Integer Programming for Automated Auctions

Arne Andersson, Mattias Tenhunen, and Fredrik Ygge

Computing Science Department
Information Technology
Uppsala University
Box 311
SE - 751 05 Uppsala, Sweden
{arnea,tein,ygge}@csd.uu.se
http://www.csd.uu.se/~{arnea,tein,ygge}

Abstract. Auctions allowing bids for combinations of items are important for (agent mediated) electronic commerce; compared to other auction mechanisms, they often increase the efficiency of the auction, while keeping risks for bidders low. The determination of an optimal winner combination in this type of auctions is a complex computational problem, which has recently attracted some research, and in this paper, we look further into the topic.

It is well known that the winner determination problem for a certain class of auctions is equivalent to what in the operations research community is referred to as *(weighted) set packing*. In this paper we compare some of the recent winner determination algorithms to traditional set packing algorithms, and study how more general auctions can be modeled by use of standard integer programming methods.

1 Introduction

There is currently a rapid growth of automated auctions on the Internet.[1] This is of core interest for the multi-agent systems community; agents may assist and represent real-world parties in automated trade. Such agent mediated trade has its biggest potential when (i) the commodities traded for are of relative low value (at each transaction), (ii) the commodities traded for are relatively complex, and (iii)—often as a consequence of (ii)—the negotiation is a time consuming process. Typical areas that fulfill these requirements are energy related markets [Ygge, 1999; Ygge and Akkermans, 1996; 1999], transport services [Sandholm, 1996], and bandwidth/network information services [Mullen and Wellman, 1995; Yamaki *et al.*, 1996]. However, for these markets to have a general commercial impact, economically and computationally efficient market mechanisms are required. In this paper we will investigate one class of such important mechanisms.

[1] For prototypical examples, see e.g. eBay (http://www.ebay.com/) and bidlet (http://bidlet.com/).

The vast majority of the current auctions are one-dimensional auctions for discrete commodities. That is, bids are placed on one commodity at a time, and bids may only request discrete amounts of the commodities.[2] In a one-dimensional auction the agents places bids at only one commodity at the time. However, agents typically have dependencies between the commodities in a trade. If commodities are not independent, an agent participating in a one-dimensional auction needs to predict the closing prices of the *other* commodities when placing a bid. Clearly, this puts a heavy computational burden as well as a significant risk on the agents.

In order to reduce these unfavorable properties, combinatorial auctions have been proposed to allow agents to place bids on combinations (bundles) of commodities, see e.g. [Rassenti *et al.*, 1982; Rothkopf *et al.*, 1995; Parkes, 1999; Wurman, 1999]. Common for all these auctions is that they require the solution of the winner determination problem, i.e. find the combination of bids that maximizes the surplus, which (for most practically relevant settings) is an $\mathcal{NP}$-complete problem [Rothkopf *et al.*, 1995]. This hard computational problem has been in focus in some recent work, particularly by Sandholm [Sandholm, 1999a] and Fujishima et al. [Fujishima *et al.*, 1999], and is also the focus of this paper. In particular, our contributions are:

- The currently best algorithms are compared to well known algorithms for the computationally identical problem of set packing, and hereby put in to a proper computer science perspective. From this exercise, we learn that many of the main features of the currently best algorithms are rediscoveries of methods invented in the late 1960's.
- We observe that the winner determination problem can be formulized as a standard integer programming problem, enabling the management of more general problems in a very efficient manner by use of standard algorithms and commercially available software. This allows for efficient treatment of highly relevant combinatorial auctions that are not supported by current algorithms.
- The significance of the probability distributions of the test bid sets used for evaluating different algorithms is discussed and exemplified. Particularly we show that some of the distributions, used for benchmarking of current algorithms, can be managed with rather trivial—yet very efficient—algorithms.

The paper is organized as follows. In Section 2 we present a well-known set partitioning algorithm, and discuss the current algorithms for optimal winner determination in the context of this algorithm. Thereafter, in Section 3, we observe that the winner determination problem can be set up as an integer programming problem and hereby be solved by standard algorithms and commercial software. In Section 4 some empirical benchmarking for standard integer programming

[2] In this paper we discuss only auctions with discrete commodities. Discussions on economical and computational aspects of auctions with continuous commodities are found elsewhere, e.g. [Cheng and Wellman, 1998; Sandholm and Ygge, 1999].

software is presented. We then discuss the significance of the probability distribution of the test sets used for benchmarking an algorithm, Section 5. Finally, Section 6 concludes.

2 Recent winner determination algorithms and traditional algorithms for corresponding problems

Before we discuss how very general versions of winner determination can be solved by general-purpose algorithms, it is interesting to discuss special purpose algorithms. We particularly study one algorithm introduced already in 1969 by Garfinkel and Nemhauser [Garfinkel and Nemhauser, 1969]. The method was originally described for solving the *(weighted) set partitioning problem*. As pointed out by the originators, "the approach is so simple that it appears to be obvious. However, it seems worth reporting because it has performed so well". Even further, some of the experimental results by Garfinkel and Nemhauser seems surprisingly good compared to recent experiments when taking the hardware performance at that time into account.

A disadvantage of the Garfinkel-Nemhauser algorithm (in the context of combinatorial auctions), as well as the work by Sandholm and Fujishima et al. [Sandholm, 1999a; Fujishima *et al.*, 1999], is that it is limited to the simple case were there is only one unit of each commodity, there are only zero reserve prices, etc. These limitations are further discussed in the next section where also more general problems are treated.

The winner determination problem of this section is trivially computationally equivalent to set packing [Sandholm, 1999a]: Given a collection of sets (bids), where each set has an associated (positive) value, find the collection of disjoint sets with the highest value. A related problem is *(weighted) set partitioning*, where each set (bid) has a (positive) cost, and the objective is to find the collection of disjoint sets that contain all commodities and which has the smallest total cost. The two problems can be transformed into each other, and therefore the set partitioning algorithm by Garfinkel and Nemhauser [Garfinkel and Nemhauser, 1969] is applicable to winner determination in combinatorial auctions.

In general, given an algorithm for set partitioning, we can use it for set packing by modifying the input in the following way: (i) In set partitioning, the objective—in its most common formulation, e.g. [Salkin, 1975]—is to minimize a cost instead of maximizing a surplus. To overcome this difference, we can compute a fictitious cost for each bid. A straight-forward way of doing this is to compute the cost of a bid as the largest valuation for any bid minus the valuation of the bid in question. (ii) In set partitioning, each commodity must be included in a solution, this is not the case in set packing. To satisfy this criterion, we can add a dummy bid for each commodity with the valuation set to zero, i.e. with the above transformation the cost is set to the highest valuation of any bid.

Based on these two modifications, we present the Garfinkel-Nemhauser algorithm as a winner determination algorithm for combinatorial auctions. The algorithm creates a list per commodity (row) and each bid (column) is stored in exactly one list. Given an ordering of the commodities, each bid is stored in the list corresponding to its first occurring commodity. Within each list, the bids are sorted according to increasing cost. The search for the optimal solution is done in the following way:

1. Choose the first bid from the first list containing a bid as the current solution.
2. Add (to the current solution) the first disjoint bid from the first list—corresponding to a commodity not included in the current solution—containing such a disjoint bid, if any.
3. Repeat Step 2 until one of the following happens: (i) *The cost for the current solution exceeds the best solution:* this branch of searching can be pruned. (ii) *Nothing more can be added:* check if this is a valid solution/the best solution so far.
4. Backtrack: Replace the latest chosen bid by the next bid of its list and go to Step 2. When no more bids can be selected from the list, back up further recursively. If no more backtracking can be done, terminate.

Note that it is trivial to modify the algorithm to suit set packing even without dummy bids. In the original Garfinkel-Nemhauser algorithm, there is an inconsistency test, rejecting combinations that do not contain all commodities (reflecting the fact that for a given in-data—without dummy bids—there need not exist a partitioning solution). By simply ignoring this test, we can avoid the dummy bids. There is also a small difference between set partitioning and set packing in (ii) of Step 3. If nothing more can be added with set partitioning, the solution either is invalid (does not cover all rows) or is optimal (otherwise it would have been pruned in (i)). With set packing, the current solution needs to be compared to the best solution so far.

As seen from the above description, the currently best performing winner determination algorithm[3], the CASS algorithm [Fujishima *et al.*, 1999], is apparently in major parts a rediscovery of the Garfinkel-Nemhauser algorithm. The main principles of both algorithms are to (i) put the bids in lists corresponding to the different commodities (called *bins* by Fujishima et al. [Fujishima *et al.*, 1999]), (ii) sort the bids in the list in some cost (valuation) related order, (iii) do pruning whenever the current combination cannot be better than the best one found so far, and (iv) do standard backtracking. There are essentially two significant differences between CASS and the Garfinkel-Nemhauser algorithm: (i) cashing of partial search results, and (ii) improved pruning.

The caching is done by storing partial search results in random access memory. As many partial allocations share the same “rest term” (i.e. remaining unassigned commodities), this often pays off [Fujishima *et al.*, 1999]. The cache can hereby also be efficiently used for pruning; whenever the “rest term” is a subset of an already cached “rest term” and the surplus of the current allocation plus the

[3] At the presentation of CASS at IJCAI 1999 in Stockholm, the CASS algorithm [Fujishima *et al.*, 1999] was reported to outperform Sandholm’s winner determination algorithm [Sandholm, 1999a] by approximately two orders of magnitude for the distributions tested.

surplus of the cached allocation is smaller than the surplus of best combination found so far, the search can safely be pruned.

Compared to the simple pruning described in the Garfinkel-Nemhauser algorithm, Sandholm's algorithm [Sandholm, 1999a] and the CASS algorithm [Fujishima *et al.*, 1999], use a more sophisticated technique which essentially is a rediscovery/specialization of the ceiling test [Salkin, 1975].

3 Combinatorial auction winner determination as a standard integer programming problem

In this section we describe in some detail how very general optimal winner determination problems can be formulated as a standard integer problem. For reading on integer problem in itself and its relation to set packing etc., we refer to the literature on combinatorial optimization and operations research, e.g. [Balas, 1965; Balas and Padberg, 1976; Geoffrion, 1969; Michaud, 1972; Garfinkel and Nemhauser, 1969; Salkin, 1975].

By properly formulating the problem we get the following advantages compared to current approaches to optimal winner determination [Sandholm, 1999a; Fujishima *et al.*, 1999][4]:

- The formulation can utilize standard algorithms and hence be run directly on standard commercially available, thoroughly debugged and optimized software, such as Cplex[5].
- There may be multiple units traded of each commodity.
- Bidders can bid on more than one unit of a commodity, and place duplicate bids enabling them to express (approximate) linear segments of their preference space as a single bid.
- Bidders can construct XOR bids.
- Sellers may have non-zero reserve prices.
- There is no distinction between buyers and sellers; a bidder can place a bid for buying some commodities and simultaneously selling some other commodities.

It should be pointed out that some of the generalizations achieved here could be expressed to fit with set packing (and hence also current winner determination algorithms). XOR bids are easily formulated by adding dummy commodities (e.g. "a XOR $b \Leftrightarrow ac$ OR bc"). Also, multiple, m, units of each commodity could be simulated by introducing m new commodities and represent each bid bidding on the commodity as an XOR bid for the m new commodities. However, this would lead to an undesired, and sometimes intractable, increase in problem size. For example, think of a situation where an agent bids for q commodities with 100

[4] The eMediator [Sandholm, 1999b] supports some of these features, and there has been some outline about how Sandholm's algorithm can be extended to more general settings [Sandholm, 1999a], but no detailed algorithm has yet been published.

[5] See `www.cplex.com`

units available for sale of each commodity. Then the agent would need to submit (or the auctioneer has to generate) a bid consisting of 100^q XOR clauses.

We now give some definitions required for formulating the winner determination problem as an integer problem.

Definition 1. *An* atomic bid*, a_i, is a tuple defined as*

$$a_i ::= \langle \mathbf{q}_i, v_i \rangle,$$

where $\mathbf{q} = [q_{i1}, q_{i2}, \ldots, q_{ik}]$ (q_{ic} is an integer, and k is the number of commodities on the market), and $v_i \in \Re$.

The interpretation of the definition of the atomic bid is perhaps most easily explained through an example. The atomic bid $\langle [1, 0, 2, -3], 15.7 \rangle$, denotes a desire to buy 1 unit of commodity 1, buy 2 units of commodity 3, and sell 3 units of commodity 4 for a payment of at most 15.7 of some monetary unit. Note that it may be that $v_i < 0$, denoting that the bid requires a monetary compensation (e.g. for selling something with a positive price).

Definition 2. *A* duplicate bid*, d_i, is a tuple defined as*

$$d_i ::= \langle a_i, I_i^{min} .. I_i^{max} \rangle,$$

where a_i is an atomic bid and I_i^{min} and I_i^{max} are integers, $I_i^{min} \leq I_i^{max}$.

Also here an example probably most easily explains the interpretation: $\langle\langle [1, 0, 2, -3], 15.7 \rangle, 1..3 \rangle$, means that the atomic bid $\langle [1, 0, 2, -3], 15.7 \rangle$ (with interpretation as above) may be duplicated 1 to 3 times.

Definition 3. *An* XOR bid*, x_i, is defined as*

$$x_i ::= a_j \;||\; x_k \; XOR \; x_l,$$

where a_j is an atomic bid, and x_k and x_l are XOR bids.

Examples of XOR bids are
$\langle [1, 0, 2, -3], 15 \rangle$,
$\langle [1, 0, 2, -3], 15 \rangle$ XOR $\langle [0, 0, -1, 1], -3 \rangle$, and
$\langle [1, 0, 2, -3], 15 \rangle$ XOR $\langle [0, 0, -1, 1], -3 \rangle$ XOR $\langle [1, 1, 0, 0], 1 \rangle$.

Using the above definition we can now formulate the problem of optimal winner determination as a standard integer programming problem.

Definition 4. *The* optimal winner combination *is the solution to*

$$\arg \max_{I_1, I_2, \ldots I_n} \sum_{i=1}^{n} v_i I_i, \textit{ (Maximize total surplus.)} \tag{1}$$

subject to

$$\sum_{i=1}^{n} q_{ic} I_i = 0, \;\; c = 1..k, \textit{ (Feasibility constr.)} \tag{2}$$

$$\sum_{i | a_i \in x_j} I_i \leq 1, \;\; j | x_j \in X, \textit{ (XOR logic constr.)} \tag{3}$$

$$I_i \in \{0, 1\}, \;\; i | x_i \in X, \textit{ (XOR range constr.)} \tag{4}$$

$$I_i^{min} \leq I_i \leq I_i^{max}, \;\; i | d_i \in D, \textit{ (Duplicate range constr.)} \tag{5}$$

where I_i is an integer corresponding to how many duplicates of the atomic bid a_i there is in the winning combination, k is the number of commodities, n is the number of atomic bids, $a_i \in x_j$ denotes that the atomic bid a_i is part of the XOR combination constituting bid x_j, X is the set of submitted XOR bids, and D is the set of submitted duplicate bids.

The equality constraint in Equation (2) should be changed to "less than or equal to" if free disposal can be assumed. All submitted bids must be either duplicate or XOR bids. Note that an atomic bid can trivially be represented either as a duplicate or an XOR bid.

From the above it is clear that a very wide class of combinatorial auctions can be managed with standard integer programming (and further extensions are conceivable). Thus, there are good reasons to believe that many practically relevant combinatorial auctions with discrete commodities can be expressed as integer programming problems and be managed very efficiently by available algorithms and software.

4 Some empirical benchmarking

In this section we give some empirical data in order to compare the new approach to optimal winner determination based on standard integer programming (and consequently tested with off-the-shelf software) to current highly specialized approaches in cases where these can be applied. We will also give some empirical data for cases where current approaches can not be used in order to give a proof of concept.

The hardware setup of the experiments has been: One standard uniprocessor 550MHz PC with 128Mb of RAM memory.[6] The software used is Cplex version

[6] Note that the software we have used also is available in versions for parallel execution. Hence, by using a high performance parallel platform, performance can be improved significantly if required.

6.5. The time required for loading the test data into memory has not been included in the results below. The time reported is "wall time" time, i.e. an upper bound on processor time used. In the tests of all distributions except for the *General binomial* distribution below, we used no seller bids and therefore changed "$= 0$" in Equation (2) to "≤ 1" representing that we have one unit available for each commodity and that all units need not be sold to have a feasible allocation (free disposal) as done by Sandholm [Sandholm, 1999a] and Fujishima et al. [Fujishima *et al.*, 1999].

The empirical benchmarks are performed on the same test data as was given by Sandholm [Sandholm, 1999a] and Fujishima et al. [Fujishima *et al.*, 1999] (where we were able to reproduce this data), i.e. the distributions used are:

– *Random* [Sandholm, 1999a]. For each bid pick the number of commodities requested randomly from 1 to the number of commodities in the market. Randomly choose the actual commodities requested without replacement. Draw an random valuation between 1 and 1000.
– *Weighted random* [Sandholm, 1999a]. Same as above, but multiply the valuation by the number of commodities requested.
– *Uniform* [Sandholm, 1999a]. Draw the same number of randomly chosen items for each bid. Pick the valuation from [1..1000].
– *Decay* [Sandholm, 1999a]. Make the bid request a random commodity. Then repeatedly add a new commodity with probability α until an item is not added or the bid requests all commodities in the market. Pick a valuation between 1 and 1000 and multiply by the number of commodities requested.
– *Binomial* [Fujishima *et al.*, 1999]. The probability distribution for a bid requesting n commodities of k commodities in the market is

$$f(n) = p^n (1-p)^{k-n} \binom{k}{n},$$

with $p = 0.2$. The valuation is drawn from 1 to 1000 and multiplied by n.
– *Exponential* [Fujishima *et al.*, 1999]. The probability distribution is defined as $f_e(n) = Ce^{-x/p}$. However, C is not given by Fujishima et al. [Fujishima *et al.*, 1999] and therefore no interpretation of their Figure 2 is possible, and hence no comparative test could be performed.[7]

We then also add a distribution of our own, which can not be managed by the current algorithms: the *General binomial* distribution. This distribution has the following characteristics: (i) Using a binomial distribution (c.f. the description of the binomial distribution above), we determine whether we can assume free disposal of a commodity or not. A $p = 0.9$ is used, and hence the probability of that we have free disposal of a commodity is 0.9. (ii) With probability 0.75 a bid is a duplicate bid (bids that are not duplicate bids are XOR bids). (iii) An atomic bid will request a commodity as described by a binomial distribution with $p = 0.2$. If a bid requests a commodity, the amount is drawn from a rectangular

[7] We have also e-mailed the authors on this topic, but have received no reply.

distribution in $[-5..5]$. (Atomic bids requesting no commodities are discarded.) (iv) The valuation is evenly drawn from [1..1000] and then multiplied by the total number of units requested (negative amounts for offerings). (That is, the valuation can be both positive and negative.) (v) For XOR bids, the number of atomic bids constituting the bid is drawn from a rectangular distribution in [1..9]. (vi) For duplicate bids I_i^{min} is rectangularly drawn from [1..5], and I_i^{max} is drawn from $[I_i^{min}..5]$.

Figure 1 to Figure 4 show the results of the respective tests. For each distribution, 10 instances have been tested. In the figures, the points denote the surplus normalized by the surplus of the optimal allocation for each test instance. The termination is also denoted by a point. In cases where the algorithm does not terminate with optimality guarantee as soon as it has reached the optimal solution, there will hence be two points at a relative surplus of one.

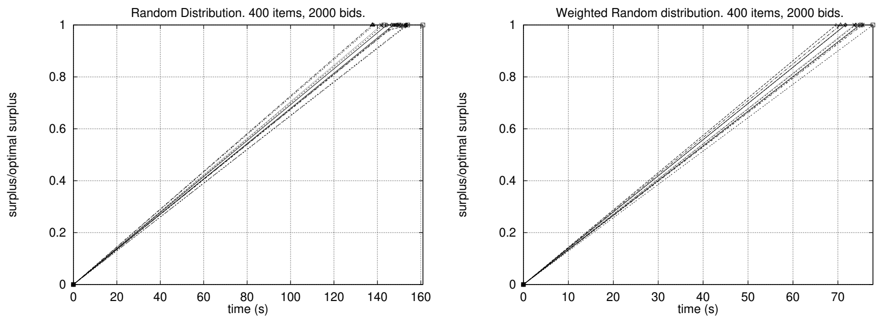

Fig. 1. Leftmost the *Random* distribution is shown for 2000 bids and 400 commodities. For this distribution the time for obtaining the optimal combination and being able to guarantee this optimality is the same for most tested instances. Worst case is in the area of 160s. Rightmost the *Weighted random* distribution is shown for 2000 bids and 400 commodities. For this distribution the time for obtaining the optimal combination and being able to guarantee this optimality is the same for *all* tested instances. Worst case is in the area of 80s.

5 Discussion on the performance for different bid distributions

It is generally recognized that it is most unfortunate that real-world test data are available. As long as no such test data is available, it is perhaps reasonable to try different types of distributions and try to identify what types distributions are “hard” and which ones are “easy” for different types of algorithms. Our experience so far is that the bid distributions chosen have an *extreme* impact on the performance of the algorithms. In fact it is safe to say that the fact that a specific algorithm does very well on a large test set often is an indication of

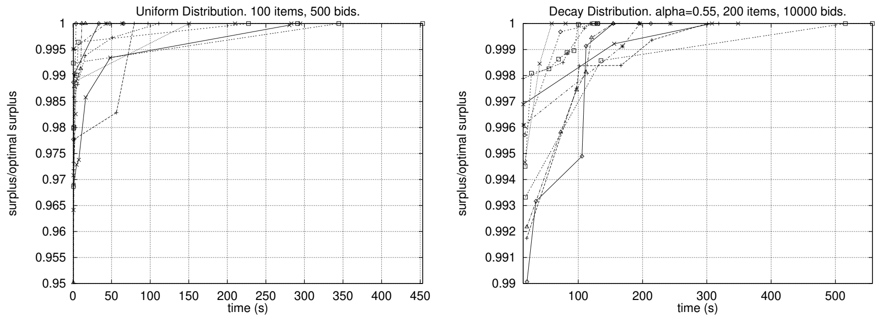

Fig. 2. Leftmost the *Uniform* distribution is shown for 500 bids and 100 commodities, where each agent bids for three commodities. The difference between the different instances vary significantly, and the execution times vary from a few seconds to almost 350s. Rightmost the *Decay* distribution is shown for 10000 bids and 200 commodities. Also here there is a significant spread in performance. Worst case is around 500s.

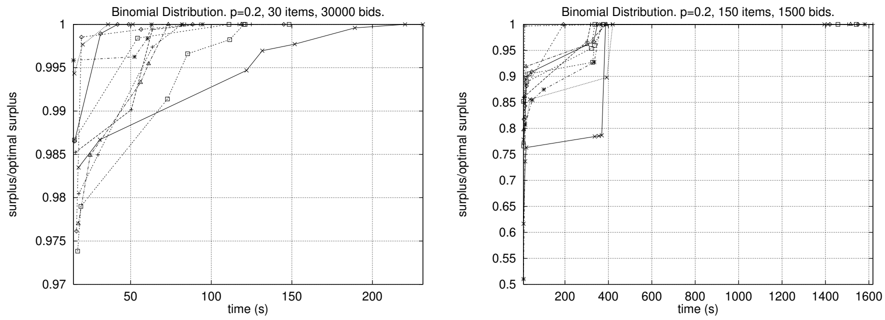

Fig. 3. Leftmost the *Binomial* distribution is shown for 30000 bids and 30 commodities. Here the relative surplus raises to the area of 97% in a few seconds. The optimal combination is normally found after between 30 and 150s. The worst case for finding a guaranteed optimal solution is around 220s. Rightmost the same distribution is shown for 1500 bids and 150 commodities. In this case the difference between the point in time where the optimal solution is found and when it can be guaranteed is significantly larger. The optimal solution is always found before 500s, but the guarantee is often not made before 1500s.

that the test set is easy rather than that the algorithm is sophisticated. It is also our experience that producing good test sets is a very delicate task. Seemingly very small differences often have a significant effect of algorithm performance. From our experiments with different families of algorithms it is furthermore clear that if the probability distribution is known to the auctioneer, it can construct algorithms that capitalize significantly on this knowledge. Our main conclusion so far is that it is very important to obtain some realistic data and investigate whether it has some special structures that can be utilized by highly specialized

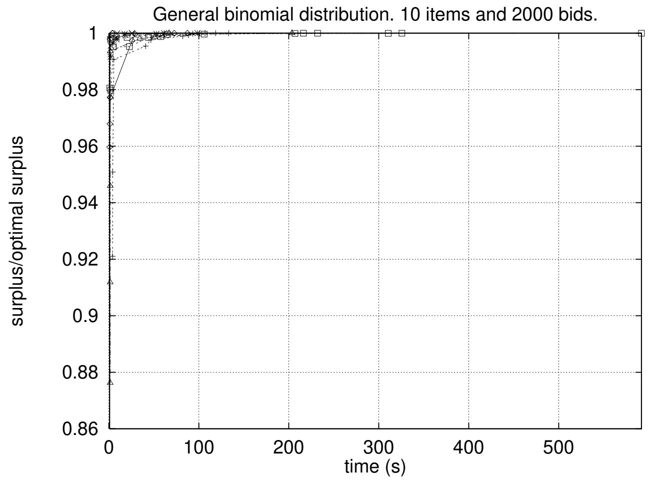

Fig. 4. The *General binomial* distribution is shown for 2000 bids and 10 commodities. We mainly include this plot as a proof of concept for that the new integer programming based method can manage very general combinatorial auctions. Without drawing too many computational conclusions, we just observe that a high quality solution is always found very rapidly, but the time to an optimal allocation varies significantly. In one of the 10 test instances, it turned out that no valid solution (apart from the trivial one of reallocating nothing) existed.

algorithms (assuming that standard algorithms fall short on practically relevant instances).

In the previous section we gave some empirical benchmarking. We now discuss some of the results from this benchmarking for the specific distributions used, and draw some general conclusions about general properties of some of them.

– *Random* [Sandholm, 1999a]. Cplex determines the optimal winner efficiently; the timings are superior to those presented by Sandholm [Sandholm, 1999a]. Furthermore, we note that this distribution is very simple in the following sense: Since the price of a bid is not weighted by the number of commodities, small bids will be dominating. A simple preprocessing, *column dominance checking* [Salkin, 1975], will decrease the problem size to a simple degenerate case. On a high level: for large test sets, the probability that any bid requesting more than one commodity is in the optimal combination is close to nil. (Hint: The expected summed valuation of two bids requesting only one commodity, is twice the expected valuation of a bid requesting two commodities, and so on.) We have implemented simple algorithms for doing column dominance checking and they have, for all instances with a number of bids larger than 3000, been able to reduce the number of bids to the number of commodities (and obtain the optimal solution without any further processing). As an example we can simply reduce 180000 bids (and 30 commodities) to 30 bids (and also find the optimal combination) in 4s with a non-optimized Java implementation of a standard column dominance checking algorithm [Salkin, 1975] on an ordinary 550MHz PC. Clearly, this suggests that a trivial approximate algorithm should be used for this distribution: for each bid, if the bid requests only one commodity and if the bid has the highest valuation for this commodity found so far, then in the current winning combination replace the highest bid so far (if any) for this commodity with the newly found bid. For all bid sets we have tried (of any significant size), this would have resulted in an optimal solution.

- *Weighted random* [Sandholm, 1999a]. Cplex performs well, and is superior to Sandholm's implementation [Sandholm, 1999a]. Also here, column dominance can be used to reduce the problem size drastically. This distribution is not as trivial as the one above from a column dominance checking point of view, but we can still reduce e.g. 60000 bids (of 30 commodities) to 800 bids in approximately half a minute with the same Java implementation as above. Still, this distribution is very efficiently managed by an algorithm as simple as the approximate algorithm for the random distribution above: The probability of a dominating large bid is very high. In all distributions tested (of any reasonable size) there is only one winning bid. Hence, the approximate algorithm of merely selecting the bid with the highest valuation finds the optimal solution for a very large number of test sets.
- *Uniform* [Sandholm, 1999a] Cplex performs well compared to Sandholm's implementation [Sandholm, 1999a]. Still this appears to be a "hard" distribution for Cplex, and we have not been able to come up with a very simple special purpose algorithm.
- *Decay* [Sandholm, 1999a] Cplex performs well compared to Sandholm's algorithm [Sandholm, 1999a], and rather large bid sets can be efficiently managed.
- *Binomial* [Fujishima *et al.*, 1999]. For 30 commodities and 3000 bids, the CASS implementation is significantly faster than Cplex. Still Cplex can manage rather big bid sets efficiently. (Note that in Figure 3, 30000 bids are used.) For 150 commodities and 1500 bids, the difference between CASS and Cplex is smaller. Furthermore, our experiments with more bids indicate that Cplex has a favorable behavior as problem size grows. However, any particular conclusion can not be made without more comparative experiments.

  It is particularly interesting to note that the binomial distribution with 150 commodities, which seems "hard" from the experiments above, is actually "easy". The reason, modulo some details, is that as the number of commodities increase, the probability that two bids are conjunct increases. Therefore, the expected number of bids in the optimal solution is small (two or three). A quickly programmed breadth-first search type of algorithm found the optimal solution in less than one second for 1500 bids and 150 commodities (which is significantly faster than CASS), although it was coded in a slower environment than CASS. (Again we used non-optimized Java code on a 550MHz standard PC.) The time for verifying optimality is below one hundred seconds. This algorithm also performs excellently as an approximate algorithm for the binomial distribution with fewer commodities. If the binomial distribution turns out to be of practical relevance, it is probably useful to study this family of algorithms further.

The three examples of the *Random* distribution, the *Weighted random* distribution, and the *Binomial* distribution with many commodities are three 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 easy to construct very simple yet very efficient algorithms for these special cases. A good way to get hints for constructing specialized algorithms is to run a few experiments with a standard integer programming software and then analyze the solutions. For example, with the *Random* distribution above it was clear from observing some solutions that the solution always consisted of 30 bids (when there were 30 commodities in the market). Similarly, for the *Weighted random* distribution and the *Binomial* distribution with 150 commodities only very few bids would be in the final solution.

More challenging is of course to construct more general and adaptive algorithms that perform excellently on all the above distributions. However, as pointed out by Sandholm [Sandholm, 1999a, Proposition 2.3], we can not even hope for being able to construct approximate algorithms with reasonable optimality guarantees that are very efficient. Furthermore, constructing very general integer programming algorithms is of a much broader interest than the one of the e-commerce community, and such work should probably be published elsewhere; the e-commerce relevance of constructing general integer programming algorithms without having any idea about what real-world distributions may look like is not completely obvious. It may actually be that the bids of certain real-world auctions have very distinctive distributions (for example caused by bidding rules), for which very simple and efficient algorithms can be constructed. But it may also be completely different; real distributions may be very hard and the performance tests used by current algorithms give no guarantee for their usefulness in real settings.

The construction of realistic probability distributions based on some of our main application areas—such as electronic power trade [Ygge, 1999] and train scheduling markets[8]—together with some reasonable agent strategies in certain attractive combinatorial auction models, such as *i*Bundle [Parkes, 1999] or the ascending bundle auction by Wurman [Wurman, 1999] is important future work. However, one brief reflection on realistic probability distributions can be given already here; there are good reasons to believe that real distributions will be *much harder* than the ones described above. For example, if the *i*Bundle auction is used and we have agents with the strategies that they only bid ϵ above the current prices (or taking the "ϵ-discount") we will have a very "tight" distribution; most bids are part of some combination which is close to optimal. This makes pruning drastically harder. Again this calls for gathering of real-world (or at least derivation of realistic) data, before focusing on heavily specialized algorithms.

[8] Confer e.g. `www.sics.se/~tuff`.

6 Conclusions

In this paper we have discussed important computational aspects of optimal winner determination in combinatorial auctions. We have compared recent approaches to this problem with a traditional approach to the computationally equivalent problem of set partitioning. The main conclusion of this comparison is that many of the features of current algorithms are rediscoveries of well-known methods.

We then presented how integer programming could be utilized to manage more general problems than the ones managed by the recent highly specialized algorithms, and that commercially available software performs excellently for many problem instances.

We also discussed and exemplified the enormous impact the probability distribution of a given test has on the computation time. It was shown that some of the distributions used in benchmarking current algorithms allow for very simple and efficient algorithms that take advantage of the structure of these distributions.

In summary our conclusions are that:

- much can be gained by *capitalizing on the achievement made in operations research and combinatorial optimization*,
- more work is needed on the *study of what real-world (or at least realistic) instances may look like*, and
- highly *specialized algorithms* are mainly of interest (from an e-commerce point of view) for real-world instances for which *standard algorithms fall short*.

Not only is it useful for electronic commerce to take advantage of existing achievements in operations research and combinatorial optimization, but it is also a concern that introducing “new” algorithms while overlooking existing theory is scientifically problematic. Furthermore, it is actually debatable if developing “new” set packing algorithms benchmarked on arbitrary distributions is a relevant e-commerce research activity. On the other hand, (i) gather real world distributions (or derive realistic ones from realistic agent preferences, agent strategies and market mechanisms), (ii) investigate state of the art of operations research and combinatorial optimization algorithms for these settings, and (iii) develop special purpose algorithms where needed, definitely is.

Acknowledgements

We thank Greger Ottosson for enlightening us on several integer programming issues, and NUTEK (www.nutek.se) and the sponsors/owners of EnerSearch (www.enerseach.se), i.e. ABB Automation Products, ECN, Iberdrola, IBM Utility and Services, PreussenElektra, and Sydkraft, for their financial support.

References

[Balas and Padberg, 1976] E. Balas and M. W. Padberg. Set partitioning: A survey. *SIAM Review*, 18:710–760, 1976.

[Balas, 1965] E. Balas. An additive algorithm for solving linear programs with zero-one variables. *The Journal of the Operations Research Society of America*, pages 517–546, 1965.

[Cheng and Wellman, 1998] J. Cheng and M. P. Wellman. The WALRAS algorithm—A convergent distributed implementation of general equilibrium outcomes. *Computational Economics*, 12:1–24, 1998. (Available from ai.eecs.umich.edu/people/wellman).

[Fujishima *et al.*, 1999] Y. Fujishima, K. Leyton-Brown, and 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*, pages 548–553, August 1999. (Available from robotics.stanford.edu/˜kevinlb).

[Garfinkel and Nemhauser, 1969] R. Garfinkel and G. L. Nemhauser. The set partitioning problem: Set covering with equality constraints. *Operations Research*, 17(5):848–856, 1969.

[Geoffrion, 1969] A. M. Geoffrion. An improved implicit enumeration approach for integer programming. *Operations Research*, 17:437–454, 1969.

[Michaud, 1972] T. Michaud. Exact implicit enumeration method for solving the set partitioning problem. *The IBM Journal of Research and Development*, 16:573–578, 1972.

[Mullen and Wellman, 1995] T. Mullen and M. P. Wellman. A simple computational market for network information services. In V. Lesser, editor, *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*, pages 283–289, San Francisco, CA, June 1995.

[Parkes, 1999] D. Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the First International Conference on Electronic Commerce*, pages 148–157. ACM Press, Box 11405, New York, NY, November 1999. (Available from www.cis.upenn.edu/˜dparkes).

[Rassenti *et al.*, 1982] S. J. Rassenti, V. L. Smith, and R. L. Bulfin. A combinatorial auction mechanism for airport time slot allocation. *Bell Journal of Economics*, 13:402–417, 1982.

[Rothkopf *et al.*, 1995] M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1995.

[Salkin, 1975] H. M. Salkin. *Integer Programming.* Addison Wesley Publishing Company, Reading, Massachusetts, 1975.

[Sandholm and Ygge, 1999] T. W. Sandholm and F. Ygge. Constructing speculative demand functions in equilibrium markets. Technical Report WUCS-99-26, Department of Computer Science, Washington University, October 1999. (Available from www.enersearch.se/ygge).

[Sandholm, 1996] T. W. Sandholm. *Negotiation among Self-Interested Computationally Limited Agents.* PhD thesis, University of Massachusetts, Amherst, 1996. (Available from www.cs.wustl.edu/˜sandholm).

[Sandholm, 1999a] 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*, pages 542–547, August 1999. (Available from www.cs.wustl.edu/˜sandholm).

[Sandholm, 1999b] T. W. Sandholm. EMediator: A next generation electronic commerce server. Technical report, AAAI Workshop Tech. Report, WS-99-01, 1999.

[Wurman, 1999] P. Wurman. *Market Structure and Multidimensional Auction Design for Computational Economies.* PhD thesis, Department of Computer Science, University of Michigan, 1999. (Available from www.csc.ncsu.edu/faculty/wurman).

[Yamaki *et al.*, 1996] H. Yamaki, M. P. Wellman, and T. Ishida. A market-based approach to allocating QoS for multimedia applications. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems, ICMAS'96*, pages 385–392. AAAI Press, Menlo Park, CA, December 9–14 1996. (Available from ai.eecs.umich.edu/people/wellman).

[Ygge and Akkermans, 1996] F. Ygge and J. M. Akkermans. Power load management as a computational market. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems, ICMAS'96*, pages 393–400. AAAI Press, Menlo Park, CA, December 9–14 1996. (Available from www.enersearch.se/ygge).

[Ygge and Akkermans, 1999] F. Ygge and J. M. Akkermans. Decentralized markets vs. central control: A comparative study. *Journal of Artificial Intelligence Research, JAIR*, 11:301–333, November 1999. (Available from www.jair.org).

[Ygge, 1999] F. Ygge. Energy resellers—An endangered species? In Moukas, Sierra, and Ygge, editors, *Proceedings of the Agent-Mediated Electronic Workshop in conjunction with the International Joint Conference on Artificial Intelligence 1999*, July 31 1999. (Available from http://www.enersearch.se/ygge).