Bidding Languages and Winner Determination for Mixed Multi-unit Combinatorial Auctions

Jesús Cerquides
University of Barcelona
Spain

Ulle Endriss
University of Amsterdam
The Netherlands

Andrea Giovannucci
Juan A. Rodríguez-Aguilar
IIIA-CSIC, Spain

Abstract

We introduce a new type of combinatorial auction that allows agents to bid for goods to buy, for goods to sell, and for transformations of goods. One such transformation can be seen as a step in a production process, so solving the auction requires choosing the sequence in which the accepted bids should be implemented. We introduce a bidding language for this type of auction and analyse the corresponding winner determination problem.

1 Introduction

A combinatorial auction is an auction where bidders can buy (or sell) entire bundles of goods in a single transaction. Although computationally very complex, selling items in bundles has the great advantage of eliminating the risk for a bidder of not being able to obtain complementary items at a reasonable price in a follow-up auction (think of a combinatorial auction for a pair of shoes, as opposed to two consecutive single-item auctions for each of the individual shoes). The study of the mathematical, game-theoretical and algorithmic properties of combinatorial auctions has recently become a popular research topic in AI. This is due not only to their relevance to important application areas such as electronic commerce or supply chain management, but also to the range of deep research questions raised by this auction model. This paper introduces a generalisation of the standard model of combinatorial auctions and discusses the issues of *bidding* and *winner determination*.

Winner determination is the problem, faced by the auctioneer, of choosing which goods to award to which bidder so as to maximise its revenue. Bidding is the process of transmitting one's valuation function over the set of goods on offer to the auctioneer. In principle, it does not matter how the valuation function is being encoded, as long as sender (bidder) and receiver (auctioneer) agree on the semantics of what is being transmitted, *i.e.* as long as the auctioneer can understand the message(s) sent by the bidder. In practice, however, the choice of bidding language is of central importance. Early work on combinatorial auctions has typically ignored the issue of bidding languages. The standard assumption used to be that if a particular bidder submits several atomic bids (a bundle together with a proposed price), then the auctioneer may accept any set of bids from that bidder for which the bundles do not overlap, and charge the sum of the specified prices. This is now sometimes called the *OR-language*. But other interpretations of a set of atomic bids are possible. For instance, we may take it to mean that the auctioneer may accept at most one bid per bidder; this is now known as the *XOR-language*.

Nisan's survey article [2006], the reference work in the field of bidding languages, covers languages for *direct single-unit* combinatorial auctions. That is, the auctioneer is selling (rather than buying) goods, and all goods are distinguishable. Our first aim in this paper is to generalise this to *multi-unit* combinatorial auctions. As a second generalisation, we show how to integrate direct and reverse auctions, *i.e.* the auctioneer will be able to both sell and buy goods within a single auction. As a third and final generalisation, we are going to show how to integrate the idea of *transformability* relationships between goods into our auction model [Giovannucci *et al.*, 2005]. For instance, if the auctioneer is interested in obtaining cars, but is also able to transform various components into a working car (at a certain cost), then it may solicit bids offering both ready-made cars and individual components. We further extend the idea of these transformability relationships by allowing agents to also bid for *transformation services*, *i.e.* an agent may submit a bid offering to transform a certain set of goods into another set of goods. We call the resulting auction model *mixed multi-unit combinatorial auctions* (MMUCA). These are not to be confused with double auctions [Wurman *et al.*, 1998]. In particular, the *order* in which agents consume and produce goods is of central importance in our model and affects the definition of the winner determination problem.

The paper is organised as follows. In Section 2 we define the notions of transformation and valuation over (multisets of) transformations, and we define an expressive bidding language to represent such valuations. Section 3 discusses the winner determination problem for MMUCAs, including its computational complexity. Finally, Section 4 concludes with a discussion of related work and an outlook on future work.

2 Bidding Languages

In this section, we first define the notions of transformation and valuation over transformations, and then define a bidding language that can be used to transmit an agent's valuation (which may or may not be their true valuation) to the auction-

eer. We also formally define the semantics of the language and introduce a number of additional language constructs that allow for the concise encoding of typical features of valuation functions. Finally, we discuss the expressive power of the language.

2.1 Transformations

Let G be the finite set of all the types of goods under consideration. A *transformation* is a pair of multisets over G: $(\mathcal{I}, \mathcal{O}) \in \mathbb{N}^G \times \mathbb{N}^G$. An agent offering the transformation $(\mathcal{I}, \mathcal{O})$ declares that it can deliver $\mathcal{O}$ *after* having received $\mathcal{I}$. In our setting, bidders can offer any number of such transformations, including several copies of the same transformation. That is, agents will be negotiating over *multisets of transformations* $\mathcal{D} \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)}$.

For example, $\{(\{\,\}, \{a\}), (\{b\}, \{c\})\}$ means that the agent in question is able to deliver a (no input required) and that it is able to deliver c if provided with b. Note that this is not the same as $\{(\{b\}, \{a, c\})\}$. In the former case, if another agent is able to produce b if provided with a, we can get c from nothing; in the latter case this would not work.

We define a *subsumption relation* $\sqsubseteq$ over transformations as follows: $(\mathcal{I}, \mathcal{O}) \sqsubseteq (\mathcal{I}', \mathcal{O}')$ iff $\mathcal{I} \subseteq \mathcal{I}'$ and $\mathcal{O} \supseteq \mathcal{O}'$. Intuitively, this means that the second transformation is at least as good as the first (for the bidder), because you receive more and have to give away less. The following definition extends this subsumption relation to multisets of transformations. It applies to multisets of the same cardinality, where for each transformation in the first set there exists a (distinct) transformation in the second set subsuming the former.

Definition 1 (Subsumption) *Let $\mathcal{D}, \mathcal{D}' \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)}$. We say that $\mathcal{D}$ is subsumed by $\mathcal{D}'$ ($\mathcal{D} \sqsubseteq \mathcal{D}'$) iff:*

(i) *$\mathcal{D}$ and $\mathcal{D}'$ have the same cardinality: $|\mathcal{D}| = |\mathcal{D}'|$.*

(ii) *There exists a surjective mapping $f : \mathcal{D} \to \mathcal{D}'$ such that, for all transformations $t \in \mathcal{D}$, we have $t \sqsubseteq f(t)$.*

For instance (using a simplified notation for the innermost sets), we have $\{(a, bb), (cc, dd)\} \sqsubseteq \{(cc, d), (aaa, b)\}$.

2.2 Valuations

In an MMUCA, agents negotiate over bundles of transformations. Hence, a *valuation* $v : \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)} \to \mathbb{R}$ is a (typically partial) mapping from multisets of transformations to the real numbers. Intuitively, $v(\mathcal{D}) = p$ means that the agent equipped with valuation v is willing to make a payment of p in return for being allocated all the transformations in $\mathcal{D}$ (in case p is a negative number, this means that the agent will accept the deal if it *receives* an amount of $|p|$). For instance, $v(\{(\{oven, dough\}, \{oven, cake\})\}) = -20$ means that I can produce a cake for 20 rupees if given an oven and some dough, and that I will return the oven again afterwards We write $v(\mathcal{D}) = \perp$ to express that v is *undefined* over the multiset $\mathcal{D}$. Again intuitively, this means the agent would be unable to accept the corresponding deal. Valuation functions can often be assumed to be both *normalised* and *monotonic:*

Definition 2 (Normalised valuation) *A valuation v is normalised iff $v(\mathcal{D}) = 0$ whenever $\mathcal{I} = \mathcal{O}$ for all $(\mathcal{I}, \mathcal{O}) \in \mathcal{D}$.*

That is, a valuation is normalised iff exchanging a multiset of items for an identical multiset does not incur any costs (this includes the special case of $\mathcal{I} = \mathcal{O} = \{\,\}$, *i.e.* the case of not exchanging anything at all). The next definitions refer to our subsumption relation $\sqsubseteq$ (cf. Definition 1).

Definition 3 (Monotonic valuation) *A valuation v is monotonic iff $v(\mathcal{D}) \leq v(\mathcal{D}')$ whenever $\mathcal{D} \sqsubseteq \mathcal{D}'$.*

That is, an agent with a monotonic valuation does not mind taking on more goods and giving fewer away. Any given valuation function can be *turned into* a monotonic valuation:[1]

Definition 4 (Monotonic closure) *The monotonic closure $\hat{v}$ of a valuation v is defined as $\hat{v}(\mathcal{D}) = \max\{v(\mathcal{D}') \mid \mathcal{D}' \sqsubseteq \mathcal{D}\}$.*

Observe that there could be infinitely many bundles an agent may want to assign a (defined) value to. As we shall see in Section 2.6, our bidding languages can only express valuations that are *finitely-peaked* (or that are the monotonic closure of a finitely-peaked valuation):

Definition 5 (Finitely-peaked val.) *A valuation v is finitely-peaked iff v is only defined over finite multisets of pairs of finite multisets and $\{\mathcal{D} \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)} \mid v(\mathcal{D}) \neq \perp\}$ is finite.*

2.3 Atomic Bids

An *atomic bid* $\text{BID}(\{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}, p)$ specifies a finite multiset of finite transformations and a price. To make the semantics of such an atomic bid precise, we need to decide whether or not we want to make a *free disposal* assumption. We can distinguish two types of free disposal:

- Free disposal *at the bidder's side* means that a bidder would always be prepared to accept more goods and give fewer goods away, without requiring a change in payment. This affects the definition of the valuation functions used by bidders.
- Free disposal *at the auctioneer's side* means that the auctioneer can freely dispose of additional goods, *i.e.* accept more and give away fewer of them. This affects the definition of what constitutes a valid solution to the winner determination problem (see Section 3).

Under the assumption of free disposal at the bidder's side, the bid $Bid = \text{BID}(\mathcal{D}, p)$ defines the following valuation:

$$v_{Bid}(\mathcal{D}') = \begin{cases} p & \text{if } \mathcal{D} \sqsubseteq \mathcal{D}' \\ \perp & \text{otherwise} \end{cases}$$

To obtain the valuation function defined by the same bid without the free disposal assumption, simply replace $\sqsubseteq$ in the above definition by equality.

2.4 Combinations of Bids

A suitable *bidding language* should allow a bidder to encode choices between alternative bids and the like. To this end, several operators for combining bids have been introduced in the literature [Nisan, 2006], which we are going to adapt to our purposes here. Informally, an OR-combination of several

[1] Here and throughout this paper, we assume that any occurrences of $\perp$ are being removed from a set before computing its maximum element, and that the maximum of the empty set is $\perp$.

bids signifies that the bidder would be happy to accept any number of the sub-bids specified, if paid the sum of the associated prices. An XOR-combination of bids expresses that the bidder is prepared to accept at most one of them. We also suggest the use of an IMPLIES-operator to express that accepting one bid forces the auctioneer to also take the second. We shall take an AND-combination to mean that the bidder will only accept the respective sub-bids together.

As it turns out, while all these operators are very useful for specifying typical valuations in a concise manner, any complex bid can alternatively be represented by an XOR-combination of atomic bids. To simplify presentation, rather than specifying the exact semantics of all of our operators directly, we are simply going to show how any bid can be translated into such a *normal form*. Firstly, any occurrences of IMPLIES and OR can be eliminated by applying the following rewrite rules:

$$\begin{aligned} X \text{ IMPLIES } Y &\rightsquigarrow (X \text{ AND } Y) \text{ XOR } Y \\ X \text{ OR } Y &\rightsquigarrow X \text{ XOR } Y \text{ XOR } (X \text{ AND } Y) \end{aligned}$$

Note that for single-unit auctions, OR cannot be translated into XOR like this (if X and Y overlap, then they cannot be accepted together; in an MMUCA this depends on the supply of the auctioneer). Next we show how to distribute AND over XOR, so as to push AND-operators to the inside of a formula:

$$(X \text{ XOR } Y) \text{ AND } Z \rightsquigarrow (X \text{ AND } Z) \text{ XOR } (Y \text{ AND } Z)$$

Finally, we need to define how to turn an AND-combination of atomic bids into a single atomic bid:

$$\text{BID}(\mathcal{D}, p) \text{ AND BID}(\mathcal{D}', p') \rightsquigarrow \text{BID}(\mathcal{D} \cup \mathcal{D}', p + p')$$

It is not difficult to see that these rewrite rules together allow us to translate any expression of the bidding language into an equivalent XOR-combination of atomic bids. We also call this the *XOR-language*. To formally define the semantics of this language, it suffices to define the semantics of the XOR-operator. Suppose we are given n bids Bid_i, with $i \in \{1..n\}$. Let $Bid = Bid_1 \text{ XOR } \cdots \text{ XOR } Bid_n$. This bid defines the following valuation:

$$v_{Bid}(\mathcal{D}) = \max\{v_{Bid_i}(\mathcal{D}) \mid i \in \{1..n\}\}$$

That is, XOR simply selects the atomic bid corresponding to the valuation giving maximum profit for the auctioneer.

2.5 Representing Quantity Ranges

As we are going to see in the next section, the XOR-language is expressive enough to describe any (finitely-peaked) valuation. Nevertheless, it may not be possible to express a given valuation in a succinct manner. From a practical point of view, it is therefore useful to introduce additional constructs that allow us to express typical features more succinctly. Here we consider the case of quantity ranges: we want to be able to express that a certain number of copies of the same transformation are acceptable to a bidder.

Let $n \in \mathbb{N}$. To express that up to n copies of the same Bid are acceptable, we use the following notation:

$$Bid^{\leq n} = \underbrace{(Bid \text{ OR } \cdots \text{ OR } Bid)}_{n \text{ times}}$$

This allows us to express *bundling constraints* in a concise manner: the bid $(\{a, a, a, b\}, -10)^{\leq 50}$ expresses that we can sell up to 50 packages containing three items of type a and one item of type b each, for 10 rupees a package (for simplicity, we omit $\mathcal{O}$ here). We also use the following shorthand:

$$Bid^n = \underbrace{(Bid \text{ AND } \cdots \text{ AND } Bid)}_{n \text{ times}}$$

Now we can express quantity ranges. Let $n_1, n_2 \in \mathbb{N}$ with $0 < n_1 < n_2$. The following expression says that we may accept between n_1 and n_2 copies of the same Bid:

$$Bid^{[n_1,n_2]} = Bid^{\leq (n_2 - n_1)} \text{ IMPLIES } Bid^{n_1}$$

These constructs also allow us to express important concepts such as quantity discounts in a concise manner. For instance, the bid $[(a, 20)^{\leq 100} \text{ IMPLIES } (a, 25)^{50}] \text{ XOR } (a, 25)^{\leq 50}$ says that we are prepared to buy up to 50 items of type a for 25 rupees each, and then up to 100 more for 20 rupees each.

2.6 Expressive Power

Next we are going to settle the precise expressive power of the XOR-language, and thereby of the full bidding language. We have to distinguish two cases, as we have defined the semantics of the language both with and without free disposal.

Proposition 1 *The XOR-language without free disposal can represent all finitely-peaked valuations, and only those.*

Proof. Let v be any finitely-peaked valuation. To express v in the XOR-language, we first compose one atomic bid for each $\mathcal{D} = \{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}$ with $v(\mathcal{D}) = p \neq \bot$:

$$\text{BID}(\{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}, p)$$

Joining all these bids together in one large XOR-combination yields a bid that expresses v. Vice versa, it is clear that the XOR-language cannot express any valuation that is not finitely-peaked. □

Proposition 2 *The XOR-language with free disposal can represent all valuations that are the monotonic closure of a finitely-peaked valuation, and only those.*

Proof. The construction of a bid representing any given valuation works in analogy to the proof of Proposition 1. Note that for the semantics with free disposal we precisely obtain the monotonic closure of the valuation we would get if we were to drop the free disposal assumption. □

These results correspond to the expressive power results for the standard XOR-language for direct single-unit combinatorial auctions. With free disposal (the standard assumption), the XOR-language can express all monotonic valuations [Nisan, 2006]; and without that assumption it can represent the complete range of valuations (note that *any* valuation is finitely-peaked if we move from multisets to sets).

3 Winner Determination

In this section, we define the winner determination problem (WDP) for MMUCAs. For lack of space we only give the

definition for the case of free disposal (at the side of the auctioneer), but this is easily adapted to the case without free disposal. We first give an informal outline of the definition, then a formal definition, and finally a formulation using integer programming, providing a basis for implementation. We also establish the computational complexity of the WDP and briefly comment on mechanism design issues.

3.1 Informal Definition

The *input* to the WDP consists of a complex bid expression for each bidder, a multiset $\mathcal{U}_{in}$ of goods the auctioneer holds to begin with, and a multiset $\mathcal{U}_{out}$ of goods the auctioneer expects to end up with.

In standard combinatorial auctions, a solution to the WDP is a set of atomic bids to accept. In our setting, however, the *order* in which the auctioneer "uses" the accepted transformations matters. For instance, if the auctioneer holds a to begin with, then checking whether accepting the two bids $Bid_1 = (\{a\}, \{b\}, 10)$ and $Bid_2 = (\{b\}, \{c\}, 20)$ is feasible involves realising that we have to use Bid_1 before Bid_2. Thus, a *solution* to the WDP will be a *sequence of transformations*. A *valid* solution has to meet two conditions:

(1) *Bidder constraints:* The multiset of transformations in the sequence has to *respect the bids* submitted by the bidders. This is a standard requirement. For instance, if a bidder submits an XOR-combination of transformations, at most one of them may be accepted.

(2) *Auctioneer constraints:* The sequence of transformations has to be *implementable:* (a) check that $\mathcal{U}_{in}$ is a superset of the input set of the first transformation; (b) then update the set of goods held by the auctioneer after each transformation and check that it is a superset of the input set of the next transformation; (c) finally check that the set of items held by the auctioneer in the end is a superset of $\mathcal{U}_{out}$. This requirement is specific to MMUCAs.

An *optimal* solution is a valid solution that maximises the sum of prices associated with the atomic bids selected.

3.2 Formal Definition

For the formal definition of the WDP, we restrict ourselves to bids in the XOR-language, which is known to be fully expressive (over finitely-peaked valuations). For each bidder i, let Bid_{ij} be the jth atomic bid occurring within the XOR-bid submitted by i. Recall that each atomic bid consists of a multiset of transformations and a price: $Bid_{ij} = (\mathcal{D}_{ij}, p_{ij})$, where $\mathcal{D}_{ij} \in \mathbb{N}^{(\mathbb{N}^G \times \mathbb{N}^G)}$ and $p_{ij} \in \mathbb{R}$. For each Bid_{ij}, let t_{ijk} be a unique label for the kth transformation in $\mathcal{D}_{ij}$ (for some arbitrary but fixed ordering of $\mathcal{D}_{ij}$). Let $(\mathcal{I}_{ijk}, \mathcal{O}_{ijk})$ be the actual transformation labelled by t_{ijk}. Finally, let T be the set of all t_{ijk}; that is, $|T|$ is the overall number of transformations mentioned anywhere in the bids.

The auctioneer has to decide which transformations to accept and in which order to implement them. Thus, an *allocation sequence* Σ (which will not necessarily be a valid solution) is a total ordering of a subset of T. We write $t_{ijk} \in \Sigma$ to say that the kth transformation in the jth atomic bid of bidder i has been selected. Furthermore, let $(\mathcal{I}^m, \mathcal{O}^m)$ be the mth transformation in the sequence Σ. That is, we have two ways of referring to a transformation: by its position in the collection of bids received (t_{ijk}) and by its position in the allocation sequence (if selected at all).

Given Σ we can obtain the set of goods held by the auctioneer after each transformation. For instance, say that the auctioneer begins with $\mathcal{U}_{in} = \{a, a, d, d\}$. If we apply the first transformation $(\mathcal{I}^1, \mathcal{O}^1) = (\{a, a\}, \{c\})$ (from two units of a produce one unit of c), the auctioneer ends up with $\mathcal{M}^1 = \{c, d, d\}$. Formally, we can express this operation as an equation over multisets:

$$\mathcal{M}^1(g) = \mathcal{U}_{in}(g) + \mathcal{O}^1(g) - \mathcal{I}^1(g)$$

The application of the transformation is possible only because two units of good a are available. This condition maps to:

$$\mathcal{U}_{in}(g) \geq \mathcal{I}^1(g)$$

Let $\mathcal{M}^m \in \mathbb{N}^G$ be the goods held by the auctioneer after applying the mth transformation. We can generalise the two equations above as follows (let $\mathcal{M}^0 = \mathcal{U}_{in}$):

$$\mathcal{M}^m(g) = \mathcal{M}^{m-1}(g) + \mathcal{O}^m(g) - \mathcal{I}^m(g) \quad (1)$$

$$\mathcal{M}^{m-1}(g) \geq \mathcal{I}^m(g) \quad (2)$$

We are now ready to define under what circumstances a sequence of transformations constitutes a valid solution:

Definition 6 (Valid solution) *Given a multiset $\mathcal{U}_{in}$ of available goods and a multiset $\mathcal{U}_{out}$ of required goods, an allocation sequence Σ for a given set of XOR-bids over transformations t_{ijk} is said to be a valid solution iff:*

(1) Σ either contains all or none of the transformations belonging to the same atomic bid. That is, the semantics of the BID-operator is being respected:

$$t_{ijk} \in \Sigma \;\Rightarrow\; t_{ijk'} \in \Sigma$$

(2) Σ does not contain two transformations belonging to different atomic bids by the same bidder. That is, the semantics of the XOR-operator is being respected:

$$t_{ijk}, t_{ij'k'} \in \Sigma \;\Rightarrow\; j = j'$$

(3) Equations (1) *and* (2) *hold for each transformation $(\mathcal{I}^m, \mathcal{O}^m)$ in Σ and each good $g \in G$. This condition ensures that all transformations have enough input goods available.*

(4) The set of goods held by the auctioneer after implementing the transformation sequence is a superset of the goods the auctioneer is expected to end up with:

$$\mathcal{M}^{|\Sigma|}(g) \geq \mathcal{U}_{out}(g)$$

The *revenue* for the auctioneer associated with a valid solution Σ is the sum of the prices associated with the selected atomic bids: $\sum\{p_{ij} \mid \exists k : t_{ijk} \in \Sigma\}$. We are now ready to define the WDP for XOR-bids:

Definition 7 (WDP) *Given a set of XOR-bids and multisets $\mathcal{U}_{in}$ and $\mathcal{U}_{out}$ of initial and final goods, respectively, the winner determination problem is the problem of finding a valid solution Σ that maximises revenue for the auctioneer.*

3.3 Integer Programming Formulation

We now show how to map the WDP defined above into integer programming (IP). Our aim is to find a solution sequence composed of transformations that leads from the initial goods to the final goods. Note that the length of the solution sequence can be at most equal to the overall number of offered transformations. Therefore, the issue is to decide for each transformation whether it is selected for the solution sequence, and if so, to choose its position in the solution sequence. Thus, we define a set of binary decision variables $x^m_{ijk} \in \{0,1\}$, where x^m_{ijk} takes on value 1 if the transformation t_{ijk} is selected at the mth position of the solution sequence, and 0 otherwise. Here and in what follows, m always ranges from 1 to the overall number of transformations $|T|$; i ranges over all bidders; j ranges for each bidder i from 1 to the number of atomic bids submitted by i; and k ranges for each atomic bid j of bidder i from 1 to the number of transformation in that atomic bid.

We also introduce several sets of auxiliary binary decision variables: x^m takes on value 1 iff any transition at all is selected at the mth position of the solution sequence; x_{ijk} takes on value 1 iff transition t_{ijk} is present anywhere in the sequence; and x_{ij} takes on value 1 iff any of the transformations in the jth atomic bid of bidder i are selected.

In what follows, we define the set of constraints that the solution sequence must fulfil:

(1) Since each atomic bid is a bundle of transformations, we want to ensure that if a transformation in an atomic bid is selected for the solution, so are the rest of the transformations in that bid (cf. condition (1) of Definition 6):

$$x_{ij} = x_{ijk} \quad (\forall ijk) \tag{3}$$

(2) We enforce that the atomic bids submitted by each bidder are exclusive (XOR). This amounts to satisfying the following constraints (cf. condition (2) of Definition 6):

$$\sum_j x_{ij} \leq 1 \quad (\forall i) \tag{4}$$

(3) We enforce that a transformation can be selected at most for a single position in the solution sequence:

$$x_{ijk} = \sum_m x^m_{ijk} \quad (\forall ijk) \tag{5}$$

(4) We also impose that at most one transformation is selected at each position of the sequence:

$$x^m = \sum_{ijk} x^m_{ijk} \quad (\forall m) \tag{6}$$

(5) Furthermore, there should be no gaps in the sequence:

$$x^m \geq x^{m+1} \quad (\forall m) \tag{7}$$

(6) Next, we capture condition (3) of Definition 6. Firstly, the multiset of goods held by the auctioneer after performing m steps of the transformation sequence can be computed recursively, by equation (1), as follows:

$$\mathcal{M}^m(g) = \mathcal{U}_{in}(g) + \sum_{\ell=1}^{m} \sum_{ijk} x^\ell_{ijk} \cdot (\mathcal{O}_{ijk}(g) - \mathcal{I}_{ijk}(g)) \quad (\forall g \in G, \forall m) \tag{8}$$

That is, we treat each $\mathcal{M}^m(g)$ as an integer decision variable. We can now formulate the constraint enforcing that enough goods must be available at step m to perform the next transformation (cf. equation (2)):

$$\mathcal{M}^{m-1}(g) \geq \sum_{ijk} x^m_{ijk} \cdot \mathcal{I}_{ijk}(g) \quad (\forall g \in G, \forall m) \tag{9}$$

(7) And finally, after having performed all the selected transformations, the set of goods held by the auctioneer must be a superset of the final goods $\mathcal{U}_{out}$ (cf. condition (4) of Definition 6):

$$\mathcal{M}^{|T|}(g) \geq \mathcal{U}_{out}(g) \quad (\forall g \in G) \tag{10}$$

This works correctly, because $\mathcal{M}^{|T|}(g) = \mathcal{M}^m(g)$ for the highest m with $x^m = 1$ (cf. equations (6) and (8)).

Now solving the WDP for MMUCAs with XOR-bids amounts to solving the following integer program:

$$\max \sum_{ij} x_{ij} \cdot p_{ij} \quad \text{subject to constraints (1)–(7)}$$

Finally, a valid solution according to Definition 7 is obtained from the solution of the IP by making transition t_{ijk} the mth element of the solution sequence Σ iff $x^m_{ijk} = 1$.

Observe that our proposed implementation can easily be amended so as to directly encode the constraints imposed by language constructs other than the XOR-operator. This would remove the need for translating into the XOR-language first and thereby greatly improve efficiency.

3.4 Computational Complexity

The (decision problem underlying the) WDP for standard combinatorial auctions is known to be NP-complete, with respect to the number of goods [Rothkopf *et al.*, 1998]. NP-hardness can, for instance, be shown by reduction from the well-known SET PACKING problem. As our mixed auction model generalises standard combinatorial auctions, winner determination remains NP-hard also here. NP-membership (and thereby NP-completeness) of the problem of checking whether there exists a solution exceeding a given revenue (for finite bids) follows from the fact that a candidate solution provided by an oracle can always be verified in polynomial time. That is, despite of the generalisations we have introduced, the computational complexity of the WDP does not increase, at least not with respect to the polynomial hierarchy.

3.5 Mechanism Design

An important issue in auction design concerns their *game-theoretical* properties. We note here that the central result in *mechanism design*, on the incentive-compatibility of the Vickrey-Clarke-Groves (VCG) mechanism, carries over from standard combinatorial auctions to MMUCAs. Recall that the VCG mechanism allocates goods in the most efficient manner and then determines the price to be paid by each bidder by subtracting from their offer the difference of the overall value of the winning bids and the overall value that would

have been attainable without that bidder taking part. That is, this "discount" reflects the contribution to the overall production of value of the bidder in question. The VCG mechanism is strategy-proof: submitting their true valuation is a (weakly) dominant strategy for each bidder. As an inspection of standard proofs of this result reveals [Mas-Colell *et al.*, 1995], this does not depend on the internal structure of the agreements that agents make. Hence, it also applies to MMUCAs.

4 Conclusions and Related Work

Our model of mixed multi-unit combinatorial auctions subsumes a range of combinatorial auction models discussed in the literature (see e.g. [Sandholm *et al.*, 2002]), in particular single-unit and multi-unit versions of the standard direct and reverse auctions (simply do not use either $\mathcal{I}$ or $\mathcal{O}$). Transformability relationships between goods (at the side of the auctioneer) as proposed by Giovannucci *et al*. [2005] can be modelled by allowing the auctioneer to submit bids representing those transformations to itself.

We should stress that there are important differences between our mixed auctions and models known as *double auction* [Wurman *et al.*, 1998] or *combinatorial exchanges* [Sandholm *et al.*, 2002]. The most important difference is that these models do not have the concept of a *sequence* of exchanges, which is required if the intention is to model some sort of production process. In the formulation of the WDP for combinatorial exchanges given by Sandholm *et al*. [2002], for instance, accepting "circular" bids such as $\text{BID}(a, b, 10)$ and $\text{BID}(b, ac, 10)$, to obtain c for 20 rupees, would be considered a solution. With our semantics in mind, however, this solution is *not* valid: the first agent needs to receive a before it can produce b, but the second agent needs to receive b before it can produce a and c. Hence, no deal should be possible. In fact, the MMUCA can be used to simulate combinatorial exchanges (and double auctions). For instance, the bid $\text{BID}(a, b, 10)$ can be rewritten as $\text{BID}(\{(a, \{\}), (\{\}, b)\}, 10)$ to express that I will only deliver b if I receive a, but that the order does not matter. Of course, if no true transformations (imposing an order) are used, then the simpler model of combinatorial exchanges is to be preferred.

Walsh and Wellman [2003] and Babaioff and Walsh [2005] tackle a similar problem to ours, focussing on supply chain formation. Although both contributions are very significant, they could be extended along three dimensions. Firstly, they do not allow a provider to submit bids on combinations of transformations. Secondly, they do not define a bidding language (in fact, their agents submit a bid with a single transformation each). Finally, the transformation net that defines the supply chain has to fulfil strict criteria: acyclicity, transformations can only produce one output good, etc. Both papers mainly adopt a game-theoretic perspective, and thus do not address either the WDP or bidding languages in depth.

The first systematic study of bidding languages is due to Nisan [2006] (an early version appeared in 2000). Nisan's paper provides an excellent introduction to the topic and has clarified a number of issues that have previously remained somewhat fuzzy. For MMUCAs, we have seen that the XOR-language is fully expressive (over finitely-peaked valuations). Future work should also address the expressive power of different fragments of the bidding language and compare the succinctness of different fragments for certain classes of valuations: which languages can express what valuations, and which languages can do so using less space than others? For the case of direct single-unit combinatorial auctions, several such results are given by Nisan [2006], and some of these results may be relatively easy to transfer to our model.

Another interesting question to consider in future work would be what exactly the auctioneer should *announce* when opening an MMUCA. In the case of direct auctions this is the set of goods to be sold. If bidding for transformations is possible, however, it may be difficult to foresee what types of goods will be relevant to a solution, as this depends on the transformation capabilities of the bidders in the market. We also envision as a possible development the application of our model to supply chain formation. Finally, our work also poses a computational challenge since the number of variables of our IP grows quadratically with the number of transformations mentioned in the bids. Thus, we plan to investigate the use of special-purpose or local algorithms.

This work has been partially supported by the Spanish Ministry of Education and Science (grants 2006-5-0I-099, TIN-2006-15662-C02-01, and TIP-2003-08763-C02-01).

References

[Babaioff and Walsh, 2005] M. Babaioff and W.E. Walsh. Incentive-compatible, budget-balanced, yet highly efficient auctions for supply chain formation. *Decision Support Systems*, 39:123–149, 2005.

[Giovannucci *et al.*, 2005] A. Giovannucci, J.A. Rodríguez-Aguilar, and J. Cerquides. Multi-unit combinatorial reverse auctions with transformability relationships among goods. In *Proc. WINE-2005*. Springer-Verlag, 2005.

[Mas-Colell *et al.*, 1995] A. Mas-Colell, M.D. Whinston, and J.R. Green. *Microeconomic Theory*. Oxford University Press, 1995.

[Nisan, 2006] N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006.

[Rothkopf *et al.*, 1998] M.H. Rothkopf, A. Pekeč, and R.M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[Sandholm *et al.*, 2002] T.W. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *Proc. AAMAS-2002*. ACM Press, 2002.

[Walsh and Wellman, 2003] W.E. Walsh and M.P. Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Journal of Artificial Intelligence Research*, 19:513–567, 2003.

[Wurman *et al.*, 1998] P.R. Wurman, W.E. Walsh, and M.P. Wellman. Flexible double auctions for electionic commerce: Theory and implementation. *Decision Support Systems*, 24:17–27, 1998.