# On the Empirical Evaluation of Mixed Multi-Unit Combinatorial Auctions

**Technical Report RR-IIIA-2007-01**

Meritxell Vinyals and Jesús Cerquides

IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
08193 Bellaterra, Spain
{meritxell,cerquide}@iiia.csic.es

**Abstract.** Mixed Multi-Unit Combinatorial Auctions (MMUCA) allow agents to bid for bundles of goods to buy, goods to sell, and transformations of goods. In particular, MMUCAs offer a high potential to be employed for the automated assembly of supply chains of agents offering goods and services, and in general the MMUCA extends and generalises all the preceding types of combinatorial auctions. However, little is known about their practical application since no empirical results have been reported on winner determination algorithms for MMUCAs. 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]). 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. In particular, supply chain formation (SCF) appears as a very promising application area where strong complementarities arise. Indeed, in [7] Walsh et al. observe

that production technologies often have to deal with strong complementarities: the inputs and outputs of a production process present a high-dependence relationship. For instance, in that context a producer may not be able to cover consumer's demands as a consequence of failing to obtain the necessary raw materials for the production process. At the same time the same provider may risk to produce goods which are not demanded by costumers in the market after investing money in its production process. Hence, a supply chain can be regarded as an intricate network of producers (entities transforming input goods into output goods at a certain cost), and consumers interacting in a complex way. Nevertheless, the complementarities arising in SCF are different from the ones we do find in CAs. The complementarities in SCF arise because of the preconditions and postconditions of production processes: precedences and dependences along the supply chain must be taken into account. Hence, whilst in CAs the complementarities can be simply represented as relationships among goods, in SCF the complementarities involve not only goods, but also interrelated *transformation* (production) relationships along several levels of the supply chain.

The work in [4] introduces a generalisation of the standard model of combinatorial auction and discusses the issues of bidding and winner determination. This new auction extends and generalises all the preceding types of combinatorial auctions: single-unit CAs, multi-unit CA, double CAs, and supply chain formation CAs. It provides a bidding language that can express several type of complex bids, and allows for bids on combinations of production processes, as well as a general winner determination problem (WDP) solver working on any network topology. This auction model is called mixed multi-unit combinatorial auction (MMUCA). Notice that this must not to be confused with a double auction [1] because the *order* in which agents consume and produce goods is of central importance. Consider as an example the assembly of a car's engine, whose structure is depicted in figure 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 a supply chain formation auction where bidders can bid over bundles of parts, bundles of transformations, or any combination of parts and transformations.

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 winner determination algorithms. These results are unlikely to come up unless, and along the lines of the research effort carried out in CAs [5], researchers are provided with algorithms or test suites to generate artificial data that is representative of the auction scenarios a winner determination algorithm is likely to encounter. Hence, winner determination 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 generate artificial data sets that is representative of the sort of scenarios a winner determination (WD) algorithm is likely to encounter. Secondly, we employ such algorithm to generate artificial data and subsequently assess the performance of an Integer Programming implementation of MMUCA on CPLEX. Based on our empirical results, we argue that there is a need for special-purpose WD algorithms for MMUCAs if these are intended to be employed in large scenarios.

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 so that they represent realistic bidding behaviour. Next, in section 4 we detail an algorithm for the generation of artificial data sets. In section 5, we analyse some empirical results for an IP formulation of the WDP for MMUCAs. Finally, we draw some conclusions and outline paths to future research in section 6.

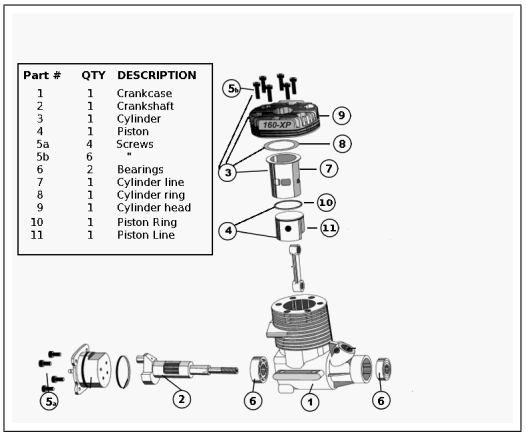


**Fig. 1.** Components of a car engine.

## 2 Background

Next, we introduce MMUCA by summarising the work in [4, 3]. Let $G$ be the finite set of all the 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 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.

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 $v(\{(\{\mathit{line}, \mathit{ring}, \mathit{head}, 6 \cdot \mathit{screws}, \mathit{screwdriver}\}, \{\mathit{cylinder}, \mathit{screwdriver}\})\}) = -10$ 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[1].

An *atomic bid* $\textsc{bid}(\{(\mathcal{I}^1, \mathcal{O}^1), \ldots, (\mathcal{I}^n, \mathcal{O}^n)\}, p)$ 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.In MMUCA we consider good free disposal *at the auctioneer's side* which means that the auctioneer can freely dispose of additional goods, *i.e.* accept more and give away fewer of them. These free disposals affect the definition of what constitutes a valid solution to the WDP. Refer to [4] for a further discussion about free disposal assumptions .

A suitable *bidding language* should allow a bidder to encode choices between alternative bids and the like[6]. Informally, an OR-combination of several 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

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

XOR-combination of bids expresses that the bidder is prepared to accept at most one of them.

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. In MMUCA transformations could form cycles and as a result given a set of transformations its order of execution is not implicit. 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. With no transformation free disposal if a bidder submits an offer over a bundle of *transformations*, all of them must be employed in the transformation sequence, whereas in the case of transformation free disposal any number of the transformations in the bundle can be included into the solution sequence, but the price to be paid is the total price of the bid.
(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 (the same set in the case of no good free disposal) of $\mathcal{U}_{out}$. This requirement is specific to MMUCAs. In case of no transformations free disposal check also that there are no transformations bought and left unused. An *optimal* solution is a valid solution that maximises the sum of prices associated with the atomic bids selected.

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. [4]). Therefore, solving the WDP for MMUCAs with XOR-bids amounts to maximise $\sum_{b \in B} x_b \cdot p_b$, while fulfilling the con-

straints informally defined above[2], where $B$ stands for the set of all atomic bids, $x_b$ is binary decision variable indicating whether bid $b$ is selected or not, and $p_b$ stands for the price of bid $b$. As noticed in [4], the number of decision variables of an integer program to solve a MMUCA WDP is of the order of $|T|^2$, where $|T|$ is the overall number of transformations mentioned anywhere in the bids. As the reader may notice, this represents a serious computational cost as the number of transformations grow. In the presence of XOR-relationships the number of variables of a problem can be reduced since in this case the maximum length of a solution sequence is the sum of the number of transformation contained in the larger bid for all XOR-bids.

## 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 [4, 3].

In this section we make explicit the requirements for a bid generation technique intended to produce artificial data sets for MMUCAs. Since bids in MMUCAs are composed of transformations, such requirements shall encompass —along the lines of artificial data set generators for CAs— the selection of which transformations and how many transformations to place in a bundle, what price to offer or request for the bundle, and which bids to combine in an XOR'ed set. Notice though that artificial data set generators for CAs consider that agents trade goods, whereas the requirements for a MMUCA generator consider that agents trade transformations.

### 3.1 A Taxonomy of Transformations

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 firslty 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:

[2] The interested reader should refer to [4] for a formal, precise definition of the WDP.

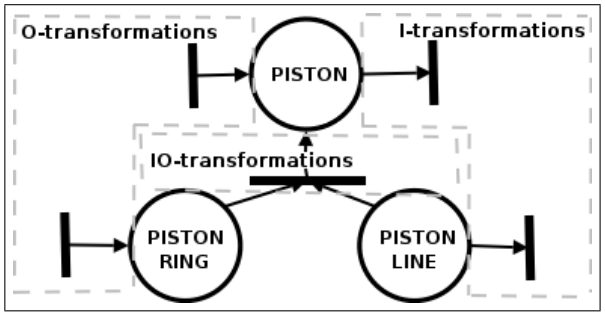


**Fig. 2.** Examples of tranformation types.

1. **Output transformations (O-transformations)** are those with no input good(s). Thus, an O-transformation represents a bidder's offer to sell some good(s). Notice that an O-transformation is equivalent to a bid in a reverse CA.
2. **Input transformations (I-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 (IO-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* $\mathcal{I}$ *after having received* $\mathcal{O}$. They can model a wide range of different processes in real-world situations (e.g. assembly, transformation, or exchange).

Fig.2 presents samples of each transformation type. In the figure, vertical, black bars stand for transformations, cercles stand for goods, and directed arrows from goods into or from transformations represent the goods input into or produced from a transformation. Thus, 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 the transformations of the types listed above. Therefore, a bid generator for MMUCA must support the generation of transformations of all these types.

### 3.2 Requirements

It is time to consider how to combine transformations of the types described in section 3.1 in order to construct bids. Since MMUCAs generalise CAs, as noticed in [4], our approach is to depart from artificial data sets generators for CAs, keeping the requirements summarised in [5], namely:

1. Certain goods are more likely to appear together than others.
2. The number of goods in a bundle is often related to which goods compose the bundle.
3. Valuations are related to which goods appear in the bundle. Where appropiate, valuations can be configured to be subadditive, additive or superadditive in the number of goods requested.
4. Sets of XOR'ed bids are constructed in a meaningful way, 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 (as defined in section 2), whereas a bidder in a CA bids over a bundle of goods. This difference leads to a fundamental issue: how should an artificial data set generator for MMUCA compose bids? Indeed, notice that a CA generator bundles goods from a given set of goods to construct bids. And hence, analogously, what is the set of transformations from which a MMUCA generator constructs bids? In order to provide a proper answer we must think of the kind of scenarios faced by buyers and providers in the real world. If so, within a given market we expect several companies to offer the very same or similar services (transformations) at different prices, as well as several companies 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 buying and providing companies request and offer. For instance, in the example in figure 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 offer to perform such transformation several times (e.g. as many times as cylinders are required), to bundle it with other transformations, or the two. Whatever the case, we can regard this sample transformation as an *atomic* transformation because it represents the minimum transformation required to perform a service. Hereafter, we shall consider the common goods and services in a given market to be represented as a collection of atomic

tranformations 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 are to be composed as combinations of market transformations. More formally, we firstly define atomic transformations as follows[3]:

**Definition 1 (Atomic transformation).** *Given a set of transformations* $T = \{t_1, \ldots, t_n\}$*, we say that transformation* $t_i = (\mathcal{I}_i, \mathcal{O}_i)$ *is minimum in* $\mathcal{T}$ *iff* $\forall t_j \in T$ *satisfying that* $\forall g_i \in \mathcal{I}_i, g_j \in \mathcal{I}_j \ g_i, g_j \in \mathcal{I} \cap \mathcal{I}'$ *and* $\forall g_i \in \mathcal{O}_i, g_j \in \mathcal{O}_j \ g_i, g_j \in \mathcal{O} \cap \mathcal{O}'$*, the following inequalities hold: (i)* $m_{\mathcal{I}_i}(g) \leq m_{\mathcal{I}_j}(g) \ \forall g \in \mathcal{I}_j$ *and; (ii)* $m_{\mathcal{O}_i}(g) \leq m_{\mathcal{O}_j}(g) \ \forall g \in \mathcal{O}_j$ .

From the definition above, we can readily provide a formal definiton of market transformations.

**Definition 2 (Market transformations).** *We say that* $T \subseteq \mathbb{N}^G \times \mathbb{N}^G$ *is a set of market transformations iff: (i) it is finite; (ii) every transformation* $t \in T$ *is minimum; and (iii)* $(\{g\}, \{\}), (\{\}, \{g\}) \in T \ \forall g \in G$.

Notice that the third condition ensures that there are at least two market transformations for every good in $G$[4], and thus ensuring that every good is individually available to buy and/or sell. Fig.3 depicts a sample of market transformations if intending to build the car engine in Fig. 1. From the discussion so far, we shall add a new requirement: *"there is a finite set of market transformations to bid for"*.

Notice too that we mentioned above that bids are composed as combinations of market transformations. If so, we must introduce the notion of *transformation multiplicity* as the counterpart of good multiplicity[5]. Indeed, 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 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 is both share the same input and output goods,

[3] Hereafter, we consider that the multiplicity (that is, the number of occurrences) of the elements in a multiset $\mathcal{A}$ is provided by a function $m_{\mathcal{A}}$ from $\mathcal{A}$ to $\mathbb{N}$.

[4] Therefore the number of market transformations is always greater or equal than $2 \cdot |G|$

[5] The number of units of a given good within an offr or a request.

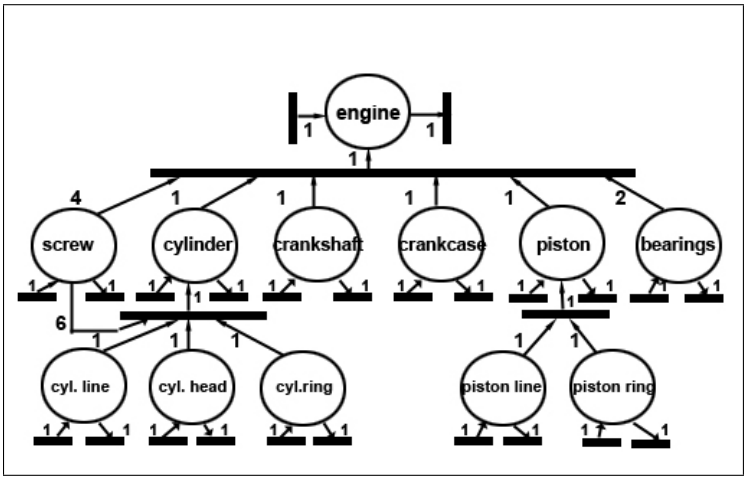


**Fig. 3.** Market transformations for a car's engine.

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 $t = (\{6 \cdot screws, 1 \cdot cylinder\_line, 1 \cdot cylinder\_rig, 1 \cdot cylinder\_head\}, \{cylinder\})$ and $t' = (\{12 \cdot screws, 2 \cdot cylinder\_line, 2 \cdot cylinder\_rig, 2 \cdot cylinder\_head\}, \{2 \cdot cylinder\})$ we way that $t'$ has multiplicity two with respect to t.Put formally, we define transformation multiplicity as follows:

**Definition 3 (Tranformation multiplicity).** *Let $t = (\mathcal{I}, \mathcal{O})$ and $t' = (\mathcal{I}', \mathcal{O}')$ be tranformations such that $\forall g \in \mathcal{I}, g' \in \mathcal{I}'$ $g, g' \in \mathcal{I} \cap \mathcal{I}'$ and $\forall g \in \mathcal{O}, g' \in \mathcal{O}'$ $g, g' \in \mathcal{O} \cap \mathcal{O}'$. We say that $t$ has multiplicity $\kappa$ with respect to $t'$ iff $m_{\mathcal{I}}(g) = \kappa \cdot m_{\mathcal{I}'}(g)$ $\forall g \in \mathcal{I}$ and $m_{\mathcal{O}}(g) = \kappa \cdot m_{\mathcal{O}'}(g)$ $\forall g \in \mathcal{O}$.*

A further issue has to do with the way bidders value transformations. Notice that performing a transformation to assemble the engine in figure 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 value already paid for the parts. Therefore, if the parts cost $850EUR$ and the expected market value of the engine is $1000EUR$, the car maker should be willing to offer to pay less than $150EUR$ for the transformation. On the other hand, any provider is expected to request less than $150EUR$ 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 expectec 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. This dicussion leads us to a further requirement: *"every transformation valuation is assessed in terms of the surplus resulting from the market price of its ouput goods with respect to the market price of its input goods"*.

Finally, a last requirement stems from the fact that, unlike auctioneer 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. 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. Hence, the new requirement goes as follows: *"Goods not requested by the auctioneer may be involved in the auction"*.

Following the analysis above, we can reformulate the requirements for an artificial data set generator for CAs and add the new requirements derived so far to finally obtain the requirements for an artificial data set generator for MMUCAs:

1. There is a finite set of market transformations to bid for.
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. Where appropiate, valuations can be configured to be subadditive, additive or superadditive in the number of transformations requested.
5. Every transformation valuation is assessed in terms of the difference between the valuation of its ouput goods with respect to the valuation of its input goods.
6. Sets of XOR'ed bids are constructed in a meaningful way, on a per-bidder basis.
7. Goods not requested by the auctioneer may be involved in the auction.

## 4 An Algorithm for Artificial Data Set Generation

In this section we describe an artificial data set generator algorithm for MMUCA that captures the requirements above. The algorithm's purpose is to generate MMUCA problems (each composed by 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 winner determination algorithm. The bid generation technique 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. Therefore, the bid generation technique described below automates the generation of artificial data sets as specified in section 3. 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 price policy. The resulting bids are further composed into XOR (mutually exclusive) bids because, as pointed out in [4], the XOR language is a fully expressive language allowing bidders to express all their preferences in a single XOR bid. Hence, our algorithm assumes that each bidder formulates a single XOR bid, being the number of bidders equal to the number of XOR bids.

### 4.1 Good Generation

The good generation algorithm (see Algorithm 1) receives 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, the algorithm returns for each good: (1) its **average market price** ($\mu[i]$) drawn from a uniform distribution $U[1, maxPrice]$ where $maxPrice$ (line 2); 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 is to be drawn later on (line 3).

**Algorithm 1** Good Generation($n_{goods}$,$maxPrice$)

1: **for** $i = 1$ to $n_{goods}$ **do**
2: $\quad \mu[i] \leftarrow U[1, maxPrice]$
3: $\quad g_{geometric}[i] \leftarrow U[0, 1)$
4: **end for**
5: return $\mu, g_{geometric}$

### 4.2 Requested Goods Generation

The requested goods generation algorithm (see Algorithm 2) assesses the number of units of each good the auctioneer requests, namely the multiset

$\mathcal{U}_{out}$. As pointed out in Section 3, not all goods involved in a MMUCA must be requested by the auctioneer. Hence, the algorithm selects a subset of goods to be part of $\mathcal{U}_{out}$. Firstly, it determines whether a good is requested by the auctioneer by comparing the value drawn from a uniform distribution with $p_{good\_requested}$, the probability of adding a new good to $\mathcal{U}_{out}$ (line 2). Once included a given good, the number of units requested for that good is drawn from a geometric distribution with the success probability $g_{geometric}$ returned by algorithm 1.

**Algorithm 2** Requested Good Generation($n_{goods}$, $p_{good\_requested}$, $g_{geometric}$)

```
1: for i = 1 to n_goods do
2:    if U[0,1) < p_good_requested then
3:       U_out[i] ← 1 + Geometric(g_geometric[i])
4:    else
5:       U_out[i] ← 0
6:    end if
7: end for
8: Return U_out
```

### 4.3 Market Transformations Generation

The market transformation generation algorithm (see Algorithm 3) generates a finite set of transformations to be employed as the building blocks to subsequently compose bids.

For each good, the algorithm constructs two market transformations (one I-transformation and one O-transformation). Each transformation, which according to definition 2 is atomic, involves a single good with multiplicity one (lines 2-11). 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 a market IO-transformation, the algorithm 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 $g_{geometric}$. (lines 16-21).

Finally, there is the matter of attaching to each market transformation a probability distribution to draw its multiplicity. Our algorithm assumes that the bid generation process, detailed by algorithm 4, is to employ

**Algorithm 3** Market Transformation Generation($n_{goods}$, $p_{good\_in\_input}$, $p_{good\_in\_output}$, $g_{geometric}$)

1: $MT \leftarrow \{\}$
2: **for** $i = 1$ to $n_{goods}$ **do**
3: $T \leftarrow EmptyTransition()$
4: $T.inputs[i] \leftarrow 1$
5: $T.t_{geometric} \leftarrow i_{multiplicity}[i]$
6: $MT \leftarrow MT \cup \{T\}$
7: $T \leftarrow EmptyTransition()$
8: $T.outputs[i] \leftarrow 1$
9: $T.t_{geometric} \leftarrow g_{geometric}[i]$
10: $MT \leftarrow MT \cup \{T\}$
11: **end for**
12: **for** $t = 1$ to $n_{market\_IO\_transformations}$ **do**
13: $T \leftarrow EmptyTransition()$
14: **for** $i = 1$ to $n_{goods}$ **do**
15: **if** $U[0,1] < p_{good\_in\_input}$ **then**
16: $T.inputs[i] \leftarrow 1 + Geometric(g_{geometric}[i])$
17: **end if**
18: **end for**
19: **for** $i = 1$ to $n_{goods}$ **do**
20: **if** $U[0,1] < p_{good\_in\_output}$ **then**
21: $T.outputs[i] \leftarrow 1 + Geometric(g_{geometric}[i])$
22: **end if**
23: **end for**
24: $T.t_{geometric} \leftarrow 1$
25: **for** $i = 1$ to $n_{goods}$ **do**
26: $a \leftarrow g_{geometric}[i]$
27: $T.t_{geometric} \leftarrow min(T.p_{multiplicity}, a^{|T.inputs[i] - T.outputs[i]|})$
28: **end for**
29: $MT \leftarrow MT \cup \{T\}$
30: **end for**
31: return MT

a geometric distribution for each market transformation to calculate its multiplicity. Hence, algorithm 3 solely assesses the success probability to be employed by such geometric distributions ($t_{geometric}$), namely the probability of adding an extra unit of a transformation already included in a bundle bid. Notice though that we cannot randomly generate $t_{geometric}$ 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 algorithm 1 (line 3). 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 $|m_{\mathcal{I}}(g) - m_{\mathcal{O}}(g)|$. We set the success probability of $t$ to the minimum of these probabilities. Formally, $t_{geometric} = min_{g \in \mathcal{G}} g_{geometric}^{|m_{\mathcal{I}}(g) - m_{\mathcal{O}}(g)|}$, where $t_{geometric}$ stands for the success probability of transformation $t$ and $g_{geometric}$ stands for the success prob-

ability of good $g$. Indeed, algorithm 1 sets the success probability for each transformation in this way (lines 25-28).

### 4.4 Bid Generation

The bid generation algorithm (algorithm 4) generates bids that are subsequently combined into XOR bids, each one encoding the offer or request of a bidder.

The bid generation algorithm (algorithm 4) generates bids that are subsequently combined into XOR bids, each one encoding the offer or request of a bidder. Firstly, for each XOR bid ($XORBid$) the algorithm composes each bid ($Bid$) by combining the market transformations ($MTS$) 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 randomly chosen from the set of market transformations ($MTS$) (line 14) and their multiplicity in the bundle bid is obtained from a geometric distribution with success probability $t_{geometric}$ (line 15). Next, the algorithm prices the transformation according to its multiplicity (lines 16-20), and bunldes it into the bid under construction (line 21). Finally, after creating the bid, the algorithm adds it to the XOR bid (line 24). At this point, notice that 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).

And yet there remains the matter of setting a valuation for each bid within an XOR bid via some pricing policy while fulfilling the requirements in section 3. At this aim, 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 $\sigma_{prices}$ 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) as stated by requirement 5 in section 3. It is time to address bid valuations while keeping in mind requirement 4 in section 3. At this aim, we propose to introduce superadditivity by applying multiplicity-based discounts to transformations. Going back to the example in Fig. (a) in table **??**, we observe that screws

**Algorithm 4** Bid Generation($MTS$, $n_{XOR\_bids}$, $\mu$, $\sigma_{prices}$, $\mu_{add\_new\_XOR\_clause}$, $\sigma_{add\_new\_transformation}$, $\mu_{add\_new\_transformation}$, $\sigma_{add\_new\_transformation}$, $\alpha$)

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:  **for** $t = 1$ to $nTransfBid$ **do**
14:  $MT \leftarrow$ Randomly select transformation from $MTS$
15:  $multiplicity \leftarrow Geometric(MT.t_{geometric})$
16:  $T.inputs \leftarrow MT.inputs \cdot multiplicity$
17:  $T.outputs \leftarrow MT.outputs \cdot multiplicity$
18:  $T.price \leftarrow \sum_{g \in T.outputs} p_{prices\_bid}[b, g] - \sum_{g \in T.inputs} p_{prices\_bid}[b, g]$
19:  $p_{offer} \leftarrow (T.t_{geometric})^{multiplicity}$
20:  $discount \leftarrow \alpha \frac{1 - e^{1 - p_{offer}}}{1 - e}$
21:  $Bid.t \leftarrow Bid.t \cup T$
22:  $Bid.price \leftarrow Bid.price + T.price \cdot (1 - discount)$
23:  **end for**
24:  $XORBid \leftarrow XORBid \cup \{Bid\}$
25:  **end for**
26:  $Bids \leftarrow Bids \cup \{XORBid\}$
27: **end for**
28: return Bids

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. 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 $\alpha$ to zero leads to no dis-

counts, and thus to no superadditvity. Finally, a bid valuation is obtained by adding the prices of its transformations (line 23).

| **Parameter** | **Description** |
|---|---|
| $n_{goods}$ | Number of goods |
| $n_{XORbids}$ | Number of XOR bids |
| $n_{IO\_market\_transformations}$ | Number of market IO-transformations |
| $maxPrice$ | Maximum average good price |
| $\sigma_{prices\_bid}$ | Variance of good prices among different bidders |
| $\mu_{add\_New\_XOR\_Clause}$ | Mean number of bids per XOR bid |
| $\sigma_{add\_New\_XOR\_Clause}$ | Variance of bids per XOR bid |
| $\mu_{add\_New\_trasnformation}$ | Mean transformations per bid |
| $\sigma_{add\_New\_transformation}$ | Variance of transformations per bid |
| $p_{good\_requested}$ | Probability for a good to be included in $\mathcal{U}_o ut$ |
| $p_{good\_in\_input}$ | Probability for a good to be included in the input set of a transformation |
| $p_{good\_in\_output}$ | Probability for a good to be included in the output set of a transformation |
| $\alpha$ | Maximum discount applied to a bid for the number of requested goods |

**Table 1.** Artificial data set generator parameters.

Table 1 summarises all parameters required by an artificial data set generator for MMUCAs.

## 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 composed bids in the XOR bidding language. Thus, transformations can be bundled into bids, which in turn may be put together into XOR bids. Hence, 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 to evaluate an IP implementation when solving artificial data sets modelling scenarios with: (1) XOR-bids composed of a single bid with a single transformation (neither AND nor XOR constraints); (2) XOR-bids composed

of two bids with one single transformation (XOR constraints); and (3) XOR-bids composed of two bids with two transformations (both AND and XOR constraints)[6]. Within each scenario, the number of transformations varies by taking on values within the set $\{40, 80, 120, 160, 200\}$. Table 2 summarises the three scenarios we have employed in our empirical evaluation. For each scenario, we set the values of the parameters required by the artificial data set generator described in section 4 (summarised in table 1) as follows: $n_{goods} = 4$, $n_{IO\_market\_transformations} = 10$, $maxPrice = 100$, $\sigma_{prices} = 0.05$, $p_{good\_requested} = 0.3$, $p_{good\_in\_input} = 0.3$, and $p_{good\_in\_output} = 0.1$.

| Scenario | Transformations | XOR-bid size | Bid size |
|---|---|---|---|
| 1 | {40,80,120,160,200} | 1 | 1 |
| 2 | {40,80,120,160,200} | 2 | 1 |
| 3 | {40,80,120,160,200} | 2 | 2 |

**Table 2.** Parameters characterising our experimental scenarios.

Considering the above-described experimental scenarios, we have run our experiments as follows. Firstly, we have generated 50 WDP instances for each configuration in table 2 (e.g. for 120 transformations in scenario 2) 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 7.0 and recorded the resulting solving times. Notice though that we have set to 3600 seconds the time deadline to solve each WDP. Furthermore, 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. All 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. 4 depicts the median of the solving times resulting when varying the number of transformations for the scenarios in table 2. The star (*) symbol stands for the median value exceeded 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 com-

[6] Notice that scenario 1 would be enough to model a CA scenario whenever no IO-transformations are generated.

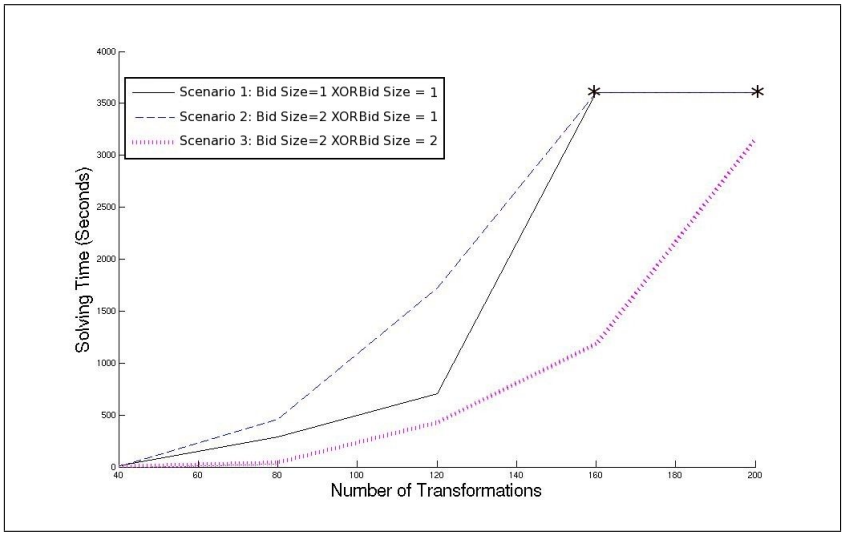


**Fig. 4.** Solving time with respect to number of transformations.

putational cost increases as the number of transformations grows. Solving times quickly, exponentially increase for all scenarios. We observe too that XOR relationships (among bids) have significantly more impact than AND relationships (among transformations bundled in bids) on computational costs. Notice that as a general rule, the length of a valid solution sequence is at most as large as the length of the sequence resulting from adding the largest (in numer of transformations) bid out of each XOR-bid. Hence, the introduction of XOR bids reduces the length of the solution sequence and, consequently, the number of decision variables of the IP.

## 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. Unlike CAs, for which special-purpose optimal algorithms cannot outperform an IP CPLEX implementation, we argue that for MMUCAs it is worth studying whether a special purpose algorithm can exploit the high redundancy among the sequences of transformations explored when calculating an optimal allocation. As to local (sub-optimal) approaches, they can be of value when anytime solutions are required. Indeed, in scenarios where supply chain formation must be agile in response to rapidly changing conditions, optimality is not as important as finding acceptable configurations within some specific time. On the theoretical side, we believe that a promising strand of research is represented by the work in [3] (or [2] in the context of multi-attribute double auctions), where the WDP is mapped into a search space where the structure of the problem is analysed in order to identify particular structures whose WDP complexity can be reduced.

## Acknowledgements

This work has been supported by projects IEA (TIN2006-15662-C02-01), OK (IST-4-027253-STP), eREP(EC-FP6-CIT5-28575) and 2006 5 OI 099.

## References

1. P. Cramton, Y. Shoham, and R. Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.
2. Y. Engel, M. Wellman, and K. Lochner. Bid expressiveness and clearing algorithms in multiattribute double auctions. In *Proceedings of the ACM Conference on Electronic Commerce*, pages 110–119, Ann Arbor, Michigan, USA, June 11-15 2006.
3. 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.
4. 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.
5. K.Leyton-Brown and Y. Shoham. *A Test Suite for Combinatorial Auctions*, chapter 18. MIT Press, 2006.

6. N. Nisan. Bidding languages for combinatorial auctions. In P. Cramton et al., editors, *Combinatorial Auctions*. MIT Press, 2006.
7. W. E. Walsh, M. P. Wellman, and F. Ygge. Combinatorial auctions for supply chain formation. In *Proc. of the 2nd ACM Conference on Electronic Commerce*, pages 260–269, Minneapolis, Minnesota, 2000.