Chapter 12
The Winner Determination Problem

Daniel Lehmann Rudolf Müller Tuomas Sandholm

1 Introduction

This part of the book gives a comprehensive overview of the computational challenges in solving the winner determination problem (WDP): given a set of bids in a combinatorial auction, find an allocation of items to bidders (the auctioneer can keep some of the items) that maximizes the auctioneer's revenue. The bids are expressions in a bidding language, by which bidders report valuations for subsets of items (see Nisan (Chapter 9)). The auctioneer's revenue is maximized by choosing an allocation that maximizes the sum, over all bidders, of the bidders' valuations for the subset of items that they receive.

In this part of the book we do not discuss the concern that bidders might not report their true valuations because of strategic considerations. They can be *motivated* to tell the truth, under certain assumptions, by using the Vickrey-Clarke-Groves (VCG) mechanism, as discussed further in Ausubel and Milgrom (Chapter 1). We also do not discuss the issue that in

some mechanisms, such as the VCG, the prices charged from winning bidders differ from the bids made, implying that the value of the optimal allocation is not equal to the revenue for the auctioneer. However, the WDP—exactly as discussed in this chapter—is the key to solving those problems as well: the VCG mechanism can be run by solving the WDP once overall, and once for each bidder removed in turn.

In this opening chapter we formally define the winner determination problem as a combinatorial optimization problem and present various alternative mathematical programming models for it. We derive several results stating that one cannot hope for a general-purpose algorithm that can efficiently solve every instance of the problem. We shall also see that restricting significantly the structure of the bids, such as allowing only bids of size of at most 3, does not help. However, negative complexity results do not exclude the ability to design algorithms that have a satisfactory performance for problem sizes and structures occurring in practice. The chapters of this part of the book will present many results in this direction: approximation methods for special cases, identification of polynomial solvable special cases, non-polynomial but nevertheless fast exact solution methods, means to evaluate the empirical hardness, well-behaved practical cases and non-computational approaches to circumvent the inherent computational complexity.

This chapter is organized as follows. Section 2 defines the winner determination problem as a combinatorial optimization problem, and lays out mathematical programming models that capture in detail variants of the problem related to specific bidding languages. Section 3 presents several computationally intractable versions of *WDP*, discusses complexity implications of side constraints, and presents exhaustive algorithms that are effective when the number of items for sale is small. Section 4 illustrates negative results in terms of approximability, and reviews approximation algorithms. Section 5 concludes.

2 Problem formulation

As elsewhere in the book the following notation is used. The set of bidders is denoted by $N = \{1, \ldots, n\}$, the set of items by $M = \{1, \ldots, m\}$. A bundle S is a set of items: $S \subseteq M$. For a bundle S and a bidder i, we denote by $v_i(S)$ the package bid that bidder i makes for bundle S, i.e., the maximal price that i announces to be willing to pay for S.

An allocation of the items is described by variables $x_i(S) \in \{0, 1\}$. The variable $x_i(S)$ is equal to one if and only if bidder i gets bundle S. An allocation $(x_i(S)|i \in N, S \subseteq M)$ is said to be feasible if it allocates no item

more than once:

$$\sum_{i \in N} \sum_{S \subseteq M, S \ni j} x_i(S) \leq 1 \quad \text{for all } j \in M, \tag{1}$$

and at most one subset to every bidder

$$\sum_{S \subseteq M} x_i(S) \leq 1 \quad \text{for all } i \in N. \tag{2}$$

Definition 2.1 (Winner Determination Problem (WDP)) *Given bids v_i, $i = 1, \ldots, n$, the* winner determination problem *is the problem to compute*

$$x \in \textit{argmax} \, (\sum_{i \in N} v_i(S) x_i(S) \mid x \textit{ is a feasible allocation}). \tag{3}$$

Definition 2.1 is not sufficient to discuss the algorithmic complexity of solving the *WDP*. It uses a number of decision variables and coefficients that is exponential in the number of items, and it does not specify how the bids are represented. The algorithmic complexity of an optimization problem is however measured in terms of the encoding length of the problem, i.e., in terms of how many binary symbols are needed in order to store an instance of the problem in computer memory. An optimization problem is said to be solvable in polynomial time or *tractable*, if the number of operations needed for any instance is bounded by a polynomial function in the encoding length of that instance. A naive representation of the *WDP* would store for every bidder i and for every subset S the bid value $v_i(S)$.

With respect to this huge representation—the size is exponential in the number of items—dynamic programming would provide a polynomial method to solve the *WDP* (see Section 3.4). Hence, in order to discuss the complexity of *WDP* we need to specify the representation of bids first.

In Nisan (Chapter 9) bidding languages are introduced as means to represent bids. We will focus here on the *OR* and *XOR* bidding languages, by which we define two refinements of *WDP*, called WDP_{OR} and WDP_{XOR}. In both frameworks, every bidder i provides atomic bids $v_i(S)$ for a set of bundles $S \subset M$. We use the notation $\mathcal{F}_i$ for the set of bundles for which bidder i submits an atomic bid. A bidding language may restrict $\mathcal{F}_i$ to subsets of items that have a particular combinatorial structure, which will have an influence on the algorithmic complexity of the *WDP*. This is briefly discussed later in this chapter and in depth in Müller (Chapter 13). For notational convenience we assume $\emptyset \in \mathcal{F}_i$ and $v_i(\emptyset) = 0$. We also make the common assumption of free-disposal: for $S, S' \in \mathcal{F}_i$ such that $S \subseteq S'$ it is $v_i(S) \leq v_i(S')$.

In the *OR* bidding language the atomic bids by bidder i are interpreted as: i is willing to pay for any combination of pairwise disjoint atomic bids a price equal to the sum of the bid prices. By the free-disposal assumption, i's bid for any other set S is then the maximum sum of bid prices for pairwise disjoint atomic bids contained in S (see Nisan (Chapter 9)).

Observe that in WDP_{OR} constraint (2) can be omitted, because, for every i, a solution with $x_i(T) = 1$ for exactly one $T \subseteq M$ has the same objective value as the solution $x_i(S) = 1$ for those $S \in \mathcal{F}_i$ which form a best packing of T. Observe also that we need only decision variables $x_i(S)$ for subsets $S \in \mathcal{F}_i$, and that for any i, j, $S_i \in \mathcal{F}_i$, $S_j \in \mathcal{F}_j$ with $S_i \subseteq S_j$ and $v_i(S_i) \geq v_j(S_j)$ we may drop the dominated bid $v_j(S_j)$. This leads to:

Definition 2.2 (WDP_{OR}) *Given a set of bids in the* OR *bidding language, with atomic bids on sets in $\mathcal{F}_i$ for every bidder i,* WDP_{OR} *is the problem to compute*

$$x \in \textit{argmax} \left(\sum_{i \in N, S \in \mathcal{F}_i} v_i(S) x_i(S) \mid x \textit{ satisfies } (1) \right). \tag{4}$$

In WDP_{XOR} the set of atomic bids by bidder i is interpreted as: bidder i is willing to receive at most one of the atomic bids. By the free-disposal assumption, i's bid for any other set S becomes then the maximum price for an atomic bid contained in S (see Nisan (Chapter 9)).

Note that in the case of WDP_{XOR} we may not drop constraint (2), unless bids are super-additive, i.e., for any two bundles S_1 and S_2 with $S_1 \cap S_2 = \emptyset$ we have $v_i(S_1 \cup S_2) \geq v_i(S_1) + v_i(S_2)$. Again we need only decision variables $x_i(S)$ for subsets $S \in \mathcal{F}_i$. This leads to:

Definition 2.3 (WDP_{XOR}) *Given a set of bids in the* XOR *bidding language, with atomic bids on sets in $\mathcal{F}_i$ for every bidder i,* WDP_{XOR} *is the*

problem to compute

$$x \in \mathit{argmax}\ (\sum_{i \in N, S \in \mathcal{F}_i} v_i(S)x_i(S) \mid x\ \mathit{satisfies}\ (1)\ \mathit{and}\ (2)). \tag{5}$$

WDP_{OR} and WDP_{XOR} are related in a similar way as *OR* and *XOR* bidding languages are related. On one hand, one can transform WDP_{XOR} to WDP_{OR} by transforming XOR bids to OR* bids. On the other hand, WDP_{OR} can be transformed into an instance of WDP_{XOR} by generating an *XOR* bid from every *OR* bid. This transformation has, however, a drastic impact on the encoding length as it calls for a potentially exponentially larger $\mathcal{F}_i$. Another reason to treat them separately is that, with respect to the same family $\mathcal{F}_i$, WDP_{OR} and WDP_{XOR} can have different computational complexity (see Section 3.2).

Sometimes several identical copies of the same type of item may be auctioned. We call this the multi-unit *WDP*. Conceptually, the multi-unit case is not of any impact since one can always consider the different copies as different items, and assume symmetric valuations. From a computational perspective, the assumption may well have an impact since the bidding language usually exploits the symmetry to derive a more compact bid representation. When we consider multiple copies, we shall assume that item $j \in M$ has multiplicity ω_j. A bundle $S = (\sigma_1(S), \ldots, \sigma_n(S))$ is, in this case, a multi-set of items, where $\sigma_j(S)$ is the multiplicity of item j in S.

Using this notation, we replace (1) by

$$\sum_{i \in N} \sum_{S \subseteq M} \sigma_j(S) x_i(S) \leq \omega_j \quad \text{for all } j \in M. \tag{6}$$

to define the feasibility of an allocation. This generalizes naturally to the *OR* and *XOR* bidding languages.

The following subsections illustrate how different classical optimization models from the field of Mathematical Programming can be used to model (special cases of) the WDP: integer linear programming, weighted stable set in graphs, knapsack and matching.

2.1 Integer linear programming and knapsack

Problem WDP_{OR} can be modeled by the integer linear program

$$(WDP_{OR}) \qquad \begin{array}{ll} \max & \sum_{i=1}^{n} \sum_{S \subseteq M} v_i(S) x_i(S) \\ & \sum_{i=1}^{n} \sum_{S \subseteq M, S \ni j} x_i(S) \leq 1 \quad \text{for all } j \in M \\ & x_i(S) \in \{0,1\} \end{array}$$

Based on our remarks above we need a variable $x_i(S)$ only for non-dominated bids on sets $S \in \mathcal{F}_i$. We use summations over all $S \subseteq M$ for notational convenience.

This model for WDP_{OR} is identical to the integer linear programming model of the weighted set packing problem (Rothkopf et al. 1998). In this problem we are given a collection of subsets of a set M, each with a weight, and the target is to find a sub-collection of non-intersecting sets of maximal total weight.

Problem WDP_{XOR} can be modeled by the integer linear program

$$
(WDP_{XOR}) \qquad
\begin{array}{lll}
\max & \sum_{i=1}^{n} \sum_{S \subseteq M} v_i(S) x_i(S) & \\
& \sum_{i=1}^{n} \sum_{S \subseteq M, S \ni j} x_i(S) \leq 1 & \text{for all } j \in M \\
& \sum_{S \subseteq M} x_i(S) \leq 1 & \text{for all } i \in N \\
& x_i(S) \in \{0,1\} &
\end{array}
$$

Again it is sufficient to restrict to variables $x_i(S)$ for $S \in \mathcal{F}_i$, but for convenience we use again the full summation.

In the more general case of multi-unit supply of items and multiplicities of items in bids, the above two integer linear programs get other than 0-1 coefficients.

Holte (2001) suggested to interpret in particular multi-item versions of WDP_{OR} and WDP_{XOR} as generalized knapsack problems. In the classical $0-1$-knapsack problem we are given a set of objects with weights α_j and values β_j, and we try to find a set of objects of maximum value with total weight less than or equal to some capacity α_0. In the case of WDP_{OR} we get a multi-dimensional knapsack problem (sometimes also called multi-item knapsack problem), and in case of WDP_{XOR} a multi-dimensional multiple-choice knapsack problem. Kellerer et al. (2004) provide a comprehensive survey of the state-of-the-art of research on knapsack problems. They also discuss the link to combinatorial auctions. In Müller (Chapter 13) dynamic programming algorithms for WDP are presented that stem from the connection to the knapsack problem.

2.2 Intersection graphs

A second way of modelling WDP_{OR} and WDP_{XOR} is by intersection graphs. We use a graph $G = (U, E)$ consisting of a finite set of nodes V and a set of undirected edges. The nodes in U are one-to-one to the bids $v_i(S), i \in N, S \in \mathcal{F}_i$, and two nodes are connected by an edge if and only if there is a conflict between the bids, i.e., an intersection of the sets of items, or, in WDP_{XOR} both bids are by the same bidder. A node u related to bid $v_i(S)$ gets a weight $w_u := v_i(S)$. Again we may restrict ourselves to relevant bids, in other words, the same bids as for which we have a column in one of the integer linear programming models above.

A subset U of nodes in a graph is called a *stable set* if no two nodes in U are connected by an edge. Synonyms for stable set to be found in the literature are *node packing* and *independent set*. The *maximum weighted stable set problem* is the problem to find a stable set of maximum total weight. It is obvious that WDP_{OR} and WDP_{XOR} coincide with weighted stable set on the intersection graph related to WDP_{OR} and WDP_{XOR}, respectively.

The maximum weighted clique problem in which we search for a maximum weighted set of nodes such that any two nodes *are* connected by an edge, is equivalent to stable set. Indeed, one can simply switch to the complement G^c of a graph G, which has an edge between two nodes if and

only if the edge is not contained in G. The (unweighted) stable set as well as the (unweighted) clique problem can thus serve as reference point to answer algorithmic questions on WDP_{OR} and WDP_{XOR}, (see, for example, Sandholm (2002)), as we will review later in this chapter. (In the unweighted case, all the weights are 1.)

2.3 Special Valuations

Special valuations allow for more succinct bidding languages, and lead also to special cases of the dicussed mathematical programming problems.

As a first example, we look at the Downward Sloping Symmetric Valuation introduced in Nisan (Chapter 9), in other words, the case of homogenous items with decreasing marginal valuations. *WDP* can be solved by sorting all marginal values of all bidders, and allocating to every bidder as many items as many marginal valuations he has among the M highest.

As a second example, we mention unit demand valuation, where every bidder is only interested in winning at most one item. In this case we can model *WDP* as weighted matching in a bipartite graph $G = (N, M, E)$ with an edge $e = (i, j) \in E$ if and only if bidder i makes a bid for item j. A graph is bipartite if its nodes can be partitioned into two disjoint subsets, such that all edges have a node in each of the two subsets. A matching is a subset of edges such that neither two of them share a common node. This

model, also called the assignment model, is the point of departure to study the connection between combinatorial auctions and Walrasian equilibria in exchange economies with indivisibilities (see Bikhchandani and Ostroy (Chapter 8)).

Further models for specialized valuations can be found in Müller (Chapter 13).

3 NP-hardness of the winner determination problem

Intuitively speaking, the *WDP* seems hard because one would need to check, *for each subset of the bids*, whether the subset is feasible (no bids within the subset share items) and how much revenue that subset of bids provides. A feasible subset that yields the highest revenue is an optimal solution. Unfortunately there are 2^k subsets of bids, k being the number of bids, so enumerating them is infeasible (except when there are fewer than 30 or so bids). The real question is: Can one do better than enumerating all the subsets of bids? To answer this question, we study the inherent complexity of the problem, that is, complexity that *any* algorithm would suffer from.

Before we present results on the computational complexity of the *WDP*, more precisely of different optimization models of *WDP*, we give a very brief introduction to complexity theory.

3.1 Complexity theory in a nutshell

An algorithm solves a problem in polynomial time if there exists a polynomial function g such that for every possible instance of the problem, the number of basic arithmetic operations needed by the algorithm is no more than $g(l)$, where l is equal to the number of binary symbols needed to represent the instance. Numbers are encoded binary, implying that an integer x has encoding length $\log_2 x$.

An algorithm that is polynomial in the *unary encoding* of some optimization problem, i.e., the length of an integer x is counted as x, is called *pseudo-polynomial*. If numbers involved are guaranteed to be small, a pseudo-polynomial running time may be practical, if numbers can become very large, they are in most cases not practical. If an algorithm for an optimization problem is polynomial, even if we measure the length of every number involved by 1, and if arithmetic operations involve only numbers that are polynomially bounded in this stricter measure, than the algorithm is said to be *strongly polynomial*.

Complexity theory investigates whether polynomial algorithms exist. It has been developed for *decision problems*, which are computational problems consisting of a class of instances and a definition of a property of these instances. The problem to be solved is to decide whether a given instance from the class has the property or not. For example, given a set of

combinatorial bids and a target revenue w, is there an allocation of items to bidders with a total revenue of at least w. $\mathcal{P}$ is the class of decision problems that have a polynomial time algorithm. $\mathcal{P}$ is contained in $\mathcal{NP}$ (non-deterministic polynomial time). This is the class of decision problems for which there exists a polynomial time algorithm that can check the validity of the property in question, if it is given as input next to the instance a certificate for the validity (regardless of how hard it is to compute such certificates). Equivalently, it is the class of problems computable by a nondeterministic Turing machine in polynomial time. The decision versions of (special cases of) *WDP* are all members of the class $\mathcal{NP}$, because we can verify in polynomial time whether a particular solution is feasible and has an objective value larger than some value w.

It is generally believed, but never proven, that $\mathcal{P} \neq \mathcal{NP}$, i.e. that there are problems in $\mathcal{NP}$ which can not be solved in polynomial time. A problem in $\mathcal{NP}$ is $\mathcal{NP}$-complete if any polynomial algorithm for it could be used as a polynomial algorithm for any other problem in $\mathcal{NP}$. Typically, one shows how to transform instances of any problem in polynomial time to instances of the $\mathcal{NP}$-complete problem, in order to prove $\mathcal{NP}$-completeness. This implies that $\mathcal{NP}$-complete problems are the hardest problems within $\mathcal{NP}$. Cook (1971) has proven the existence of an $\mathcal{NP}$-complete problem by showing that any problem in $\mathcal{NP}$ can be

transformed into the satisfiability problem. Based on Cook's theorem it became relatively easy to prove that other problems are $\mathcal{NP}$-complete. We first have to show that they are in $\mathcal{NP}$, and second we have to define a polynomial transformation of an $\mathcal{NP}$-complete problem to them.

A computational problem for which the existence of a polynomial time algorithm would imply tractability of all problems in $\mathcal{NP}$ is called $\mathcal{NP}$-hard. By definition, NP-complete problems are NP-hard, but also problems not contained in NP can be NP-hard. For example, optimization problems whose decision version is $\mathcal{NP}$-complete.

$\mathcal{ZPP}$ is a sub-class of $\mathcal{NP}$ which we will mention in the context of approximation algorithms. It consists of those decision problems for which there exists an algorithm and for every instance a certificate, such that the algorithm terminates in *expected* polynomial time, and outputs 1 if and only if the correct answer to the decision question is *yes*. The question of whether $\mathcal{NP}$=$\mathcal{ZPP}$ is as well an open question, related to the $\mathcal{P}$=$\mathcal{NP}$ question. $\mathcal{NP}$=$\mathcal{ZPP}$ is not known to imply $\mathcal{P}$=$\mathcal{NP}$.

3.2 Complexity of the winner determination problem

Almost every paper on combinatorial auctions mentions that the winner determination problem is $\mathcal{NP}$-hard. This can be simply derived by the fact that the $\mathcal{NP}$-hard weighted set packing problem is equivalent to WDP_{OR}

(Rothkopf et al. 1998). We take here a little closer look in order to convince the reader that the problem remains $\mathcal{NP}$-hard if we restrict our attention to instances with rather simplistic bids.

Given some number k, we define the decision version of the winner determination problem as to decide whether there exists a feasible allocation of the items with total revenue greater than or equal to k.

Our first reductions make use of the intersection graph formulation of *WDP* from Section 2.2. The decision version of the stable set problem is $\mathcal{NP}$-complete (Garey and Johnson, 1979). Given a graph $G = (V, E)$ we can construct an instance of WDP_{OR} as follows. We set $N = V$, $M = E$ and assume a bid $b_i(S) = 1$ for every $i \in N$ with $S = \{e \in E \mid i \in e\}$. Note that this is also an instance of WDP_{XOR}. It is obvious that G has a stable set of size k if and only if the instance of WDP_{OR} (or, equivalently, WDP_{XOR}) has an allocation with total bid value greater than or equal to k. This proves:

Theorem 3.1 *The decision versions of WDP_{OR} and WDP_{XOR} are $\mathcal{NP}$-complete, even if we restrict to instances where every bid has a value equal to 1, every bidder submits only one bid, and every item is contained in exactly 2 bids.*

The same transformation can be used to prove the complexity of further special cases of WDP_{OR} and WDP_{XOR}. For example, the stable set

problem remains $\mathcal{NP}$-complete if we restrict ourselves to *planar cubic graphs* (Garey and Johnson, 1979). Planar graphs are graphs which can be drawn in the plane such that edges become continuous curves which intersect only in their end nodes. In cubic graphs every edge has exactly degree 3. Using this in the transformation we get $\mathcal{NP}$-hardness of WDP_{OR} and WDP_{XOR} for the case that every bid offers a price of 1, every bid contains exactly 3 items, every item is contained in exactly two bids, and every bidder makes at most one bid.

The general pattern of such proofs is to take an optimization problem as restrictive as possible, but which is still $\mathcal{NP}$-complete and to map instances of that problem to instances of winner determination. Other problems than stable set can be used, too. For example, Rothkopf et al. (1998) use *3-set packing* to show

Theorem 3.2 (Rothkopf et al. 1998) *The decision version of WDP_{OR} is $\mathcal{NP}$-complete even if we restrict to instances where every bid has a value equal to 1, and every bidder bids only on subsets of size of at most 3.*

Similarly, van Hoesel and Müller (2001) transform *3-dimensional matching* to WDP_{XOR} to show:

Theorem 3.3 (van Hoesel and Müller 2001) *The decision version of WDP_{XOR} is $\mathcal{NP}$-complete even if we restrict to instances where every bid*

has a value equal to 1, and every bidder bids only on subsets of size of at most 2.

Let us next assume that the items in M can be linearly ordered into a sequence $(i_1, \ldots, i_m)$, and that bidders bid only on subsets of items that are adjacent in this list. In Müller (Chapter 13) we will see that WDP_{OR} restricted to this case is solvable in polynomial time. However, the problem becomes $\mathcal{NP}$-hard in case of WDP_{XOR}. Indeed, the problem is then identical to the *job interval selection problem*. In this problem we are given a number l, n jobs, and for every job l intervals on a discrete time scale in which this job may be scheduled on a single processor. Viewed as a combinatorial auction we may interpret jobs as bidders who bid for time intervals. A feasible allocation assigns at most one interval to every job, such that intervals do not overlap. Keil (1992) has shown that, even for the case $l = 3$, i.e., three bids per bidder, the problem to decide whether n jobs can be assigned, is $\mathcal{NP}$-complete (for further references on the complexity of this problem see Spieksma (1999)).

So far we restricted the instances by restricting the sets $\mathcal{F}_i$ for which bids are submitted. From an economic perspective it seems to be more natural to restrict bid values, because this better reflects a restriction of bidders' types. A prominent restriction in economics is to assume submodularity, which translates in the context of bids to the following

condition. For any two subsets $S_1, S_2 \subseteq M$ we have

$$v_i(S_1 \cup S_2) \leq v_i(S_1) + v_i(S_2) - v_i(S_1 \cap S_2).$$

A special case of submodular bids is that of additive valuations with budget limits. Additive bids with budget constraint are OR bids for single items, with the additional constraint that the bid for a set S is never larger than the budget. Thus the bid for S is the minimum of the following two values: the sum of bids for individual items in S and the budget. Note that such bids can be represented very efficiently by OR bids for individual items together with the limit q_i. At the same time, it is a special case of WDP_{XOR}, although an explicit representation as an instance of WDP_{XOR} would be very large. The complexity of winner determination for additive valuations with budget limits, and thus for the case of submodular bids is as follows.

Theorem 3.4 (Lehmann et al. 2003; Sandholm and Suri 2001b) *The winner determination problem with two bidders, each submitting additive bids (that is, OR-bids on individual items only) with budget constraints is $\mathcal{NP}$-hard.*

Proof. We present the proof given in Lehmann et al. (2003), where $\mathcal{NP}$-hardness is proven by reducing the knapsack problem to this problem. (The decision version of the knapsack problem is $\mathcal{NP}$-complete (Garey and

Johnson, 1979).) An instance of the knapsack problem is given by a sequence of integers $a_1, \ldots, a_m$ and a desired total t. We want to decide whether there exists a subset T of the integers whose sum is equal to t. Let $A = \sum_{j=1}^{m} a_j$.

Given an instance of knapsack we construct the following bids on m items:

$$\begin{aligned} v_1(S) &= \textstyle\sum_{j \in S} a_j \\ v_2(S) &= 2 \min(t, \textstyle\sum_{j \in S} a_j) \end{aligned}$$

We show that the answer to the decision is "yes" if and only if the winner determination instance has an allocation of value $A + t$.

If the answer is "yes", we can allocate the elements in T to bidder 2, and the elements in the set complement T^c to bidder 1, to obtain an allocation of value

$$\sum_{j \in T^c} a_j + 2t = A - t + 2t = A + t.$$

Assume the constructed winner determination problem has an allocation of value $A + t$. If bidder 2 would get a subset of value strictly less than t, say t', then the total value would be $A - t' + 2t' < A + t$, a contradiction. If bidder 2 would get a subset of value $t' > t$, then the total value would be $A - t' + 2t = A + t - (t' - t) < A + t$, again a contradiction. Thus the items allocated to bidder 2 exactly sum up to $A + t$, and the answer is "yes". ■

Interestingly, with a similar constraint where the *number of items sold to each bidder* is constrained, WDP_{OR} is solvable in polynomial time (see Müller (Chapter 13)). Other restricted versions of WDP_{OR} with certain types of structural side constraints are solvable in polynomial time as well, using b-matching (Penn and Tennenholtz, 2000).

3.3 Complexity for other bidding languages

Nisan (Chapter 9) discusses, next to *OR* and *XOR*, other bidding languages, as to combine OR and XOR. Most of them add *side constraints* to feasible allocations, either from the buyer or from the bid takers side. The bid taker (auctioneer) may have legal constraints, prior contractual obligations, or business rules that he may want to incorporate into the winner determination in the form of side constraints. A practical example is the one where the auctioneer does not want the hassle of dealing with more than a certain number of winners. Also, the budget constraint from Theorem 3.4 is an example.

Sandholm and Suri (2001b) provide a rich collection of results on the impact of side constraints on the complexity of winner determination. For example, it is shown that if at most k winners are allowed, WDP_{OR} is $\mathcal{NP}$-complete even if bids are on indiviudal items only. Also it is shown that if we allow for arbtriray collections of *XOR* constraints, *WDP* is

$\mathcal{NP}$-complete even if bids are on individual items only. This case is a special case of *WDP* for bids submitted in the *XOR-of-OR* bidding language (see Nisan (Chapter 9)). In both examples, the problem remains $\mathcal{NP}$-complete even if bids could be accepted partially.

Specific side constraints (by the bid taker) may induce a combinatorial structure with *favorable* algorithmic properties, as well. For example, Bikhchandani et al. (2001) and Nisan and Ronen (2001) study the setting of a reverse auction, where the auctioneer is interested to purchase a set of edges in a network that form a path from a source s to a target t. Such a path can be thought of as a communication link. Similarly, Bikhchandani et al. (2001) study the case where the buyer wants to purchase a spanning tree. In both examples, bids are on individual items only, side constraints of the bid taker restrict the set of acceptable combinations of winning bids, but a rich combinatorial structure allows for efficient winner determination.

WDP_{OR} is solvable in polynomial time using linear programming if bids can be accepted partially. There are a host of practical side constraints that do not affect this. Examples are the budget constraint, not giving any bidders more than $k\%$ of the total dollar value in the auction, or making minority bidders win at least $k\%$ (Sandholm and Suri, 2001b). On the other hand XOR-constraints and maximum winners constraints make WDP_{OR} $\mathcal{NP}$-complete even if bids can be accepted partially.

Beyond side constraints, in some markets multiple *attributes* (such as reputation, weight, volume, color, delivery time, etc.) should affect the winner determination. Attributes can be integrated into market designs with package bidding and side constraints (Sandholm and Suri, 2001b), but that is not totally straightforward. The preceding discussion has omitted a number of important aspects of the WDP, including combinatorial reverse auctions and two-sided exchanges, removing the free-disposal assumption, and allowing of partial bid fulfilment. The reader is referred to Kothari et al. (2003), Sandholm and Suri (2001b), and Sandholm et al. (2002), for discussion of these issues.

3.4 WDP_{OR} when the number of items is small or the number of bids is huge

While WDP_{OR} is $\mathcal{NP}$-complete, there are algorithms that always—that is, regardless of the number of bids—find an optimal allocation quickly if the number of items m is small. They also run in polynomial time in the size of the input if the number of bids is very large (exponential) compared to the number of items.

For example, WDP_{OR} can be solved by enumerating all partitions of items (Sandholm, 2002), evaluate them, and pick the one of highest value. This yields an $O(m^{m+1})$ algorithm (Sandholm, 2002).

Dynamic programming is a more efficient approach for achieving the

same goals. Rothkopf et al. (1998) suggest the following dynamic programming approach for WDP_{OR}. For all $S \subseteq M$ they compute the best allocation that allocates only elements out of S. This is done iteratively for increasing cardinality of S. For subsets of size 1, they get the information from the bids on single elements. For a subset S with size larger than 1, $w(S)$ is initialized by the largest bid for S, and then updated by the following formula:

$$w(S) = \max(w(S') + w(S \setminus S') \mid S' \subseteq S, |S'| \geq |S|/2)$$

The algorithm runs in $O(3^m)$ time (Rothkopf et al. 1998), It can be improved by using in the recursion only S' such that there exists a bid for S' (see Müller (Chapter 13)).

Interestingly, it constitutes an efficient algorithm (in the sense that it runs in polynomial time in the size of its input) if the number of bids l is very large in comparison to the number of items:

Theorem 3.5 (Sandholm 2002) *Let l be the number of (non-dominated) bids. If the dynamic program for* WDP_{OR} *runs in $O((l+m)^\rho)$ time for some constant $\rho > 1$, then $l \in \Omega(2^{\frac{m}{\rho}})$. If $l \in \Omega(2^{\frac{m}{\rho}})$ for some constant $\rho > 1$, then the dynamic program runs in $O(l^{\rho \log_2 3})$ time.*

We may interpret this result as follows. If the auctioneer receives a number of bids that is very large in comparison to the number of items,

then receiving the bids (and, accordingly, bidding for the bidders) is the computational bottleneck, and not solving the *WDP*. In Müller (Chapter 13) further dynamic programming algorithms are given, in particular for multi-item versions of *WDP*.

4 Approximation

Since the winner determination problem can not be solved in polynomial time to optimality unless $P = NP$ one may hope for efficient algorithms that compute an "almost" optimal solution. An approximation algorithm is a polynomial time algorithm with a provable performance guarantee. More precisely, let $\mathcal{I}$ be the set of instances of a maximization problem, $|I|$ the encoding length of $I \in \mathcal{I}$, $\mathcal{F}_\mathcal{I}$ the set of feasible solutions of $\mathcal{I}$, $c : \mathcal{F}_\mathcal{I} \to \mathbb{R}$ the objective, x^* the optimal solution, and $g : \mathbb{N} \to \mathbb{N}$. We say that an algorithm *approximates* the maximization problem *within* g, if it finds for every instance $I \in \mathcal{I}$ a feasible solution $x \in \mathcal{F}_\mathcal{I}$ with $c(x^*) \leq g(|I|)c(x)$.

4.1 Inapproximability results

From the close relation between the winner determination problem and other combinatorial optimization problems we can derive several impossibility results with respect to approximation. The first was observed by Sandholm (2002) (and independently by Lehmann et al. (2002)) and is

based on the following theorem:

Theorem 4.1 (Håstad 1999) *If $\mathcal{NP} \neq \mathcal{ZPP}$, then for any $\epsilon > 0$ there is no polynomial algorithm that approximates maximum clique for a graph $G = (V, E)$ within $|V|^{1-\epsilon}$.*

Corollary 4.2 (Sandholm 2002) *If $\mathcal{NP} \neq \mathcal{ZPP}$, then for any fixed $\epsilon > 0$ there is no polynomial algorithm that approximates* WDP_{OR} *or* WDP_{XOR} *within* $\min(l^{1-\epsilon}, m^{1/2-\epsilon})$*, where l equals the number of bids and m is the number of items, even when restricted to instances where every item is contained in at most 2 bids, and bid prices are all equal to 1.*

Proof. Using the construction reviewed in Section 2.2, we can first transform the maximum clique problem into a stable set problem by taking the complement of the edge set, then construct an instance of WDP_{OR} (or WDP_{XOR}) where all bid prices are 1. The later has $l = |V|$ bids. A polynomial time $l^{1-\epsilon}$ approximation algorithm for the winner determination problem would thus immediately give rise to a $|V|^{1-\epsilon}$ approximation algorithm for the maximum clique problem, which would contradict Theorem 4.1. For $m^{1/2-\epsilon}$ the proof is similar to Halldórsson et al. (2000) for maximum clique. ■

This negative result generalizes to a specific multi-unit case of WDP_{OR} and WDP_{XOR}.

Theorem 4.3 (Bartal et al. 2003) *For every fixed $k \geq 1$, consider the* WDP_{OR} *(or* WDP_{XOR}*) with $\sigma_i = k$ for all $i \in M$ and every bidder wants to obtain at most* 1 *unit of each type of item. For any fixed $\epsilon > 0$, approximating this multi-unit version of* WDP_{OR} *(or* WDP_{XOR}*) within a factor of $O(m^{\frac{1-\epsilon}{k+1}})$ is $\mathcal{NP}$-hard unless $\mathcal{NP}$=$\mathcal{ZPP}$.*

A polynomial time approximation scheme (PTAS) for a maximization problem is a family of algorithms A_ϵ, $\epsilon > 0$, such that A_ϵ approximates within a factor $1 + \epsilon$, and such that, for fixed ϵ, A_ϵ is polynomial. Berman and Fujito (1999) show:

Theorem 4.4 (Berman and Fujito 1999) *Unless $\mathcal{P} = \mathcal{NP}$ there is no PTAS for the maximum stable set problem on graphs with degree at most 3.*

Again by the same transformation this shows that

Corollary 4.5 *Unless $\mathcal{P} = \mathcal{NP}$ there is no PTAS for* WDP_{OR} *or* WDP_{XOR} *even when restricted to instances with bids of size at most 3, and where every item is contained in at most 2 bids, and bid prices are all equal to 1.*

4.2 Approximation algorithms

Fast approximation algorithms for the *WDP* can be obtained by translating results for combinatorial problems related to winner determination (mainly the weighted set packing problem and the weighted stable set problem) to the context of winner determination (Sandholm 2002). Algorithms for these problems come very close to the bound of the inapproximability results.

The asymptotically best approximation algorithm for WDP_{OR} establishes a bound $O(l/(\log l)^2)$, where l is the number of bids (Halldórsson 2000).

From the same paper, we can establish a polynomial time algorithm for WDP_{OR} with a bound that depends only on the number of items, m. The auctioneer can choose the value for c. As c increases, the bound improves but the running time increases. Specifically, steps 1 and 2 are $O(\binom{l}{c})$ which is $O(l^c)$, step 3 is $O(l)$, step 4 is $O(1)$, step 5 can be naively implemented to be $O(m^2l^2)$, and step 6 is $O(1)$. So, overall the algorithm is $O(\max(l^c, m^2l^2))$, which is polynomial for any given c.

Algorithm 4.6 (Greedy winner determination) *Given an integer c and a set of bids $\{v_i(S) \mid i \in N, S \subset M\}$.*

1. *Let $\mathcal{P}_c$ be the feasible allocations of bids consisting of no more than c bids.*

2. *Let $x_{\mathcal{P}}$ be the optimal alloction within $\mathcal{P}_c$.*

3. *Let $\mathcal{B}_c$ be the subset of bids $v_i(S)$ such that $|S| \leq \sqrt{m/c}$.*

4. *Compute a greedy feasible allocation $x_{\mathcal{B}}$ with respect to bids in $\mathcal{B}$ by sequentially selecting highest value bids which do not overlap with previously selected bids.*

5. *Choose the better of the allocations $x_{\mathcal{P}}$ and $x_{\mathcal{B}}$.*

Another greedy algorithm for winner determination simply inserts bids into the allocation in largest $\frac{v(S)}{\sqrt{|S|}}$ first order (if the bid shares items with another bid that is already in the allocation, the bid is discarded) (Lehmann et al. 2002). This algorithm establishes a bound $\sqrt{m}$. If $c > 4$, the bound that Algorithm 4.6 establishes is better than that of this algorithm ($2\sqrt{m/c} < \sqrt{m}$). On the other hand, the computational complexity of Algorithm 4.6 quickly exceeds that of this algorithm as c grows.

The bound $m^{1/2-\epsilon}$ is so high that it is likely to be of limited value for auctions. Similarly, the bound $2\sqrt{m/c} \geq 2$. Even a bound 2 would mean that the algorithm might only capture 50% of the available revenue. However, the bound that Algorithm 4.6 establishes is about the best that one can obtain, recall Corollary 4.2. If the number of items is small compared to the number of bids ($2\sqrt{m/c} < l$ or $\sqrt{m} < l$), as will probably

be the case in most combinatorial auctions, then these algorithms establish a better bound than that of Corollary 4.2.

One can do even somewhat better in special cases where the bids have special structure. For example, there might be some cap on the number of items per bid, or there might be a cap on the number of bids with which a bid can share items. For a detailed overview on specialized bounds available from the combinatorial optimization literature we refer to the review part in Sandholm (2002), or to the original literature (Chandra and Halldórsson, 1999; Halldórsson and Lau, 1997; Halldórsson, 1998; Hochbaum, 1983). Another family of restrictions that will very likely lend itself to approxiamation stems from limitations on the prices. Two examples will be presented next.

The first algorithm is due to Bartal et al. (2003) and approximates the WDP_{XOR} for a sub-class of the multi-item case. Its approximation ratio is close to the best lower bound possible. The algorithm has to make two assumptions. The first is that bids are given such that valuation and demand oracle queries can be computed efficiently. Recall that a demand oracle for bidder $i's$ bid v_i computes for item prices $p_j, j \in M$ a set

$$S \in \ \arg\max(v_i(S) - \sum_{j \in S} p_j).$$

The second assumption that has to be made says essentially that no bidder

has a demand for a major share of the total number of items available of each type. More precisely we assume that there exists numbers $\Theta > \theta > 0$ such that the following hold. For each multiset S and every j for which $\sigma_j < \theta\omega_j$, $b_i(S) = b_i(S')$, where S' is the multi-set with $\sigma'_k = \sigma_k$ for $k \neq j$, and $\sigma_j = 0$. Similarly, multi-sets with a $\sigma_j > \Theta\omega_j$ have the same valuation as the multi-set in which we reduce the demand for item j to $\Theta\omega_j$. For example, if k copies of each item are in supply, and $\theta = \Theta = \frac{1}{k}$, this models the case that each multi-set for which a bidder makes a bid is an ordinary set, i.e., all $\sigma_j = 1$.

The algorithm is parameterized by a price p_0 and a constant r. It works as follows:

Algorithm 4.7 (Approximate multi-unit winner determination)

For each good j *set* $l_j = 0$.

For each bidder $i = 1 \ldots, n$

for each good j *set* $p_j = p_0 r^{l_j}$

choose $S' = (\sigma'_1, \ldots, \sigma'_m) \in \ arg\max(b_i(S) - \sum_{j \in S} \sigma_j(S) p_j)$.

set bidder $i's$ *payment equal to* $P_i = \sum_{j \in S'} \sigma'_j p_j$.

update $l_j = l_j + \sigma'_j$.

Thus, the basic idea is to increase prices fast enough such that the solution computed by the algorithm is feasible, i.e., numbers of allocated items do

not exceed supply. Bartal et al. (2003) show that for appropriate choices of p_0 (namely $np_0 \leq \frac{Opt}{2}$ and $r^{1-\Theta} \geq v_{\max}\theta p_0$) the solution computed is indeed feasible and achieves an approximation ratio of $2(1 + \frac{r^{\Theta}-1}{\Theta})$. Bartal et al. (2003) show further that with a small modification a sealed bid auction using this algorithm becomes incentive compatible. Furthermore the algorithm can be modified to turn the auction into an incentive compatible online auction, i.e., an auction that can be used when bids arrive over time. Despite these achievements, a combinatorial auction based on this algorithm makes more a theoretical than a practical contribution to the auction literature because of the undesirable property that prices for later bidders are much higher than for earlier bidders, and that in practice an integer linear programming based heuristic does certainly better in terms of the value of the solution computed.

We show next that WDP_{XOR} can be approximated up to a factor of 2 if all $v_i : 2^M \rightarrow \mathbb{R}$ are submodular functions. We use the notation $v_i(j|S) := v_i(S \cup \{j\}) - v_i(S)$ for the marginal bid for item j given subset S is owned alreadyby i. The following algorithm is due to Lehmann et al. (2003).

Algorithm 4.8 (Approx. WDP algorithm for submodular bids)

1. *Set* $S_i = \emptyset$ *for* $i = 1, \ldots, n$, *and* $S = M$.

2. *Let* $(i,j) \in arg\max(v_i(j|S_i)|i \in N, j \in S)$

3. *Set* $S_i = S_i \cup \{j\}, S = S \setminus \{j\}$.

4. *If* $S = \emptyset$ *stop, else go to 2.*

Theorem 4.9 (Lehmann et al. 2003) *The above winner determination algorithm computes a 2-approximation for* WDP_{XOR} *with submodular bids.*

Proof. Our proof is by induction on the number of items. For $m = 1$ the claim is obvious.

For $m > 1$ let us assume w.l.o.g. that item m is assigned to bidder n. The algorithm can be thought of as being recursively invoked on the remaining instance. Bids become:

$$v_i'(S) = \begin{cases} v_i(S) & \text{if } 1 \leq i \leq n-1, \\ v_i(S|\{m\}) & \text{if } i = n. \end{cases}$$

It can easily be shown that v' is again submodular. Let us denote by $A(v)$ the revenue computed by the algorithm, and by $Opt(v)$ the optimal value. By the induction hypothesis we get

$$A(v) = v_n(m) + A(v') \geq v_n(m) + \frac{1}{2} Opt(v').$$

We are done if we can show: $Opt(v') \geq Opt(v) - 2v_n(m)$.

Let $T_1, \ldots, T_n$ be an optimal solution with respect to v. Assume $m \in T_k$. By deleting m from T_k we get a feasible solution T' with respect to

bids v'. We show that this solution differs at most by $2v_n(m)$ from $Opt(v)$, from which the previous inequality follows.

If $k = m$, $Opt(v)$ and the value of solution T' differ at most $v_n(m)$. If $k \neq m$, we get by the definition of Step 2 of the algorithm

$$v'_k(T_k \setminus \{m\}) = v_k(T_k \setminus \{m\}) \geq v_k(T) - v_k(m) \geq v_k(T) - v_n(m),$$

and by the definition of v':

$$v'_n(T_n) = v_n(T_n|\{m\}) = v_n(T_j) - v_n(m).$$

The claim follows now since for all other i it is $v'_i(T_i) = v_i(T_i)$. ■

Put together, considerable work has been done on approximation algorithms for *WDP* and for special cases of combinatorial optimization problems related to *WDP*. However, the worst case guarantees provided by the current algorithms are so far from optimum that they are of limited importance for auctions in practice, even if the approximation factor is 2.

5 Conclusion

The *WDP* is $\mathcal{NP}$-complete and inapproximable. So, is there a useful way to tackle the *WDP*?

Since 1998, there has been a surge of research into addressing this. Three fundamentally different approaches have been pursued:

1. Designing algorithms that are provably fast (polynomial time in the size of the problem instance) but fail to find an optimal solution to some problem instances. Work along this avenue will was briefly reviewed in this chapter in the form of the approximation algorithms. Because the problem is inapproximable, such algorithms yield solutions that are extremely far from optimal (yielding less than 1% of the available revenue) in some cases. Approximation compromises economic value and generally also ruins the incentive properties of the auction. One can think of local search (and stochastic local search, e.g. Hoos and Boutilier (2000)) algorithms as falling within this approach, too, except that they generally do not provide any guarantees on solution quality or run time.

2. Restricting the bundles on which bids can be submitted, or the bid prices, so severely that the problem can be solved optimally and provably fast. This approach is discussed in Müller (Chapter 13) of the book. While computationally attractive, this approach may suffer from similar economic inefficiencies, incentive problems, and exposure problems as noncombinatorial auctions because the bidders cannot fully express their preferences. In fact, they can only bid on a vanishingly small fraction of possible bundles: the number of

allowable bundles has to be polynomial in m, while the total number of bundles is exponential in m (specifically, 2^m). Imposed restrictions by the auctioneer should therefore always be motivated by bidders' valuations.

3. Designing *tree search algorithms* that provably find an optimal solution. This approach is discussed in Sandholm (Chapter 14) of the book. Because the problem is $\mathcal{NP}$-complete, any optimal algorithm for the problem will be slow on some problem instances (unless $\mathcal{P} = \mathcal{NP}$). Even so, such algorithms can be fast in practice. Furthermore, most of them are *anytime algorithms*: they can be terminated early (if they happen to be taking too long) with usually a good feasible solution in hand. To this class of approaches belongs also using commercial integer linear programming solvers. Andersson et al. (2000) have investigated how well such solvers scale. Similarly, we may employ algorithms for the multi-dimensional (multiple-choice) knapsack problem, as suggested by Holte (2001). However, dedicated solvers generally improve significantly on general-purpose commercial packages (see Sandholm (Chapter 14)).

References

Andersson, Arne, Mathias Tenhunen and Fredrik Ygge (2000), "Integer programming for combinatorial auction winner determination," in *Proc. Fourth International Conference on Multi-Agent Systems*, 39–46.

Bartal, Yair, Rica Gonen and Noam Nisan (2003), "Incentive compatible multi unit combinatorial auctions," in M Tennenholtz (ed.), *Proceedings of the 9th Conference on Theoretical Aspects of Rationality and Knowledge*, ACM, 72–87.

Berman, Piotr and Toshihiro Fujito (1999), "On approximation properties of the independent set problem for low degree graphs," *Theory of Computing Systems*, 32, 115–132.

Bikhchandani, Sushil, Sven de Vries, James Schummer and Rakesh R. Vohra (2001), "Linear programming and Vickrey auctions," in Brenda Dietrich and Rakesh R. Vohra (eds.) *Mathematics of the Internet - E-Auctions and Markets*, Springer, 75–117.

Chandra, Barun and Magnús M. Halldórsson (1999), "Greedy local search and weighted set packing approximation," in *Proc. of 10th Annual SIAM-ACM Symposium on Discrete Algorithms (SODA)*, 169–176.

Cook, Stephen A. (1971), "The complexity of theorem proving procedures,"

in *Proc. 3rd Annual ACM Symp. on Theory of Computing*, ACM Press, 151–158.

Garey, Michael R. and David S. Johnson (1979), *Computers and Intractability*, W. H. Freeman and Company.

Halldórsson, Magnús M. (1998), "Approximations of independent sets in graphs," in Klaus Jansen and José D. P. Rolim (eds.), *Proceedings of The First International Workshop on Approximation Algorithms for Combinatorial Optimization Problems (APPROX)*, LNCS Vol. 1444, Springer, 1–14.

Halldórsson, Magnús M. (2000), "Approximations of weighted independent set and hereditary subset problems," *J*ournal of Graph Algorithms and Applications, 4(1), 1–16. Early versions appeared in *Computing and Combinatorics, Proceedings of the 5th Annual International Conference (COCOON)*, Tokyo, Japan, 1999, and in LNCS Vol. 1627, Berlin: Springer, 1999, 261–270.

Halldórsson, Magnús M. and Jan Kratochvíl and Arne Telle (2000), "Independent sets with domination constraints," *Discrete Applied Mathematics*, 99, 39-54. Also appeared in *Proceedings of the 25th International Conference on Automata, Languages, and Programming (ICALP)*, LNCS Vol. 1443, Berlin, Springer, 1998.

Halldórsson, Magnús M. and Hong Chuin Lau (1997), "Low-degree graph partitioning via local search with applications to constraint satisfaction, max cut, and 3-coloring," *Journal of Graph Algorithms and Applications*, 1, 1–13.

Håstad, Johan (1999), "Clique is hard to approximate within $n^{1-\epsilon}$," *Acta Mathematica*, 182, 105–142.

Hochbaum, Dorit S. (1983), "Efficient bounds for the stable set, vertex cover, and set packing problems," *Discrete Applied Mathematics*, 6, 243–254.

Hoesel, Stan van, and Rudolf Müller (2001), "Optimization in electronic markets: examples in combinatorial auctions," *Netnomics*, 3, 23–33.

Holte, Robert C. (2001), "Combinatorial Auctions, Knapsack Problems, and Hill-climbing search," in Eleni Stroulia, Stan Matwin (eds.), *Advances in Artificial Intelligence, Proceedings of 14th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, AI 2001, LNCS Vol. 2056, Springer, 57–66.

Hoos, Holger and Craig Boutilier (2000), "Solving combinatorial auctions using stochastic local search," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 22–29.

Keil, J. Mark (1992), "On the complexity of scheduling tasks with discrete starting times," *Operations Research Letters*, 12, 293–295.

Kellerer, Hans, Ulrich Pferchy and David Pirsinger (2004), *Knapsack Problems*, Springer Verlag.

Kothari, Anshul, Tuomas Sandholm and Subhash Suri (2003), "Solving combinatorial exchanges: Optimality via a few partial bids," in *Proceedings of the ACM Conference on Electronic Commerce (ACM-EC)*, 236–237. Early version in *Proc. AAAI-02 workshop on Artificial Intelligence for Business.*

Kuhn, Harold W. (1955), "The Hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, 2, 83–97.

Lehmann, Daniel, Benny Lehmann and Noam Nisan (2003), "Combinatorial auctions with decreasing marginal utilities," *Games and Economic Behavior*, to appear.

Lehmann, Daniel, Liadan Ita O'Callaghan and Yoav Shoham (2002), "Truth revelation in approximately efficient combinatorial auctions," *Journal of the ACM*, 49, 577–602.

Nisan, Noam and Amir Ronen (2001), "Algorithmic mechanism design," *Games and Economic Behavior*, 35, 166–196.

Penn, Michal and Moshe Tennenholtz (2000), "Constrained multi-object auctions and b-matching," *Information Processing Letters*, 75, 29–34.

Rothkopf, Michael H., Aleksander Pekeč and Ronald M. Harstad (1998), "Computationally manageable combinational auctions," *Management Science*, 44, 1131–1147.

Sandholm, Tuomas (2002), "Algorithms for optimal winner determination in combinatorial auctions," *Artificial Intelligence*, 135:1–54. Earlier version in *Proc. Intl. Joint Conf. on Artificial Intelligence (IJCAI)*, 542–547, 1999.

Sandholm, Tuomas and Subhash Suri (2001), "Side constraints and non-price attributes in markets," in *Proc. of IJCAI-2001 Workshop on Distributed Constraint Reasoning*, 55–61.

Sandholm, Tuomas, Subhash Suri, Andrew Gilpin and David Levine (2001), "Winner determination in combinatorial auction generalizations," in *Proc. International Conference on Autonomous Agents and Multi-Agent Systems*, 69–76. Early version in *AGENTS-01 Workshop on Agent-Based Approaches to B2B*, 35–41, 2001.

Spieksma, Frits C. R. (1999), "On the approximability of an interval scheduling problem," *Journal of Scheduling*, 2, 215–227.