four different curves are plotted, corresponding to a different number of goods. Our experimental data was collected on a Pentium III-450 running Linux, using 640K of memory. Figure 2 is comparable to the *Number of Bids vs. Time* figure in [7]. Our running times are larger, by orders of magnitude, than theirs. For this reason, we were not even capable of solving the smallest auctions they considered. Figure 2 shows very clearly a sub-linear (on a logarithmic scale) growth in execution time, indicating that the growth is sub-exponential. This feature is also found, but less clearly, in [7]'s graph. We intend to look into the reasons for the huge gap in running times between our algorithm and CAMUS. The huge discrepancy between the amount of memory they use (25 MB) and ours (640K) is certainly part of the explanation.

Figure 3 examines the dependence of the running time on the number of goods. The number of bids is fixed at 50, and the number of goods is described on the x axis. CAMUS [7] exhibits an exponential sensitivity to the number of goods (as opposed to the number of bids). Figure 3 clearly shows a sub-exponential growth.

8. CONCLUSIONS AND FURTHER WORK

We proposed a simple branch-and-bound framework to solve MUCAs. We succeeded in characterizing the theoretically optimal method for sorting bids. Two main directions for research are left open:

- characterize the power of different methods for bounding from above the value of MUCAs

- run more extensive experiments to compare the performance of different heuristics both for bounding and for ordering and to compare our proposal to others such as in [7].

- consider more efficient ways of using Linear Programming, using the results of previous computations to speed up the search for a solution.

9. ACKNOWLEDGEMENTS

Conversations with Michel Bercovier and Michael Ben-Or and remarks from Amir Ronen are gratefully acknowledged. The first author is partially supported by Grant 15561-1-99 from the Israel Ministry of Science, Culture and Sport and by the Jean and Helene Alfassa fund for research in Artificial Intelligence.

10. REFERENCES

[1] Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceedings of IJCAI'99*, Stockholm, Sweden, July 1999. Morgan Kaufmann.

[2] M. M. Halldórsson. Approximation of weighted independent set and hereditary subset problems. In *Proc. of COCOON'99*, number 1627 in Lecture Notes in Computer Science. Springer Verlag, 1999.

[3] J. Håstad. Clique is hard to approximate within $n^{1-\epsilon}$. *Acta Mathematica*, 182:105–142, 1999.

[4] R. Holte, August 2000. private communication.

[5] E. Lawler and W. D.E. Branch-and-bound methods: A survey. *Operations Research*, 14(4):699–719, 1966.

[6] D. Lehmann, L. I. O'Callaghan, and Y. Shoham. Truth revelation in rapid, approximately efficient combinatorial auctions. In *Proceedings of the First ACM Conference on Electronic Commerce. EC'99*, pages 96–102, Denver, Colorado, November 1999. SIGecom, ACM Press.

[7] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. Unpublished draft, January 2000.

[8] N. Nisan. Bidding and allocation in combinatorial auctions. Presented at Northwestern's Summer Workshop in Microeconomics, July 1999.

[9] A.H.G. Rinnooy Kan, L. Stougie, and C. Vercellis. A class of generalized greedy algorithms for the multi-knapsack problem. *Discrete Applied Mathematics*, 42:279–290, 1993.

[10] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. Technical Report 95-09, DIMACS, April 1995.

[11] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *IJCAI-99*, pages 542–547, Stockholm, Sweden, July 1999.

[12] M. Tennenholtz. Some tractable combinatorial auctions. Unpublished draft: January 2000.

[13] R. R. Vemuganti. Applications of set covering, set packing and set partitioning models: A survey. In D.-Z. Du and P. M. P., editors, *Handbook of Combinatorial Optimization*, volume 1, pages 573–746. Kluwer Academic Publishers, 1998.

[14] W. S. Vickrey. Counterspeculation, auctions and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.