Towards a Realistic Bid Generator for Mixed Multi-Unit Combinatorial Auctions

Meritxell Vinyals[1], Andrea Giovannucci[1], Jesús Cerquides[2], Pedro Meseguer[1], and Juan Antonio Rodriguez-Aguilar[1]

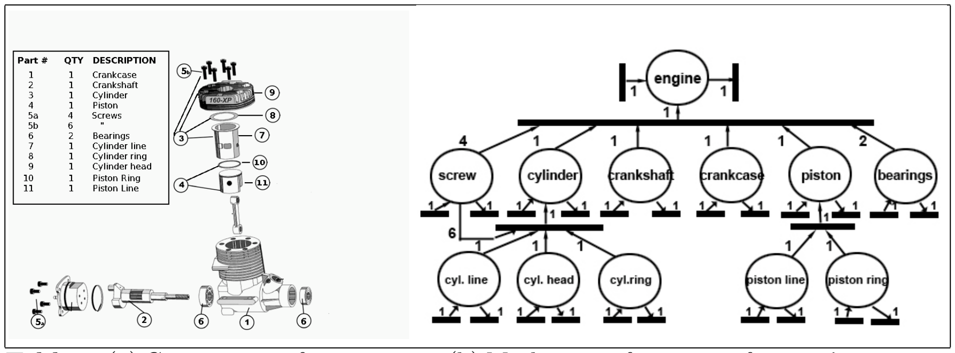
[1] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
Campus UAB, 08193 Bellaterra, Spain
{meritxell|andrea|pedro|jar}@iiia.csic.es
[2] WAI, Dep. Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
Gran Via 585, 08191 Barcelona, Spain
cerquide@maia.ub.es

Abstract. Mixed Multi-Unit Combinatorial Auctions extend and generalise all the preceding types of combinatorial auctions. In this paper, we try to make headway on the practical application of MMUCAs by: (1) providing an algorithm to generate artificial data that is representative of the sort of scenarios a winner determination algorithm is likely to encounter; and (2) subsequently assessing the performance of an Integer Programming implementation of MMUCA on CPLEX.

1 Introduction

A combinatorial auction (CA) is an auction where bidders can buy (or sell) entire bundles of goods in a single transaction ([1]). Selling goods in bundles has the great advantage of eliminating the risk for a bidder of not being able to obtain complementary goods 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.

Central to CAs are the issues of *winner determination*(WD) and *bidding*. Winner determination is the problem, faced by the auctioneer, of choosing which goods to award to which bidder so as to maximise its revenue. The WD problem for standard CAs is known to be $\mathcal{NP}$-complete, with respect to the number of goods [6]. $\mathcal{NP}$-hardness can, for instance, be shown by reduction from the well-known Set Packing problem. Bidding is the process of transmitting one's valuation function over the set of goods on offer to the auctioneer through some *bidding language*[5]. Under an *OR-language*, if a particular bidder submits several



Part #	QTY	DESCRIPTION
1	1	Crankcase
2	1	Crankshaft
3	1	Cylinder
4	1	Piston
5a	4	Screws
5b	6	"
6	2	Bearings
7	1	Cylinder line
8	1	Cylinder ring
9	1	Cylinder head
10	1	Piston Ring
11	1	Piston Line

Table 1. (a) Components of a car engine. (b) Market transformations for a car's engine.

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. If we use an *XOR-language* instead, that means that only one of the atomic bids can be accepted. The advantage of an XOR-language is that it allows to express not only *complementarity* between goods (value of a bundle being greater than the values of its parts), likewise an OR-language, but also *substitutability* (value of a bundle being less than the sum of its parts). Although an XOR bidding language is known to be fully expressive [5, 3], for some types of CAs its usage makes that even finding a feasible solution is $\mathcal{NP}$-complete [7].

In [3] we introduce a generalisation of the standard model of CA. This new auction model integrates direct and reverse auctions, i.e. the auctioneer can buy and sell goods within a single auction. It also incorporates the idea of *transformability* relationships between goods by allowing agents not only to bid for goods but also 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). To illustrate the operation of MMUCA, consider as an example the assembly of a car's engine, whose structure is depicted in Fig. (a) in table 1. Notice that each part in the diagram, in turn, is produced form further components or raw materials. For instance, a cylinder ring (part 8) is produced by transforming some amount of stainless steel with the aid of an appropriate machine. Therefore, there are several production levels involved in the making of a car's engine. A MMUCA allows to run an auction where bidders can bid over bundles of parts, bundles of transformations, or any combination of parts and transformations. Notice that the result of an MMUCA WD algorithm would be an ordered sequence of bids making explicit how bidders coordinate to progressively trasform goods till producing engines as final products. Therefore, an MMUCA would allow to assemble a supply chain from bids.

Despite its potential for application, and unlike CAs, little is known about the practical application of MMUCAs since no empirical results have been reported on any WD algorithms. These results are unlikely to come up unless, and along the lines of the research effort carried out in CAs [4], researchers are provided with algorithms or test suites to generate artificial data that is representative of the auction scenarios a WD algorithm is likely to encounter. Hence, WD algorithms could be accurately tested, compared, and improved. In this paper, we try to contribute to the practical application of MMUCAs along two directions. Firstly, we provide an algorithm to generate artificial data sets that are representative of the sort of scenarios a WD algorithm is likely to encounter. Secondly, we employ such algorithm to generate artificial data and subsequently assess the performance of an Integer Programming (IP) implementation of a WD solver for MMUCA on CPLEX.

The paper is structured as follows. In section 2 we provide some background on MMUCAs. Next, in section 3 we analyse the required features of an artificial data set generator for MMUCAs whose algorithm is detailed in section 4. In sections 5 and 6, we analyse some early, empirical results of an IP formulation of the WDP, draw some conclusions and outline paths to future research.

2 Background

Next, we introduce MMUCA by summarising the work in [3, 2]. Let G be the finite set of all types of goods. 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 negotiate 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.

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, valuation $v(\{(\{\mathit{line}, \mathit{ring}, \mathit{head}, \mathit{6} \cdot \mathit{screws}, \mathit{screwdriver}\}, \{\mathit{cylinder}, \mathit{screwdriver}\})\}) = -10$ means that some agent can assemble a cylinder for \$10 when provided with a cylinder line, a cylinder ring, a cylinder head, six screws, and a screwdriver, and returns the screwdriver once done[3].

An *atomic bid* $\text{BID}(\{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}, p)$ specifies a bundle (as a finite multiset) of finite transformations and a price. For instance, following the example above, $\text{BID}(\{(\{\mathit{line}, \mathit{ring}, \mathit{head}, \mathit{6} \cdot \mathit{screws}, \mathit{screwdriver}\}, \{\mathit{cylinder}, \mathit{screwdriver}\})\}, -10)$ stands for an atomic bid to assemble a cylinder for \$10. Therefore, atomic bids are the basic expressions of the bidding language that a bidder can employ to transmit his valuations to the auctioneer. More complex bids can be obtained

[3] We use $6 \cdot \mathit{screws}$ as a shorthand to represent six identical elements in the multiset.

by composing atomic bids with the aid of an XOR-language. Thus, an XOR-combination of bids expresses that the bidder is prepared to accept at most one of them. Notice that the XOR language is fully expressive since it allows to express both substitutability and complementarity. On the one hand, substitutability is readily achieved by combining atomic bids into XOR combinations; whereas complementarity is achieved by putting together transformations into the very same bundle in an atomic bid (the valuation of the bundle allows to express either super-additivity, sub-additivity, or simply the addition of the valuations of the goods involved).

The *input* to the MMUCA 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. A *valid solution* to the WDP will be a *sequence of transformations* satisfying that: (1) the multiset of transformations in the sequence has to *respect the bids* submitted by the bidders; and (2) the set of goods held by the auctioneer in the end is a superset of $\mathcal{U}_{out}$. Namely, the first requirement involves that a valid solution must guarantee that for every transformation in the sequence the preceding transformations provide its input goods, otherwise the transformation cannot be employed (e.g. in figure (a) in Table 1 a transformation to assemble an engine cannot be employed unless other transformation provide its parts). Furthermore, the second requirement demands that the auctioneer ends up the auction with the goods he requested (e.g. if the auctioneer requests 2 engines, a sequence of transformations leading to the production of 1 engine would not be enough).

For the formal definition of the WDP, we restrict ourselves to bids in the XOR-language, which is known to be fully expressive (as proved by Cerquides et al. [3]). Therefore, solving the WDP for MMUCAs with XOR-bids amounts to maximise the following objective function:

$$\sum_{b \in B} x_b \cdot p_b$$

subject to requirements (1) and (2) informally defined above[4]. Here B stands for the set of all atomic bids, x_b stands for a binary decision variable indicating whether bid b is selected or not, and p_b stands for the price of bid b. As noticed in [3], the number of decision variables of an IP to solve a MMUCA WDP is of the order of $|T|^2$, where $|T|$ is the overall number of transformations within all bids. As the reader may notice, this represents a serious computational cost as the number of transformations grow.

Bids in MMUCAs are composed of transformations. Each transformation expressses either an offer to buy, to sell, or to transform some good(s) into (an)other good(s). Since transformations are the building blocks composing bids, we must firstly characterise the types of transformations a bid generator may need to construct in order to produce bids. Our analysis of transformations has led to a classification into three types, namely:

[4] The interested reader should refer to [3] for a thorough account of how these requirements map into a set of side constraints for the optimisation problem.

1. **Output transformations** are those with no input good(s). Thus, an O-transformation represents a bidder's offer to sell some good(s). Besides, an O-transformation is equivalent to a bid in a reverse CA.
2. **Input transformations** are those with no output good(s). Thus, an I-transformation represents a bidder's offer to buy some good(s) . Notice that an I-transformation is equivalent to a bid in a direct CA.
3. **Input-Output transformations** are those whose input and output good(s) are not empty. An IO-transformation stands for a bidder's offer to deliver some good(s) after receiving some other good(s): *I can deliver $\mathcal{O}$ after having received $\mathcal{I}$*. They can model a wide range of different processes in real-world situations (e.g. assembly, transformation, or exchange).

Figure (b) in Table 1 presents samples of each transformation type. In the figure, horizontal, black bars stand for transformations, circles stand for goods, and directed arrows from goods into or from transformations represent the goods input into or produced out of a transformation. Thus, for instance, we differentiate an I-transformation to consume a piston, an O-transformation to give away a piston, and an IO-transformation giving away a piston after receiving a piston ring and a piston line. Notice that any bid in a MMUCA results as a combination of transformations of the above-listed types. Therefore, a bid generator for MMUCA must support the generation of transformations of all these types.

3 Bid Generator Requirements

In order to test and compare MMUCA WD algorithms, researchers must be provided with algorithms or test suites to generate artificial data that is representative of the auction scenarios a WD algorithm is likely to encounter. Hence, WD algorithms can be accurately tested, compared, and improved. Unfortunately, we cannot benefit from any previous results in the literature since they do not take into account the notion of transformation introduced in [3, 2]. In this section we make explicit the requirements for a bid generation technique considering that in MMUCA agents trade transformations instead of goods.

A naive approach to artificial bid generation would be to create bids totally at random. It is easy to see that this approach would generate unrealistic bids, from which it would be hard to draw conclusions. Let us consider a random bid $b = \{\mathcal{I}, \mathcal{O}, p\}$. If goods appearing in sets $\mathcal{I}$ and $\mathcal{O}$ are selected randomly, there is little chance that they will represent a realistic transformation. Also, if p is chosen randomly, it will not be related with the actual values of the goods in the sets $\mathcal{I}$ and $\mathcal{O}$ and consequently the transformation would be either too profitable or too expensive for the auctioneer, unrealistically easing the problem.

If individual bids randomly generated may be unrealistic, sets of random bids also present similar drawbacks. Let us consider two bids b_1 and b_2 by different bidders. In a real MMUCA, there is a high chance that two bids involve the same goods (or the same type of goods; both are offering close transformations with small variations in price). However, if b_1 and b_2 are generated totally at random, there is little chance that they will contain similar goods or that their prices are related.

These two simple examples clearly show that totally random bid generation originates unrealistic scenarios. Testing WD algorithms on these scenarios is basically useless, because any extracted conclusion cannot be used in real settings. The bid generator has to satisfy a number of requirements to make the artificial bids close to the bids that are likely to appear in a real-world auction.

Since MMUCAs generalise CAs, as discussed in [3], our approach is to depart from artificial data sets generators for CAs, keeping the requirements summarised in [4], namely: (1) There is a finite set of goods ; (2) Certain goods are more likely to appear together than others; (3) The number of goods in a bundle is often related to which goods compose the bundle; (4) Valuations are related to which goods appear in the bundle, (5) Valuations are related to which goods appear in the bundle; (6) Valuations can be configured to be subadditive, additive or superadditive in the number of goods requested; and (6) Sets of XOR'ed bids are constructed on a per-bidder basis.

Notice though that the requirements above must be reformulated, and eventually extended, in terms of transformations since a bidder in a MMUCA bids over a bundle of transformations, whereas a bidder in a CA bids over a bundle of goods. Hence, in what follows we discuss the CA requirements listed above reformulated for MMUCA and some new requirements derived from trading with transformations:

- **There is a finite set of transformations** A CA generator bundles goods from a given set of goods to construct bids. What is the set of transformations from which a MMUCA generator constructs bids? In order to provide a proper answer we must take inspiration on realistic scenarios faced by buyers and providers. If so, within a given market we expect several producers to offer the very same or similar services (transformations) at different prices, as well as several consumers to require the very same or similar services (transformations) valued at different prices. In other words, within a given market we can identify a collection of common services that companies request and offer. For instance, in the example in Fig. (a) in table 1, several providers may offer to assemble a cylinder through the very same transformation: $t = (\{6 \cdot screws, 1 \cdot cylinder\_line, 1 \cdot cylinder\_rig, 1 \cdot cylinder\_head\}, \{cylinder\})$. Eventually, a provider may either offer to perform such transformation several times (e.g. as many times as cylinders are required), or to bundle it with other transformations, or the two. Hereafter, we shall consider the common goods and services in a given market to be represented as a collection of transformations that we shall refer to as market transformations. Therefore, market transformations represent the "goods" providers and buyers can request and bid for. Hence, bids for MMUCAs shall be composed as combinations of market transformations. In our generator, the set of market transformations is always finite and includes at least two market transformations for every good in G, ensuring that every good is individually available to buy and/or sell. As an example, Fig. (b) in table 1 depicts a sample of market transformations if intending to build the car engine in Fig. (a) in table 1.

– **Certain transformations are more likely to appear together than others.** In any market services and goods are related to each other. For example, the production process for a good can also generate some other products that can be sold with it or used in another industrial process. Also, some services or products are usually bought together by the final customer. In order to deal with this requirement, our generator uses a simple Markov model, that modifies the probabilities of the next transformation to include to the bundle depending on the last transformation that added.
– **The number of transformations in a bundle is often related to which goods compose the bundle.** Since bids are composed as combinations of market transformations, we must introduce the notion of *transformation multiplicity* as the counterpart of good multiplicity (the number of units of a given good within an offer or a request). Say that in a CA a bidder submits a bid for the goods in multi-set $\{engine, engine, piston\}$. It is clear that the multiplicity in this bundle of good *engine* is two, whereas the multiplicity of good *piston* is one. But things become more complicated when we consider transformations because the multiplicity of a given transformation must be defined in terms of another transformation, which in turn is composed of multiple input and output goods. Intuitively, we say that a transformation is a multiple of another one if both share the same input and output goods and the former has more input and output goods than the latter but keeping the same ratio between input and output goods. For instance, given transformations $t = (\{6 \cdot screws, 1 \cdot cylinder\_line, 1 \cdot cylinder\_ring, 1 \cdot cylinder\_head\}, \{cylinder\})$ and $t' = (\{12 \cdot screws, 2 \cdot cylinder\_line, 2 \cdot cylinder\_ring, 2 \cdot cylinder\_head\}, \{2 \cdot cylinder\})$ we way that t' has multiplicity two with respect to t. Thus, in our generator we fulfill the requirement that *the number of transformations in a bundle is often related to which goods compose the bundle* by providing each market transformation with a probability distribution over the set of multiplicities in which it is likely to appear.
– **Valuations are related to which transformations appear in the bundle; furthermore transformation valuations keep consistency with respect to bidder valuations for goods involved in each transformation.** A further issue has to do with the way bidders value transformations and bundles of transformations. Notice that performing a transformation to assemble the engine in Fig. (a) in table 1 results in a new product that has more market value than its parts. Therefore, a car maker values the transformation according to his expected benefits, namely the difference between the expected market value of the engine and the cost of its parts. Therefore, if the parts cost \$850 and the expected market value of the engine is \$1000, the car maker should be willing to offer to pay less than \$150 for the transformation. On the other hand, any provider is expected to request less than \$150 in order to perform the transformation. In general, buyers and providers in a MMUCA should value a transformation on the basis of the difference between the expected market value of its output goods and the

cost of its input goods. Notice though that we are not assuming here that such difference must be always positive. Likewise bidder should value bundles of transformations considering the prices of transformations included in it.

- **Appropiate valuations can be configured to be subadditive, additive or superadditive in the number of transformations requested** This requirement tries to capture the multiplicity-based (volume-based) discounts policies that are applied in real world. Significant discounts are applied in real markets when goods and services are traded at certain number of units. For example in Fig. (a) in table 1, we observe that screws are usually traded in higher quantities than full engines. Thus, not surprisingly the same (percentage) discount may apply to an offer for 100 screws than to an offer for 5 engines. Hence, an offer to produce more than 5 engines, though more unlikely, should reflect higher discounts.
- **Sets of XOR'ed bids are constructed on per-bidder basis** When a bidder submits different bids in an XOR bid he declares they are mutually exclusive offers, expressing substitutability. For example, the following offer $\text{BID}_1(\{(\{\textit{engine}\}\{\})\}, 100)$ XOR $\text{BID}_2(\{(\{\textit{2} \cdot \textit{engine}\}\{\})\}, 190)$ stands for a bidder that offers to buy two engines or one engine but in any case three engines. On the other hand when a bidder expresses complementarity he translates the OR bids as XOR bids. For example if a bidder wants to buy one engine or one cylinder he submits the following XOR-bid: $\text{BID}_1(\{(\{\textit{engine}\}\{\})\}, 100)$ XOR $\text{BID}_2(\{(\{\textit{cylinder}\}\{\})\}, 30)$ XOR $\text{BID}_3(\{(\{\textit{cylinder}, \textit{engine}\}\{\})\}, 140)$. As you can observe in both cases bids submitted in the same XOR bid are likely to have similarities and , consequently, combining bids into XOR bids randomly does not capture this property.
- **Unrequested goods by the auctioneer may become involved in the auction.** Finally, we add a last requirement stems from the fact that, unlike auctioneers in CAs, not all goods involved in a MMUCA must be requested by the auctioneer. Back to our example of a car maker in need of engines as depicted in Fig. (a) in table 1, it can run a MMUCA only requesting engines. Thereafter, bidders may offer already-assembled engines, or other goods (e.g. parts like crankcases, crankshafts, or screws) that jointly with transformations over such goods help produce the requested goods.

Requirements summary. Following the analysis above, we can reformulate the requirements for an artificial data set generator for CAs to obtain the requirements for an artificial data set generator for MMUCAs:

1. There is a finite set of transformations
2. Certain transformations are more likely to appear together than others
3. The number of transformations in a bundle is often related to which transformations compose the bundle
4. Valuations are related to which transformations appear in the bundle; furthermore, transformation valuations keep consistency with respect to bidders valuations for goods involved in each transformation

5. Valuations can be configured to be subadditive, additive or superadditive in the number of transformations requested
6. Sets of XOR'ed bids are constructed on a per-bidder basis
7. Unrequested goods by the auctioneer may become involved in the auction

These requirements are the extended versions for MMUCAs of the requirements for CAs detailed at the beggining of this section.

4 An Algorithm for Artificial Data Set Generation

In what follows we describe a bid generation algorithm that automates the generation of artificial data sets for MMUCA while capturing the requirements above. The algorithm's purpose is to generate MMUCA WDP (each one composed of a collection of XOR bids and the set of goods available to and requested by the auctioneer) that can be subsequently fed into an MMUCA WD algorithm.
The algorithm starts by generating the set of goods involved in MMUCA. Next, it generates the goods the auctioneer requests. After that, it creates a subset of atomic transformations, which are the market transformations to employ for bid generation. Thereafter, it generates bids as linear combinations of market transformations, which are subsequently priced according to a pricing policy. The resulting bids are further composed into XOR (mutually exclusive) bids because the XOR language is a fully expressive language allowing bidders to express all their preferences in a single XOR bid [3][5]. Hence, our algorithm assumes that each bidder formulates a single XOR bid, being the number of bidders equal to the number of XOR bids.

Good Generation. This process requires the number of different goods (n_{goods}) involved in an auction along with the maximum price any good can take on ($maxPrice$). Based on these values, it assesses for each good g: (1) its average market price (μ_g) drawn from a uniform distribution $U[1, maxPrice]$ where $maxPrice$ stands for the maximum market price any good can take on; and (2) the distribution to assess its multiplicity, or more precisely, the success probability ($g_{geometric}$) of a geometric probability distribution from which the good multiplicity can be drawn.

Requested Goods Generation. This process assesses the number of units of each good the auctioneer requests, namely the multiset $\mathcal{U}_{out}$. Since the auctioneer must not request all goods, this process selects a subset of the goods in G to be part of $\mathcal{U}_{out}$. Firstly, it determines whether a good g is requested by the auctioneer by comparing the value drawn from a uniform distribution $U[0, 1]$ with $p_{good\_requested}$, the probability of adding a new good to $\mathcal{U}_{out}$. Once included a given good g, the number of units requested for g is drawn from a geometric distribution with the success probability $g_{geometric}$ obtained by the good generation process. Notice that by selecting a subset of the goods we fulfil

[5] Here we only provide the bid generation algorithm. The interested reader must refer to [8] for a description of all algorithms required by the artificial data set generator.

the requirement 7 listed in section 3 *unrequested goods by the auctioneer may be involved in the auction.*

Market Transformations Generation. This process generates *a finite set of transformations* to be employed as the building blocks to subsequently compose bids and consequently fullfilling the requirement 1 listed in section 3. For each good, this procedure constructs two market transformations (one I-transformation and one O-transformation). Each transformation involves a single good with multiplicity one. For instance, $(\{engine\},\{\})$ and $(\{\},\{engine\})$ stand respectively for the I-transformation and O-transformation for good *engine*. After that, the algorithm generates a limited number of market IO-transformations ($n_{IO\_market\_transformations}$). In order to generate each market IO-transformation, this procedure chooses the goods to include in its input and output set employing the probabilities of adding some good to the input and output set respectively ($p_{good\_in\_input}$ and $p_{good\_in\_output}$). Once included a good to either the input or output set, its multiplicity is calculated from a geometric distribution parametrised by $g_{geometric}$.

Finally, there is the matter of attaching to each market transformation a probability distribution to draw its multiplicity. We assume that the bid generation process, detailed by algorithm 1, is to employ a geometric distribution for each market transformation to calculate its multiplicity. Hence, the generation of market transformations assesses the success probability to be employed by such geometric distributions, namely the probability of adding an extra unit of a transformation already included in a bundle bid. Thus, each transformation t is assigned a success probability $t_{geometric}$. However, we cannot randomly generate success probabilities because transformations are defined over multisets of goods, and therefore we must keep consistency with respect to the success probabilities assigned to each good by the good generation process. Therefore, we propose to set the success probability for each transformation as follows. Given a transformation $t = (\mathcal{I}, \mathcal{O})$, for each good involved in the transformation, g, we assess its probability of having multiplicity $|m_{\mathcal{I}}(g) - m_{\mathcal{O}}(g)|$, where $m_{\mathcal{I}}(g)$ (respectively $m_{\mathcal{O}}(g)$) stands for the number of occurrences of g in $\mathcal{I}$ (respectively $\mathcal{O}$). We set the success probability of t to the minimum of these probabilities as follows: $t_{geometric} = min_{g \in \mathcal{G}} g_{geometric}^{|m_{\mathcal{I}}(g) - m_{\mathcal{O}}(g)|}$.

Bid Generation. The bid generation algorithm (algorithm 1) generates bids that are subsequently combined into XOR bids, each one encoding the offer or request of a bidder. This process makes explicit which transformations and how many of them to offer/request in a bundle, how to price the bundle, and which bids to combine in an XOR bid.

Which transformations are requested in a bundle and in what number of units. Firstly, for each XOR bid ($XORBid$) the algorithm composes each bid (Bid) by combining the market transformations ($\mathcal{MTS}$) returned by the market transformation generation process. The number of market transformations ($nTransfBid$) to compose each bid is obtained from a normal distribution $\mathcal{N}(\mu_{add\_new\_transformation}, \sigma_{add\_new\_transformation})$ (line 12). Market transformations are chosen from the set of market transformations ($\mathcal{MTS}$) and their multiplicity in the bundle bid is obtained from a geometric distribution with success probability $t_{geometric}$ (line 15-16). By assessing the number of units to include in a bundle using a probabilistic distribution that depends on each transformation we fulfil the requirement 3: *the number of transformations in a bundle is often related to which transformations compose the bundle.* We also consider that given an existing bundle not all transformations are equally likely to be requested because *certain transformations (for which some complementarities exist) will be more likely to appear together than others* as stated by requirement 2 in section 3. To ease these complementarities we assume the Markov property that the probability of adding a new market transformation to an existing bundle only depends on the last transformation added and not on the whole bundle. The markov model is a matrix $\#\mathcal{MTS} \times \#\mathcal{MTS}$ with the following properties: An state $\mathcal{MT}_i$ is defined as an state where the last market transformation added

Algorithm 1 Bid Generation(MTS, n_{XOR_bids}, μ, σ_{prices}, $\mu_{add\_new\_XOR\_clause}$, $\sigma_{add\_new\_transformation}$, $\mu_{add\_new\_transformation}$, $\sigma_{add\_new\_transformation}$, α)

```
1: for $g = 1$ to $n_{goods}$ do
2:    for $b = 1$ to $n_{XOR\_bids}$ do
3:       $p_{prices\_bid}[b, g] \leftarrow \mu[g] \cdot N(1, \sigma_{prices})$
4:    end for
5: end for
6: $Bids \leftarrow \{\}$
7: for $b = 1$ to $n_{XOR\_bids}$ do
8:    $XORBid \leftarrow EmptyXORBid()$
9:    $nXORClauses \leftarrow \mathcal{N}(\mu_{add\_new\_XOR\_clause}, \sigma_{add\_new\_XOR\_clause})$
10:   for $x = 1$ to $nXORClauses$ do
11:      $Bid \leftarrow EmptyCombinatorialBid()$
12:      $nTransfBid \leftarrow \mathcal{N}(\mu_{add\_new\_transformation}, \sigma_{add\_new\_transformation})$
13:      if $x == 1$ then
14:         for $t = 1$ to $nTransfBid$ do
15:            $MT \leftarrow$ Select a transformation using Markov model from $MTS$ with state $MT$
16:            $multiplicity \leftarrow Geometric(MT.t_{geometric})$
17:            $T.inputs \leftarrow MT.inputs \cdot multiplicity$
18:            $T.outputs \leftarrow MT.outputs \cdot multiplicity$
19:            $T.price \leftarrow \sum_{g \in T.outputs} p_{prices\_bid}[b, g] - \sum_{g \in T.inputs} p_{prices\_bid}[b, g]$
20:            $p_{offer} \leftarrow (T.t_{geometric})^{multiplicity}$
21:            $discount \leftarrow \alpha \frac{1-e^{1-p_{offer}}}{1-e}$
22:            $Bid.t \leftarrow Bid.t \cup T$
23:            $Bid.price \leftarrow Bid.price + T.price \cdot (1 - discount)$
24:         end for
25:      else
26:         $model$ ←Randomly generate a number betwen 1 and x-1
27:         $Bid \leftarrow XORBid(model)$
28:         if $nTransfBid \geq length(XORBid(model).t)$ then
29:            $Bid \leftarrow removeRandomTransition(Bid)$
30:            $Bid \leftarrow recalculatePrices(Bid)$
31:         end if
32:         if $nTransfBid \leq length(XORBid(model).t)$ then
33:            $Bid \leftarrow addRandomTransition(Bid)$
34:            $Bid \leftarrow recalculatePrices(Bid)$
35:         end if
36:      end if
37:      $XORBid \leftarrow XORBid \cup \{Bid\}$
38:   end for
39:   $Bids \leftarrow Bids \cup \{XORBid\}$
40: end for
41: return Bids
```

to the bundle is $\mathcal{MT}_i$ and $\sum_{\mathcal{MT}_i \in \mathcal{MTS}, j=1}^{j=\#\mathcal{MTS}} P(\mathcal{MT}_i|\mathcal{MT}_j) = 1$. First market transformation is chosen randomly, next transformations are chosen using the probability distribution of the row i in the matrix (i: current state).

How to price the bundle. Next, the algorithm prices the transformation according to its multiplicity (lines 17-21). To fulfil valuations requirements listed in section 3, a pricing policy must provide the means to price a good, a transformation, multiple units of the very same transformation, and a bundle of transformations in a realistic manner. As to pricing goods, in order to vary prices among bidders, our algorithm generates a price for bidder b for good g, represented as $p_{prices\_bid}[b, g]$, from a normal distribution $\mathcal{N}(\mu[g], \sigma_{prices})$, where $\mu[g]$ stands for good g's average price in the market and σ_{prices} for the variance among bidders' prices (lines 2-4). Thereafter, a transformation's price for bidder b is assessed in terms of the difference from his valuation of its output goods with respect to his valuation of its input goods (line 19). Accordingly *transformation valuations keep consistency with respect to bidder valuations for goods involved in each transformation* as stated by requirement 5 in section 3. Each bid valuation is obtained by adding the prices of its transformations (line 23). Hence *valuations are related to which transformations compose the bundle* as stated by requirement 6 although varying among different bidders.
Furthermore we propose to introduce superadditivity by applying multiplicity-based discounts to transformations addressing the requirement that *valuations can be configured to be subadditive, additive o superadditive in the number of transformations requested.* In other words, as a general rule the more unlikely for a transformation to be traded at certain units (multiplicity), the higher the discount to apply to its overall price. In this way we try to capture in a realistic manner the way multiplicity-based (volume-based) discounts are applied in the real world. Therefore, given transformation t, we firstly assess the probability p_{offer} of the transformation to be traded with multiplicity m from a geometric distribution with success probability $t_{geometric}$ as follows: $p_{offer} = t_{geometric}{}^{multiplicity}$ (line 20). Secondly, we compute the discount to apply ($discount$) as follows: $discount = \alpha \frac{1-e^{1-p_{offer}}}{1-e}$. Indeed, in this way we manage to apply higher discounts to more unlikely offers within the range $[0, \alpha]$. Notice too that setting α to zero leads to no discounts, and thus to no superadditivity.

Which bids to combine in an XOR bid. Finally, after creating each bid, the algorithm adds it to the XOR bid under construction (line 37). The number of bids that compose an XOR bid is obtained from a normal distribution $\mathcal{N}(\mu_{add\_new\_XOR\_clause}, \sigma_{add\_new\_XOR\_clause})$ (line 9). We consider here the requirement 7 listed in section 3 and since different bids in XOR-relationships stand for different alternatives or options for the bidder we propose to generate similar bids for the same XORBid. First bid of each XORBid is generated randomly (lines 13-24) whereas the rest of bids are created applying some modifications over one existing bid in the bundle (lines 25-36). The number of modifications depends on the difference between the number of transformations assigned to the new bid and the existent one: while it is less we remove randomly one transformation; while it is greater we add randomly new transformations and if it is equal we apply once both operations. In all cases we finally recalculate the prices following the proposed price policy. Hence the requirement that *sets of XOR'ed bids are constructed on a per-bidder basis* is fulfilled.

5 Experimental results

In this section we shall illustrate the computational cost of solving the WDP for MMUCA. At this aim, we present our first experimental results for MMUCA by assessing the performance of an IP implementation on CPLEX.

As explained at the end of section 2, the number of decision variables of an IP to solve a MMUCA WDP depends on the overall number of transformations. Thus, the number of transformations must be considered as one dimension when measuring the time complexity of a WD algorithm for MMUCA. However, transformations are subsequently combined in several ways in order to finally compose bids in the XOR bidding language. Thus, transformations can be bundled into bids (adding AND-constraints), which in turn may be put together into XOR bids (adding XOR-constraints) giving as a result problems with different levels of constriction. Here we list the significant changes that involve the introduction of different constraints in a MMUCA problem : (1) the introduction of AND-relationships reduces the resulting number of bids ; (2) the introduction of XOR-relationships reduces the resulting number of XOR-bids ; (3) the introduction of XOR-relationship reduces the length of the solution sequence and, consequently, the number of decision variables of the IP since the maximum length of a solution sequence is the sum of the number of transformations contained in the largest bid of each XOR bid.

Hence, we believe that the size of bids (transformation bundles) as well as the size of XOR bids must be regarded as further dimensions when measuring the time complexity of a WD algorithm. Considering the dimensions mentioned so far, we propose a first experiment to evaluate an implementation of the IP formulated in [3] when solving artificial data sets in scenarios with: (1) XOR-bids composed of a single bid with a single transformation (neither AND nor XOR constraints); (2) XOR-bids composed of a single bid with two transformations (AND constraints) (3) XOR-bids composed of two bids with one single transformation (XOR constraints); and (4) XOR-bids composed of two bids with two transformations (both AND and XOR constraints).

Considering the above-described experimental scenarios, we have run our experiments as follows. Firstly, we have generated 50 WDP instances for each configuration using a MATLAB implementation of the artificial data set generator detailed in section 4 whose source code is publicly available at `http://www.iiia.csic.es/~meritxell/material/MMUCA_problem_generator.zip` . We have solved each WDP with an IP implementation of MMUCA on CPLEX 10.1 and recorded the resulting solving times. Notice though that we have set to 3600 seconds the time deadline to solve each WDP. We have only considered feasible WDP instances to calculate solving times since the time required by CPLEX to prove infeasibility is (usually) significantly lower than time required to find an optimal solution. Our tests have been run on a Dell Precision 490 with double processor Dual-Core Xeon 5060 running at 3.2 GHz with 2Gb RAM on Linux 2.6.

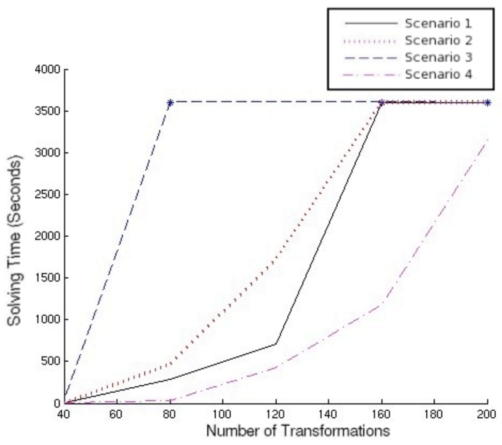


Fig.5 Solving time with respect to number of transformations.

Scen.	**XOR bid size**	**Bid size**	**Num. Transf**	**Sol. Length**	**Num. Bids**	**Num. XOR-bids**
1 **	1	1	40	40	40	40
			80	80	80	80
			120	120	120	120
			160	160	160	160
			200	200	200	200
2 **	1	2	40	40	20	20
			80	80	40	40
			120	120	60	60
			160	160	80	80
			200	200	100	100
3 **	2	1	40	20	40	20
			80	40	80	20
			120	60	120	60
			160	80	160	80
			200	100	200	100
4 **	2	2	40	20	20	10
			80	40	40	20
			120	60	60	30
			160	80	80	40
			200	100	100	50

Parameters characterising our experimental scenarios.

** Rest of parameters fixed as:$n_{goods} = 4$,$n_{IO\_market\_transformations} = 10$, $maxPrice = 100$,$\sigma_{prices} = 0.05$,$p_{good\_requested} = 0.3$, $p_{good\_in\_input} = 0.3$,$p_{good\_in\_output} = 0.1$

Fig. 5 depicts the median of the solving times resulting when varying the number of transformations for the above-described scenarios. The star(*) symbol stands for the median value exceeding the time limit (3600s). If that is the case, we cannot know the exact median value, but only that it exceeded the time limit. Observe that indeed the MMUCA computational cost increases as the number of transformations grows. Solving times exponentially increase for all scenarios. However, significant differences appear among different scenarios. Observing results of scenario 2 and 3 it stands out that CPLEX solving times increase with the introduction of restrictions with respect to a unconstrained problem (scenario 1). In these scenarios, the space search reduction provided by the addition of the constraints does not compensate CPLEX for considering these restrictions. We also observe that considering XOR-relationships significantly increases the complexity of the problem with respect to considering AND-relationships. However, as we observe in scenario 4, the introduction of constraints can involve important reductions in solving times.

We hypothesize that the introduction of a larger number of constraints (in this scenario we introduce AND-constraints and XOR-constraints) leads to a larger reduction of the search space with respect to scenario 2 and 3. In this case, although CPLEX has to operate with two types of constraints, the reduction of the search space compensates for the introduction of more complexity and produces better results. Nonetheless, it stands out the need for more research in order to accurately understand and explain the significant variations obtained in these experiments through the different scenarios above.

6 Conclusions and future work

In this work, we have attempted at making headway in the practical application of MMUCAs along two directions. Firstly, we have provided an algorithm to generate artificial data sets for MMUCA that are representative of the sort of scenarios a WD algorithm is likely to encounter. A distinguishing feature of the algorithm is that it pursues to capture in a realistic manner how bidders trade transformations. Our algorithm reformulates and extends in terms of transformations the requirements for an artificial data set generator for CAs. Secondly, we provide the first empirical tests for MMUCAs by assessing the performance of a CPLEX IP implementation. These tests assess the computational cost of solving the WDP as transformations grow for different bid expressions in the XOR bidding language. Our results indicate that the scalability of an IP implementation of MMUCA is seriously compromised by the exponential growth of computational cost as the number of transformations increases. Hence, we argue in favour of special-purpose optimal and local algorithms that improve the current performance of an IP implementation.

Acknowledgements

This work has been partially funded by the projects TIN2006-15662-C02-01 and TIN2006-5-0I-099. The work of Meritxell Vinyals is supported by the Ministry of Education of Spain (FPU grant AP2006-04636).

References

1. P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions.* MIT Press, 2006.
2. A. Giovannucci, J. A. Rodríguez-Aguilar, J. Cerquides, and U. Endriss. On the winner determination problem in mixed multi-unit combinatorial auctions. In *Proceedings of the Sixth International Conference on Autonomous Agents and Multiagent Systems*, Honolulu, Hawaii, USA, May 14-18 2007. In press.
3. J.Cerquides, U.Endriss, A.Giovannucci, and J.A Rodríguez-Aguilar. Bidding languages and winnder determination for mixed multi-unit combinatorial auctions. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, pages 1221–1226, India, January 2007.
4. K.Leyton-Brown and Y. Shoham. *A Test Suite for Combinatorial Auctions*, chapter 18. MIT Press, 2006.
5. N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton et al., editors, *Combinatorial Auctions.* MIT Press, 2006.
6. Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.
7. T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *2nd International Joint Conference on Autonomous Agents and Multiagent Systems*, Bologan, Italy, July 2002.
8. M. Vinyals, J. Cerquides, and J. A. Rodriguez-Aguilar. On the empirical evaluation of mixed multi-unit combinatorial auctions. Technical Report RR-IIIA-2007-01, IIIA-CSIC, February 2007.