Multi-unit Combinatorial Reverse Auctions with Transformability Relationships among Goods

Andrea Giovannucci[1], Juan A. Rodríguez-Aguilar[1], and Jesús Cerquides[2]

[1] Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain
{andrea,jar}@iiia.csic.es
[2] Dept. de Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
Gran Via, 585
08007, Barcelona, Spain
cerquide@maia.ub.es

Abstract. In this paper we extend the notion of multi-unit combinatorial reverse auction by adding a new dimension to the goods at auction. In such a new type of combinatorial auction a buyer can express transformability relationships among goods: some goods can be transformed into others at a transformation cost. Transformability relationships allow a buyer to introduce his information as to whether it is more convenient to buy some goods or others. We introduce such information in the winner determination problem (WDP) so that not only does the auction help allocate the optimal set of offers —taking into account transformability relationships—, but also assesses the transformability relationships that apply. In this way, the buyer finds out what goods to buy, to whom, and what transformations to apply to the acquired goods in order to obtain the required ones.

1 Introduction

Since many reverse (or direct) auctions involve the buying (or selling) of a variety of different assets, combinatorial auctions [3, 7] (CA) have recently deserved much attention in the literature. In particular, a significant amount of work has been devoted to the problem of selecting the winning set of bids [12, 2]. Nonetheless, to the best of our knowledge, while the literature has considered the possibility to express relationships among goods on the bidder side —such as complementarity and transformability (e.g. [4],[13])—, the impact of the eventual relationships among the different assets to sell/buy on the bid-taker side has not been conveniently addressed so far.

Consider that a company devoted to the assembly and repairing of personal computers (PCs) requires to assembly new PCs in order to fulfil his demand. Figure 1 graphically represents the way a PC is assembled. Our graphical description largely borrows from the representation of Place/Transition Nets (PTN) [6], a particular type of Petri Net. Each circle (corresponding to a PTN *place*) represents a good. Horizontal bars connecting goods represent assembly/disassembly operations, likewise *transitions* in a PTN. Assembly and disassembly operations are labelled with an indexed *t*, and shall be referred to as *transformability relationships*. In particular t_1 and t_2 represent disassembly operations whereas t_3 and t_4 stand for assembly operations. An arc connecting a good to a transformation indicates that the good is an *input* to the transformation, whereas an arc connecting a transformation to a good indicates that the good is an *output* from the transformation. In our example, a motherboard is an *input good* to t_2, whereas CPU, RAM, USB and empty motherboard are *output goods* of t_2. Thus, t_2 represents the way a motherboard is taken into pieces (disassembled). The labels on the arcs connecting *input goods* to transitions, and the labels on the arcs connecting *output goods* to transitions indicate the units required of each *input good* to perform a transformation and the units generated per *output good* respectively. In figure 1, the labels on the arcs connected to t_3 indicate that 1 motherboard is assembled from 1 CPU, 4 RAM units, 3 USBs and 1 empty motherboard at a cost of 8 EUR. Each transformation has an associated cost every time it is carried out. In our example, assembling a motherboard via t_3 costs 8 EUR, while taking a motherboard into pieces via t_2 costs 7 EUR.

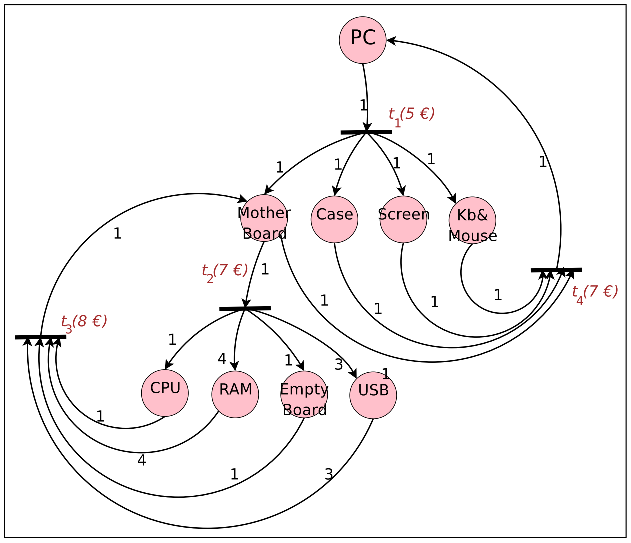

Fig. 1. Graphical representation of an RFQ with t-relationships.

Say that the company's warehouse contains most of the components composing each PC. However, there are no components to assemble motherboards. Therefore, the company would have to start a sourcing [5] process to acquire such components. For this purpose, it may opt for running a combinatorial reverse auction [13] with qualified providers. But before that, a professional buyer may realise that he faces a decision problem: shall he buy the required components to assemble them in house into motherboards, or buy already-assembled motherboards, or opt for a *mixed purchase* and buy some components to assemble them and some already-assembled motherboards? This concern is reasonable since the cost of components plus transformation (assembly) costs may eventually be higher than the cost of already-assembled motherboards. Hence, the buyer requires a combinatorial reverse auction mechanism that provides: (a) a language to express required goods along with the relationships that hold among them; and (b) a winner determination solver that not only assesses what goods to buy and to whom, but also the transformations to apply to such goods in order to obtain the initially required ones. In this paper we try to provide solutions to both issues.

Firstly, since commercial e-sourcing tools [11] only allow buyers to express fixed number of units per required good as part of the so-called *Request for Quotation* (RFQ), we have extended this notion to allow for the introduction of transformation relationships (*t-relationships* henceforth) among goods. Thus, we introduce a formal definition of a *Transformability Network Structure* (TNS) that largely borrows from Place/Transition Nets [6], where transitions stand for t-relationships and places stand for goods.

Secondly, we extend the formalisation of multi-unit combinatorial reverse auction (MUCRA), departing from the model in [12], to introduce transformability by applying the expressiveness power of multi-set theory. Additionally, we provide a mapping of our formal model to integer programming that takes into account t-relationships to assess the winning set of bids along with the transformations to apply in order to obtain the buyer's initial requirements.

Finally, we empirically analyse how the introduction of t-relationships affects scalability with respect to a classical multi-unit combinatorial reverse auction.

The paper is organised as follows. In section 2 we provide some background knowledge on place/transition nets and multi-sets. In section 3 we present a formal model of multi-unit combinatorial reverse auctions with t-relationships among goods, along with its winner determination problem and its mapping to integer programming. Section 4 is devoted to illustrate some preliminary, experimental results. Finally, section 5 draws some conclusions and outlines directions for future research.

2 Background

In this section we introduce some background knowledge on multi-sets and place/transition nets.

A *multi-set* is an extension to the notion of set, considering the possibility of *multiple appearances* of the same element. A *multi-set* $\mathcal{M}_X$ over a set X is a function $\mathcal{M}_X : X \rightarrow \mathbb{N}$ mapping X to the cardinal numbers. For any $x \in X$, $\mathcal{M}_X(x) \in \mathbb{N}$ is called the *multiplicity* of x. An element $x \in X$ *belongs* to the multi-set $\mathcal{M}_X$ if $\mathcal{M}_X(x) \neq 0$ and we write $x \in \mathcal{M}_X$. We denote the set of multi-sets over X by X_{MS}. Given the multi-sets $\mathcal{M}_S, \mathcal{M}'_S \in S_{MS}$, their union is defined as: $\mathcal{M}_S \cup \mathcal{M}'_S(x) = \mathcal{M}_S(x) + \mathcal{M}'_S(x)$.

Following [6], a *Place/Transition Net Structure* (PTNS) is a tuple $N = (G, T, A, E)$ such that: (1) G is a set of *places*; (2) T is a finite set of *transitions* such that $P \cap T = \emptyset$; (3) $A \subseteq (G \times T) \cup (T \times G)$ is a set of *arcs*; (4) $E : A \rightarrow \mathbb{N}^+$ is an *arc expression* function. A *marking* of a PTNS is a multi-set over G. A PTNS with a given initial marking $\mathcal{M}_0 \in G_{MS}$ is denoted by $PTN = (N, \mathcal{M}_0)$ and it is called a *Place/Transition Net* (PTN). The graphical representation of a PTNS is composed of the following graphical elements: places are represented as circles, transitions are represented as bars, arcs connect places to transitions or transitions to places, and E labels arcs with values (see figure 1).

A *step* is a non-empty and finite multi-set over T. A step $\mathcal{S} \in T_{MS}$ is *enabled* in a marking $\mathcal{M} \in G_{MS}$ if the following property is satisfied: $\forall g \in G \sum_{t \in \mathcal{S}} E(g,t)\mathcal{S}(t) \leq \mathcal{M}(g)$.

Let step $\mathcal{S}$ be enabled in a marking $\mathcal{M}_1$. Then, $\mathcal{S}$ may *occur*, changing the $\mathcal{M}_1$ marking to another $\mathcal{M}_2 \in G_{MS}$ marking. Setting $Z(g,t) = E(t,g) - E(g,t)$ $\mathcal{M}_2$ is expressed as: $\forall g \in G\ \mathcal{M}_2(g) = \mathcal{M}_1(g) + \sum_{t \in \mathcal{S}} Z(g,t)\mathcal{S}(t)$. Moreover, we say that marking $\mathcal{M}_2$ is *directly reachable* from marking $\mathcal{M}_1$ by the occurrence of step $\mathcal{S}$, and we denote it by $\mathcal{M}_1[\mathcal{S} > \mathcal{M}_2$.

A *finite occurrence sequence* is a finite sequence of steps and markings:$\mathcal{M}_1[\mathcal{S}_1 > \mathcal{M}_2 \ldots \mathcal{M}_n[\mathcal{S}_n > \mathcal{M}_{n+1}$ such that $n \in \mathbb{N}$ and $\mathcal{M}_i[\mathcal{S}_i > \mathcal{M}_{i+1}\ \forall i \in \{1, \ldots, n\}$. $\mathcal{M}_1$ is called the *start marking*, while $\mathcal{M}_{n+1}$ is called the *end marking*. The *firing count multi-set* $\mathcal{K} \in T_{MS}$ associated to a finite occurrence sequence is the union of all its steps: $\mathcal{K} = \bigcup_{i \in \{1,2,\ldots,n\}} \mathcal{S}_i$.

A marking $\mathcal{M}''$ is *reachable* from a marking $\mathcal{M}'$ iff there exists a finite occurrence sequence having $\mathcal{M}'$ as start marking and $\mathcal{M}''$ as end marking. We denote it as $\mathcal{M}'[\mathcal{S}_1\ \ldots \mathcal{S}_n > \mathcal{M}''$, where $n \in \mathbb{N}$. Furthermore the start and end markings are related by the following equation:

$$\forall g \in G \qquad \mathcal{M}''(g) = \mathcal{M}'(g) + \sum_{t \in \mathcal{K}} Z(g,t)\mathcal{K}(t). \tag{1}$$

The set of all possible markings reachable from a marking $\mathcal{M}'$ is called its *reachability set*, and is denoted as $R(N, \mathcal{M}')$.

In [10], Murata shows that in an *acyclic* Petri Net a marking $\mathcal{M}''$ is *reachable* from a marking $\mathcal{M}'$ iff there exists a multi-set $\mathcal{K} \in T_{MS}$ such that expression 1 holds (which is equivalent to say that the state equation associated to a PTN admits an integer solution). As a consequence, when a Petri Net is acyclic, the reachability set $R(N, \mathcal{M}')$ is represented by

$$R(N, \mathcal{M}') = \{\mathcal{M}'' \mid \exists \mathcal{K} \in T_{MS} : \forall g \in G\, \mathcal{M}''(g) = \mathcal{M}'(g) + \sum_{t \in \mathcal{K}} Z(g,t)\mathcal{K}(t)\}. \tag{2}$$

3 MUCRA with t-Relationships

3.1 Transformability Network Structures

A Transformability Network Structure describes the different ways in which our business is allowed to transform goods and at which cost. More formally, a *transformability network structure* (TNS) is a pair $S = (N, C_T)$, where $N = (G, T, A, E)$ is a Place-Transition Net Structure and $C_T : T \rightarrow \mathbb{R}^+$ is a cost function. The cost function associates a *transformation cost* to each *t-relationship*. In this context we associate: (1) the *places* in G to a set of goods to negotiate upon; (2) the *transitions* in T to a set of *t-relationships* among goods; (3) the *directed arcs* in A along with their weights E to the specification of the number of units of each good that are either consumed or produced by a transformation.

The values of C and the values of E label respectively transitions (between parenthesis) and arcs in figure 1.

In the following example, we formally specify the Transformability Network Structure $S = (N, C_T)$, graphically represented in figure 1: (1) $G = \{$PC, Motherboard, Case, Screen, Kb&Mouse, CPU, RAM, Empty Board, USB$\}$; (2) $T = \{t_1, t_2, t_3, t_4\}$; (3) $A = \{(PC, t_1), (t_1, motherboard), (t_1, case), (t_1, screen), (t_1, kb\&mouse), (motherboard, t_2), (t_2, CPU), (t_2, RAM), (t_2, EmptyBoard), (t_2, USB), \ldots\}$; (4) $E(PC, t_1) = 1, E(t_1, motherboard) = 1, E(t_1, case) = 1, E(t_1, screen) = 1, E(t_1, kb\&mouse) = 1, E(motherboard, t_2) = 1, E(t_2, CPU) = 1, E(t_2, RAM) = 4, E(t_2, EmptyBoard) = 1, E(t_2, USB) = 3, \ldots$; (5) $C_T(t_1) = 5$ EUR, $C_T(t_2) = 7$ EUR, $C_T(t_3) = 8$ EUR, $C_T(t_4) = 7$ EUR.

Given a Place/Transition net $PTN = (N, \mathcal{M}_0)$, if we consider $\mathcal{M}_0$ as a good configuration, PTN defines the space of good configurations *reachable* by applying tranformations to $\mathcal{M}_0$. The application of tranformations is obtained by firing transitions on PTN. Hereafter, we consider the concepts of *transformation step*, *enabling of a transformation step*, *occurrence of a transformation step* and *transformation sequence* as the counterparts to, respectively, *step*, *enabling of a step*, *occurrence of a step*, and *finite occurrence sequence* on a PTN.

We also need to define the concept of transformation cost, taking into account the cost of transforming good configuration $\mathcal{M}_0$ into another good configuration $\mathcal{M}_1 \in R(N, \mathcal{M}_0)$ by means of some transformation sequence $J = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$. The $\mathcal{K}$ firing count multi-set associated to J accounts for the number of times a transition in the sequence is fired. Thus, the cost of transforming good configuration $\mathcal{M}_0$ into good configuration $\mathcal{M}_1$ amounts to adding the transformation cost of each transition in the firing count multi-set $\mathcal{K}$ associated to J. We assess the transformation cost associated to J as $C_{TS}(J) = \sum_{\mathcal{S} \in J} \sum_{t \in \mathcal{S}} C_T(t)\mathcal{S}(t) = \sum_{t \in \mathcal{K}} C_T(t)\mathcal{K}(t)$. Notice that the transformation cost of a transformation sequence only depends on its firing count multi-set.

3.2 Winner Determination Problem (WDP) for MUCRA with t-relationships

In a classic MUCRA scenario, an RFQ can be expressed as a multi-set $\mathcal{U} \in G_{MS}$ whose multiplicity indicates the number of units required per good. In the example of figure 1, if $\mathcal{U}(motherboard) = 1, \mathcal{U}(CPU) = 1, \mathcal{U}(RAM) = 4, \mathcal{U}(EmptyBoard) = 1, \mathcal{U}(USB) = 3$, $\mathcal{U}$ would be representing a buyer's need for 1 motherboard (M), 1 CPU (C), 1 empty board (E), 4 RAM units (R), and 3 USB (U) connectors. Nonetheless, since t-relationships hold among goods, the buyer may have different alternatives depending on the bids he receives. If we represent each bid as a multi-set $\mathcal{B} \in G_{MS}$, whose multiplicity indicates the number of units offered per good, the buyer might, for example, have the following alternatives:

1. $\mathcal{M}_0 = \{M, C, R, R, R, R, E, U, U, U\}$. Buy all items as requested.
2. $\mathcal{M}_1 = \{M, M\}$. Buy 2 motherboards, and then disassemble 1 motherboard into 1 CPU, 4 RAM units, 1 Empty Board, and 3 USB connectors at cost $C_T(t_2) =$ 7EUR. The overall cost of the purchase results from the cost of the acquired units plus the additional transformation cost.

Notice that both alternatives allow the buyer to obtain his initial requirement, though each one at a different cost. The goal of the WDP is to assess what alternative to select.

We begin by defining the set of possible auction outcomes. Given a set of bids B, a possible auction outcome is a pair (W, J), where $W \subseteq B$, and $J = (\mathcal{S}_1, \ldots, \mathcal{S}_n)$ is a transformation sequence, such that the application of J to $PTN = (N, \cup_{\mathcal{B} \in W} \mathcal{B})$ allows a buyer to obtain a good configuration that fulfils his requirements in $\mathcal{U}$. More formally, the set of possible auction outcomes is defined as[3]:

$$\Omega = \{(W, J), W \subseteq B \mid \exists \mathcal{X} \in G_{MS} \ (\bigcup_{\mathcal{B} \in W} \mathcal{B})[J > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \tag{3}$$

To each auction outcome (W, J) we associate an *auction outcome cost* as follows:

$$C_O(W, J) = \sum_{\mathcal{B} \in W} C_B(\mathcal{B}) + C_{TS}(J) \tag{4}$$

where $C_B : B \rightarrow \mathbb{R}^+$ stands for the bid cost function.

Definition 1 (Winner Determination Problem). *Given a set of bids B, an RFQ $\mathcal{U} \in G_{MS}$, and a transformability network structure $S = (N, C_T)$, the winner determination problem for a MUCRA with t-relationships amounts to assessing the auction outcome $(W^{opt}, J^{opt}) \in \Omega$ that minimises the auction outcome cost function C_O. Formally,*

$$(W^{opt}, J^{opt}) = \arg \min_{(W,J) \in \Omega} C_O(W, J) \tag{5}$$

3.3 Mapping to Integer Programming

In section 2, we defined the reachibility set according to equation 2 for the case of acyclic Petri nets. Thus, if we restrict to the case of acyclic TNS, a finite occurrence sequence J is completely specified by its firing count vector $\mathcal{K}$. Then, we can rewrite expressions 3 and 4 respectively as follows:

$$\Omega = \{(W, \mathcal{K}), W \subseteq B, \mathcal{K} \in T_{MS} \mid \exists \mathcal{X} \in G_{MS} \ (\bigcup_{\mathcal{B} \in W} \mathcal{B})[\mathcal{K} > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \tag{6}$$

$$C_O(W, \mathcal{K}) = \sum_{\mathcal{B} \in W} C_B(\mathcal{B}) + C_{TS}(\mathcal{K}) \tag{7}$$

where $C_{TS}(\mathcal{K}) = \sum_{t \in \mathcal{K}} C_T(t)\mathcal{K}(t)$. Hence, the WDP when considering acyclic TNSs can be restated, from equation 5, to assess:

$$(W^{opt}, \mathcal{K}^{opt}) = \arg \min_{(W,\mathcal{K}) \in \Omega} C_O(W, \mathcal{K}) \tag{8}$$

We can model the problem of assessing $(W^{opt}, \mathcal{K}^{opt})$ as an Integer Programming problem. For this purpose, we need to associate integer variables to the elements in: (1) a generic subset of bids ($W \subseteq B$); and (2) a generic firing count multi-set ($\mathcal{K}$).

[3] Assuming free disposal.

In order to represent W we assign a binary decision variable $x_{\mathcal{B}}$ to each bid $\mathcal{B} \in B$, standing for whether $\mathcal{B}$ is selected ($x_{\mathcal{B}} = 1$) or not ($x_{\mathcal{B}} = 0$) in W. A multi-set is uniquely determined by its mapping function $\mathcal{K} : T \rightarrow \mathbb{N}$. Hence, we represent a multi-set $\mathcal{K} \in T_{MS}$ by considering an integer decision variable q_t for each $t \in T$. Each q_t represents the multiplicity of element t in the $\mathcal{K}$ multi-set. Thus, the translation into integer programming of expression (8) becomes:

$$min[\sum_{\mathcal{B} \in B} x_B p(\mathcal{B}) + \sum_{t \in T} q_t c(t)] \tag{9}$$

subject to $x_{\mathcal{B}} \in \{0, 1\}$. Notice that leaving the $q_t (t \in T)$ decision variables unbounded is utterly unrealistic because it is equivalent to say that the buyer has got the capability of applying as many transformations as required to fulfil $\mathcal{U}$. In practice, a buyer's production capacities are constrained, and therefore it is realistic to assume that the number of in-house transformations that he can apply are constrained. Hence, we add the following constraints to equation 9: $\forall t \in T \; q_t \in \{0, 1, \ldots, max_t\}$, where $max_t \in \mathbb{N}$.

Besides, we capture the side constraints enforcing that the selected bids, along with the transformations applied to them, fulfil $\mathcal{U}$ by translating expression 6 into linear programming. We consider a set of PTNs such that $PTN = (N, \mathcal{L})$, where $\mathcal{L} = \cup_{\mathcal{B} \in W} \mathcal{B}$. Moreover, we consider all the finite occurrence sequences of $PTN = (N, \mathcal{L})$ that transform $\mathcal{L}$ into a configuration that at least fulfils $\mathcal{U}$. Under the hypothesis of N being acyclic we can express the reachability set of $\mathcal{L}$ as follows:

$$\forall g \in G \;\; \mathcal{M}(g) = \mathcal{L}(g) + \sum_{t \in \mathcal{K}} Z(g, t)\mathcal{K}(t). \tag{10}$$

Next, we select the elements in the reachability set $[\mathcal{L} >$ that at least fulfil $\mathcal{U}$:

$$\forall g \in G \;\; \mathcal{L}(g) + \sum_{t \in \mathcal{K}} Z(g, t)\mathcal{K}(t) \geq \mathcal{U}(g) \tag{11}$$

Hence, substituting marking $\mathcal{L}$ by $\sum_{\mathcal{B} \in B} x_{\mathcal{B}} \mathcal{B}(g)$ we finally obtain the following side constraints:

$$\forall g \in G \;\; \sum_{\mathcal{B} \in B} x_{\mathcal{B}} \mathcal{B}(g) + \sum_{t \in T} Z(g, t) q_t \geq \mathcal{U}(g).$$

4 Experiments

The main purpose of our preliminary experiments is to empirically evaluate the benefits and drawbacks of introducing transformability relationships. With this aim we compared the scalability of the MUCRATR solver with respect to a traditional MUCRA solver on large instances.

The solvers for the MUCRATR WDP and MUCRA WDP have been developed with the aid of ILOG's[1] CPLEX 9.0. The benchmark has been generated with the aid of MATLAB 7.0 [9]. The solver for MUCRA's WDP uses a state-of-the-art Integer Programming formulation, that exploits the analogy of a multi-unit combinatorial auction WDP with a well known optimisation problem: the Multi Dimensional Knapsack Problem (MDKP). For a complete explanation refer to [3].

A problem instance for a MUCRA is composed of a a multi-unit RFQ, a set of multi-unit multi-item bids, whereas a MUCRATR additionally needs a TNS. Thus, firstly we built some problem instances for the MUCRATR and we solved them with the MUCRATR solver. Next, we solved the very same instances with the MUCRA solver considering only bids and RFQ.

In [8], Leyton-Brown specifies an algorithm to create MUCA instances whose purpose is to test WDP algorithms. We have adapted his algorithm to generate MUCRA instances. It is well known from [13] that a MUCRA is the dual case to a MUCA.

The existence of a TNS has led us to change some aspects of Leyton-Brown's algorithm. Firstly, instead of assigning an independent average price to each good, we have to take into account the t-relationships connecting goods. We assign goods' prices so that the sum of input good costs plus the transformation cost equals the sum of the output good costs (*adapted price distribution*). Consider, for instance, the example depicted in figure 1: the default price distribution can generate problem instances in which a PC price is lower than its USB's prices, whereas our pricing policy creates a sort of equilibrium among prices. Next, we consider more realistic to weight the average price of each bid via a normal probability distribution instead of a uniform one (concretely we used a normal distribution with mean 1 and variance 0.1).

In the following we describe the parameter settings of our experiments. We performed a single experiment in which the only parameter varying was the number of bids generated, ranging from 1000 to 270000.

The number of negotiated items was set to 20, the maximum number of units of a single item that a buyer can ask for was set to 15. The maximum number of units a bidder can offer for a single item was set to 20. The decaying probabilities employed to generate the number of goods per bid and the number of units offered per bit per item were both set to 0.8. The number of t-relationships imposed among the goods was set to 8.

Figure 2 depicts the results of this preliminary scalability test. Notice that we obtained very similar results to a state-of-the-art solver that does not take into account t-relationships. Thus, we can conclude that the introduction of t-relationships does not suppose a significant time overload with respect to a traditional combinatorial auction.

5 Conclusions and Future Work

In this paper we have presented a formalisation and an integer programming solution to the winner determination problem of a new type of multi-unit combinatorial reverse auction that allows for expressing t-relationships on the buyer side. Several advantages derive from such a new type of auction. On the one hand, it allows a buyer to incorporate his uncertainty as to whether it is better to buy a required bundle of goods, or alternatively buy some goods to transform them into the former ones, or even opt for a mixed purchase and buy some goods as required and some others to be transformed. This is achieved by introducing t-relationships among goods into the winner determination problem. Therefore, not only does the winner determination solver assess what goods to buy and to whom, but also the transformations to apply to such goods in order to obtain the initially required ones. To the best of our knowledge, this is the first type of

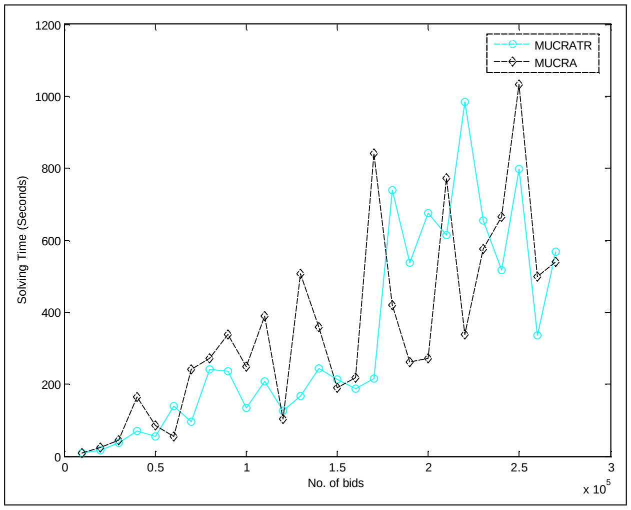

Fig. 2. Comparation of MUCRATR and MUCRA solvers for a normal distribution.

auction aimed at also handling buyers' uncertainty. As a side effect, the introduction of t-relationships is expected to increase competitiveness among bidders, and thus obtain better deals since bidders that otherwise would not be competing are put together to compete. Finally, our integer programming solution can be readily implemented with the aid of linear programming libraries.

We also performed some preliminary experiments comparing our solver for the WDP for MUCRATR with a state-of-the-art MUCRA solver. We compared the differences in terms of solving time and auction outcome cost. The results showed two main issues: (1) there is no significant, computational overload when solving a MUCRATR WDP with respect to solving a MUCRA WDP; and (2) there are always savings in terms of costs when running a MUCRATR, being outstanding for small-medium auction scenarios (less than 100 bids). Nonetheless, notice that the preliminary experiments we have run deserve further elaboration in order to thoroughly validate our early hypothesis.

As future work, it is our aim to further elaborate along several directions. Firstly, we aim at theoretically analysing the benefits in terms of savings that our mechanism provides with respect to multi-unit combinatorial reverse auctions. Secondly, we believe that it is important to research on the theoretical properties of our mechanism from a mechanism design point of view. And finally, the complexity of bidding in MUCRATRs along with decision support mechanisms for bidders shall be studied.

Acknowledgements

This work has been supported by project Web-i(2) (TIC-2003-08763-C02-01). Andrea Giovannucci enjoys the BEC.09.01.04/05-164 CSIC scholarship.

References

1. Ilog. http://www.ilog.com.
2. A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, pages 39–46, Boston, MA, 2000.
3. Sven de Vries and Rakesh Vohra. Combinatorial auctions: A survey. *INFORMS Journal of Computing*, 15(3):284–309, 2003.
4. Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI'99)*, pages 548–553, August.
5. Aberdeen Group. Making e-sourcing strategic: from tactical technology to core business strategy. Technical report, Aberdeen Group, 2002.
6. Kurt Jensen. *Coloured Petri Nets, Basic Concepts, Analysis Methods and Practical Use*, volume 1, chapter 2, pages 78–80. Springer, 1997.
7. Jayant Kalagnanam and David C. Parkes. *Supply Chain Analysis in the eBusiness Era*, chapter Auctions, Bidding and Exchange Design. 2003.
8. K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *American Association for Artificial Intelligence (AAAI)*, 2000.
9. MATLAB. http://www.mathworks.com.
10. T. Murata. Petri nets: Properties, analysis and applications. In *IEEE*, volume 77, pages 541–580, 1989.
11. Antonio Reyes-Moro, Juan Antonio Rodríguez-Aguilar, Maite López-Sánchez, Jesús Cerquides, and David Gutiérrez-Magallanes. Embedding decision support in e-sourcing tools: Quotes, a case study. *Group Decision and Negotiation*, 12:347–355, 2003.
12. Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.
13. Tuomas Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.