Multi-unit Combinatorial Reverse Auctions with Substitutability Relationships among Goods

Andrea Giovannucci
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain
andrea@iiia.csic.es

Juan A. Rodríguez-Aguilar
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain
jar@iiia.csic.es

Jesús Cerquides
Dept. de Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
Gran Via, 585
08007, Barcelona, Spain
cerquide@maia.ub.es

Giuseppe Carilli
Artificial Intelligence Research Institute, IIIA
Spanish Council for Scientific Research, CSIC
08193 Bellaterra, Barcelona, Spain
giuseppe@iiia.csic.es

July 11, 2005

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 substitutability relationships among goods: some goods can be substituted with others at a substitution cost. Substitutability 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 substitutability relationships—, but also assess the substitutability relationships that apply. In this way, the buyer finds out what goods to buy, to whom, and what *operations* (substitutions) to apply to the acquired goods in order to obtain the initially required ones.

1 Introduction

Since many auctions involve the selling or buying[1] 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, 13, 4]. 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 substitutability (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. Say that its warehouse contains most of the components composing each PC.

[1] Depending on whether the auction is direct or reverse respectively.

However, there are no components to assemble motherboards[2]. 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. To tackle this issue, the buyer could think of running separate auctions for motherboards and their components, and after that decide whether to buy the whole or the parts. Notice though that besides impractical and costly (in general, the more transformation relationships among goods we consider, the larger number of auctions would be required) this method would be missing the opportunity represented by mixed purchases. 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, notice that we can resort to a more general semantics when referring to relationships among goods: the semantics of *substitutability*. In the example above, if a buyer requires a motherboard, we can say that it can be *substituted* with 1 CPU, 4 RAM units, and 3 USB connectors at a certain *substitution* (transformation in our example) cost. Notice though that this notion of substitutability among goods is different from the classic notion of substitutability on the bidder side that we find in the CA literature [13]. 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 definition of substitutability relationships among goods. Thus, we introduce a formal definition of a *Substitutability Network Structure* (SNS) that largely borrows from Place/Transition Nets [6], where transitions stand for substitution relationships and places stand for required goods.

Secondly, we extend the formalisation of multi-unit combinatorial reverse auction (MUCRA), departing from the model in [12], to introduce substitutability by applying the expressiveness power of multi-set theory. Complementarity, we provide a mapping of our formal model to integer programming that takes into account substitutability relationships to asses the winning set of bids along with the substitutions to apply in order to obtain the buyer's initial requirements. Notice that although our example above depicts a very simple scenario where only a substitution applies (from components to motherboard), much more complex scenarios where a larger number of substitutability relationships are defined (see for instance the example in section 2) do require that the winner determination solver does find the substitutions to apply as well as the winning bids. Finally, we empirically analyse how the introduction of substitutability relationships helps increase competitiveness among bidders, and thus obtain better deals, with respect to a classical multi-unit combinatorial reverse auction. We believe that this is an important issue because the introduction of relationships among goods has the effect of putting together to compete bidders that otherwise would not be competing (e.g. CPU, memory, and USB manufactures compete with motherboard manufacturers).

The paper is organised as follows. In section 2 we introduce an extended version of the above-described example that founds our definition of RFQ with substitutability relationships. In section 3 we provide some background knowledge on multi-sets and Place/Transition Nets. In section 4 we present a formal model of multi-unit combinatorial reverse auctions with substitutability relationships among goods, along with its winner determination problem and its mapping to integer programming. Section 5 is devoted to illustrate some experimental results. Finally, section 6 draws some conclusions and outlines directions for future research.

2 Example

In this section we provide an extended version of the example introduced in section 1 to illustrate the type of substitutability relationships that we are interested in representing. 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 to negotiate upon. Assem-

[2] In this particular case, we consider that a motherboard is composed of 1 CPU, 4 RAM units, and 3 USB connectors.

bly/splitting operations are represented as horizontal bars connecting goods, likewise *transitions* in a PTN. The assembling and splitting operations are labelled with an indexed capital T, and shall be referred to as *substitutability relationships*. In particular $T1$ and $T2$ represent the effects of splitting operations whereas $T3$ and $T4$ stand for assembling operations. The values in parentheses, labelling good transformations, stand for the cost of each transformation every time it is *fired* (carried out). The arcs connecting a set of goods $G1$ to a transformation $T1$ indicates that the goods in $G1$ are an *input* to transformation $T1$. The arcs connecting a transformation $T1$ to a set of goods $G2$ indicates that goods in $G2$ are an *output* from transformation $T1$. In the example in figure 1, the $T2$ transformation, representing the way a motherboard is taken into pieces, has a motherboard as *input good* and CPUs, RAM memories, USBs and empty motherboards as *output goods*. We call *input weights* the labels on the arcs connecting *input goods* to transitions, and *output weights* the labels on the arcs connecting *output goods* to transitions. They indicate the units required of each *input good* to perform a transformation and the units generated per *output good* respectively. For instance, the labels on the arcs connected to $T3$ in figure 1 indicate that 1 motherboard is composed of 1 CPU, 4 RAM units, 3 USBs and 1 empty motherboard at a cost of 8 EUR.

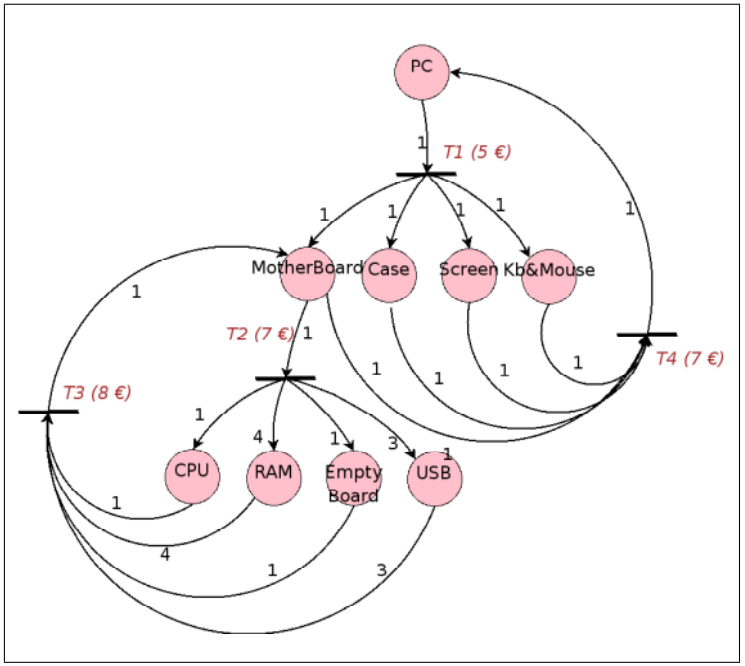

Figure 1: Graphical representation of an RFQ with substitutability relationships.

3 Background

3.1 Multi-sets

Multi-sets are a very powerful mathematical modelling tool, since they allow to represent, at the same time, the elements of a set and their multiplicity. We introduce multi-sets in order to formally represent both *multi-unit* offers and requirements, as well as substitutability relationships. Multiplicities of elements shall help us model the multi-unit nature of our problem.

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. We formally represent a multi-set $\mathcal{M}_X$ by a sum as follows:

$$\sum_{x \in X} \mathcal{M}_X(x) \grave{} x$$

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:

$$\mathcal{M}_S \cup \mathcal{M'}_S = \sum_{s \in S} (\mathcal{M}_S(s) + \mathcal{M'}_S(s)) \grave{} s$$

Operations over multi-sets (addition, multiplication, subtraction,...etc.) amount to the standard operations on their mapping functions.

3.2 Place/Transition Nets

In what follows we recall some definitions for *Place/Transition Nets* (PTN) [6]:

Definition 3.1 (Place/Transition Net). A *Place/Transition Net* (PTN) is a tuple $PTN = (G, T, A, E, \mathcal{M}_0)$ satisfying the requirements below:

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. $E : A \to \mathbb{N}^+$ is an *arc expression* function.
5. $\mathcal{M}_0 \in G_{MS}$ is the *initial marking*.

Definition 3.2 (Place/Transition Net Structure). A *Place/Transition Net Structure* $N = (G, T, A, E)$ does not specify any initial marking. A Place/Transition Net with a given initial marking $\mathcal{M}_0$ is denoted by $PTN = (N, \mathcal{M}_0)$.

The graphical representation of a PTN structure 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.

Definition 3.3 (Marking). A *marking* is a multi-set over G. The *initial marking* $M_0 \in G_{MS}$ denotes the initial distribution of tokens.

Definition 3.4 (Step, Enabling of a step). 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: $\forall g \in G \ \sum_{t \in \mathcal{S}} E(g,t)\mathcal{S}(t) \leq \mathcal{M}(g)$.

Definition 3.5 (Occurrence, Direct Reachability). 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 marking $\mathcal{M}_2 \in G_{MS}$, defined as follows:

$$\forall g \in G \, \mathcal{M}_2(g) = \mathcal{M}_1(g) + \textstyle\sum_{t \in \mathcal{S}} Z(g,t)\mathcal{S}(t)$$

where $Z(g,t) = E(g,t) - E(t,g)$. Moreover, we say that the $\mathcal{M}_2$ marking is *directly reachable* from the $\mathcal{M}_1$ marking by the occurrence of step $\mathcal{S}$, and we denote it by $\mathcal{M}_1[\mathcal{S} > \mathcal{M}_2$.

Definition 3.6 (Finite Occurrence Sequence). A *finite occurrence sequence* is a finite sequence of steps and markings:

$$\mathcal{M}_1[\mathcal{S}_1 > \mathcal{M}_2 ... \mathcal{M}_n[\mathcal{S}_n > \mathcal{M}_{n+1} \tag{1}$$

such that $n \in \mathbb{N}$ and $\mathcal{M}_i[\mathcal{S}_i > \mathcal{M}_{i+1} \ \forall i \in \{1, .., n\}$. $\mathcal{M}_1$ is called the *start marking*, while $\mathcal{M}_{n+1}$ is called the *end marking*.

We also define the *firing count multi-set* $\mathcal{K} \in T_{MS}$, associated to the finite occurrence sequence, as the union of all its steps: $\mathcal{K} = \bigcup_{i \in \{1,2,..,n\}} \mathcal{S}_i$.

Definition 3.7 (Reachability). 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. In this case we say that $\mathcal{M}''$ is *reachable* from $\mathcal{M}'$ in n steps and we denote it as $\mathcal{M}'[\mathcal{S}_1\mathcal{S}_2...\mathcal{S}_n > \mathcal{M}''$, where $\bigcup_{i=1..n} \mathcal{S}_i = \mathcal{K}$. Furthermore the start and end markings are related by the following equation:

$$\forall g \in G \, \mathcal{M}''(g) = \mathcal{M}'(g) + \sum_{t \in \mathcal{K}} Z(g,t)\mathcal{K}(t) \tag{2}$$

The set of all possible markings reachable from a marking $\mathcal{M}$ is called its *reachability set*, and is denoted as $[\mathcal{M} >$.

Murata in [10] shows that a necessary an sufficient condition for reachability in an acyclic Petri net (with no directed circuits) is that the state equation admits an integer solution. In fact there are various classes of Petri nets for which such condition holds. Anyway in this paper we limit our scope to the most restrictive of such classes: the *acyclic* or *circuit-free* Petri nets. This condition maps to the following proposition:

Proposition 3.8. *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 2 holds.*

As a consequence, when a petri net is acyclic, the reachability set $[M_0 >$ is represented by:

$$[M_0 >= \{\mathcal{M} \mid \exists \mathcal{K} \in T_{MS} : \forall g \in G \ \ \mathcal{M}(g) = \mathcal{M}_0(g) + \sum_{t \in \mathcal{K}} Z(g,t)\mathcal{K}(t)\} \tag{3}$$

4 Multi-Unit Combinatorial Reverse Auctions with Substitutability Relationships

4.1 Substitutability network structures

A Substitutability Network Structure describes the different ways in which our business is allowed to transform goods and at which cost. More formally, we define it as follows:

Definition 4.1 (Substitutability network structure). A *Substitutability network structure* (SNS) is a pair $S = (N, C)$, where:

- N is a Place-Transition Net Structure $N = (G, T, A, E)$ such that:
  1. The *places* in G represent a set of goods to negotiate upon.
  2. The *transitions* in T represent a set of possible *substitutability relationships* among goods.
  3. The *directed arcs* in A connect goods to substitutability relationships.
  4. The weights assigned by the *arc expression* function E indicate the number of units of each good that are either consumed or produced by a substitution. The values of E are the arc labels in figures 1 and 2.
- $C : T \rightarrow \mathbb{R}^+ \cup \{0\}$ is a cost function that associates a *substitution cost* to each *substitutability relationship*. The values of C are enclosed in parenthesis next to each transition in figures 1 and 2.

Notice that T represents the set of possible substitutions among subsets of G. The arcs in A relate either goods to substitutions or substitutions to goods. The goods connected to a substitutability relationship by incoming arcs (*input goods*) can substitute the goods connected to the very same substitutability relationship by outgoing arcs (*output goods*). The weights on the arcs connected to a substitutability relationship indicate the number of units of input and output goods consumed and produced respectively by the substitution.

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 means of applying substitutions to $\mathcal{M}_0$. The application of substitutions is obtained by firing transitions on the PTN. Henceforth, we define the concepts of *substitution step*, *enabling of a substitution step*, *occurrence of a substitution step* and *substitution 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 substitution cost, taking into account the cost of transforming good configuration $\mathcal{M}_0$ into another good configuration $\mathcal{M}_1 \in [M_0 >$ by means of some substitution sequence. The $\mathcal{K} = \bigcup_{i=1,..,n} \mathcal{S}_i$ multi-set associated to substitution sequence $J = (\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_n)$ accounts for the number of times a transition is fired. Thus, the cost of transforming the $\mathcal{M}_0$ good configuration into the $\mathcal{M}_1$ good configuration amounts to adding the substitution cost of each transition in the $\mathcal{K}$ substitution sequence, namely:

Definition 4.2 (Substitution Cost). Given a substitution sequence $J = (\mathcal{S}_1, \mathcal{S}_2, .., \mathcal{S}_n)$ and its associated firing count multi-set $\mathcal{K} = \bigcup_{i=1,..,n} \mathcal{S}_i$, we define the substitution cost associated to it as:

$$C_{sub}(J) = \sum_{\mathcal{S} \in J} \sum_{t \in \mathcal{S}} c(t)\mathcal{S}(t) = \sum_{t \in \mathcal{K}} c(t)\mathcal{K}(t) = C_{sub}(\mathcal{K}) \qquad (4)$$

Thus, the substitution cost only depends on the firing count multi-set.

In the following example we formally specify a Substitutability Network Structure $S = (N, C)$, graphically represented in figure 2:

- $G = \{g_1, g_2, g_3, g_4\}$;
- $T = \{T_1\}$
- $A = \{(g_1, T_1), (g_2, T_1), (T_1, g_3), (T_1, g_4)\}$
- $E(g_1, t_1)$=3, $E(g_2, t_1)$=4, $E(t_1, g_3)$=2, $E(t_1, g_4)$=1
- $C(T_1) = 200$ EUR.

It describes a buyer's capacity of transforming a pair of goods $(g1, g2)$ into a pair $(g3, g4)$ by means of substitution $t1$. The arc labels indicate that 3 units of good g_1 and 4 units of item g_2 can be transformed into (substituted with) 2 units of good g_3 and one unit of good g_4. C sets the substitution cost of T_1 to 200 EUR.

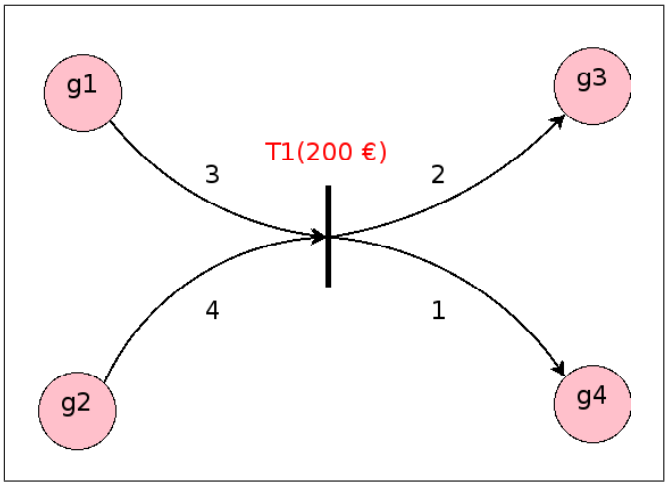

Figure 2: Graphical representation of a substitutability relationship

Say that we assign an initial marking $\mathcal{M}_0$ to S: $\mathcal{M}_0(g_1) = 6$, $\mathcal{M}_0(g_2) = 8$, $\mathcal{M}_0(g_3) = 0$, $\mathcal{M}_0(g_4) = 0$. The underlying PTN allows to transform it via substitutability relationship t_1 into a new good configuration $\mathcal{M}_1$: $\mathcal{M}_1(g_1) = 0$, $\mathcal{M}_1(g_2) = 0$, $\mathcal{M}_1(g_3) = 4$, $\mathcal{M}_1(g_4) = 2$. In such a case, the firing count multi-set would be $\mathcal{K} = \{t_1, t_1\}$ (t_1 is fired twice), and the substitution cost would amount to $C(\mathcal{K}) = 2 * 200$ EUR$= 400$ EUR.

4.2 Winner Determination Problem for MUCRASGs

In a classic multi-unit combinatorial reverse auction scenario, a Request For Quotation (RFQ), a buyer's requirement, 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 2, if $\mathcal{U}(g_1) = 2$,$\mathcal{U}(g_2) = 2$,$\mathcal{U}(g_3) = 2$,$\mathcal{U}(g_4) = 1$, $\mathcal{U}$ would be representing a buyer's need for 2 units of g_1, g_2, and g_3, and 1 unit of g_4. Nonetheless, since substitutability relationships hold among goods, the buyer may have different alternatives depending on the bids he receives:

1. $\mathcal{M}_0 = \{g_1, g_1, g_2, g_2, g_3, g_3, g_4\}$. Buy all items as requested.
2. $\mathcal{M}_1 = \{g_1, g_1, g_1, g_1, g_1, g_2, g_2, g_2, g_2, g_2, g_2\}$. Buy 5 units of item g_1 and 6 units of item g_2 and then to transform 2 units of g_1 and 4 units of g_2 into 2 units of g_3 and 1 unit of g_4 at cost $c = 200$ EUR. The overall cost results from the cost of the acquired units plus an additional transformation cost c.

Notice that both possibilities allow the buyer to obtain his initial requirement, namely 2 units of g_1, 2 units of g_2, 2 units of g_3, and 1 unit of g_4, each one at a different cost. Notice also that a bid can be represented as a multi-set $\mathcal{B} \in G_{MS}$, whose multiplicity indicates the number of units offered per good.

Definition 4.3 (Winner Determination Problem). Given a set of bids B, their costs $p : B \rightarrow \mathbb{R}^+ \cup \{0\}$, an RFQ $\mathcal{U} \in G_{MS}$, and a substitutability network structure $S = (N, C)$, the winner determination problem amounts to selecting a subset of bids ($W \subseteq B$) and to assessing a substitution sequence to apply to them in order to fulfil the requirements in $\mathcal{U}$ while minimising the total cost of the substitution sequence plus W.

We begin by defining the set of possible auction outcomes. A possible auction outcome is a pair (W, J), where $W \subseteq B$ contains a set of bids, and $J = (\mathcal{S}_1, \mathcal{S}_\in, ..., \mathcal{S}_n)$ is an ordered set of substitution steps composing a substitution sequence. The application of J to the $PTN = (N, \cup_{\mathcal{B}\in B}\mathcal{B})$ allows a buyer to obtain a good configuration that fulfils its requirement $\mathcal{U}$. More formally, the set of possible auction outcomes is defined as[3]:

$$\Omega = \{(W, J), W \subseteq B, J = (\mathcal{S}_1, \mathcal{S}_2, ..., \mathcal{S}_n), n \in \mathbb{N} \mid \exists \mathcal{X} \in G_{MS} \ (\bigcup_{\mathcal{B}\in W} \mathcal{B})[\mathcal{S}_1\mathcal{S}_2...\mathcal{S}_n > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \quad (5)$$

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

$$c(W, J) = \sum_{\mathcal{B}\in W} p(\mathcal{B}) + \sum_{\mathcal{S}\in J} \sum_{t\in S} c(t)\mathcal{S}(t) \quad (6)$$

Given a set of auction outcomes, the aim of the WDP for a MUCRASG is to find the optimal outcome $(W^{opt}, J^{opt}) \in \Omega$ that minimises the outcome cost $c(W, J)$. Formally,

$$(W^{opt}, J^{opt}) = \arg \min_{(W,J)\in\Omega} c(W, J) \quad (7)$$

[3] Assuming free disposal.

We mentioned above that proposition 3.8 holds in the case of acyclic Petri nets. Thus, if we restrict to the case of acyclic SNS, the firing count vector $\mathcal{K}$ completely specifies the finite occurrence sequence J. We can then rewrite expressions 5 and 6 as follows:

$$\Omega = \{(W, \mathcal{K}), W \subseteq B, \mathcal{K} \in T_{MS} \mid \exists \mathcal{X} \in G_{MS} \ (\bigcup_{\mathcal{B} \in W} \mathcal{B})[\mathcal{K} > \mathcal{X}, \mathcal{X} \supseteq \mathcal{U}\}. \tag{8}$$

and

$$c(W, \mathcal{K}) = \sum_{\mathcal{B} \in W} p(\mathcal{B}) + C(\mathcal{K}) \tag{9}$$

4.3 Mapping to Integer Programming

We model the problem of assessing (W^{opt}, J^{opt}) as an Integer Programming problem. Our problem, in the case of acyclic SNSs, amounts to finding:

$$(W^{opt}, \mathcal{K}^{opt}) = \arg \min_{(W,\mathcal{K}) \in \Omega} c(W, \mathcal{K}) \tag{10}$$

For this purpose, we need to express as integer variables:

- a generic subset of bids ($W \subseteq B$).
- a generic firing vector multi-set ($\mathcal{K}$) associated to a substitution sequence.

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 being included ($x_{\mathcal{B}} = 1$) 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 bounded 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 (10) becomes:

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

subject to $x_{\mathcal{B}} \in \{0, 1\}, q_t \in \{0, 1, ..., max_t\}$

Now we have to capture the side constraints enforcing that the selected bids, along with the transformations applied to them, fulfil $\mathcal{U}$, the initial buyer's requirement. For this purpose we translate expression 8 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 must 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 explicit the reachability set of $\mathcal{L}$ as follows:

$$\forall g \in G \ \mathcal{M}(g) = \mathcal{L}(g) + \sum_{t \in \mathcal{K}} Z(g, t)\mathcal{K}(t). \tag{11}$$

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

$$\forall g \in G \ \mathcal{L}(g) + \sum_{t \in \mathcal{K}} Z(g, t)\mathcal{K}(t) \geq \mathcal{U}(g) \tag{12}$$

Hence, expressing $\mathcal{L}$ as $\sum_{\mathcal{B} \in B} x_{\mathcal{B}} \mathcal{B}(g)$ we finally obtain the side constraints:

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

5 Experiments

The main purpose of our experiments is to empirically compare MUCRASG with respect to MUCRA. In other words, our aim is to compare the benefits and drawbacks of introducing substitutability relationships. Our analysis runs along two lines. On the one hand, we compare the differences in cost of the optimal solutions assessed by the two mechanisms when solving the very same problems. On the other hand, we compare the differences in computational time required by the winner determination solvers employed by both mechanisms. In this way, we can empirically quantify the savings that we may achieve by introducing substitutability relationships and whether such savings come at an extra computational cost.

5.1 Winner Determination Solver

The solvers for the MUCRASG WDP and MUCRA WDP have been developed with the aid of ILOG's[1] OPL Studio and CPLEX 9.0. The test

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].

5.2 Experimental Settings

Our experiments were performed over four types of artificially-generated problem instances. Each problem instance is composed of an RFQ and a SNS along with a set of multi-unit bids. In what follows we detail how to generate different types of problem instances.

5.2.1 Uniform Price Distributions for MUCRAs

A problem instance for a MUCRA is composed of a a multi-unit RFQ and a set of multi-unit bids. 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. Thus, the problem instances generated by Leyton-Brown's algorithm can be also used for MUCRAs. In the following we recall the way this algorithm work. Next, we recall the way this algorithm works.

On the one hand, an RFQ is artificially generated from a number of goods (n) and the maximum number of units that can be request per good (RFQ_{max_units}). Then, the algorithm composes an RFQ containing n goods such that each good $i \in \{1, \ldots, n\}$ is assigned a multiplicity $m_i \in \{1, 2, .., RFQ_{max\_units}\}$ via a uniform discrete distribution.

On the other hand, the algorithm generates a set of bids from the following input parameters:

1. num_{bids}. Number of bids.
2. $units_{max}$. Maximum number of units that a bidder can offer for a single item.
3. μ_{price} and σ_{price}. Each good i is assigned a price p_i uniformly drawn from the following interval:

$$p_i \in [\mu^{upd}_{price} - \sigma^{upd}_{price}, \mu^{upd}_{price} + \sigma price^{upd}] \quad (13)$$

4. $prob_1$ and $prob_2$ parametrise two decay probabilities, which are employed to generate the number of goods per bid and the number of units offered per bid per item respectively.
5. $price_{var}$. The cost per bid is assessed according to the following formula[4]:

$$price = \sum_{i \in Bids} (\mathcal{N}(1, price_{var}) * p_i * units_i)$$

where $\mathcal{N}(1, price_{var})$ draws random values following a normal distribution with mean 1 and variance $price_{var}$.

We refer to a problem instance for MUCRA generated as explained above as a *uniform price distribution* since we employ a uniform distribution to calculate the price of each good (see expression 13 above).

5.2.2 Uniform Price Distributions for MUCRASG

A problem instance for a MUCRASG is composed of a multi-unit RFQ, a substitutability network structure, and a set of multi-unit bids. Whereas the RFQ and the set of bids are inherited from section 5.2.1, the substitutability network structure is added as a new element to problem instances.

In section 4.3 we showed that it is possible to use a linear programming approach to solve the WPD for MUCRASG when some conditions hold. The most restrictive condition is that the PTN to which a substitutability network structure belongs is *acyclic* (and thus the substitutability network structure itself). Thus, in our experiments we have only considered acyclic substitutability network structures. The easiest way to construct such type of structures is to generate tree-like structures based on a special type of substitutability relationship: a single parent good is connected to several children goods. Figure 3 depicts the type of substitutability relationship used for constructing tree-like substitutability network structures. Notice that all transitions in these structures represent either *decomposition* (the parent good is substitutable with its children goods) or *composition* (children goods can be substituted with their

[4]The way we calculate prices is slightly different from [8]. In fact, in this work prices are obtained as follows:

$$price = U(1 - price_{var}, 1 + price_{var}) * \sum_{i \in Bids} (p_i * units_i)$$

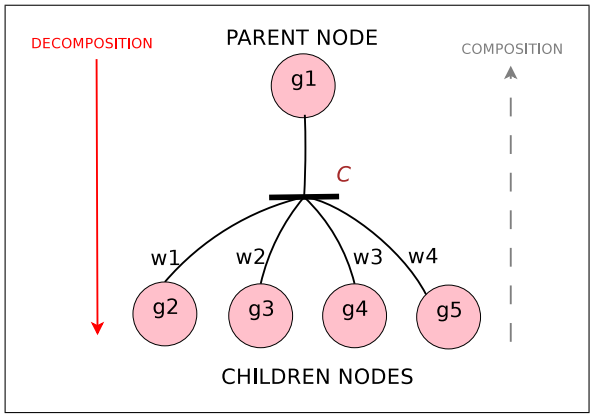

Figure 3: Graphical representation of a decomposition/composition transition

parent good). Otherwise, mixing decomposing transitions with composing transitions may easily lead to cycles.

Figure 4 shows an example of the tree-like substitutability network structures employed in our experiments. Notice that no weight is assigned to the arc connecting parent nodes to transitions because we set them all to 1. The following input parameters are employed to generate a tree-like substitutability network structure like the one in 4:

1. n_T: number of transitions.
2. c_T: maximum number of children per transition (the minimum is 1).
3. w_T: The maximum child weight (the minimum is 1). The weight of the arc connecting a parent node to a transition is always set to 1.
4. d_T: depth factor controlling the a structure's shape: lower values make the structure wider whereas higher values make it deeper.
5. sc_T: maximum *substitution cost*. It represents the maximum cost we can associate to each transition (c_1, c_2, c_3 in figure 4). Each c is set according to a uniform distribution as follows:

$$c = U(1, sc_T) \tag{14}$$

5.2.3 Adapted Price Distributions for MUCRASG

A problem instance for a MUCRASG is composed of a multi-unit RFQ, a SNS, and a set of multi-unit bids. Whereas the RFQ is inherited from section 5.2.2, the SNS is added as a new element to problem instances. As to bids, they result from adapting the bids in section 5.2.2. Recall that we employed expression 13 in order to calculate a good's average price in section 5.2.2 . We realised though that defining some substitutability relationships among goods without also introducing relationships among their prices generated paradoxical results. Consider, for instance, the example depicted in figure 2: a uniform price distribution can generate problem instances in which a PC price is lower than its USB's prices. It is highly unlikely (though not impossible) to acquire a PC at a lower price than a USB. To overcome this issue we employ an *adapted price distribution* to generate goods' average prices. Relationships among goods' prices are built based on substitutability relationships and their associated transformation costs. For this purpose, we employ the following recursive algorithm to compute each good average price[5]:

Algorithm 1 *function* compute_price ($(\boldsymbol{good})$)

1: **if** $\boldsymbol{g} = top\_good$ **then**
2: $price(\boldsymbol{g}) = rand(min\_top\_good\_price, max\_top\_good\_price)$:
3: **for** $\boldsymbol{i} \in \boldsymbol{g}$.children **do**
4: compute_price($\boldsymbol{i}$);
5: **end for**
6: **else**
7: $price(\boldsymbol{g}) = \frac{\boldsymbol{g}.parent.price \pm \boldsymbol{g}.parent.transition\_cost}{\boldsymbol{g}.parent.children\_number \cdot \boldsymbol{g}.weight}$
8: **for** $\boldsymbol{i} \in \boldsymbol{g}$.children **do**
9: compute_price($\boldsymbol{i}$);
10: **end for**
11: **end if**

The way prices are calculated in a SNS like the one in figure 8 is straightforward. We choose the price for the top good (the good at the top of the structure) from a uniform distribution over $[min\_top\_good\_price, max\_top\_good\_price]$. Given a parent good, we set

[5]The $\pm$ sign at step 7 means that the operation to apply depends on the type of transitions in the SNS: $+$ when transitions stand for decompositions and $-$ when they stand for compositions.

the price of each one of its children taking into account the number of units the parent good can generate (from the weights of its children), their parent's price, and the cost of the transition relating parent to children. This operation is recursively repeated along the SNS.

To summarise, we shall generate the bid prices to compose a problem instance for a MUCRASG likewise we explained in section 5.2.2, the only difference being that the average, unitary price per good is assessed using algorithm 1.

5.2.4 Adapted Price Distribution for MUCRAs

A problem instance for a MUCRA using an adapted price distribution results from merging problem instances produced in sections 5.2.2 and 5.2.3. Thus, a problem instance contains an RFQ along with a set of bids generated as described in section 5.2.3. In this way, we can compare more realistically the benefits of MUCRASG (using substitutability relationships) with respect to MUCRA (not using substitutability relationships).

5.3 Experiment Results

We have run some early tests to compare MUCRA and MUCRASG in terms of solving times and auction outcome costs. More precisely, we have compared MUCRA and MUCRASG along two scenarios: (1) over uniform price distributions; and (2) over adapted price distributions.

In order to generate problem instances for both MUCRA and MUCRASG considering the uniform and adapted price distributions, we have followed the directions established in section 5.2 using the following parameters:

[RFQ]
- n=10.
- $RFQ_{max\_units} = 15$

[Bids]
- $units_{max} = 20$
- $\mu_{price} = 25$ and $\sigma_{price} = 10$
- $prob_1 = 0.90$ and $prob_2 = 0.85$
- $price_{var} = 0.3$

[SNS]
- $n_T = 3$
- $c_T = 4$
- $w_T = 4$
- $d_T = 70$
- $sc_T = 15$

Notice that we did not run tests for different values of the parameters above. However, we did compare MUCRA and MUCRASG when varying the number of bids in their problem instances. Notice too that given a number of bids, we constructed 20 problem instances for both MUCRA and MUCRASG and for each type of price distribution. Hence, the result in 5.3.1 and 5.3.2 come from averaging the solving time and auction cost for both MUCRA and MUCRASG over the 20 problem instances each one's solver had to handle.

Finally, figure 4 illustrates the SNS along with the RFQ that we have used for our experiments.

5.3.1 Solving time

Figures 5 and 6 graphically compare the solving times for MUCRA and MUCRASG over uniform and adapted price distributions respectively. Observe that using different price distributions affects the relationship between the two curves.

Some interesting results arise from the analysis of these results. Although we expected the solving of the WDP for MUCRASG to be higher than the one for MUCRA, we observe that MUCRASG outperforms MUCRA for problem instances created with uniform price distributions (see figure 5). We cannot state the same as to problem instances created with adapted price distributions, since MUCRA slightly outperforms MUCRASG. We explain these results considering that a MUCRASG WDP integer programming formulation creates some more decision variables (the number of substitutability relationships: 3 in the example of figure 4), but the same number of constraints with respect to the MUCRA case. Thus, we conjecture that the branch and cut algorithm used by the optimiser can prune more efficiently unsuitable configurations. Consider, for instance, the problem presented in section 5.2.3. An unbalanced price distribution may generate an easily prunable search space. This explains also why the solving time difference is reduced for the APD case (see figure 6). In this case the possibility of finding very convenient transformations that prune effectively the search space is much more difficult because the prices are always balanced.

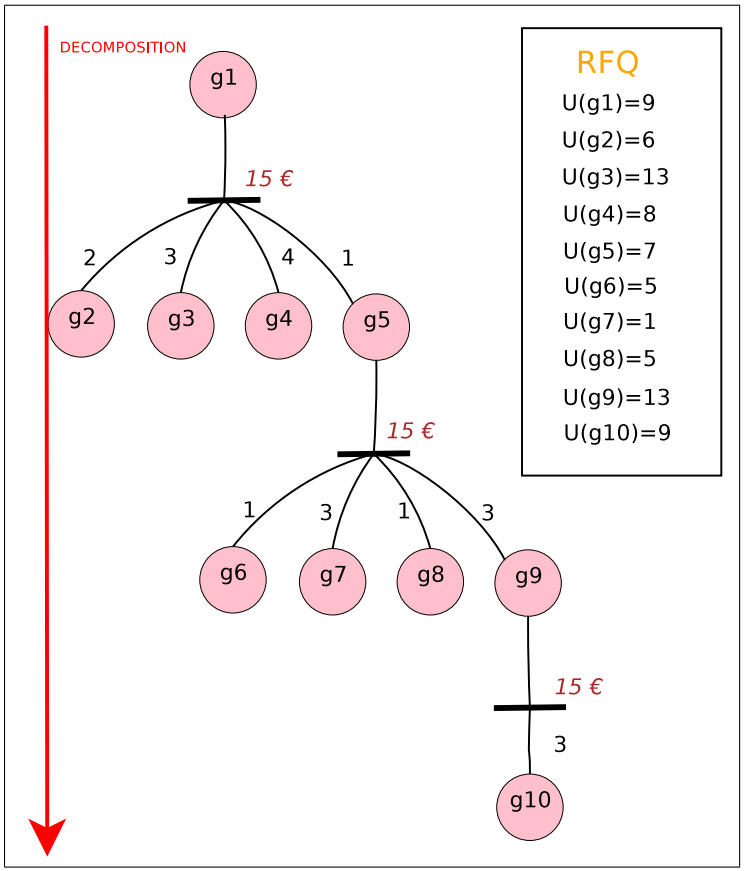

Figure 4: Representation of the Substitutability Network Structure employed during the experiments along with the RFQ

Notice though that we leave a thorough analysis of these hypothesis for future research.

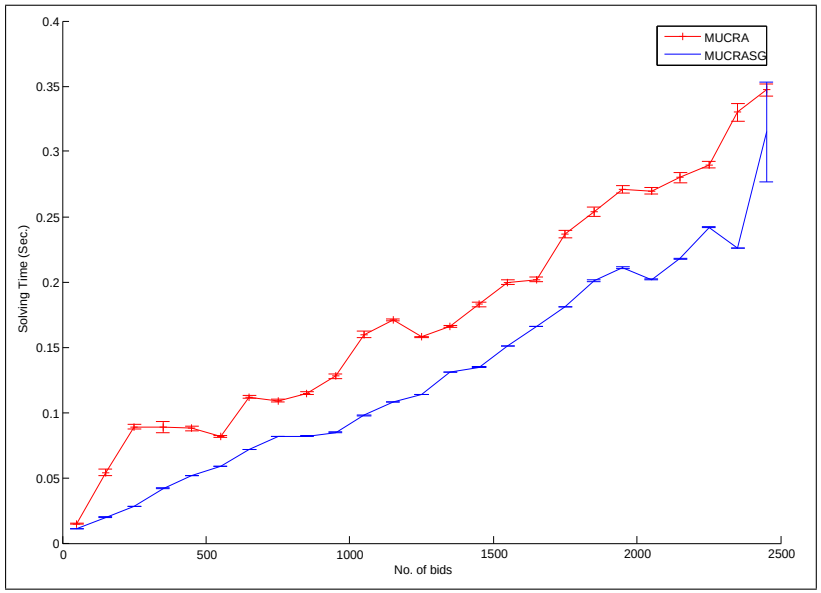

Figure 5: Comparison of solving times over uniform price distributions.

5.3.2 Optimal Outcome cost

Figures 7 and 8 compare the optimal outcome costs for problem instances created with adapted price distributions. The former figure compares the optimal outcome costs, whereas the latter depicts the relative margin differences obtained using an MUCRASG instead of an MUCRA. Such margins are computed according to the following expression:

$$\%_{margin} = \frac{\text{optimal\_cost(MUCRA) - optimal\_cost(MUCRASG)}}{\text{optimal\_cost(MUCRA)}}$$

Notice that the differences in the figures are variable, although always present. We measured differences ranging from around 2 to 32 %. Nonetheless, differences are larger for small-medium negotiation scenarios. Notice that for scenarios with less than 100 bids we obtain differences ranging from 15% to 30%.

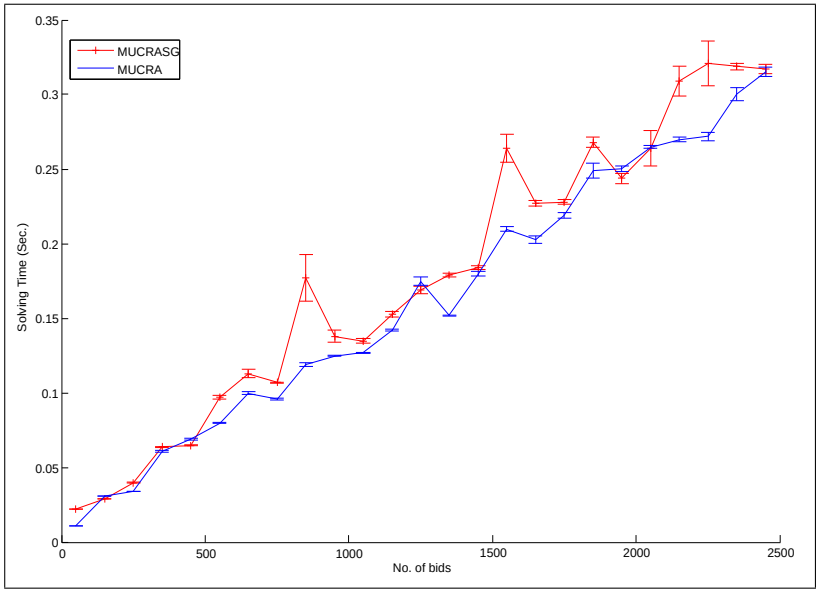

Figure 6: Comparison of solving times over adapted price distributions.

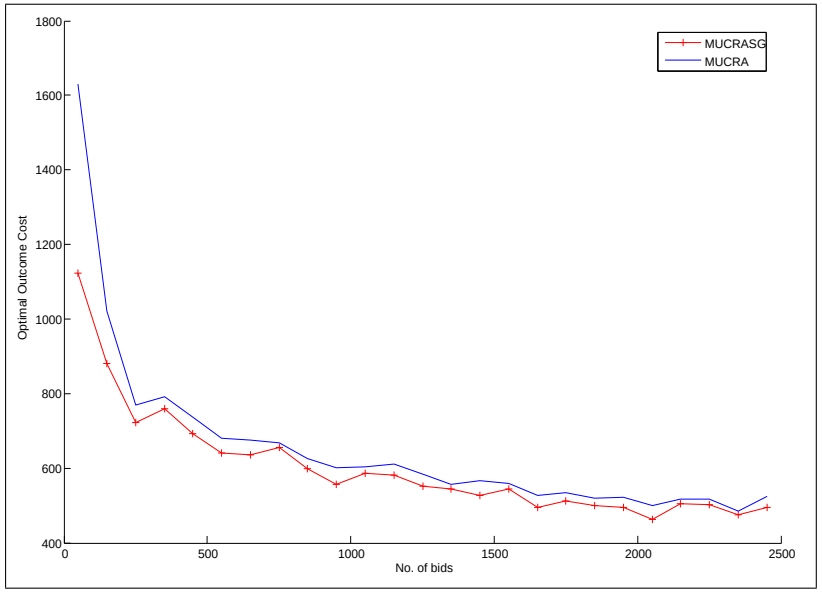

Figure 7: Comparing optimal outcome costs for adapted price distributions.

Notice also that the distribution that we are considering (APD) corresponds to quite pessimistic a case. In fact, the way we built APD implicitly implies that bidders have in average the same transformation costs than the bid-takers. This is indeed a strong assumption.

6 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 substitutability 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 substitutability 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 auction aimed at also handling buyers' uncertainty. As a side effect, the introduction of substitutability 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 MUCRASG with a state-of-the-art MUCRA solver. We compared the differences in terms of solving time and auction outcome cost. The

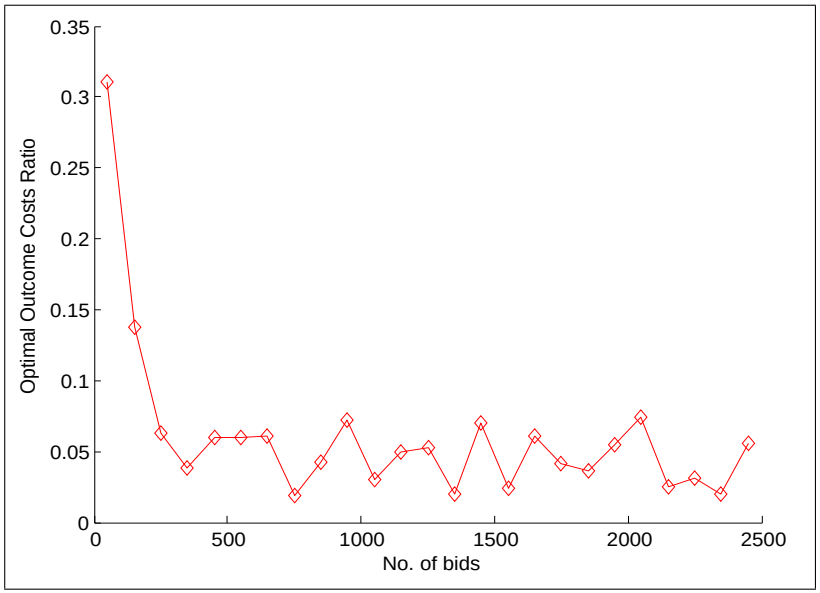

Figure 8: Relative difference in optimal outcome costs for adapted price distributions.

results showed two main issues: (1) there is no significant, computational overload when solving a MUCRASG WDP with respect to solving a MUCRA WDP; and (2) there are always savings in terms of costs when running a MUCRASG, 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 MUCRASGs 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.