Matrix Bidding in Combinatorial Auctions

Robert W. Day
Operations and Information Management
School of Business
University of Connecticut
Storrs, CT 06269-1041

S. Raghavan
Decision and Information Technologies
The Robert H. Smith School of Business
University of Maryland
College Park, MD 20742-1815

Abstract

In a combinational auction in which bidders can bid on any combination of goods, bid data can be of exponential size. We describe an innovative new combinatorial auction format in which bidders submit "matrix bids". The advantage of this approach is that it provides bidders a mechanism to compactly express bids on every possible bundle. We describe many different types of preferences that can be modeled using a matrix bid, which is quite flexible, supporting additive, subadditive, and superadditive preferences simultaneously. To utilize the compactness of the matrix bid format in a more general preference environment, we describe a logical language with matrix bids as "atoms", and show that matrix bids compactly express preferences that require an exponential number of atoms in other bidding languages, and is as expressive as the most sophisticated languages in the literature, (Boutilier, 2002). We model the $\mathcal{NP}$-hard winner-determination problem as a polynomially-sized integer program, specifically an assignment problem with side constraints. We show the strength of this formulation with which we rapidly solve winner determination problems with 72 unique items, indicating that this model may be well suited for practical implementation.

1 Introduction

Combinatorial Auctions (auctions in which bidders may express bids on combinations or bundles of goods) are motivated by the presence of synergy in bidders' valuations for sets of goods. The synergy associated with a set of goods S and bidder j may be defined as $\sigma_j(S)$, where:

$$\sigma_j(S) = v_j(S) - \sum_{i \in S} v_j(\{i\})$$

and $v_j(S)$ is bidder j's value for the set of goods S.

The presence of synergistic valuations impedes the generalization of the single-good auctions to efficient multiple-good auctions. When synergy is positive there are complementarity effects; the bidder would like to tell the auctioneer, "I would give you more money if I could be guaranteed to get these goods together." When synergy is negative, substitution effects dominate; the bidder would say, "Here are my prices, but I want to pay less if I get goods that are substitutes." Preferences for a bundle may be referred to as *subadditive*, *additive*, or *superadditive* when synergy is negative, zero, or positive, respectively.

Clearly the auctioneer benefits from taking positive synergy effects into consideration; bidders will promise to pay extra if they are guaranteed certain combinations of goods. Though it is less immediately clear, the

auctioneer may benefit if he is willing to consider negative synergy information as well. This is because a bidder will be more willing to bid up to his true value on small bundles if the risk of paying too much for a combined bundle is reduced or eliminated. In addition to the possibility of increased revenue for the auctioneer, taking negative synergies into account can increase the economic efficiency of an auction, a feature desirable in governmental auctions of electricity, radio-spectrum, oil-drilling rights, etc. which constitute an area of major interest in the auction literature.

In a combinatorial auction, the auctioneer collects bids $b_j(S)$ from each bidder j on potentially any subset S of the items in the auction. In an *efficient* combinatorial auction, the auctioneer then solves a combinatorial optimization problem, the winner-determination problem, which finds an allocation of items to bidders that maximizes the total value of accepted bids.[1] For the most general context, this may be modeled as an Integer Program (IP), related to the set-packing problem, and described, for example, by de Vries and Vohra (2003). This *General Winner-Determination* problem (GWD) for the allocation of N items in the set $I = \{1, 2, ...N\}$ among M bidders in the set $J = \{1, 2, ...M\}$ can be formulated as follows, with binary variables $x_j(S)$ that equal 1 if and only if bidder j is awarded bundle $S \subseteq I$:

$$\max \sum_{j \in J} \sum_{S \subseteq I} b_j(S) \cdot x_j(S) \tag{GWD}$$

$$\text{subject to} \quad \sum_{S \supseteq \{i\}} \sum_{j \in J} x_j(S) \leq 1, \qquad \forall i \in I \tag{1}$$

$$\sum_{S \subseteq I} x_j(S) \leq 1, \qquad \forall j \in J \tag{2}$$

$$x_j(S) \in \{0, 1\}, \qquad \forall S \subseteq I, \forall j \in J \tag{3}$$

Constraint set (1) ensures that each item is assigned to at most one bidder, while constraint set (2) prevents the auctioneer from accepting multiple bids from the same bidder, preventing the auctioneer from recombining multiple bids to get a different bid on a subset than the one submitted by the bidder.

In addition to the $\mathcal{NP}$-hardness of the general winner-determination problem (see, for example Rothkopf et al., 1998), bidders in a combinatorial auction experience a fundamental difficulty expressing their preferences over an exponential number of bundles; a bid can be submitted for each of the $2^N - 1$ nonempty bundles in an auction of N distinct items. Some attempts made in the literature to alleviate these problems fall into a category we refer to collectively as the *restricted-subset* methods (Rothkopf et al., 1998, provide several examples). Such methods restrict the number of subsets that may receive a positive bid from a given bidder, and very often limit bidding behavior by disallowing any bids on certain subsets.

Eschewing this approach to the "exponential bundles problem", we introduce a new submission format

[1] The argument for the use of an efficient mechanism is provided by Ausubel and Cramton (1999). In this paper, we restrict our attention to expressive languages and the solution of (efficient) winner-determination problems. Though we do not discuss strategic properties in the present text, we recommend the use of a "core" pricing mechanism. As described in Day and Raghavan (2007), such a mechanism relies heavily on our ability to rapidly solve several winner-determination problems, further justifying our focus on efficiency based on submitted bids.

for submitting preferences in an auction for several unique items, referred to as "matrix bidding." Unlike restricted-subset approaches to preference submission, matrix bidding allows for positive bidding on any subset of interest to the bidder, while simultaneously allowing for a positive incremental offer on any bundle. Stated differently, a matrix bid can make positive bids on every bundle simultaneously, such that the bid on any bundle is greater than the bid on any strict subset of that bundle. The "bid statements" which can be expressed with matrix bids have the ability to model preferences requiring an exponential number of atoms in a "flat-bid" language (see, for example Nisan, 2000), and offer an alternative bidding paradigm to the other logical languages which address this problem (see, for example, Boutilier, 2002). While a single matrix bid restricts the precision with which prices are expressed on every bundle, we show that a logical language of matrix bids is fully expressive.

The rest of the paper is organized as follows. Before moving ahead, §1.1 provides a brief guide to the literature. In §2 we discuss the basics of matrix bids and motivate how they can be used with a few examples. In §3 and §4 we provide a deeper exploration of the preference expression afforded by the matrix bid format and compare the logical language of matrix bids to other logical languages from the literature. In §5, we provide an IP formulation of the winner determination problem for a Matrix Bid Auction (MBA) and show that in general this problem is $\mathcal{NP}$-hard. Also, we discuss the properties of the formulation that make it particularly strong, allowing for rapid winner-determination. In §6 we describe our computational experience solving simulated MBAs using our IP formulation. These experiments verify a fundamental strength of matrix bids, that they allow us to conduct auctions that would require too much time or memory using a set-packing approach.

1.1 Related Literature

de Vries and Vohra (2003) provide a compact survey of recent combinatorial auction research, while Cramton et al. (2006) provide a more in depth collection of combinatorial auction essays. Included in the latter, Lehmann et al. (2006) discuss the complexity issues of winner determination, and Sandholm (2006) discusses winner-determination algorithms. Other prominent articles on winner-determination methods include Günlük et al. (2005), and Sandholm et al. (2005), based on XOR-of-OR and XOR logical languages of flat bids, respectively.

Rothkopf et al. (1998) provide a seminal paper in the study of combinatorial auctions, including some of the first detailed complexity results and the first discussion of "restricted-subset" combinatorial auctions for which winner determination is tractable. More recently, Müller (2006) provides a survey of tractable winner-determination instances based on both subset restrictions and preference-type restrictions (e.g. preferences satisfying the "substitutes property.") Relative to the combinatorial auction formats discussed there, the matrix bid format enforces a preference restriction that is not strong enough to result in a tractable instance, but is strong enough to drastically reduce the size of formulation relative to an unrestricted approach.

As used elsewhere in the literature, we use the term *flat-bid*, denoted (S, p), for an all-or-nothing bid of p

monetary units on the set of items S. Nisan (2000) provides the seminal study of flat-bid logical languages in which flat bids are joined with logical connectives (AND, OR, XOR, etc.) to form more complex expressions of preferences. As noted by Nisan (2000), certain natural expressions of preference can require exponentially long bid sentences in a logical language with only flat-bids as atoms[2]. The work of Boutilier and Hoos (Boutilier, 2002; Boutilier and Hoos, 2001; Hoos and Boutilier, 2000) explore the possibility of expanding the bid language to include a "k-of" operator, handling one of the most natural expressions that is tiresome to convey in flat-bids by simply adding the expression in explicitly as a new atom. The k-of operator allows a bidder to offer a price premium if at least k elements from a list of other bids (clauses) is satisfied, allowing the bidder to specify, "I want 10 out of the following 15 items for $100," for example. At first, this addition is not very rewarding in terms of the impact on winner-determination, because the authors simply expand a k-of expression back into flat-bids prior to solving the winner-determination problem. But Boutilier (2002) later shows how to explicitly model the k-of operator in an IP formulation of winner-determination with a new class of constraints to handle the new operator. A more recent summary on bidding languages is provided by Nisan (2006).

In §3 and §4 we will use many of these concepts in describing a logical language of matrix bids. The notation $\mathcal{L}_{GB}$ will be used for the bidding language described by Boutilier (2002), in which bidders form bids by joining goods or bundle bids together with logical connectives, including the k-of operator. This bidding language also uses a colon and a number at any point in the sentence to denote a bid amount if the corresponding clause is satisfied. In the notation from Boutilier (2002), $A : 22$ denotes, for example, a bid of 22 on item or expression A. Also, one may use A by itself in place of $A : 0$, and can in general omit ": 0" from any expression without confusion.

Additionally, let the symbols $\wedge$, $\vee$, and $\oplus$, denote the standard logical operators AND, OR, and XOR, respectively, with their typical interpretations. These symbols are used to join together clauses to form sentences in the language, where each clause is either a singleton bid like $A : 22$, or, recursively, any other sentence in the language. Thus, $(RedBike : 40 \vee RedHat : 10) \oplus (BlueBike : 40 \vee BlueHat : 10)$ would be submitted by a bidder bidding $40 on a $RedBike$ or $10 on a $RedHat$ (or $50 for both), $40 on a $BlueBike$ or $10 on a BlueHat (or $50 for both), with an assurance that she will not get both a red item and a blue item, by virtue of the exclusive $\oplus$.

Formally, $\mathcal{L}_{GB}$ is defined as the set of sentences obtained recursively as follows: for any set of items S and any monetary amount p, the flat-bid $(S : p)$ is a sentence in $\mathcal{L}_{GB}$. Further, if S_1 and S_2 are sentences in $\mathcal{L}_{GB}$, then $S_1 \wedge S_2 : p$, $S_1 \vee S_2 : p$, and $S_1 \oplus S_2$ are in $\mathcal{L}_{GB}$ for any monetary amount p. $S_1 \wedge S_2 : p$ bids p monetary units in addition to the amounts bid by S_1 and S_2 if both subclauses are satisfied. Similarly, $S_1 \vee S_2 : p$ bids p additional units if one or both of the subclauses are satisfied, while $S_1 \oplus S_2$ states that only bids from one subclause may be satisfied (i.e., the auctioneer may only collect bids from one subclause.) Finally for any collection of sentences S_1, S_2, ... S_n in $\mathcal{L}_{GB}$, the sentence $\langle k, \{S_1, S_2, ...S_n\}, p\rangle$ is in $\mathcal{L}_{GB}$

[2]In logic, *atoms* refer to the terms joined by logical connectives at the most basic level. Clauses are made by joining atoms with connectives, larger clauses are built by joining those clauses with connectives, etc.

for any monetary amount p and positive integer $k \leq n$. This notation represents the k-of operator and is interpreted as an additional bid of p units if at least k-of the n subclauses are satisfied. For a more detailed explanation see Boutilier (2002).

In the current work, we introduce matrix bids, each of which bids on several bundles simultaneously, like the previously described sentences in the language $\mathcal{L}_{GB}$. In fact, one strength of matrix bids is that each one places a bid on every subset, with the potential for a positive marginal bid for any item received. Because of this dense expression of preferences in a single matrix bid, we demonstrate how to use matrix bids as an alternative atomic unit for a logical bidding language, with no loss of compact expressability relative to $\mathcal{L}_{GB}$, the most compactly expressive of the languages in the literature.

2 Matrix Bids

A bidder in this model specifies her preferences with a value for each item in each of its possible rankings in the final bundle. The bid offered for item i by bidder j given that it is the kth best item she receives will be denoted b_{ijk}. Bidder j's bid on a bundle S may then be computed as:

$$b_j(S) = \sum_{i \in S} b_{ijK(i,S)}$$

where $K(i, S)$ gives the ordinal ranking of item i among the items in S. For example, $K(\bar{\imath}, S) = 1$ if no item in S has a higher rank than item $\bar{\imath}$. Similarly, $K(\bar{\imath}, S) = 2$ if exactly one item in bundle S has a higher rank than item $\bar{\imath}$, etc. A bidder interprets each matrix bid entry as an incremental bid on an item; the row indicates which item is bid on, while the column tells the ranking of the item within the bundle it brings value to. The matrix bid itself is interpreted as a collection of bids on any possible subset, each bid equal to the sum of incremental values for the items. The preference restriction imposed in the matrix bid model is therefore that the value an item brings to a bidder is determined *only* by its ranking relative to the other items in the bundle. Although this may at first seem restrictive, we will show that when used effectively a great variety of preferences can be expressed by each bidder.

Together with the b_{ijk} entries, each bidder j in the matrix bid format must convey a numerical ranking r_{ij} for each of the N items at auction. We say that $r_{ij} = 1$ if item i is the first (highest ranked) item in bidder j's ordering of the items, $r_{ij} = 2$ if item i is the second item in bidder j's ordering, etc. We require that this ranking be strict and total for each bidder j: if $i_1 \neq i_2$, then $r_{i_1 j} \neq r_{i_2 j}$, and $\forall n \leq N$, $\exists i \in I$ with $r_{ij} = n$. It is important to note that the rankings specified by the values of r_{ij} are intended only to interpret the matrix bid and do not necessarily reflect any preference ordering among the items for that bidder. Stated differently, $r_{i_1 j} < r_{i_2 j} \Rightarrow v_j(\{i_1\}) \geq v_j(\{i_2\})$ is not always the case.

Thus each bidder submits an ordered list of the items to establish the values of r_{ij} and a matrix containing non-negative values of b_{ijk} (for simplicity we assume integer values throughout). The matrix of b_{ijk} entries

together with the precedence ordering r_{ij} is referred to as a *matrix bid*. In each matrix bid a row will be associated with a particular item, and each column will be associated with a particular ranking. The item associated with a given row is written to the left of that row, with the row of the highest ranked item on top, the second highest item beneath it, etc. Since the kth item in the ordering can be at most the kth item in any bundle, the matrix of b_{ijk} values will be lower triangular.

The use of matrices to hold bid information suggests that the preferences of each bidder may be represented by an assignment network, and indeed it was this interpretation that initially motivated the matrix bid format. As such it may be useful for the reader to consider this network interpretation in order to build intuition and to connect the current exposition to earlier research on unit-demand agents (for example, Demange et al., 1986).

In the network interpretation, a matrix bid may be viewed as representing each bidder by a collection of N "unit-demand" agents, each of which can potentially consume a single item. This suggests an assignment network, with N origin nodes (one for each item) and MN destination nodes (one for each of the bidders' agents). Each agent is represented by a particular column of a matrix bid, and we can therefore refer to agents and columns interchangeably. Thus the ordering specified by a bidder determines the order in which the agents may receive items. Specifically, an agent may not receive an item unless the next higher ranked agent receives a higher ranked item.

For each agent k representing bidder j (denoted as agent (j,k)), define the binary acceptance variable $x_{ijk} = 1$ if agent (j,k) receives item i, and $x_{ijk} = 0$ otherwise. With this terminology defined, we may now state the *cooperation restriction* among a particular bidder's agents as:

$$\forall k > 1,\ x_{ijk} = 1 \Rightarrow x_{\bar{\imath}jk-1} = 1 \text{ for some } \bar{\imath} \text{ with } r_{\bar{\imath}j} < r_{ij}$$

This says that any agent (other than agent 1) may not receive an item unless the next more highly ranked agent receives a more highly ranked item.

The following simple rules summarize how to interpret a matrix bid:

- When an item is awarded to a bidder, the auctioneer receives a single bid from the corresponding row in that bidder's matrix bid.
- Only a single bid may be taken from any column
- Except for bid entries in the first column, a bid may not be used unless a bid in the previous column and a higher row is also used.

Example 1: Suppose a bidder is submitting preferences for the following entertainment choices on a specific date: a ticket to the afternoon baseball game, a coupon for dinner at a nearby restaurant, a day-pass to a water-park (outside of town), and a ticket to a matinee at the local theatre. The bidder reasons that the matinee and baseball game conflict; she cannot go to both, but can make it to dinner after either one. She

decides that if she gets any of the other items she will not leave town to go to the water-park. Her matrix bid may appear as follows:

baseball	40			
matinee	10	0		
dinner	25	25	25	
water-park	30	0	0	0

The order r_{ij} is given in the outside column with baseball being ranked first, the matinee second and so forth. The first column inside the matrix (always) gives the bid on the item in that row if it is the highest ranked (or only) item received. The second column gives the price for each item in the row given that it is the second highest item received etc.

If she receives a baseball ticket she is willing to pay 0 for the matinee which she cannot attend due to conflict. If she receives either baseball or matinee in the first column she would being willing to pay 25 for the meal (in the second column). Although the seller is unlikely to give away the matinee ticket, the mathematical formulation does not necessarily rule this out. If the auctioneer gives her the baseball ticket at 40 and the matinee ticket at 0, she is still willing to pay 25 for the dinner, and expresses this with a 25 in the third column; a free matinee ticket does not change her preferences for the dinner. The fourth row shows that she would pay 30 for the water-park pass by itself, but would pay 0 for it if any other items are won. The bid of 30 cannot be accepted with any other bid by the rules outlined above.

Example 2: A major television network is hosting a worldwide television event with a limited number of available advertising time-slots; for simplicity we assume there are four time-slots, A, B, C, and D, chronologically. They attract companies X, Y, and Z who have very different marketing strategies determining their preferences for these time-slots.

BidderX has developed a two-part "gag" commercial which they will only use if they can secure exactly two time-slots. They are indifferent to which two. *BidderY* feels that slots B and C, occurring in the middle of the program are more effective than A and D at the beginning and end of the program when less viewers are watching. The effectiveness of their ads diminishes with repeat viewing so they are uninterested in purchasing more than two slots. *BidderZ* agrees that slots B and C are superior to A and D. Their hypnotic ad campaign will show increased effectiveness with each viewing, as the increasing values in each row of their matrix bid submission suggest.

The matrix bids these companies submit are:

Bidder X

A	0			
B	0	30		
C	0	30	0	
D	0	30	0	0

Bidder Y

B	20			
C	20	6		
D	10	4	0	
A	10	4	0	0

Bidder Z

B	7			
C	6	7		
A	5	6	7	
D	4	5	6	7

These examples show the ability of the matrix bid format to express several types of preferences[3]. The

[3]Bidder X and Bidder Y provide examples of Nested k-of bidding and Diminshing Returns bidding, respectively, used in

first example shows the ability to model a relationship in which certain items preclude one another while others do not. In the second example the items are thought of as substitutes by Bidder Y, complements by Bidder Z, and somewhere in between by Bidder X who shows preference for a specific quantity[4]. With these examples as motivation, we now look at a detailed discussion of preference expression in §3 and §4.

3 Preference Expression using Matrix Bids

In this section we explore how matrix bids may be used to convey preferences. We suggest the use of matrix bids as "atomic bids" to be combined with logical connectives, offering a new notational system for writing down preferences that are difficult to express with flat-bids alone. We show that a language with matrix bids as atoms is more expressive in a polynomial amount of input than a language with just flat-bids as atoms, and equally as expressive as a flat-bid language augmented with k-of or exactly-k-of bids.

Relative to other logical bidding languages we show that *more* can be said with a single atom, and that some things can be said *easily* in matrix bids that are cumbersome in other languages. We also hope to convince the reader that with some experience matrix bids are an easier to read format than the equivalent collection of bids needed to express the same preferences in a language with k-of and exactly-k-of operators applied to flat-bids only. Matrix bids provide a structured way to string together several k-of, exactly-k-of, and flat-bid sentence fragments to be read in a meaningful way after deleting many of the logical connectives. In §6, we will see that this allows us to quickly generate simulated auction data (with no logical connectives) which would have required exponentially long bid sentences in any of the flat-bid auction simulations commonly found in recent literature (for example Günlük et al., 2005; Leyton-Brown et al., 2002; Sandholm et al., 2005).

We follow in the spirit of Boutilier (2002) who confronts the difficulty of expressing the k-of preferences in flat-bid languages. Are there other natural preference types that are difficult to handle in a logical language of flat-bids? Should the list of connectives be augmented to accept other forms of preferences like the k-of operator? One example of another natural connective is the "exactly k-of" operator, where the k-of operator discussed elsewhere could be called more precisely the "at least k-of" operator (though we will continue to refer to this one as the k-of operator). It is not difficult to show that the exactly k-of operator can be handled in an IP winner-determination formulation with specialized constraints as the k-of operator is handled in Boutilier (2002).

Adding more and more basic connectives and new classes of constraints seem to complicate the matter. Reading and writing bids in such a language becomes difficult (for a human user) with many connectives and complicated nestings to consider when putting together an entire bid sentence or a complete profile of preferences. Is there instead an atomic format which can accept a wide variety of preferences including in

our simulations and described in detail in Appendix B.

[4]The optimal value for the winner-determination problem in this second example is 57. Bidder X receives items A and D, contributing 30 to the objective function. Bidder Y receives item C for 20 units, while Bidder Z pays 7 units for B.

particular those that are difficult to express in flat-bids alone? We demonstrate that matrix bids offer such a format and that a logical language of matrix bids can be made at least as expressive in a polynomial amount of input space as any of the other languages in the literature. This latter statement is justified by comparison to the language $\mathcal{L}_{GB}$ of Boutilier (2002), which in turn is at least as expressive as any language that precedes it. In particular, let $\mathcal{L}^{flat}$ denote the language $\mathcal{L}_{GB}$ of Boutilier (2002) without the k-of operator, and $\mathcal{L}_{MB}$ denote the language formed with matrix bids as atoms using the same logical connectives as in $\mathcal{L}_{GB}$. We will show that the preferences which can be expressed compactly in $\mathcal{L}^{flat}$ are properly contained in $\mathcal{L}_{MB}$, while the preferences which can be expressed compactly in $\mathcal{L}_{MB}$ are exactly equal to the preferences which can be expressed compactly in $\mathcal{L}_{GB}$.

3.1 Capacity Constraints and Capacity Costs

We begin with an illustrative demonstration of one of the strengths of matrix bids relative to the language $\mathcal{L}^{flat}$, which is composed of flat bids joined by the logical connectives AND, OR, and XOR. We show how simple capacity-constraints are naturally expressed in a matrix bid, and compare with the equivalent formulation in $\mathcal{L}^{flat}$.

Suppose we have an auction with $N = 6$ and a bidder wants to express an additive valuation over the items with a constraint that he cannot consume more than 3 items. A matrix bid expressing this (with arbitrary values given for each item) is as follows:

A	22					
B	18	18				
C	17	17	17			
D	16	16	16	0		
E	14	14	14	0	0	
F	12	12	12	0	0	0

With this bid, he pays the same cost for each item regardless of what other items he gets, but will never pay a positive amount for a fourth, fifth, or sixth item, reflected by the zeros in the fourth, fifth, and sixth columns. For a comparison of size and readability, consider the previous matrix bid expressed in the $\mathcal{L}^{flat}$ language and notation of Boutilier (2002):

$$(A:22 \vee B:18 \vee C:17) \oplus (A:22 \vee B:18 \vee D:16) \oplus (A:22 \vee B:18 \vee E:14) \oplus$$
$$(A:22 \vee B:18 \vee F:12) \oplus (A:22 \vee C:17 \vee D:16) \oplus (A:22 \vee C:17 \vee E:14) \oplus$$
$$(A:22 \vee C:17 \vee F:12) \oplus (A:22 \vee D:16 \vee E:14) \oplus (A:22 \vee D:16 \vee F:12) \oplus$$
$$(A:22 \vee E:14 \vee F:12) \oplus (B:18 \vee C:17 \vee D:16) \oplus (B:18 \vee C:17 \vee E:14) \oplus$$
$$(B:18 \vee C:17 \vee F:12) \oplus (B:18 \vee D:16 \vee E:14) \oplus (B:18 \vee D:16 \vee F:12) \oplus$$
$$(B:18 \vee E:14 \vee F:12) \oplus (C:17 \vee D:16 \vee E:14) \oplus (C:17 \vee D:16 \vee F:12) \oplus$$
$$(C:17 \vee E:14 \vee F:12) \oplus (D:16 \vee E:14 \vee F:12)$$

If this example does not yet convince the reader of exponential growth for this type of preference expression in $\mathcal{L}^{flat}$, note that if we were to add another item to the auction and maintain the capacity constraint of three items, we would have to add 15 new XORed clauses to the above statement of preferences, while

only adding 7 new numbers (one new row) to the matrix bid. In general, additive preferences for n items with a capacity constraint of k takes $\binom{n}{k}$ clauses, each containing k atomic bids in the language of $\mathcal{L}^{flat}$, or a single matrix bid of size $\binom{n}{2} = \frac{n(n-1)}{2}$, verifying that $\mathcal{L}_{MB}$ can contain preferences in a single matrix bid that require a sentence of exponential length in $\mathcal{L}^{flat}$.

This example demonstrates how to apply a capacity constraint using a matrix bid and is not limited to a situation where the underlying preferences are additive. Though we used an additive structure in this example for simplicity, one could start within any matrix bid expression and "zero out" columns that exceed capacity. Further, matrix bids easily handle a capacity cost structure in place of a hard capacity constraint. Imagine a firm that experiences an underlying additive structure (again for simplicity) for items received (for example, in terms of potential revenue generated), but then wishes to capture storage and maintenance costs based on the number of items in their possession.

Suppose, for example, that in a 6 item auction they experience storage and maintenance costs of 0 for the first two items, but have to employ a new worker (at a cost of 3 units) to maintain every two items they purchase after that, and further must rent a storage area at 1 unit each for the fifth and sixth items. If their additive revenue structure is captured in the following matrix bid,

A	22					
B	18	18				
C	17	17	17			
D	16	16	16	16		
E	14	14	14	14	14	
F	12	12	12	12	12	12

then they can subtract the entries of the following row vector from each entry in the corresponding column to capture the storage and maintenance costs, (0 0 3 0 4 1). Thus if revenue accrues additively with the values from the above matrix bid, and costs are as described above, the preferences of this firm are captured in the following matrix bid:

A	22					
B	18	18				
C	17	17	14			
D	16	16	13	16		
E	14	14	11	14	10	
F	12	12	9	12	8	11

Thus, they have compactly submitted a bid with positive value for each of the $2^N - 1$ possible non-empty bundles, while $\mathcal{L}^{flat}$ cannot express the generalized version of these preferences compactly. To be certain, note that the information conveyed in the first three columns requires at least as much space in a $\mathcal{L}^{flat}$ sentence as in the previous example, and that both examples easily generalize due to the underlying additive preference structure.

Note also that with this example the bidder has conveyed a positive incremental value for any item when added to any bundle that does not contain it. This allows for a more aggressive bidding strategy than is

possible (compactly) in the most general language, XOR-of-flat-bids, where the statement "I will pay an incremental amount p if item a is added to any of my previously submitted flat-bids" potentially requires the bidder to double the number of submitted flat-bids.

3.2 Characterization of $\mathcal{L}_{MB}$

Having established that $\mathcal{L}_{MB}$ affords expression not available in a polynomially sentence in $\mathcal{L}^{flat}$, we proceed to show that a language with matrix bids as atoms, $\mathcal{L}_{MB}$, is equally as expressive as the language $\mathcal{L}_{GB}$ of Boutilier (2002) in an amount of input that is polynomially sized in the number of items. To do this, we must show that an arbitrary $\mathcal{L}_{GB}$ bid sentence can be expressed compactly in $\mathcal{L}_{MB}$, and that an arbitrary $\mathcal{L}_{MB}$ bid sentence may be expressed compactly in $\mathcal{L}_{GB}$. The first statement follows by noting that an arbitrary flat-bid $(S, p$ can be held compactly in a matrix bid that places the items of S to the top of the matrix bid ordering with an entry of p placed in the $|S|$th diagonal entry, and all other entries equal to 0. We admit however that this doesn't make good use of the space in a matrix bid. We do, however, present techniques for injecting $\mathcal{L}_{GB}$ preferences into matrix bids more efficiently in §4. For the reverse statement, we need only show that an arbitrary matrix bid can be expressed in $\mathcal{L}_{GB}$. We accomplish this first in the language $\mathcal{L}^{+}_{GB}$ which we form by adding the "exactly k-of" operator to $\mathcal{L}_{GB}$. It turns out the translation of a matrix bid into $\mathcal{L}^{+}_{GB}$ is slightly more readable than the equivalent expression given in $\mathcal{L}_{GB}$.

To write these expressions more succinctly, let $\langle 2, \{A:2, B:4, C:8\}, 3\rangle$ denote, for example, the k-of bid of 3 monetary units if at least 2 of the bids $A:2$, $B:4$, or $C:8$ are satisfied, with A, B, C receiving a bid of 17. Similarly, let $[2, \{A:2, B:4, C:8\}, 3]$ denote the exactly-k-of bid of 3 monetary units if exactly 2 of the subclauses $A:2$, $B:4$, or $C:8$ are satisfied; in this case, the set A, B, C receives a bid of 14.

Given a general matrix bid from a four-item auction:

$$\left|\begin{array}{llll} A & a_1 & & \\ B & b_1 & b_2 & \\ C & c_1 & c_2 & c_3 \\ D & d_1 & d_2 & d_3 & d_4 \end{array}\right|$$

an equivalent bid in the language $\mathcal{L}^{+}_{GB}$ is given by:

$$\begin{array}{l} (A:a_1) \\ \vee \\ (B \wedge [0,\{A\},0]:b_1) \oplus (B \wedge [1,\{A\},0]:b_2) \\ \vee \\ (C \wedge [0,\{A,B\},0]:c_1) \oplus (C \wedge [1,\{A,B\},0]:c_2) \oplus (C \wedge [2,\{A,B\},0]:c_3) \\ \vee \\ (D \wedge [0,\{A,B,C\},0]:d_1) \oplus (D \wedge [1,\{A,B,C\},0]:d_2) \oplus \\ (D \wedge [2,\{A,B,C\},0]:d_3) \oplus (D \wedge [3,\{A,B,C\},0]:d_4) \end{array}$$

where we remind the reader that, for example, B on its own is equivalent to $B:0$. We note that a few minor truncations are possible in this expression (for example, $(D \wedge [3,\{A,B,C\},0]) : d_4$ is equivalent to

$A \wedge B \wedge C \wedge D : d_1$) but that no major truncations are possible (since each entry is used exactly once). We therefore leave it in this form which illustrates that any matrix bid can be expressed polynomially within $\mathcal{L}^+_{GB}$ as $\bigvee_{i:r_{ij}=1}^{i:r_{ij}=n} \bigoplus_{k=1}^{r_{ij}} (i \wedge [k-1, S_i, 0] : b_{ijk})$, where $S_i = \{\bar{\imath} \in I | r_{\bar{\imath}j} < r_{ij}\}$ and b_{ijk} are the matrix bid entries. To show the equivalence in polynomially expressability between $\mathcal{L}_{GB}$ and $\mathcal{L}_{MB}$, we simply note that $[k, S, p]$ is equivalent to $\langle k, S, p\rangle \wedge \langle k+1, S, *\rangle$, where we use $*$ to signify a sufficiently large negative value, making $\mathcal{L}^+_{GB}$ and $\mathcal{L}_{GB}$ equivalent up to a polynomial transformation. (The use of $*$ will be discussed in the next section; it may be interpreted as a negative number whose absolute value is larger than the sum of all positive bids in the auction.)

4 Using the Matrix Bid Language Efficiently

The characterization of $\mathcal{L}_{MB}$ as equivalent to $\mathcal{L}_{GB}$ in terms of polynomial expressability suggests that matrix bids may be thought of as a notational compactification of preferences that could before only be expressed with several k-of expressions. In particular, our capacity constraint example above showed that a matrix bid readily houses several k-of expressions, while simultaneously allowing for price differentiation among particular items to be expressed in the rows of each item. As noted above, however, our claim that any $\mathcal{L}_{GB}$ sentence could be expressed in a polynomial size in $\mathcal{L}_{MB}$ made poor use of the space afforded to us by a matrix bid. We now show a few techniques for injecting several general forms of preferences into a single matrix bid, demonstrating more efficient use of the freedom allowed by the matrix bid format. We begin by showing that an arbitrary AND, OR, or XOR clause of singleton bids can be contained in a single matrix bid, as can an arbitrary k-of-singletons bid or flat-bid.

Recall that in the matrix bid language, a flat-bid (S, p) can be written:

$$\begin{array}{c|ccccc|} i_1 & 0 & & & & \\ i_2 & 0 & 0 & & & \\ \vdots & \vdots & & \ddots & & \\ i_{|S|} & 0 & 0 & \cdots & p & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \\ \end{array}$$

where $i_1, i_2, ...i_{|S|}$ are the items in S written in any order, and the ellipses in this example represent regions containing only zeros. This expression represents a very inflexible offer, paid only if the exact package S is received.

A more flexible offer is demonstrated by the arbitrary OR bid, $(i_1 : p_1 \vee i_2 : p_2 \vee ... \vee i_n : p_n) : b$. This bid offers p_l for item i_l taken in any combination with other items, and offers a bonus b if any one of the items is awarded (i.e. if any of the singleton subclauses is satisfied). This OR bid is captured in the matrix bid:

$$\begin{array}{c|cccccc|} i_1 & p_1+b & & & & & \\ i_2 & p_2+b & p_2 & & & & \\ \vdots & \vdots & & \ddots & & & \\ i_n & p_n+b & p_n & \cdots & p_n & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{array}$$

Here the items $i_1, i_2, ...i_n$ in the OR statement are placed above (in higher ranked rows) than items not in the statement, while again the ordering among the items in the statement is inconsequential. We form this matrix bid by starting with an additive valuation (a p_l placed in every entry of the row for each i_l), and add the bonus amount b to the first entry in each of these rows. This ensures that the bonus must be awarded if at least one item in $i_1, i_2, ...i_n$ is received, and is only awarded once. The rows below i_n contain the items $I \backslash \{i_1, i_2, ...i_n\}$ and are filled with zeros. (Such rows of zeros need not be submitted in practice.)

Modeling the arbitrary AND of singleton bids combines the previous two approaches. The $\mathcal{L}_{GB}$ sentence $(i_1 : p_1 \wedge i_2 : p_2 \wedge ... \wedge i_n : p_n) : b$ offers an additive valuation over single items $i_1, i_2, ...i_n$, with a bonus amount b offered only if every item in $\{i_1, i_2, ...i_n\}$ is received. We therefore begin with constant rows to express the additive valuation, and add the bonus amount b to the last element on the diagonal:

$$\begin{array}{c|cccccc|} i_1 & p_1 & & & & & \\ i_2 & p_2 & p_2 & & & & \\ \vdots & \vdots & & \ddots & & & \\ i_n & p_n & p_n & \cdots & p_n+b & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{array}$$

where again the order of the items is inconsequential, as long as all the items not belonging to the AND statement are placed in zero rows below those that are in the statement.

The arbitrary XOR of n singleton bids $(i_1 : p_1 \oplus i_2 : p_2 \oplus ... \oplus i_n : p_n)$ makes an offer of p_l on each singleton set $\{i_l\}$, but places no bid on any bundle containing more than one item. This is captured in the following matrix bid, in which we only place positive amounts (specifically the bid p_l for the item in that row) in the first column:

$$\begin{array}{c|cccccc|} i_1 & p_1 & & & & & \\ i_2 & p_2 & 0 & & & & \\ \vdots & \vdots & & \ddots & & & \\ i_n & p_n & 0 & \cdots & 0 & & \\ \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \end{array}$$

where as always, items not involved in the bid are placed at the bottom of the matrix bid with zero rows.

To model the k-of preference $\langle k, S, p \rangle$, a bid of p on at least k items from the set S, consider the matrix bid:

$$
\begin{array}{c|cccccccc}
i_1 & 0 & & & k\text{th} & \text{col} & & & \\
i_2 & 0 & \ddots & & \downarrow & & & & \\
\vdots & \vdots & & \ddots & & & & & \\
i_k & \vdots & & & p & & & & \\
\vdots & \vdots & & & p & \ddots & & & \\
\vdots & \vdots & & & \vdots & \vdots & \ddots & & \\
i_{|S|} & 0 & \cdots & & p & \cdots & \cdots & \ddots & \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots &
\end{array}
$$

where the only non-zero entries are the values of p in the kth column in rows that contain elements of S.

We next note that several ORed k-of bids for the same set S can be contained in the same matrix bid with "constant column" bids as illustrated in the previous example. For example, if the bids $\langle 2, \{a, b, c, d, e, f\}, p_2\rangle$, $\langle 4, \{a, b, c, d, e, f\}, p_4\rangle$, and $\langle 5, \{a, b, c, d, e, f\}, p_5\rangle$ are to be submitted with an OR relationship, we would form the matrix bid:

$$
\begin{array}{c|cccccc}
a & 0 & & & & & \\
b & 0 & p_2 & & & & \\
c & 0 & p_2 & 0 & & & \\
d & 0 & p_2 & 0 & p_4 & & \\
e & 0 & p_2 & 0 & p_4 & p_5 & \\
f & 0 & p_2 & 0 & p_4 & p_5 & 0
\end{array}
$$

If at least two items are received the amount p_2 is promised to the auctioneer, if at least four items are received an additional amount p_4 is promised, etc.

Just as several k-of bids can be nested in a single matrix bid, we now show that a chain of exclusive (i.e., XORed) flat-bids $(S_1, p_1), (S_2, p_2), \ldots (S_n, p_n)$ with $S_1 \subset S_2 \ldots \subset S_n$ can be nested along the diagonal as in the following matrix bid:

$$
\begin{array}{c|cccccccc}
S_1 & 0 & & & & & & & \\
\downarrow & \vdots & p_1 & & & & & & \\
S_2 & & 0 & & & & & & \\
\downarrow & & & p_2 - p_1 & & & & & \\
\vdots & & & & \ddots & & & & \\
S_n & & & & & 0 & & & \\
\downarrow & & & & & & p_n - p_{n-1} & & \\
\vdots & \vdots & & & & & & \ddots &
\end{array}
$$

Here we have ordered the items so that all the items in S_1 are ranked higher than all the items in S_2, which are all ranked higher than the items in S_3, etc. We place the bid of p_1 for the set S_1 on the diagonal in the row of the last (lowest ranked) element in S_1, and a bid of $p_2 - p_1$ in the row of the last element in set S_2. Similarly, we put a bid of $p_l - p_{l-1}$ on the diagonal entry of the row with the last element of S_l. It is easily verified that this matrix bid yields the appropriate expression of preference, as assigning the last item in S_l

causes the total accepted bid to reach p_l.

4.1 Expressing a Set of Bids as a Single Matrix Bid

We first examine the question of expressing a set of preferences (bids) within a single matrix bid. A naive approach would be to simply go through the $N!$ possible orderings and check whether the ordering can express the 2^N preferences in a single matrix bid. This approach takes $O(N!2^N)$ time. A better approach, while still exponential in its running time (but polynomial in the size of the $O(2^N)$ input), is based upon checking the key condition on preference restriction imposed in the matrix bid model. That is, b_{ijk}, the incremental value for item i when added to any of the $\binom{r_{ij}-1}{k-1}$ possible bundles[5] containing $k-1$ more highly ranked items, is constant. In other words

$$b_{ijk} = b_j(S \cup i) - b_j(S) \qquad \forall S \in SB(i,j,k) \tag{4}$$

where $SB(i,j,k) = \{S \subset I | r_{\bar{\imath}j} < r_{ij}, \ \forall \bar{\imath} \in S, \text{ and } |S| = k-1\}$. We describe two approaches to check this condition. The first is a column based approach, working to fill the matrix bid entries column by column from left to right. The second is a row based approach, working to fill the matrix bid entries row by row from bottom to top. Both approaches either provide an ordering of items or indicate that it is not possible to express the preference as a matrix bid.

In the column based approach we work from left to right keeping track for each item its highest possible position in the ordering. In particular we use a position index pos[i] for each item i. In the first column every bid is a bid on exactly one item. Thus for the first column (without any checking) we can set pos[i] to N for all items. Next, in the second (and later columns) we work from bottom to top using a position index t that goes from N down to k (the column index). We check for position index t and column k whether the value that item i brings to the $\binom{t-1}{k-1}$ possible bundles of items with position index t or lower is constant. If true this indicates that pos[i] can remain at value t. Otherwise, the value of pos[i] is reduced. At each iteration, the procedure also checks whether the number of items with pos[i] $\geq t$ is less than $N - t + 1$. If so, we cannot express the preference with a matrix bid, and the procedure stops. The column based procedure is described in Figure 1. The running time of the procedure is easily computed by noting that in each position of the matrix bid the procedure considers $t \times \binom{t-1}{k-1}$ sets. Since the sum of all entries in Pascal's triangle is 2^N, and t is bounded above by N the running time is $\mathcal{O}(N2^N)$. We note that the column-based procedure makes use of the following property that we prove in the online appendix.

Claim 4.1. *Suppose while considering column k and position index t there are l items that satisfy condition (4) (only considering items with position index t or lower). Then, we can immediately lower the position index of items with pos[i] value strictly between t and $t-l$ to $t-l$ (i.e., there can be no items in position*

[5]We note the interesting property that the number of bundles corresponds exactly to the term in the same position in Pascal's triangle (see Weisstein, 2005).

```
For i = 1 to N set pos[i] = N
For k = 2 to N − 1 {
    t = N;
    Repeat {
        count = 0;
        For each i with pos[i] = t{
            Is b_j(S ∪ i) − b_j(S) = constant for all S that contain
                k − 1 items (not including i) with position index ≤ t?
            If yes, count = count + 1;
            If no, pos[i] = t − 1;
        }
        If ((# of items with pos[i] ≥ t) < N − t + 1) STOP;
        For i = 1 to N
            If ((pos[i] < t)AND(pos[i] > t−count))
                set pos[i] = t−count;
        t = t− count;
    } until (t ≤ k);
}
```

Figure 1: Column-based algorithm to determine whether a preference set can be represented in a single matrix bid.

index strictly between values $t - l$ and t).

Proof. See online appendix. □

The row based approach (independently suggested by Goossens, 2006) works similarly to the column-based approach except that condition (4) is checked row by row (working from left to right in a row) and bottom to top. Initially, each item has pos$[i] = N$. We check for position index t and for each $k = 2$ to t, whether the value that item i brings to the $\binom{t-1}{k-1}$ possible bundles of items with position index t or lower is constant (the constant can be different for different k). If true this indicates that pos$[i]$ can remain at value t. Otherwise, the value of pos$[i]$ is reduced. The row-based procedure is described in Figure 2 (it also makes use of Claim 4.1). The running time of the procedure is computed identically as the column-based procedure by noting that in each position of the matrix bid the procedure considers $t \times \binom{t-1}{k-1}$ sets. Thus, the running time is $\mathcal{O}(N2^N)$.

4.2 Information Loss in a Single Matrix Bid

Because of its widespread use in simulations and general treatments of combinatorial auctions, it is interesting to consider the relationship of matrix bidding to an XOR-of-flat-bids language. In particular, we are interested in the following question: if a bidder cannot express his/her preference in a single matrix bid, is there a way to fill in matrix bid entries that makes safe (i.e. does not expose the bidder to paying too much for a bundle) but effective (i.e. is not dominated by another safe value) use of the space afforded by the matrix bid format?

```
For i = 1 to N set pos[i] = N
t = N;
Repeat {
    count = 0;
    For each i with pos[i] = t{
        cond=true;
        For k = 2 to t − 1 {
            While (cond=true) do {
                Is b_j(S ∪ i) − b_j(S) = constant for all S that contain
                    k − 1 items (not including i) with position index ≤ t?
                If no, set pos[i] = t − 1 and cond=false;
            }
        }
        If (cond=true) set count = count + 1;
    }
    If ((# of items with pos[i] ≥ t) < N − t + 1) STOP;
    For i = 1 to N
        If ((pos[i] < t)AND(pos[i] > t−count))
            set pos[i] = t−count;
    t = t− count;
} until (t ≤ 2);
```

Figure 2: Row-based algorithm to determine whether a preference set can be represented in a single matrix bid.

Suppose, for example, that bidder j can quickly compute her desired bid on any bundle, $b_j(S)$, and is filling out a matrix bid to convey her preferences. For a given ordering, observing that every bid on the diagonal and every bid in the first column are bids on exactly one subset (shown in the previous matrix bid example and the XOR of singletons example, respectively) she may simply fill these entries in. For example, in a four item auction we have the following matrix bid, where we abbreviate, for example, the set $\{A, B, C\}$ by ABC:

$$\left|\begin{array}{llll} A & b_j(A) & & \\ B & b_j(B) & b_j(AB) - b_j(A) & \\ C & b_j(C) & x & b_j(ABC) - b_j(AB) \\ D & b_j(D) & y & z \qquad\qquad b_j(ABCD) - b_j(ABC) \end{array}\right|$$

Our question from the previous paragraph now becomes: is there a way to choose entries x, y, and z that is safe and effective? The matrix bid paradigm makes simultaneous bids on several sets by assuming that the incremental value an item provides to a bundle does not vary over all bundles containing only higher ranked items. If in fact the incremental value an item provides to a bundle does vary over bundles containing only higher ranked items, the bid information in the interior entries (i.e. those not in the first column or on the diagonal) is not exact. These entries can be employed in a safe and effective way, however, by using the *minimum possible incremental value* for that entry. For this example we have:

$$x = \min(b_j(AC) - b_j(A), b_j(BC) - b_j(B))$$
$$y = \min(b_j(AD) - b_j(A), b_j(BD) - b_j(B), b_j(CD) - b_j(C))$$
$$z = \min(b_j(ABD) - b_j(AB), b_j(ACD) - b_j(AC), b_j(BCD) - b_j(BC))$$

Filling in the corresponding matrix bid entries with these values assures that the bidder has bid a safe amount, but could not have bid a higher safe amount. Again note the connection between the number of terms in each minimization and the term in the same position in Pascal's triangle, thus providing a characterization of the accuracy of each entry. This emphasizes that bid accuracy is exact in the left column and diagonal, and is least accurate in the middle of the bottom row. In general, this technique of finding a safe value for an arbitrary entry is given by $b_{ijk}^{safe} = \min_{S \in SB(i,j,k)} b_j(S \cup i) - b_j(S)$.

This "safe bid" technique allows a bidder to safely "pack in" more bid information into a matrix bid with a particular ranking or ordering of items already chosen. Consequently, if one wanted to choose an ordering that maximizes some measure of bid accuracy (or minimizes a measure of error) then in general the bidder must optimize over the $N!$ possible orderings, a problem that in general seems to require a great deal of enumeration. On the other hand the column-based and row-based approach to determine whether a preference can be expressed as a matrix bid can easily be adapted as heuristics to approximate a general bid function by a matrix bid.

For example, in a column-based approximation, when the column-based approach cannot find any item i with $\text{pos}[i] = t$ (at a given t and k) that satisfies condition (4), we can select amongst all items considered for the given t and k. We choose the item that minimizes the difference between the maximum and minimum value that the item brings to the bundles considered. This item is given position index t as an approximation and the procedure continues (ties are broken arbitrarily).

On the other hand, in a row-based approximation, when the row-based approach cannot find any item i with $\text{pos}[i] = t$ (at a given t) that satisfies condition (4), we can select amongst all items considered for the given t. Observe that in the row-based procedure we check an entire row at a time to determine if an item can be given that row as its position index. Consequently, a simple heuristic is to choose the item that minimizes the sum of the differences between the maximum and minimum values that the item brings to each of the bundles considered in columns $k = 2$ to t respectively (ties are broken arbitrarily). Goossens (2006) describes this procedure in greater detail.

4.3 Contingency Expression Using $*$-entries

We now demonstrate another technique for preference expression with matrix bids that allows for a contingency relationship among "blocks" in the matrix. We will see that this allows for another form of "compactification" in which two or more "basic" expressions of preference can be combined in the same matrix bid, rather than spreading these basic pieces into separate matrix bids and joining them with logical connectives.

Further, the ability to express this contingency relationship in a matrix bid allows us to omit a contingency operator from our logical language.

Observe that if a particular entry in a matrix bid is assigned a significantly negative number, then a revenue maximizing auctioneer will never assign the item in that row to the agent represented by that column. Representing this large negative number as $*$, we can block off regions of a matrix bid forcing entries to only be accepted in a certain way. To demonstrate the use of $*$ in a matrix bid, consider a bidder who wishes to express a contingency relationship between two "basic" valuations which can each be written in its own matrix bid. Suppose, for example, that the bidder is interested in purchasing a set of three items $\{A, B, C\}$ as a package at a price of 40, and would like to express an additive valuation over items $\{D, E, F\}$ contingent upon winning the essential set $\{A, B, C\}$. Using one block for the flat-bid portion of these preferences and an additive block for the other, this bidder may place the following matrix bid:

A	0					
B	0	0				
C	0	0	40			
D	$*$	$*$	$*$	16		
E	$*$	$*$	$*$	14	14	
F	$*$	$*$	$*$	12	12	12

Note, for example, that the set $\{B, C, D, E\}$ cannot be awarded to this bidder at a price of 14 (as it could if the $*$s were replaced with zeros) because in pricing this bundle a $*$-entry is accepted, representing a penalty great enough to overcome any generated benefit.

This ability to block-off "basic" expressions of preference quickly generalizes: any expressions that can be made in a separate matrix bid can be combined with a contingency relationship, as long as the subsets of items effectively bid on in each matrix bid is disjoint (where the items effectively bid on are those for which there is a positive entry in its row or below). A simple and powerful example of this is given by the following type of preferences which we call a *grocery-list bidding*. In this scenario a bidder perceives some list of necessary *ingredients* each of which must be obtained in a certain quantity and each of which has a list of substitute items.

For example, suppose the manager of a pizza company is purchasing ingredients at a wholesale auction (suppose perhaps that each item is a truckload of some ingredient). He may divide the set of auctioned items into categories of items, each containing several substitutable alternatives: doughs, sauces, cheeses, and toppings. Suppose further that his preferences are as follows. (i) He needs to obtain exactly one of the three types of dough, two of the three types of cheese, and one of the three types of sauce, or else he is not willing to purchase anything. (ii) Further, he perceives different quality levels for the various sauces, making him willing to pay 8 additional monetary units if his bundle contains sauce B, and 12 units if his bundle contains sauce C. (iii) Finally, he has additive preferences over three toppings, given that he has a bundle that satisfies his dough, cheese and sauce requirements. His preferences may be compactly expressed in the matrix bid Figure 3.

Figure 3: A grocery-list Matrix Bid

dough A	0						
dough B	0	$*$					
dough C	0	$*$	$*$				
cheese A	$*$	0	$*$	$*$			
cheese B	$*$	0	0	$*$	$*$		
cheese C	$*$	0	0	$*$	$*$	$*$	
sauce A	$*$	$*$	$*$	40	$*$	$*$	$*$
sauce B	$*$	$*$	$*$	48	$*$	$*$	$*$
sauce C	$*$	$*$	$*$	52	$*$	$*$	$*$
topping A	$*$	$*$	$*$	$*$	7	$*$	$*$
topping B	$*$	$*$	$*$	$*$	6	6	$*$
topping C	$*$	$*$	$*$	$*$	5	5	5

Applying the basic assumption for matrix bids that matrix bid entries may only be accepted in a down and to the right fashion, it is easy to see that the bid of Figure 3 expresses no positive bid except on bundles that contain exactly three doughs, two cheeses, and one sauce, and that topping preferences are additive thereafter. The preferences expressed by the bidder in Figure 3 also provides a first example of a Partition Bidder as used in our simulations and described in Appendix B.

4.4 Logical Language of Matrix Bids

Up to this point, this section has focused on expression of preferences within a single matrix bid. Matrix bidding is, however, a restrictive format and we emphasize that in practice a logical language of matrix bids may be required to convey preferences accurately. Though the previous section demonstrates a technique for joining basic preference expressions within a single matrix bid, many relationships can only be expressed by combining several matrix bids with logical connectives, as the following telecommunications example demonstrates.

Suppose we are auctioning city-wide licenses for mobile communication rights. One company may submit the two bids in Figure 4 for California cities. The metropolitan areas of $SanFrancisco$, $Oakland$ and $SanJose$ are close enough geographically that it would be impractical to purchase rights to one city without the other two; Bid 1 offers a "single-minded" bid of 50 for all three, but otherwise make no offers on any cities. Bid 2 introduces a different type of behavior. The bidding company is most concerned with purchasing the rights to $L.A.$, but feels that they will experience increased returns for each of the neighboring cities they obtain. Furthermore, they cannot support a communications network in the smaller neighboring cities without at least controlling the hub, $L.A.$ They bid 30 on $L.A.$ by itself, 35 on $L.A.$ with any one of its neighbors, 45 on $L.A.$ with any two of its neighbors, and 60 on $L.A.$ with all three of its neighbors. In general, if a bidder considers one item to be essential, with a set of n other items as substitutable accessories or add-ons, she may make bids on the distinguished item with 0 accessories, 1 accessory, . . . n accessories

Figure 4: Matrix Bids for spectrum licenses

Bid 1

SanJose	0			
Oakland	0	0		
San Fran.	0	0	50	

Bid 2

Pasadena	0			
LongBeach	0	0		
Anaheim	0	0	0	
L.A.	30	35	45	60

by bidding with an increasing row (we call such a bidder an *add-on bidder* in our simulations, see Appendix B).

If the two bids in this example are treated as if the single bidder had two bidders acting on his behalf, the submission is interpreted as "Bid 1 OR Bid 2". The OR is inclusive; the company submitting these bids may receive a subset which is the union of subsets obtained by each of its matrix bids. It may be the case, however, that the company is only interested in purchasing the rights for one of the two regions; they wish to bid "Bid 1 XOR Bid 2". In Appendix A, we demonstrate how to model both OR-of-XOR-of-Matrix-Bid and XOR-of-OR-of-Matrix-Bid languages within our IP formulation of the general matrix bidding format (which will now be presented in §5).

5 Formulation of the Winner-Determination Problem

The winner-determination problem, in this context, is to find the maximum revenue assignment of items to columns, satisfying supply of one for each item and demand of at most one for each column, as well as upholding the ordering or ranking of items. According to the ranking, a feasible assignment may assign an item to a given column only if the columns for a particular bidder with a lower value of k are assigned items with a lower value of r_{ij}. These constraints are formalized in the following integer program which we will refer to as MBA-IP:

$$\max \sum_{i \in I} \sum_{j \in J} \sum_{k \leq r_{ij}} b_{ijk} \cdot x_{ijk} \tag{MBA-IP}$$

$$\text{subject to} \quad \sum_{j \in J} \sum_{k \leq r_{ij}} x_{ijk} \leq 1 \text{ for each } i \in I \tag{5}$$

$$\sum_{i \in I | r_{ij} \geq k} x_{ijk} \leq 1 \text{ for each } (j,k) \in J \times K \tag{6}$$

$$\sum_{l \in I | k \leq r_{lj} \leq r_{ij}} x_{ljk} - \sum_{l \in I | k-1 \leq r_{lj} < r_{ij}} x_{ljk-1} \leq 0 \text{ for each } (i,j,k) \text{ with } k > 1 \tag{7}$$

$$x_{ijk} \in \{0,1\}, \quad \forall i,j,k$$

The decision variable x_{ijk} is a binary acceptance variable; $x_{ijk} = 1$ if the bid b_{ijk} is accepted, while $x_{ijk} = 0$ if it is not. Constraint set (5) is a set of supply constraints for each item, while demand constraints (6) assure that at most one item is received by each column. Constraints (7) enforce the ordering; $x_{ijk} = 0$ unless some $x_{lj(k-1)} = 1$ where $r_{lj} < r_{ij}$. This is not the only way to formulate this set of constraints and not the most obvious; these inequalities are chosen to eliminate as many fractional solutions as possible in the LP relaxation, while enforcing the ordering by including more than is necessary in the first summation. To see that this formulation of the ordering constraints is superior to the "natural" version (i.e. $x_{ijk} - \sum_{l \in I | k-1 \leq r_{lj} < r_{ij}} x_{ljk-1} \leq 0$ for each (i, j, k) with $k > 1$) observe that in a three item auction for items A, B, and C with $r_{A1} = 1$, $r_{B1} = 2$, and $r_{C1} = 3$ it is possible to have a feasible fractional solution to the LP relaxation of MBA-IP with $x_{A11} = x_{B12} = x_{C12} = \frac{1}{2}$ using the "natural" constraints while the constraints (7) do not admit such a solution.

Constraint sets (5) and (6) correspond to assignment constraints. Thus, without constraints (7) that enforce the ordering, the LP relaxation of constraints (5) and (6) is integral and the problem is polynomially solvable. Adding constraints (7) greatly complicates the matter, however. In particular, it is not hard to find examples in which the LP-relaxation of MBA-IP has a fractional optimal solution. Indeed, we show in Theorem 5.1 that the winner-determination problem for an MBA is $\mathcal{NP}$-hard, while it follows from the Padberg and Alevras (1994) study[6] of "order preserving assignment problems" that a single matrix bid may be evaluated in polynomial time.

We now verify one strength of the formulation MBA-IP: it is polynomially-sized in the number of items and bidders, unlike the general winner-determination problem (GWD) which requires an exponential number of variables, one for every bidder/bundle pair. This suggests that for some values of N a GWD auction cannot be performed due to memory restrictions, while an MBA may still be held for the same set of N items. To verify this, we note that each bidder must submit $\sum_{l=1}^{N} l = \frac{N(N+1)}{2}$ bid entries and an ordered list of N items, $\frac{1}{2}N^2 + \frac{3}{2}N$ pieces of information. The total number of x variables is equal to the total number of matrix bid entries $M \cdot \frac{N(N+1)}{2}$. Counting the constraints similarly according to their indices we find that there are N of type (5), MN of type (6), and $M \cdot \frac{N(N-1)}{2}$ of type (7). Thus, MBA-IP has $\frac{1}{2}N^2M + \frac{1}{2}NM$ variables and $\frac{1}{2}N^2M + \frac{1}{2}NM + N$ constraints. Our computational results (§6) suggest that for reasonable N and M, MBA-IP instances of this size can be solved quickly enough for auction implementation purposes, despite the following theorem.

Theorem 5.1. *The winner-determination problem for a matrix bid auction is $\mathcal{NP}$-hard.*

Proof. We employ the $\mathcal{NP}$-hard Weighted Set-Packing Problem (WSP): Given a set S, and a list of subsets of S: S_1, S_2, ... S_p, each with a corresponding weight w_1, w_2, ... w_p, problem WSP asks for the maximum weight collection of mutually-disjoint S_is.

The transformation of this general instance of WSP into a matrix bid auction proceeds in a straightforward

[6]Padberg and Alevras (1994) study the order preserving assignment problem, which corresponds exactly to an instance of MBA-IP with only a single bidder, and potentially less than N columns.

manner. Identify S from the WSP with I and create p bidders, one for each S_i. Each bidder will submit a matrix bid of w_p on the set S_p (recall the compact matrix bid representation of a flat bid of w_p on a set of items S_p is illustrated on page 12.) An optimal solution to the winner-determination problem for this auction MBA-IP provides a maximum weight set-packing of S_1, S_2, ... S_p, and vice-versa. □

Despite this classification of the decision problem as $\mathcal{NP}$-hard, our experience tells us that the formulation MBA-IP is fairly well-behaved. (In §6 we provide additional computational evidence.) Recall that with the assignment constraints (5) and (6) the feasible region has integer extreme points. With the addition of constraints (7) many of these extreme points are no longer feasible and fractional extreme points may be introduced. Exactly how many of each type are generated or eliminated by (7) varies among instances (even of the same size) depending on the orderings chosen by bidders. Every integer solution to MBA-IP, however, is also a feasible assignment and thus an extreme point of the region defined without (7), partially explaining the proliferation of LP-relaxation solutions that are integer optimal as seen in our experiments. A stronger explanation is suggested by the following lemma.

Lemma 5.2. *For any bidder j, if $\sum_k x_{ijk} \in \{0,1\}$ for all items $i \in I$, then $x_{ijk} \in \{0,1\}$ for all i and k.*

Proof. The implication is trivial for all i such that $\sum_k x_{ijk} = 0$, so we consider the case in which $\sum_k x_{ijk} = 1$.

Let $i_1 = \arg\min_{I_j} r_{ij}$, where $I_j = \{i | \sum_k x_{ijk} = 1\}$ (i.e. I_j is the set of all items awarded to j). Similarly, let $i_2 = \arg\min_{I_j \setminus \{i_1\}} r_{ij}$, $i_3 = \arg\min_{I_j \setminus \{i_1, i_2\}}$ and so forth, yielding in general that i_n is the nth ranked item in I_j.

First note that since there is no item l with $r_{lj} < r_{i_1 j}$ and $x_{ljk} \neq 0$, constraints (7) for i_1 imply that $x_{i_1 jk} \leq 0$ for all $k > 1$, and therefore $x_{i_1 jk} = 0$ for all $k > 1$. Since $\sum_k x_{i_1 jk} = 1$, it must be the case that $x_{i_1 j1} = 1$. By (6) it also follows that $x_{ij1} = 0$ for all $i \neq i_1$.

Next, we claim that if $x_{i_n jn} = 1$ for all $n < \overline{k}$, then $x_{i_{\overline{k}} j \overline{k}} = 1$. This is easily shown since $x_{i_n jn} = 1$ for all $n < \overline{k}$ implies that:

$$x_{ijk} = 0 \text{ for all } k \geq \overline{k} \text{ and } i \text{ with } r_{ij} < r_{i_{\overline{k}} j} \tag{8}$$

$$x_{i_{\overline{k}} jk} = 0 \text{ for all } k < \overline{k} \tag{9}$$

The first of these two facts (8), together with (7), implies that $x_{i_{\overline{k}} jk} = 0$ for all $k > \overline{k}$. Using this with (9) and $\sum_k x_{i_{\overline{k}} jk} = 1$ we obtain the desired result, $x_{i_{\overline{k}} j \overline{k}} = 1$. The lemma now follows by induction. □

Lemma 5.2 states roughly that if a bidder j receives integer amounts of all items (among all her columns), the constraints of MBA-IP dictate that the assignment of items to bidder j's collection of columns must be integral. Alternatively, items may not be split fractionally among the columns of a single bidder j, unless there is some item which j shares fractionally with a different bidder.

The polynomial-solvability of the order-preserving assignment problem (see Padberg and Alevras, 1994) now follows as a special case of Lemma 5.2, which implies that the polyhedron defined by constraints (5),

(6), and (7) with just a single bidder has integer extreme points. Note that for this special case, Lemma 5.2 offers an alternate and much more succinct proof than Padberg and Alevras (1994).

Lemma 5.2 suggests the possibility of an advantageous branching strategy when using a branch-and-bound technique to solve MBA-IP. Rather than branching on a single variable at each node in the branch-and-bound tree, it seems possible to benefit by branching on a bidder/item pair, dictating that the specific item definitely be awarded to bidder j on one branch, and that the item definitely not be awarded to bidder j on the other branch. It is advantageous to focus on bidder/item pairs corresponding to items split fractionally among several bidders, since by Lemma 5.2, if these bidder/item pairs don't exist, then items will not be split fractionally among the columns of the same bidder, and the resulting solution must therefore be integer.

6 Computational Benefits of Matrix Bidding

Having explored the expressability of the matrix bids format as a compact language for conveying a wide variety of preference information and categorizing the winner-determination problem for an MBA as $\mathcal{NP}$-hard, we are interested to see how efficiently we can solve instances of our formulation MBA-IP. The purpose of our experiments was to demonstrate the strength of the MBA-IP formulation, verifying that we can indeed rapidly solve MBA instances, and to illustrate the benefits of matrix bidding relative to the XOR-of-flat-bids language.

To achieve these goals, data must be generated randomly, as little empirical data is publicly available for actual combinatorial auctions. The type of data simulation used here is common in the combinatorial auction literature. Sandholm et al. (2005) provide a prominent example, while Boutilier (2002) notes that the available benchmark data (i.e., the CATS test suite discussed by Leyton-Brown et al., 2002) may not always be beneficial for the testing of new language paradigms. In our case, we applied the algorithm of §4.1 to two-hundred of the CATS instances used for the computational experiments in Day and Raghavan (2007). (In particular, we used CATS instances with 16 items and 10, 25, 50, or 100 package bids, with 50 instances of each auction size.) We found that about 29% of all bidders had preferences that could be expressed exactly with a single matrix bid, but there was only a single auction instance for which *all* bidders could be thus converted, making a head-to-head comparison with the CATS data impractical. This should not be a great surprise, since the CATS architecture is geared toward finding a few high valued bundles for a simulated bidder, while matrix bidding emphasizes the simultaneous bidding on a large number of bundles. Therefore, we may conclude that the CATS data does not provide an easily usable set of data for matrix bid testing. Instead, and as in Boutilier (2002), we used random generation of "typical sentences" in our own language as a test-bed for an initial evaluation of our ability to solve winner-determination problems stated in the matrix bidding language.

In the early phases of our investigation, we began with some small examples generated on an ad-hoc basis, using our own imagination and economic intuition. Next we generated problems randomly, selecting

each entry in each matrix bid from the same set of possible values with equal probability, however, this *totally* random data was not entirely useful. Indeed, in these initial experiments it was uncommon to find instances in which the LP-relaxations yielded fractional solutions, and we decided that it would be beneficial to generate simulated auction data in a more sophisticated way. The reasoning for this was that the very "lumpy" preferences, those with significant positive synergies, were not very likely under a scheme in which each matrix bid entry was drawn with equal probability from the same support; *totally* random instances seemed too easy. Since computational difficulties in combinatorial auction winner-determination arise exactly from this lumpiness, we decided instead to simulate bidders who would emulate "typical" behavior, including several lumpy type preferences like the k-of and flat-bidding described in §3.

Since many available combinatorial auction techniques focus on flat atomic bids and preferences built as an aggregation of flat-bids, the data generated to test these techniques is naturally comprised of randomly drawn bids attached to randomly drawn subsets or bundles of items. Since matrix bidding is an auction format in which bidders assign values to bundles in a more orderly fashion, it should seem natural that our method is to randomly simulate expected behavior patterns.

To maintain simplicity in our experiments, we restrict to the situation in which each bidder is allowed only a single matrix bid (e.g. there is no XORing of matrix bids.) The input for each set of experimental runs is N (the number of items), M (the number of matrix bids), H (a parameter bounding the highest bid per item), and A (the number of auctions to simulate with these parameters).

Using these parameters, the simulation first randomly selects a bid profile for each matrix bid from one of the seven types described below with equal probability. For each bid type a subroutine randomly selects a nonnegative integer (or $*$) for each of the appropriate matrix bid entries based on the value H. Appendix B describes the exact details for each of the following seven bidder types used in our experiments: (i) Additive Preference Bidder, (ii) Single-Minded Bidder, (iii) Nested Flat-Bid Bidder, (iv) Nested k-of Bidder, (v) Partition Bidder, (vi) Add-on Bidder, and (vii) Diminishing Returns Bidder.

The descriptions given in Appendix B show that matrix bid entries in our random experiments were chosen so that the incremental value an item brings to a bundle is never more than the value H, and that when a particular entry effectively bids on multiple items, the random seed used to price the incremental value of a single item is multiplied by the number of items effectively bid on so that the bid entry is not drastically small. These measures assure that the preferences conveyed by each bid entry is "in the right ballpark" relative to all other bids in the auction. Further, they provide the right notion of "lumpiness" that experience suggests is the cause of computational difficulty in combinatorial auction winner-determination.

Throughout the experiments presented here we maintain $H = 20$, though our glimpses at performance for other values of H suggest similar results. A more elaborate model might incorporate a different value H_i for each item i (i.e. a different support for each item's value), though here we assume that the value any item brings to a matrix bid entry is drawn from the same support.

While Appendix B describes the exact method for simulating each behavior type, and we do note here that

our matrix-bid-based method provides an interesting alternative to the CATS data (see Leyton-Brown et al., 2000). While both methods generate data based on economic models of synergy and predefined preference structures, the compactness of matrix bidding allows for the expression of economically simple preferences that require an exponential number of flat-bids in the XOR language used by CATS. Thus the technique for data generation used here may in some cases provide access to data sets not available through CATS due to memory constraints. (Day's website, http://users.business.uconn.edu/bday/index.html, provides a link to the executable used to generate matrix bid auction instances, as well as the actual 1,800 instances generated for the current treatment.)

Using this technique for bidder simulation, we generated fifty instances ($A = 50$) of each instance size taken from combinations of $N \in \{4, 8, 16, 24, 72\}$ and $M \in \{5, 10, 25, 50, 75, 100\}$. For each of these instances, after generating the matrix bids we performed a conversion into XORed flat-bids. To do this, we simply searched through each possible bundle for each matrix bid and assigned a flat-bid to that bundle equal to the matrix bid on that bundle. In order to eliminate unnecessary bids, no flat-bid was generated unless the lowest ranked item in that bundle contributed a strictly positive marginal amount to the value of the bundle. For example, a matrix bid with all zero entries except for a single positive entry on the diagonal would generate only one flat-bid. This makes the comparison more fair, so that the number of flat-bids is minimized.

With identical auction instances captured in two different formats, we next solved each instance using the MBA-IP formulation and the GWD formulation, for the matrix bid and the XOR-of-flat-bids languages, respectively. Unless otherwise stated, we used CPLEX 9.0 on a 3.2 GHz Pentium 4 processor Dell desktop with 1 gigabyte of RAM for our computational work. Table 1 provides the results in terms of average run time, providing some indication of the horizon of practical implementation for matrix bids. The entry of ** indicates that *some* instances of that size could not be solved due to RAM overflow in CPLEX's branch-and-cut process, while *** indicates that no instance of that size could be solved, again due to RAM overflow[7].

Though the XOR language may provide a faster winner-determination problem for a small number of items, the computational time for the GWD formulation grows much faster than for the MBA-IP formulation as the number of items increases. Indeed, under matrix bidding we do not see 4+ minutes of average run time until around 48 items and 75 bidders, whereas the XOR formulation requires this long on average for just 16 item and 25 bidders.

As Table 1 indicates, however, the greater strength of the matrix bid approach is reduced memory. When

[7]For two of the XOR/GWD cases, ($N = 16$, $M = 75$), and ($N = 16$, $M = 100$), marked with † and ‡, respectively, we could not compute solutions for any instances due to RAM overflow. We re-tried these instances on a different machine (2.66 GHz Intel Xeon processor Dell desktop with 2 gigabytes of RAM) to see how much more could be done with a greater amount of RAM. The average run times given reflect the average over the 43 out 50 instances for †, and 16 out of 50 instances for ‡ that could be completed prior to RAM overflow with 2 GB. For †, 4 of the 7 overflow instances yielded usable best integer solutions at around 96% optimality on average, while only 4 out of the 34 overflow cases yielded usable best integer solutions, at about 95% optimality on average for ‡. For all other instances of these sizes no feasible integer solution was found prior to RAM overflow. For $N = 24$ and larger we encountered RAM overflow even with the 2GB RAM machine

Table 1: Average run time (in seconds) for winner-determination

MBA-IP

	M=5	10	25	50	75	100
N=4	0.00506	0.00382	0.00884	0.01608	0.02628	0.03870
8	0.00646	0.01244	0.03358	0.07736	0.12110	0.18050
16	0.09112	0.0895	0.2271	0.62284	0.88516	1.21684
24	0.19328	0.48246	1.40398	4.09132	5.56776	7.19489
48	13.5852	15.5009	67.2408	149.861	250.890	456.343
72	63.7875	260.139	1337.05	4411.98	4275.91	**

XOR / GWD

	M=5	10	25	50	75	100
N=4	0.00160	0.00480	0.01012	0.01302	0.01826	0.02458
8	0.01380	0.02374	0.05630	0.14470	0.2085	0.31032
16	20.5628	28.8746	156.269	602.694	1522.67†	815.426‡
24	***	***	***	***	***	***
48	***	***	***	***	***	***
72	***	***	***	***	***	***

Note: ** indicates that *some* instances of that size could not be solved due to RAM overflow while *** indicates that no instance of that size could be solved. † and ‡ indicate results on a 2GB machine.

we expand out all the bid information from each matrix bid into flat-bids there becomes such a large number of decision variables (non-trivial bids) that there is typically not enough memory to hold all the bases stored at each node in a CPLEX branch-and-cut tree. As we see in Table 1, the XOR language paradigm could not be applied for more than 16 items due to memory overload, while matrix bidding solved most of the problems with 72 items and 100 bidders (33 out of 50 instances) and every instance of a smaller size without overloading the memory.

The run times indicated also give an indication of the applicability of the matrix bidding approach in a practical context. Multi-round auctions for several items often provide an hour or so per round to accommodate human deliberation, making run times of a few seconds or minutes for a small number of items seem adequate. Though larger run times for a larger number of items suggest that matrix bidding may not be appropriate for each round of an iterative combinatorial auction for more than 48 items, the approach may still be suitable for a sealed-bid round of an auction (such as the clock-proxy auction of Ausubel et al., 2006) for around 72 items, since winner-determination that may take a day or two of run time may be acceptable after bidder participation has concluded.

Since the XOR language becomes too cumbersome with huge data files ($>$ 2GB) starting around size $N = 24$, we performed a limited-bundle-size experiment for $N = 24$ and each of the previously used values for M. To show the increased effectiveness of the MBA-IP formulation relative to GWD as the number of items increases, we truncated a set of bid data to allow only for bundles of size 5 or less. (In some applications, such as spectrum license auctions, the auctioneer may want to limit bundle size to decrease market concentration, for example.) Note that limiting bundle size to 5 or less is easy with matrix bids, just

Table 2: Median and Worst-case run time (in seconds) over 10 winner-determination instances for each size, with 24 total items and bundle-size limited to 5 items or less in each case.

Median	M=5	10	25	50	75	100
MBA-IP	0.078	0.188	0.359	0.812	1.343	1.976
XOR / GWD	0.452	5.148	18.765	59.859	130.305	423.867

Worst-Case	M=5	10	25	50	75	100
MBA-IP	0.109	0.266	1.422	1.64	2.437	3.453
XOR / GWD	5.079	1088.484	1883.25	148.078	722.391	681.907

"zero out" the sixth through final columns. Using the same conversion procedure as before, this produced XOR data files of manageable size, allowing for comparisons of the speed of the two formulations at $N = 24$. We performed this smaller experiment with just 10 instances of each size, truncated from the same set of randomly generated instances discussed above.

The results are shown in Table 2. Not only can we see that the MBA-IP formulation is continuing to perform a few orders of magnitude better for the median instances at each size, but we also see that the disparity between the typical case and the worst case is growing much faster for the GWD formulation. Indeed, for one particularly difficult instance (the worst for size $M = 25$), the run time is more than one hundred times the median instance of that size under the XOR formulation, compared to a factor of about four times the median instance of that size for the same difficult instance formulated using MBA-IP.

The results presented in Tables 1 and 2 represent the use of matrix bidding in its most straightforward implementation. Thus the computational results presented here may be viewed as a first step in establishing the potential of the matrix bidding language. Many promising computational areas remain to explored, including both the investigation of specialized heuristics to reduce memory usage and runtimes, and a thorough exploration of how CPLEX's various cutting and branching parameters may be tuned to provide better performance. Although a more detailed discussion of these possibilities is beyond the scope (and page limit) of the current submission, we note that positive results using heuristics presented by Day (2004) and Goossens (2006) indicate that more specialized computational techniques may further strengthen the applicability of the matrix bidding paradigm. Indeed this is the focus of much of our current research, and we anticipate a sequel paper focused solely on the computational aspects of solving combinatorial auctions with the matrix bid format.

7 Conclusions

Is there a way to express a variety of preferences in a two-dimensional array, containing just numbers, that allows for expression of both substitutes and complements? In this paper we introduce matrix bidding and assert that it is such a format. We offer this new compact representation of preferences for combinatorial

auctions based on the theory of "price-vector agents" (see Day, 2004, for a more general treatment of this approach). Matrix Bids can indeed be expressed in a standard two-dimensional array, giving them the valuable property of being able to be written in a spreadsheet or standard text editor, with only numbers and no need for special symbols or commands. A survey of the literature on bid languages shows that few if any have this property.

Indeed, Theorem 5.1 shows that matrix bid expression is general enough for the winner-determination problem to be $\mathcal{NP}$-hard, and thus fundamentally difficult. Despite this difficulty, Lemma 5.2 (which generalizes a result of Padberg and Alevras, 1994) assures us that the problem of evaluating the bid on an arbitrary bundle can be performed in polynomial-time for any single matrix bid. This tight formulation of the matrix bid auction winner-determination problem makes it a "good" way to solve the $\mathcal{NP}$-hard problem, with a high percentage of simulated instances solving to integral optimality as the LP-relaxation.

In §3 and §4 we investigated some of the strengths and weaknesses of matrix bidding in terms of preference expression. We showed that a logical language of matrix bids is equally as expressive in a polynomially sized format as any known bidding language, but with an added level of visual intuitive-ness. We demonstrated that matrix bids are particularly useful for the expression of several k-of bids, perhaps with price differentiation among the items, as well as intuitively applying a capacity cost structure any other matrix bid expression, or for grocery-list bidding, to name just a few. Indeed, the wide range of preference expression possible under matrix bidding is among its greatest strengths. We also described a method that allows for the expression of a bidder's preferences in a single matrix bid when possible, or lets us know that expression of the bidder's preferences in a single matrix bid is not possible. This naturally led to the question of how much information is lost in a single matrix bid, which we answered, and the notion of safe-bidding when bidders must submit bids in a single matrix bid. An interesting question that arises in the context of a matrix bid is whether the bids expressed satisfy various microeconomic properties (e.g., free disposal). In this regard, we must mention the novel work of Goossens (2006) where he shows that checking whether the valuations represented by a matrix bid satisfies any of the following microeconomic properties—free disposal, subadditivity, superadditivity, submodularity, supermodularity, gross substitutes—can be done by solving shortest path problems on polynomially sized graphs.

We also emphasized that a logical language of matrix bids would be necessary for a reasonable generality of preference expression, and propose that an XOR-of-OR of matrix bid language admits a great variety of preference expression that can be written in a spreadsheet environment without the explicit use of connectives (by separating XORed scenarios into different sheets.). The development and empirical testing of user-friendly interfaces (such as a spreadsheet implementation of an XOR-of-OR of matrix bid language) remain goals for future research, of which we have taken a few initial steps.

Our method for data simulation (discussed in Appendix B) elucidates another strength of the format. Matrix bidding provides a quick way to generate complex examples and simulated data using only numerical information. Matrix bidding can therefore be a quick way to generate examples from a fairly general class of

preferences, including preferences (such as the k-of preferences) that would take an exponential amount of bid information in other combinatorial auction data sets, such as those using flat-bids (for example CATS, Leyton-Brown et al., 2000). However, some care must be taken that data is generated in an orderly fashion for the resulting matrix bids to represent economically meaningful behaviors.

Finally, using this simulation approach, we demonstrated the computational benefits of matrix bidding. Not only did computational processing time grow much slower for matrix bids, but because of the compactness of the language it took much larger problem instances to exhaust our memory capabilities under matrix bidding relative to the XOR language. As a result, we could solve instances with three times as many items and three times as many bidders as the largest instance we could solve under the XOR language. With the recently proposed "clock-proxy" auction of Ausubel et al. (2006), it is evident that there are indeed great benefits to conveying a bidder's *entire* set of preferences for as many bundles as possible.

Altogether we hope to have convinced the reader that matrix bidding is an interesting and useful technique for combinatorial auctions. Its advantages are its compactness, expressability, and relative computational ease.

Acknowledgments

The authors gratefully acknowledge the support of the National Science Foundation award numbers DMI-0205489 and DMS-0240049, as well as the Robert H Smith School of Business Strategic Research Fund. We would also like to thank the referees for invaluable comments that have helped improve this work.

References

Ausubel, L., P. Cramton. 1999. The optimality of being efficient. World Bank Policy Research Working Paper 1985, University of Maryland.

Ausubel, L., P. Cramton, P. Milgrom. 2006. The clock-proxy auction: A practical combinatorial auction design. P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, chap. 5. MIT Press, Cambridge, MA, 115–138.

Boutilier, C. 2002. Solving concisely expressed combinatorial auction problems. *Proc. 18th National Conference on Artificial Intelligence AAAI-02*. 359–366.

Boutilier, C., H. Hoos. 2001. Bidding languages for combinatorial auctions. *Proc. 17th International Joint Conferences on Artificial Intelligence IJCAI-01*. 1211–1217.

Cramton, P., Y. Shoham, R. Steinberg, eds. 2006. *Combinatorial Auctions*. MIT Press, Cambridge, MA.

Day, R. 2004. Expressing preferences with price-vector agents in combinatorial auctions. Ph.D. thesis, Applied Mathematics and Scientific Computation, University of Maryland, College Park. Url: http://users.business.uconn.edu/bday/thesis.pdf.

Day, R., S. Raghavan. 2007. Fair payments for efficient allocations in public sector combinatorial auctions. *Management Science* **53**(9) 1389–1406.

de Vries, S., R. Vohra. 2003. Combinatorial auctions: A survey. *INFORMS J. of Computing* **15**(3) 284–309.

Demange, G., D. Gale, M. Sotomayor. 1986. Multi-item auctions. *J. of Political Economy* **94**(4) 863–872.

Goossens, D. 2006. Exact methods for combinatorial auctions. Ph.D. thesis, Economics and Applied Economic Sciences, the Catholic University of Leuven.

Günlük, O., L. Ladányi, S. de Vries. 2005. A branch-and-price algorithm and new test problems for spectrum auctions. *Management Science* **51**(3) 391–406.

Hoos, H., C. Boutilier. 2000. Solving combinatorial auctions using stochastic local search. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*. AAAI Press / The MIT Press, 22–29.

Lehmann, D., R. Müller, T. Sandholm. 2006. The winner determination problem. P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, chap. 12. MIT Press, Cambridge, MA, 297–318.

Leyton-Brown, K., E. Nudelman, Y. Shoham. 2002. Learning and the empirical hardness of optimization problems: The case of combinatorial auctions. *8th International Conference, CP 2002, Ithaca, NY*. 556–572.

Leyton-Brown, K., M. Pearson, Y. Shoham. 2000. Towards a universal test suite for combinatorial auction algorithms. *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*. ACM Press, New York, NY, USA, 66–76.

Müller, R. 2006. Tractable cases of the winner determination problem. P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, chap. 13. MIT Press, Cambridge, MA, 319–336.

Nisan, N. 2000. Bidding and allocation in combinatorial auctions. *EC '00: Proceedings of the 2nd ACM conference on Electronic commerce*. ACM Press, New York, NY, USA, 1–12.

Nisan, N. 2006. Bidding languages in combinatorial auctions. P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, chap. 9. MIT Press, Cambridge, MA, 215–232.

Padberg, M., D. Alevras. 1994. Order-preserving assignments. *Naval Research Logistics* **41**(3) 395–421.

Rothkopf, M., A. Pekeč, R. Harstad. 1998. Computationally manageable combinatorial auctions. *Management Science* **44**(8) 1131–1147.

Sandholm, T. 2006. Optimal winner determination algorithms. P. Cramton, Y. Shoham, R. Steinberg, eds., *Combinatorial Auctions*, chap. 14. MIT Press, Cambridge, MA, 337–368.

Sandholm, T., S. Suri, A. Gilpin, D. Levine. 2005. CABOB: A fast optimal algorithm for winner determination in combinatorial auctions. *Management Science* **51**(3) 374–390.

Weisstein, E. 2005. Pascal's triangle. From MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/PascalsTriangle.html.

Appendix A: Modeling OR-of-XOR and XOR-of-OR Matrix-Bid Languages

In this section we discuss how to model the OR-of-XOR and XOR-of-OR Matrix-Bid Languages using the integer program MBA-IP.

One way to quickly accomplish an XOR of Matrix bids is to add constraints on the set of slack variables s_{jk} from each constraint in the set 6 in the integer program MBA-IP. (To generate slack variables, replace "≤ 1" with "$+s_{jk} = 1$" in constraint set 6, so that $s_{jk} = 0$ for any integer feasible solution in which j receives at least k items, and $s_{jk} = 1$ for any integer feasible solution in which j receives less than k items.)

Observe that if slack variable $s_{j1} = 1$, bidder j's first column is completely without an item. According to the ordering constraints if there is no item at bidder j's first column there can be no item at bidder j's second column, and so forth; $s_{j1} = 1$ therefore implies that bidder j receives no items. Since each bidder represents a single matrix bid we may achieve "Bid 1 XOR Bid 2" by adding the constraint:

$$s_{11} + s_{21} \geq 1$$

to the IP formulation, where $j = 1, 2$ refer to Bid 1 and Bid 2 respectively. For the remainder of this subsection we use the convention that each $j \in J$ refers to a single matrix bid, rather than a bidder in the auction.

In general, the XOR of matrix bids $1, 2...m$ may be expressed by:

$$\sum_{j=1}^{m} s_{j1} \geq m - 1 \tag{10}$$

If a bidder submits a request to XOR matrix bids $1, 2...m$, and a separate request to XOR matrix bids $(m+1), (m+2)...n$, these submissions do not mutually exclude one another. This allows each bidder's submission to easily take the OR-ofXOR general form:

$$\begin{array}{l}(MatrixBid1\ XOR\ MatrixBid2...\ XOR\ MatrixBid\ M_1)\ OR \\ (MatrixBid\ M_1 + 1\ XOR\ ...MatrixBid\ M_2)\ OR... \\ (MatrixBid\ M_{Q-1} + 1\ XOR\ ...MatrixBid\ M_Q)\end{array}$$

specifying an OR-of-XOR of matrix bids language, where $M_q = \sum_{l=1}^{q} m_l$, and each m_l equals the number of XORed matrix bids in the lth of the Q different ORed clauses. For each of the Q clauses, a constraint of the form (10) is introduced.

Also studied in the auction literature are XOR-of-OR bidding languages, which include the format initially proposed for the FCC's Auction #31 (Günlük et al., 2005). This can be modeled in the MBA-IP formulation as follows. Add a binary variable for each OR string, and define $\mathcal{Q} = \{1, 2, ...Q\}$ as the XORed set of clauses, each an ORed string of matrix bids. A bidder's complete preference submission will be of the form:

$$\begin{array}{l}(MatrixBid1\ OR\ MatrixBid2...\ OR\ MatrixBid\ M_1)\ XOR \\ (MatrixBid\ M_1 + 1\ OR\ ...MatrixBid\ M_2)\ XOR... \\ (MatrixBid\ M_{Q-1} + 1\ OR\ ...MatrixBid\ M_Q)\end{array}$$

where $M_q = \sum_{l=1}^{q} m_l$, and each m_l is now equal to the number of ORed matrix bids in the lth of the Q different XORed clauses. For each $q \in \mathcal{Q}$ define $g_q = 0$ if any matrix bid in the ORed string q receives items, $g_q = 1$ otherwise. Then add definitional constraints to MBA-IP for each string q, together with an

XOR constraint:

$$\begin{aligned} g_q &\leq s_{j1}, \forall\ j \in q \\ \sum_{q=1}^{Q} g_q &\geq Q-1 \end{aligned}$$

The first set of constraints ensures that if $g_q = 1$, then none of the matrix bids j in string q receives items. The second constraint ensures that among all Q strings at most one has one or more matrix bids receiving items.

Though we provide here several examples of potentially useful logical languages of matrix bids, many other are possible. Indeed, any of the logical bidding structures put forth by Boutilier (2002) may be adapted to create a more or less complex logical language of matrix bids. Our examples show that in certain cases a formulation may be expressed in terms of the slack variables for a single column from each matrix bid, rather than on a large collection of x_{ijk} assignment variables. We note that the particular choice of an appropriate logical language may vary from application to application, and that one direction for future research is to empirically measure the trade-off between expressability and computational burden in the selection of a logical language.

Appendix B: Methodology used for Simulating Bidder Preferences

Here we describe the prototypes used to simulate bidder preferences using matrix bids for our experiments. In each case (except for the partition bidder), before the entries in a matrix bid are filled in, a ranking r_{ij} is chosen at random from among the $N!$ possibilities.

Additive Preference Bidder: For each item in the auction, an integer is chosen from the set $\{0, 1, 2, ..., H\}$ with equal probability. Every row is then filled in with the same number equal to value chosen for the item in that row.

Single-minded Bidder: Included to simulate very inflexible or lumpy preferences, this bidder places a single positive entry on the diagonal. A column is chosen at random with equal probability, and a seed for the entry is chosen from the set $\{1, 2, ..., H\}$ with equal probability. The seed is then multiplied by the column number to reflect the number of items it is effectively bidding on.

Nested Flat-Bid Bidder: For each entry on the diagonal, an integer is chosen from the set $\{-H, ..., H\}$ with equal probability. Next, every negative selection is set to equal zero; there is thus a $\frac{H}{2H+1}$ probability of a positive entry, and probability of $\frac{1}{H}$ for each positive value conditional upon the entry being positive. Finally each positive entry is multiplied by the number of zero entries preceding it along the diagonal, weighting each entry by the number of items it has effectively bid on.

Nested k-of Bidder: Begin with a nested flat-bid bidder and simply fill in each column with the entry chosen for the diagonal entry in that column. An example is given by Figure 5. If the numbers 6, 4, and

Figure 5: A nested k-of Matrix Bid

D	0					
B	0	0				
C	0	0	18			
A	0	0	18	0		
F	0	0	18	0	8	
E	0	0	18	0	8	7

7 are drawn for the columns 3, 5, and 6, respectively, each column generated by multiplying the random number by the number of consecutive zero columns immediately to the left plus one. Each positive bid then has an average bid per item equal to the randomly drawn number for that column.

Partition Bidder: This bidder divides the set of items into g groups of substitutes and gives a price for receiving one item from the group, given that one item from each previous group has been received. First we choose the number of groups g with equal probability from 2 to $\frac{N}{2}+1$. Then each item is assigned to a group with equal probability. Next the rankings are chosen so that each item in an earlier group has an earlier ranking (ranking within a group will not matter). Then, the value for any item in the group is chosen from the set $\{-H, ..., H\}$ with equal probability. The value is set to zero if negative, and multiplied by the number of consecutive immediately preceding groups with value of zero. Again this is to ensure that each positive bid value reflects the number of items it is effectively bidding on. Finally, the matrix bid column corresponding to each group number is filled in with the value of the group for any row corresponding to an item in the group, and $*$ for every other row. The result is a special case of the grocery-list bidding from §4.2, in which only one item is demanded from each group of substitutes. We note that when using $*$-entries in a matrix bid, we can simply remove the decision variable associated with that entry from the IP formulation.

Add-on Bidder: This bidder bids on an essential item and some nonnegative amount for any number of add-on items. First the row of the item to be considered essential is chosen at random with equal probability. Next an initial value is chosen for the essential item which is placed in the first column of the row. For each following entry in the row an integer between zero and the initial value is chosen, and added to the previous value in the row until the row is filled.

Diminishing Returns Bidder: This bidder simulates weakly diminishing marginal returns for every item received, and thus is generated to have weakly decreasing rows and columns. The entry in the first row and first column (i.e., the value for the highest ranked item) is chosen from $\{0, 1, 2, ..., H\}$ with equal probability. Each entry in the first column is given a value between the value above it and half of that value, with a 50% probability of being equal to the entry above it and an equal probability of taking any integer value between if not equal. To start the next column once a column is filled, the diagonal entry from that column is generated in the same way, but adjusting down to the entry on its left if that number is smaller. The other entries in the new column are generated in the same way as the first column, based on the entry above, but now adjusted down to the value of the entry on its left if necessary to ensure weakly decreasing rows.

Online Appendix

Proof of Claim 4.1.

Proof. By induction on the columns. This is clearly true when $k = 1$ and $k = 2$. Suppose it is true for column $k = \alpha$, and we are now working on column $k = \alpha + 1$ in the column-based procedure. Observe, as a consequence of the inductive argument, the only items with pos$[i]$ values strictly between t and $t - l$ must have had pos$[i] = t$ in column α.

Let f be an item with position index values strictly between t and $t - l$. Let g be one of the l items that satisfy condition (4) for row t and denote the set of l items as L. Consider any set S_g of size $k - 1$ that contains items with position index t or lower and (i) does not contain f, and (ii) contains g. Note there are $\binom{t-2}{k-2}$ such sets that we denote by Ω_g and observe

$b_j(S_g \cup f) = b_j(S_g \cup f \backslash g)+$ constant, since g has pos$[g] = t$ in column k;

$b_j(S_g) = b_j(S_g \backslash g)+$ constant, since g has pos$[g] \geq t - 1$ in column $k - 1$;

and

$b_j(S_g \cup f \backslash g) = b_j(\{S_g \cup f \backslash g\} \backslash f)+$ constant, since f has pos$[f] = t$ in column $k - 1$.

Resulting in the observation that

$b_j(S_g \cup f) - b_j(S_g) = b_j(\{S_g \cup f \backslash g\} \backslash f) - b_j(S_g \backslash g)+$ constant.

But, $\{\{S_g \cup f \backslash g\} \backslash f\} = \{S_g \backslash g\}$, and thus $b_j(S_g \cup f) - b_j(S_g) =$ constant for all $S_g \in \Omega_g$.

The above argument remains true for any item $h \in L$. Thus for every $h \in L$

$b_j(S_h \cup f) - b_j(S_h) =$ constant for all $S_h \in \Omega_h$.

Further,

$b_j(S_h \cup f) - b_j(S_h) = b_j(S_g \cup f) - b_j(S_g) \qquad \forall g, h \in L$, since $S_h \cap S_g \neq \phi \qquad \forall g, h \in L$.

In simple terms the above argument says for any set T of $k - 1$ items (i) not containing f, and (ii) containing at least one item from L, the incremental value item f brings to the set is the same.

With the above result in hand, we now prove inductively that f cannot be given a position index c strictly between t and $t - l$ (i.e., $t - l < c < t$). Let I_t denote all items with position index less than or equal to t. Suppose c the position index of f is equal to $t - x$ with $x = 1$. Then it must be true for the $\binom{t-2}{k-1}$ sets S with elements chosen from $(I_t \backslash L \backslash f) \cup (L \backslash g)$, for some choice of $g \in L$, the incremental value f brings to the set S is constant. Note none of the sets $S \in \Omega_g$, but some of the sets in S are also elements of Ω_h, $h \in L \backslash g$. Thus, it must be true that the incremental value f brings to sets $S_g \in \Omega_g$ is identical to the incremental value f brings to the $\binom{t-2}{k-1}$ sets S. Or there are $\binom{t-2}{k-1} + \binom{t-2}{k-2} = \binom{t-1}{k-1}$ sets of $k - 1$ elements from $I_t \backslash f$ to which the incremental value f brings is constant. And consequently f must have position index equal to t resulting in a contradiction.

Repeating this argument with $x = 2$, i.e., $c = t - 2$, we find that it must be true for the $\binom{t-1-x}{k-1}$ sets S with elements chosen from $(I_t \backslash L \backslash f) \cup (L \backslash \{g, h\})$, for some choice of $\{g, h\} \in L$, the incremental value f brings to the set S is constant. Again, none of the sets $S \in (\Omega_g \cup \Omega_h)$, but some of the sets in S

are also elements of Ω_e, $e \in L \backslash \{g, h\}$. Thus, it must be true that the incremental value f brings to sets $S_g \in \Omega_g$ and $S_h \in \Omega_h$ is identical to the incremental value f brings to the $\binom{t-1-x}{k-1}$ sets S. Or there are $\binom{t-1-x}{k-1} + \sum_{y=1}^{x} \binom{t-1-y}{k-2} = \binom{t-1}{k-1}$ sets of $k-1$ elements from $I_t \backslash f$ for which the incremental value f brings is a constant. Consequently f must have position index equal to t resulting in a contradiction. Continuing in this fashion until $x = l - 1$ completes the proof. $\square$