Integer Programming for Combinatorial Auction Winner Determination

Arne Andersson Mattias Tenhunen Fredrik Ygge
Computing Science Department
Information Technology
Uppsala University
Box 337
SE - 751 05 Uppsala, Sweden
{arnea,tein,ygge}@csd.uu.se
http://www.csd.uu.se/~{arnea,tein,ygge}

Abstract

Combinatorial auctions are important as they enable bidders to place bids on combinations of items; compared to other auction mechanisms, they often increase the efficiency of the auction, while keeping risks for bidders low. However, the determination of an optimal winner combination in combinatorial auctions is a complex computational problem.

In this paper we (i) compare recent algorithms for winner determination to traditional algorithms, (ii) present and benchmark a mixed integer programming approach to the problem, which enables very general auctions to be treated efficiently by standard integer programming algorithms (and hereby also by commercially available software), and (iii) discuss the impact of the probability distributions chosen for benchmarking.

1 Introduction

Combinatorial auctions are important as they enable bidders to place bids on combinations of items; compared to other auction mechanisms, they often increase the efficiency of the auction, while keeping risks for bidders low [Rassenti *et al.*, 1982; Rothkopf *et al.*, 1995; Parkes, 1999; Wurman, 1999]. The determination of an optimal winner combination in combinatorial auctions is an $\mathcal{NP}$-hard problem [Rothkopf *et al.*, 1995], which has recently attracted some research, e.g. [Rothkopf *et al.*, 1995; Nisan, 1999; Fujishima *et al.*, 1999; Sandholm, 1999]. In this paper we look further into the topic. In particular, our contributions are:

- The recent algorithms by Fujishima et al. [Fujishima *et al.*, 1999] and Sandholm [Sandholm, 1999] are compared to traditional algorithms for the computationally identical problem of set packing, and hereby put into a proper computer science perspective. From this exercise, we learn that many of the main features of recently presented algorithms are rediscoveries of traditional methods in the operations research community.
- We observe that the winner determination problem can be expressed as a standard mixed integer programming problem, cf. [Nisan, 1999; Wurman, 1999], and we show that this enables the management of very general problems 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 sets used for evaluating different algorithms is discussed and exemplified. Particularly we demonstrate that some of the distributions used for benchmarking in recent literature [Fujishima *et al.*, 1999; Sandholm, 1999] can be efficiently managed with rather trivial algorithms.

The paper is organized as follows. In Section 2 we present a well-known set partitioning algorithm by Garfinkel and Nemhauser [Garfinkel and Nemhauser, 1969], 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 a mixed integer programming problem and hereby be solved by standard algorithms and commercial software. In Section 4 some empirical benchmarking for standard mixed integer programming software is presented, and we discuss the significance of the probability distribution of the test sets used for benchmarking. Finally, Section 5 concludes.

2 Recent winner determination algorithms and traditional algorithms for corresponding problems

Before discussing how very general versions of winner determination can be solved by general purpose algorithms, we investigate the basic case in which a bid states that a bundle of commodities, $\mathbf{q}_i = [q_{i1}, q_{i2}, \ldots, q_{ik}]$, $q_i \in \{0, 1\}$ (k is the number of commodities) is valued at $v_i \in \Re$. Given a collection of such bids, the surplus maximizing combination is the solution to the integer programming problem [Wurman, 1999; Nisan, 1999]:

$$\begin{array}{l} \max \sum_{i=1}^{n} v_i B_i \\ s.t. \sum_{i=1}^{n} q_{ij} B_i \leq 1, 1 \leq j \leq k \end{array} \tag{1}$$

where n is the number of bids and B_i is a binary variable representing whether bid i is selected or not.

We focus this presentation around a set partitioning algorithm introduced by Garfinkel and Nemhauser [Garfinkel and Nemhauser, 1969]. (For definitions of the set partitioning problem and related problems, cf. Balas and Padberg [Balas and Padberg, 1976], and Salkin [Salkin, 1975].) 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". Indeed, some of the experimental results reported by Garfinkel and Nemhauser seem surprisingly good compared to recent experiments when taking the hardware performance at that time into account.

The principles of the Garfinkel-Nemhauser algorithm are as follows. The algorithm creates one list per row (i.e. commodity) and each column (set/bid) is stored in exactly one list. Given an ordering of the rows, each set is stored in the list corresponding to its first occurring row. Within each list, the sets are sorted according to increasing cost. The search for the optimal solution is done in the following way:

1. Choose the first set from the first list containing a set as the current solution.

2. Add (to the current solution) the first disjoint set from the first list—corresponding to a row not included in the current solution—containing such a disjoint set, if any.

3. Repeat Step 2 until one of the following happens: (i) *The cost for the current solution exceeds the cost for the best solution:* this branch of the search 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 set by the next valid set in its list and go to Step 2. When no more sets can be selected from the list, back up further recursively. If no more backtracking can be done, terminate.

Since the problem of Equation (1) is equivalent to the definition of set packing and the problems of set packing and set partitioning can be transformed into each other [Balas and Padberg, 1976], the Garfinkel-Nemhauser algorithm can be used for winner determination in combinatorial auctions. It is also clear that it is trivial to modify the algorithm to be suited for set packing without any modification of the input; it is only a minor modification of the pruning/consistency test. Specifically, the sets need to be renamed as bids, cost has to be replaced by valuation, and item 3 needs to be replaced by:

> Repeat Step 2 until one of the following happens: (i) *The value of the current solution can not exceed the value of the best combination found so far:* this branch of the search can be pruned. (ii) *Nothing more can be added:* check if this is the best solution so far.

As seen from the above description, the currently best performing winner determination algorithm[1], 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.), (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) caching of partial search results, and (ii) improved pruning.

The caching, normally referred to as dynamic programming, is done by storing partial search results in a table. This is reported to often pay off, as many partial allocations share the same "rest term" (i.e. remaining unassigned commodities) [Fujishima *et al.*, 1999]. The cache can hereby also be 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 surplus of the cached allocation is smaller than the surplus of best combination found so far, this branch of the search can be pruned.

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

[1] 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, 1999] by approximately two orders of magnitude for the distributions tested.

3 Combinatorial auction winner determination as a mixed integer programming problem

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

As discussed below, by properly formulating the problem, we get a large number of very attractive features. These include:

- The formulation can utilize standard algorithms and hence be run directly on standard commercially available, thoroughly debugged and optimized software, such as CPLEX.[2]
- There may be multiple units traded of each commodity.
- Bidders are not restricted to bid for integer amounts.
- Bidders can construct advanced forms of mutually exclusive bids.
- Sellers may have non-zero reserve prices.
- There need not be a distinction between buyers and sellers; a bidder can place a bid for buying some commodities and simultaneously selling some other commodities.
- Complicated recursive bids with the above features can be expressed.
- Very general constraints can be expressed.
- Settings without free disposal (for some or all commodities) can be managed.

It should be pointed out that CASS and Sandholm's algorithms can handle some of the generalizations above. For example, mutually exclusive (XOR) bids are easily formulated by adding dummy commodities (e.g. "a XOR $b \Leftrightarrow (a \wedge c) \vee (b \wedge c)$"), but such transformations often give rise to a combinatorial explosion of bids [Nisan, 1999].

It is noteworthy that the formulation of Equation (1) can be run directly by commercially available software, and in Section 4 some empirical comparison between recent algorithms and the standard CPLEX software is shown. (Note that the formulation of Equation (1) only is applicable to the simple case discussed in Section 2 and that the linear programming form of the winner determination problem in general will look different.) However, the possibility of using off-the-shelf software has been overlooked and current benchmarked algorithms [Fujishima *et al.*, 1999; Sandholm, 1999] are written from scratch. (The formulations of this problem given by Rothkopf et al. [Rothkopf *et al.*, 1995], Fujishima et al. [Fujishima *et al.*, 1999], and Sandholm [Sandholm, 1999], are however not suited as direct input for standard software.)

The formulation used here conforms with the formulations by Wurman [Wurman, 1999] and Nisan [Nisan, 1999]. Compared to these formulations, we observe that much more general combinatorial auctions than the ones treated so far can be expressed as mixed integer problems, and that they can be successfully managed by standard operations research algorithms and commercially available software.

With standard MIP methods, any constraint that can be expressed in linear terms in the involved variables can be used when defining a bid. Thus in the general case, the objective function will consist of terms representing the value of a (certain part of a) bid, times the degree to which it is accepted. That is, we need not restrict the auction to only binary choices. Correspondingly, the feasibility constraint need in the general case not be restricted to the cases where there is only one unit for sale of each commodity, free disposal can be assumed, etc.[3] It is also possible to use the MIP approach for the minimal winning bid problem [Rothkopf *et al.*, 1995], i.e. the problem of replying to the question "If I request these and these amounts of these and these commodities, how much do I have to pay to get my bid accepted?".

Clearly, requiring that each bidder should give its bids as terms to be added to the objective function and the feasibility constraints together with a number of constraints may be a too heavy burden put on a bidder, cf. [Nisan, 1999]. Therefore it makes sense to construct different high level support for expressing bids and using the combinatorial auction.

4 Empirical benchmarking

In this section we give some empirical data in order to compare the new approach to optimal winner determination based on standard MIP (and consequently tested with off-the-shelf software) to current highly specialized approaches in cases where these can be applied.

It is generally recognized that it is most unfortunate that real-world test data are not available. As long as no such

[2]See `www.cplex.com`

[3]The free disposal assumption (i.e. that non-allocated resources can be disposed without a cost) typically has a very drastic impact on "anytime behavior"; without the free disposal assumption, finding a feasible allocation is significantly harder.

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. The empirical benchmarks are performed on the same test data as was given by Sandholm [Sandholm, 1999] and Fujishima et al. [Fujishima *et al.*, 1999]. We use these tests not because we are convinced that they are the most relevant, but because they are the only ones for which we have data for competing algorithms. Our experience so far is that (seemingly small differences of) the bid distributions chosen have an *extreme* impact on the performance of the algorithms. It is theoretically shown that (unless $\mathcal{P} = \mathcal{NP}$) no efficient general optimal algorithm can be constructed [Rothkopf *et al.*, 1995] (not even if a certain approximation error can be tolerated [Sandholm, 1999, Proposition 2.3]). Therefore one should be very careful when arguing about the practical usefulness of an algorithm without access to real data.

The software used is CPLEX version 6.5. The hardware setup of the experiments has been one standard uniprocessor 550MHz PC with 128Mb of RAM memory.[4] The time required for loading the test data into memory has not been included in the results below. The time reported is "wall time", i.e. an upper bound on processor time used.

For the sake of reproducibility, all test data and programs required for generating the CPLEX input format, as well as detailed descriptions of the CPLEX settings used is available from the Internet.[5]

Figure 1 to Figure 5 show the results of the respective tests. For each distribution, 10 instances have been tested, which is sufficient for obtaining a basic illustration. The instance sizes have been selected to match the sizes tested in the literature and/or to give reasonable computation time.

During the search for the optimal solution, CPLEX reports the best solution found so far (i.e. works as what is sometimes referred to as an anytime algorithm). In the figures, the curves denote the surplus of the currently best solution normalized by the surplus of the optimal solution. For each moment in time, the worst, average and best solution is plotted. The point at which optimality is verified is marked as a special point ($\triangleright$ and $\triangleleft$ for minimum and maximum time respectively, $\diamond$ for the average, and * for all instances other than min and max).

We have tested the following distributions.

Random [Sandholm, 1999]

Definition: 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 a random integer valuation between 1 and 1000. (For this and all other distributions we have used integer valuations for simplicity of parsing etc. However, our experience is that changing to real numbers increases computation time by less than 10%; a negligible number for our purposes.)

Results and discussion: CPLEX determines the optimal winner efficiently, see Figure 1; the timings are superior to those presented by Sandholm [Sandholm, 1999]. 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 significant number of bids, 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—instead of using CPLEX—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 heuristic column dominance checking algorithm on an ordinary 550MHz PC. Clearly, this suggests that a trivial approximate algorithm should be used for this distribution: Select the highest bid—requesting only one commodity—for each commodity. For all bid sets we have tried (of any significant size), this would have resulted in an optimal solution.

The special character of the computation—only one "step" in the surplus of the best and worst solutions and 10 steps for the average solution—is explained by the nature of the distribution. From the above reasoning, it is easy to see that one can establish one price per commodity supporting the optimal allocation. When this holds, an LP solution to the problem is the optimal allocation [Nisan, 1999]. The principle of CPLEX (which is to first establish an LP solution and then do a branch and bound) then explains this behavior.

The characteristics of the **weighted random** distribution [Sandholm, 1999] are similar both in terms of computation time as well as in the possibility to use a trivial highly efficient approximate algorithm. (In this case the trivial algorithm is to simply select the bid with the highest valuation.)

[4] 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.

[5] See `www.docs.uu.se/~tein/IPForComb.html`.

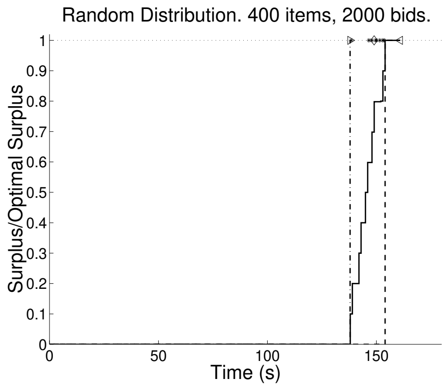

Figure 1. The *Random* distribution for 2000 bids and 400 commodities. For this distribution the time for obtaining the optimal combination and being able to guarantee this optimality is the same for most tested instances. Worst case is in the area of 160s. In comparison, Sandholm's algorithm is reported to manage 1000 bids and 400 commodities in approximately 6000s on average.

Uniform [Sandholm, 1999]

Definition: Draw the same number of randomly chosen items for each bid. Pick an integer valuation from [500..1500] and multiply by the number of commodities.

Results and discussion: CPLEX performs well compared to Sandholm's implementation [Sandholm, 1999], cf. Figure 2. Still this appears to be a "hard" distribution for CPLEX.

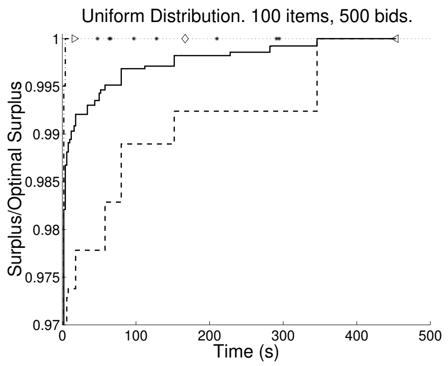

Figure 2. The *Uniform* distribution for 500 bids and 100 commodities, where each bidder bids for three commodities. The difference between the different instances vary significantly, and the execution times vary from a few seconds to around 470s. Sandholm's algorithm is reported to manage 150 bids and 100 commodities in approximately 40000s on average. (For this latter instance size, CPLEX finds the verified optimal solution in around 400ms.)

Decay [Sandholm, 1999].

Definition: 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 random integer valuation between 1 and 1000 and multiply by the number of commodities requested.

Results and discussion: CPLEX performs well compared to Sandholm's algorithm [Sandholm, 1999], and rather large bid sets can be efficiently managed, see Figure 3.

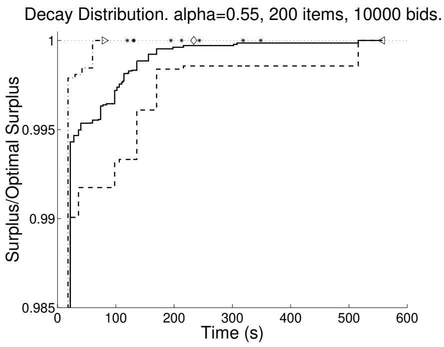

Figure 3. The *Decay* distribution for 10000 bids and 200 commodities with $\alpha = 0.55$. Worst case is around 580s. For the same distribution with 200 bids and 200 commodities the execution time with Sandholm's algorithm is reported to be around 40000s.

Binomial [Fujishima *et al.*, 1999]

Definition: The probability distribution for a bid requesting n commodities of k commodities in the market is

$$f(n) = p^n(1-p)^{k-n} \left(\begin{array}{c} k \\ n \end{array} \right),$$

with $p = 0.2$. An integer valuation is drawn from 500 to 1500 and multiplied by n.

Results and discussion: Under the assumption that the benchmarks given by Fujishima et al. [Fujishima *et al.*, 1999] denotes the time required for finding a verified optimal solution[6], the CASS implementation is faster than CPLEX (around 20 times) for 30 commodities and 3000 bids. Still CPLEX can manage rather big bid sets efficiently. (Note that in Figure 4, 30000 bids are used.)

The CASS algorithm was also reported to be tested on 1500 bids and 150 commodities [Fujishima *et al.*, 1999]. However, a later version of the paper suggests that there was

[6]It is an important distinction between the time required for finding an optimal solution and verifying that it is so. We have not been able to tell from the paper by Fujishima et al. [Fujishima *et al.*, 1999] which of the two they present.

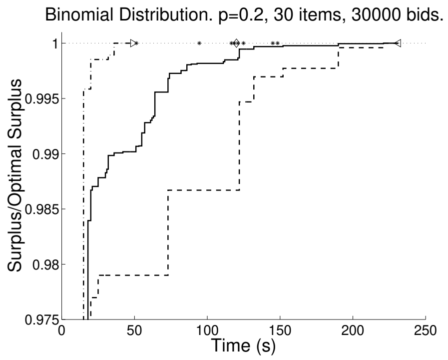

Figure 4. The *Binomial* distribution is shown for 30000 bids and 30 commodities. Here the relative surplus raises to the area of 97% in a few seconds. The optimal combination is normally found after between 30 and 150s. The worst case for finding a guaranteed optimal solution is around 240s. (For smaller instances, 3000 bids, the timing reported for CASS are approximately 20 times better than the ones of CPLEX.)

a typo in the published version. Instead, the number of bids seem to be 15000. While CPLEX can manage 1500 bids efficiently, it fails to handle 15000 bids. On the other hand, it is interesting to note that this problem instance, which at a first glance appears to be "hard", actually turns out to be "easy". Indeed, there is a simple algorithm which from our experience outperforms CASS.

The reason that the problem is simple is, modulo some details, 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 algorithm, enumerating all non-colliding combinations (i.e. it uses neither pruning nor ranking heuristics) finds the verified optimal solution in approximately 15s on a standard 450MHz PC with 256Mb RAM. This is around 15 times faster than CASS, under the assumption that Figure 3 in the Fujishima et al. paper [Fujishima *et al.*, 1999] denotes 15000 bids. (If the number of bids is 1500 as stated in the paper the difference is of course larger.) This algorithm works as follows:

1. For each bid, construct a list of all non-colliding bids. Assuming a certain probability distribution and a certain number of commodities, the time for this is quadratic in the number of bids.

2. Combine bids in a depth first manner. No bid comparisons are made, instead we use the lists of non-colliding bids. As we combine bids, we also combine their lists by taking their intersection. The short length of the lists of non-colliding bids gives the fast execution. (For an input of 15000 bids, 150 commodities, and $p = 0.2$, a typical number of valid combinations of two bids is 250000, three bids 40000, and four bids 200.)

Of the 15s spent on each test set, around 14s were spent in step 1. Hence, neither pruning nor ranking heuristics can improve this algorithm significantly here.

Exponential [Fujishima *et al.*, 1999]

Definition: The probability distribution is defined as $f_e(n) = Ce^{-n/p}$ (with C is assumed to be implicitly defined from $\sum_{n=1}^{k} f_e(n) = 1$, where k as before is the number of commodities). The valuation is an integer, rectangularly drawn from $[500..1500]$ and multiplied by the number of requested commodities.
Results and discussion: CASS appears to be around a factor of two faster than CPLEX for 3000 bids and 30 commodities, cf. Figure 5. In an extended on-line version of their paper, Fujishima et al. report that the CASS algorithm finds the optimal solution (though it is not clearly stated that optimality is verified) in around 550s for 4500 bids and 45 commodities. CPLEX finds the verified optimal solution for the corresponding instance size in around 8s on average (with small variation).

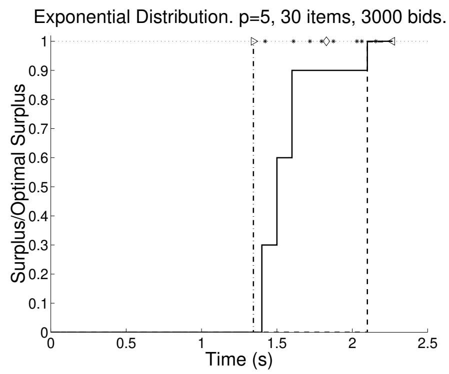

Figure 5. The *Exponential* distribution for 3000 bids and 30 commodities. The verified optimal solution is found in at most around 2.3s. The corresponding timing for the CASS algorithm is slightly above 1s.

4.1 Discussion

In sum, CPLEX performs very well for many of the tested distributions. Under most reasonable assumptions of the collection of bids, the computation time is relatively small. Furthermore, if the bids are submitted in sequence under some amount of time, CPLEX can do subsequent searches starting from the best solution found up to the point of the arrival of a new bid. For the harder distributions—which hence are of main interest—CPLEX is around five orders of magnitude faster than Sandholm's algorithm. As CASS has been reported to outperform Sandholm's algorithm by around two orders of magnitude for the harder distributions, there is an indication that CPLEX also is faster than CASS is here. Still, as a consequence of the $\mathcal{NP}$-hardness of the problem (and as indicated by our empirical study), nor CPLEX nor any other known algorithm is a silver bullet for this problem. Even though CPLEX outperformed the recent algorithms for the sparser distributions (sparse in the sense that few bids collide), implicit enumeration algorithms—for example in the spirit of the Nemhauser-Garfinkel algorithm, cf. Section 2—becomes highly competitive for denser distributions. Again, we should bare in mind that CPLEX is a general-purpose software and that the comparisons only are performed for the very special cases where Sandholm's algorithm and the CASS implementation can be used.

From our experiments with different families of algorithms it is clear that if the probability distribution is known to the auctioneer, it is sometimes able to 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 algorithms (assuming that standard algorithms fall short on practically relevant instances).

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.

The construction of realistic probability distributions based on some of our main application areas—such as electronic power trade and train scheduling markets—together with some reasonable agent strategies in certain attractive combinatorial auction models, such as *i*Bundle [Parkes, 1999] or AkBA [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. For example, we tried the Uniform distribution, but with 100000 added to each valuation (i.e. the valuations vary in only 1%), and this increased the execution time of CPLEX by some factor 20. But there are also other aspects of the hardness of real-world distributions [Nisan, 1999]. Again this calls for gathering of real-world (or at least derivation of realistic) data, before focusing on heavily specialized algorithms.

5 Conclusions

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

We then discussed how mixed 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 believe that this can enable the application of combinatorial auctions to applications to which there are not yet any winner determination algorithms available. The approach introduced here can be used in combination with many different forms of combinatorial algorithms and is of interest regardless of whether bids are sealed or open, whether the auction is iterative or one-shot, and whether the computation is centralized or decentralized (e.g. let the bidders suggest better solutions).

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) gathering real world distributions (or derive realistic ones from realistic agent preferences, agent strategies and market mechanisms), (ii) investigating state of the art of operations research and combinatorial optimization algorithms for these settings, and (iii) developing special purpose algorithms where needed, definitely is.

A final—more fundamental issue—is whether the approach used in this and other papers is at all useful. One view is that the bids should be restricted in such a way that polynomial time algorithms can be used to find the optimal allocation [Rothkopf *et al.*, 1995]. Another view is that it is unnecessary to restrict the bids; if the bids happen to be restricted in the ways Rothkopf et al. suggest [Rothkopf *et al.*, 1995], then an appropriate algorithm will rapidly find the optimal solution anyway [Nisan, 1999]. We very much agree with this latter view. Our arguments for this are as follows. If it is the case that there only are a few important dependencies (which is required for implying that the restrictions proposed by Rothkopf et al. do not decrease the surplus significantly), most bids will reflect these dependencies. The ones that reflect other (less important) dependencies will be non-competitive and pruned from the search at an early stage. Furthermore, we argue that if there are so many dependencies (of comparable importance) that the winner determination indeed is computationally intractable, then it seems safe to conjecture that it is always better to allow bids on all combinations and use an approximate algorithm, than to restrict the bids. The following heuristic algorithm supports this point of view. First identify the most important dependencies (i.e. the ones that would have been used if we would have taken the restricting approach by Rothkopf et al. [Rothkopf *et al.*, 1995]), then as first heuristics in the search, only consider bids that fulfill the restrictions (and let this heuristics be common knowledge). Then once the optimal solution of this restricted bid set has been found, add all other bids and search until a certain dead-line is met, and take the best solution found up to that point. Under the assumption that the time for extracting only the bids fulfilling the restrictions is negligible (it must indeed be small for the restricting approach to be successful—otherwise checking the validity of bids will be too hard), this approach can safely be expected to always do as least as well as the approach of only treating the restricted bid space.

Rothkopf et al. [Rothkopf *et al.*, 1995] discuss another attractive idea; to let bidders suggest winning allocations. As each bidder will prioritize its own bids heavily in such a search (in order to find a winning combination of which its own bids is a part), this may serve as a very efficient parallelization of the search and utilization the computational power of the participating agents. In auctions with high values in which all bidders can propose better combinations, and in which highly optimized software (as the one described in this paper) is used, we can therefore expect that very large bid spaces can be searched in reasonable time. If—despite good heuristics, considerable computational power (optionally with the bidders participating in the search), and highly efficient algorithms (such as CPLEX)—the problem of finding a good solution still is computationally intractable, then it is probably generally very challenging to construct a simple and computationally efficient auction with any significant economic efficiency.

Acknowledgements

We gratefully thank Greger Ottosson, David Parkes, Peter Wurman, Makoto Yokoo, Michael Wellman and his group, and Lars Rasmusson for important discussions.

References

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

[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.

[Nisan, 1999] N. Nisan. Bidding and allocation in combinatorial auctions. Working paper. Presented at the 1999 NWU Microeconomics Workshop. (Available from http://www.cs.huji.ac.il/ noam/), 1999.

[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, 1999] 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).

[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).