Auctioning Transformable Goods

Andrea Giovannucci
Juan A. Rodríguez-Aguilar
IIIA-CSIC, Campus UAB, 08193 Bellaterra, Spain
{andrea, 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

ABSTRACT

In this paper we explore whether an auctioneer/buyer may benefit from introducing his transformability relationships (some goods can be transformed into others at a transformation cost) into multi-unit combinatorial reverse auctions. Thus, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider tranformability relationships.

1. INTRODUCTION

Consider a company devoted to sell manufactured goods. It can either buy raw goods from providers, transform them into some other goods via some manufacturing process, and sell them to customers; or it can buy already-transformed products and resell them to customers. Thus, either the company buys raw goods to transform via an in-house process at a certain cost, or it buys already-transformed goods. Figure 1 graphically represents an example of a company's inner manufacturing process, more formally *Transformability Network Structure* (TNS), fully described in [1]. This graphical description largely borrows from the representation of Place/Transition Nets (PTN), a particular type of Petri Net [2]. Each circle (corresponding to a PTN *place*) represents a good. Horizontal bars connecting goods represent manufacturing operations, likewise *transitions* in a PTN. Manufacturing operations are labeled with a numbered *t*, and shall be referred to as *transformation relationships* (*t-relationships* henceforth). 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, g_2 is an *input good* to t_2, whereas g_6, g_7, and g_8 are *output goods* of t_2. Thus, t_2 represents the way g_2 is transformed. 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_2 indicate that 1 unit of g_6, 7 units of g_7, and 1 unit of g_8 are obtained after processing 1 unit of g_2. Each transformation has an associated cost every time it is carried out. In our example, transformation t_2 costs €7.

AAMAS'06 May 8–12 2006, Hakodate, Hokkaido, Japan.

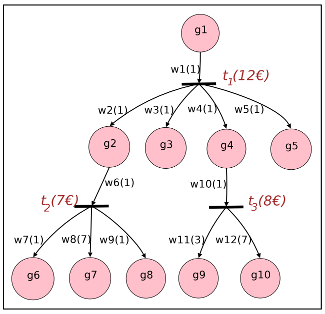

Figure 1: Example of a Transformability Network Structure.

Say that a buying agent requires to purchase a certain amount of goods g_3, g_5, g_6, g_7, g_8, g_9, and g_{10}. For this purpose, it may opt for running a combinatorial reverse auction with qualified providers. But before that, a buying agent may realise that he faces a decision problem: shall he buy g_1 and transform it via an in-house process, or buy already-transformed goods, or opt for a *mixed-purchase* solution and buy some already-transformed goods and some to transform in-house? This concern is reasonable since the cost of g_1 plus transformation costs may eventually be higher than the cost of already-transformed goods.

The work in [1] addresses the possibility of expressing transformability relationships among the different assets to sell/buy on the bid-taker side in a multi-unit combinatorial reverse auction. The new type of combinatorial reverse auction (the Multi-Unit Combinatorial Reverse Auction with Transformability Relationships among Goods (MUCRAtR)) provides to buying agents: (a) a language to express required goods along with the relationships that hold among them; and (b) a winner determination problem (WDP) 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. It is shown that, if the TNS representing the relationships among goods is acyclic, the associated WDP is modeled by the following integer program:

$$min[\sum_{j=1}^{m} x_j p_j + \sum_{k=1}^{r} q_k c(t_k)] \tag{1}$$

$$\forall\, 1 \leq i \leq n \quad \sum_{j=1}^{m} a_j^i x_j + \sum_{k=1}^{r} q_k m_k^i \geq u_i \tag{2}$$

where $x_j \in \{0,1\}\ \forall\, 1 \leq j \leq m$ stands for whether bid b_j is selected or not, p_j is the price associated to bid b_j, q_k is a decision variable taking into account how many times transformation t_k is fired, a_j^i is the number of units of good i offered in bid b_j, u_i is the number of required unit of good i, m_k^i is obtained from the *incidence matrix* [2] of the place-transition net within a TNS,

$c(t_k)$ stands for the cost associated to transformation t_k, m is the number of bids, n is the number of different negotiated goods, and r is the number of *t-relationships*. Expression (1) minimises the sum of the costs of the selected bids plus the cost of the transformations to apply, and equation (2) enforces that the selected bids plus the transformations applied at least fulfill a buyer's requirements. We will assume a finite production capacity, that is $q_k \in \{0, 1, \ldots, max_k\}, 1 \leq q_k \leq r$.

Notice that the integer program above can be clearly regarded as an extension of the integer program associated to a Multi Unit Combinatorial Reverse Auction (MUCRA) WDP as formalised in [3]. Thus, the second component of expression (1) changes the overall cost as transformations are applied, whereas the second component of expression (2) makes sure that the units of the selected bids fulfill a buyer's requirements, taking into account the units consumed and produced by transformations.

The purpose of this paper to quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider tranformability relationships.

2. EMPIRICAL EVALUATION

Our experiments artificially generate different data sets. Each data set shall be composed of: (1) a TNS; (2) a Request for Quotations (RFQ) detailing the number of required units per good; and (3) a set of combinatorial bids. Then, we solve the WDP for each auction problem regarding and disregarding *t-relationships*. This is done to quantitatively assess the potential savings that a buyer/auctioneer may obtain thanks to *t-relationships*. Thus, the WDP for a MUCRA will only consider the last two components of the data set, whereas the WDP for a MUCRAtR will consider them all. In order to solve the WDP for a MUCRA we exploit its equivalence with the multi-dimensional knapsack problem [3]. The WDP for a MUCRAtR is modeled by the integer program represented by expressions (1) and (2). In what follows we describe the way to artificially generate such data set.

2.1 Data Set Generation

In order to create a data set, the most delicate task is concerned with the generation of a collection of combinatorial bids. Unfortunately, we cannot benefit from any previous methods for artificially generating auction data sets in the literature since they do not take into account the novel notion of t-relationship.

TNS generation. Firstly, we consider the creation of a TNS. As explained in the introduction, if we restrict to the case of an acyclic TNS, then the WDP for a MUCRAtR can be formulated as an integer program. Thus, we shall focus on generating acyclic TNSs for our data sets. For this purpose, we create TNSs fulfilling the following requirements: (a) each transition receives a single input arc; (b) each place can have no more than one input and one output arc; and (c) there exists a place, called *root place*, that can only have output arcs. Figure 1 depicts an example of a TNS that satisfies such requirements. A distinguishing feature of our algorithm is that, since we aim at empirically assessing the potential savings when considering *t-relationships* independently of TNSs' shapes, it is capable of constructing acyclic TNSs that may largely differ in their shapes, and in the combination of weights assigned to arcs. Our generator randomly constructs TNSs receiving as inputs: (1) a number of places n (the number of goods); (2) a number of *t-relationships* r; (3) the minimum/maximum arc weight w_{min}/w_{max} (each arc weight is chosen from a uniform discrete distribution $U[w_{min}, w_{max}]$); and (4) the minimum/maximum transformation cost c_{min}/c_{max} (a transformation cost for each *t-relationship* is drawn from a uniform distribution $U[c_{min}, c_{max}]$).

RFQ generation. An RFQ is represented as a set $U = \{u_1, \ldots, u_n\}$ where u_i stands for the number of units requested of good g_i. We generate each $u_i \in U$ from a uniform discrete distribution $U[u_{min}, u_{max}]$, where u_{min} and u_{max} stand for the minimum and maximum number of units required per item respectively.

Bid generation. Finally, we complete the artificial generation of a data set by generating a set of plausible bids. Each bid $b_j \in B$ can be represented as a pair $\langle p_j, [a_j^1, \ldots, a_j^n] \rangle$ where p_j stands for the bid price and $[a_j^1, \ldots, a_j^n]$ for the units offered per good. For each bid b_j our generator firstly obtains the number of jointly offered goods from a binomial distribution with parameters $(p_{offered\_goods}, n)$, (say z goods); then it randomly selects z goods in G (the set of required goods). For each one of the z selected good g_i, the number of offered units is obtained from another binomial distribution parameterised by $(p_{offered\_units}, u_i)$. We employ binomial distributions since our aim is to maintain a proportionality relationships among: (1) the number of negotiated goods and the cardinality of offers; and (2) the number of required units and the number of offered units per good. This is done since we would like to analyse separately the effects of such parameters on savings, and we want to avoid inter-dependency effects. For instance, employing a geometric distribution to describe the number of offered units would implicitly create a dependency effect among the number of required units and the number of offered units, since increasing the number of required units would have the equivalent effect of lowering the number of offered units. Instead, a binomial distribution allows to analise, *ceteris paribus*, the effect of increasing the number of required units.

After generating the units to offer per good for all bids, we must assess all bid prices. This process is rather delicate when considering *t-relationships* if we want to guarantee the generation of plausible bids. In general, it is unrealistic to think of a market scenario wherein raw goods are more expensive than transformed goods. Hence, we assume that all providing agents produce goods in a *similar* manner (they share similar TNSs). However, goods' prices and transformation costs differ from provider to provider. In practice, our providing agents use the same TNS as the buying agent, though each one has his own transformation costs. Thus, for each provider we compute the unitary price for each good in the TNS. Thereafter, for each provider, we use his unitary prices to construct his bids.

Next, we describe how to calculate the unitary prices for each good for a given provider. We depart from the value of the p_{root} parameter, standing for the average unitary price of the root good (e.g. the root good in figure 1 is g_1). The first step of our pricing algorithm calculates the unitary price of the root good for each provider under the assumption that all providers have similar values for such good. Thus, for each provider P_j, his unitary price for the root good is assessed as $\pi_{root,j} = p_{root} \cdot \lambda$, where λ is sampled from a normal distribution $N(\mu_{root\_price}, \sigma_{root\_price})$. After that, our pricing algorithm recursively proceeds as follows. Given a provider and a good whose unitary price has been already computed, this is *propagated down* the provider's TNS through the transition it is linked to towards its output goods. We compute the value to propagate by weighting the unitary price by the value labelling the arc connecting the input good to the transition, and adding the provider's particular transformation cost of the transition. The resulting value is unevenly distributed among the output goods according to a *share factor* randomly assigned to each output good. For instance, consider the TNS in figure 1 and a provider P_j such that its unitary cost for g_2 is $\pi_{g_2,j} =$ €50, *his* transformation cost (different from the buying agent's one) for t_2 is €10, and $w_6 = 1$.

Table 1: **Parameters characterising our experimental scenario**

Parameter	*Explanation*	*Value*
n	The number of items	20
r	The number of transitions	8
u_{min}, u_{max}	The minimum/maximum number of units required per item	10/10
w_{min}, w_{max}	Minimum/Maximum arc weight	1/5
c_{min}, c_{max}	Minimum/Maximum Transformation cost	10/10
m	The number of bids to generate	1000
$p_{offered\_goods}$	Statistically sets the number of items simultaneously present in a bid	0.2 - 0.3 0.4 - 0.5
$p_{offered\_units}$	Statistically sets the number of unit offered per item	0.2 - 0.3 0.4 - 0.5
p_{root}	Average price of the $root$ good	1000
μ_{root_price} σ_{root_price}	Parameters of a Gaussian distribution weighting the $root$ price p_{root}	1 0.01
$\mu_{production\_cost}$ $\sigma_{production\_cost}$	Parameters of a Gaussian distribution setting the production costs difference between buyer and providers	0.8:0.1:1.8 0.1

In such a case, the value to split down through t_2 towards g_6, g_7, and g_8 would be $50 \cdot 1 + 10 =$ €60. Say that g_7 is assigned 0.2 as share factor. Thus, $60 \cdot 0.2 =$ €12 would be allocated to g_7. Finally, that amount should be split further to obtain g_7 unitary price since $w_8 = 7$. Then, the final unitary price for g_7 is €$1.7142 = \frac{12}{7}$. Hence, we can provide a general way of calculating the unitary price for any good for a given provider. Let P_j be a provider and g a good such that $a_j^g \neq 0$. Let t be a transition such that g is one of its output goods, and $father(g)$ is its single input good. Besides, we note as G' the set of output goods of t. Then, we obtain $\pi_{g,j}$, the unitary price for good g as follows:

$$\pi_{g,j} = \frac{\pi_{father(g),k} \cdot |M[father(g), t]| + c(t) \cdot \nu}{M[g, t]} \omega_g \qquad (3)$$

where $\pi_{father(g),k}$ is the unitary price for good $father(g)$ for a provider $P_k \neq P_j$; $|M[father(g), t]|$ indicates the units of good $father(g)$ that are input to transition t; ν is a value obtained from a normal distribution $N(\mu_{production\_cost}, \sigma_{production\_cost})$ that weighs transformation cost $c(t)$; $M[g, t]$ indicates the number of units of good g that are output by transition t; and ω_g is the share factor for good g. Notice that after applying our pricing algorithm we obtain Π, an $n \times m$ matrix storing all unitary prices.

Several remarks apply to equation 3. Firstly, the share factors for output goods must satisfy $\sum_{g' \in G'} \omega_{g'} = 1$. Secondly, it may surprise the reader to realise that the value to propagate down the TNS ($\pi_{father(g),k}$) is collected from a different provider. We enforce this *crossover* operation among unitary prices of different providers to avoid undesirable *cascading effects* that occur when we start out calculating unitary prices departing from either high or low unitary root prices. In this way we avoid to produce non-competitive and extremely competitive bids respectively that could be in some sense regarded as noise that could eventually lead to diverting results. Finally, from equation 3 we can readily obtain the bid price for a bid $b_j \in B$ as $p_j = \sum_{i=1}^{n} a_j^i \cdot \pi_{i,j}$.

After generating a complete data set, in a MUCRA scenario the Winner Determination Algorithm (WDA) shall solely focus on finding an optimal allocation for the required goods, whereas in a MUCRAtR scenario, the WDA shall assess whether an optimal allocation that considers the buying agent's *t-relationships* can be obtained. Therefore, the difference is that a MUCRAtR WDA does consider and exploit both the buying agent's *t-relationships* along with the implicit transformation cost within each bid, while a MUCRA WDA does not.

To summarise, the parameters we must set to create an auction data set are reported in table 1.

2.2 Experimental Settings and Results

In order to measure the benefits provided by the introduction of t-relationships among goods we compute the cost of the optimal outcome, that is, the cost of the winning bid set for MUCRA (C^{MUCRA}) and the cost of the winning bid set plus transformations for MUCRAtR ($C^{MUCRAtR}$). We define the *saving index* (SI) as: $SI = 100 \cdot \frac{C^{MUCRA} - C^{MUCRAtR}}{C^{MUCRA}}$. The larger the index, the higher the benefits that a buyer can expect to obtain by using a MUCRAtR instead of a MUCRA. Our experimental hypothesis is that the SI index will increase as the buyer's transformation costs increase wrt the providers' ones. Therefore, we will analyse how the difference in production costs between the buyer and the providers affects SI. The differences among the transformation costs of the buyer and the average transformation costs of the providers are controlled by the $\mu_{production\_cost}$ parameter. We expect that, as the average transformation costs of the providers increases with respect to the buyer's ones, so do the benefits of using a MUCRAtR instead of a MUCRA. In fact, the experimental results do strongly agree with our hypothesis. Figure 2 depicts the results when varying $\mu_{production\_cost}$ from 0.8 to 1.8. The legend reports the value of the $p_{offered\_goods}(=)p_{offered\_units}$ parameter. As $\mu_{production\_cost}$ increases, so do savings.

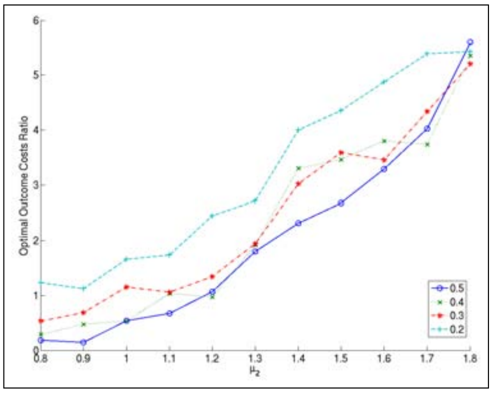

Figure 2: Varying the $\mu_{production\_cost}$ parameter

3. CONCLUSIONS AND FUTURE WORK

We have performed an experiment to empirically evaluate under which market conditions it is convenient to employ the new auction defined in [1]. We found that the best benefits in terms of savings are obtained when: (1) the manufacturing costs of a buyer/auctioneer are higher than the providers' ones; and (2) a buyer cannot access the raw materials at a price lower or equal than the providers.

Acknowledgements. This work was partially funded by the Spanish Science and Technology Ministry as part of the Web-i-2 project (TIC-2003-08763-C02-00). Giovannucci Andrea enjoys the BEC.09.01.04/05-164 CSIC scholarship.

4. REFERENCES

[1] A. Giovannucci, J. A. Rodrguez-Aguilar, and J. Cerquides. Multi-unit combinatorial reverse auctions with transformability relationships among goods. In *WINE 2005*, volume 3828 of *LNCS*, Hong Kong, China, December 2005. www.iiia.csic.es/∼andrea/papers/WINE318a.pdf.

[2] T. Murata. Petri nets: Properties, analysis and applications. In *IEEE*, volume 77, pages 541–580, 1989.

[3] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *First International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pages 69–76, Bologna, Italy, July 2002.