

Benefits of Combinatorial Auctions with Transformability Relationships

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 transformability relationships. Furthermore, we empirically identify the market conditions under which it is worth for the auctioneer/buyer to exploit transformability 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 [3]. 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.

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 trans-

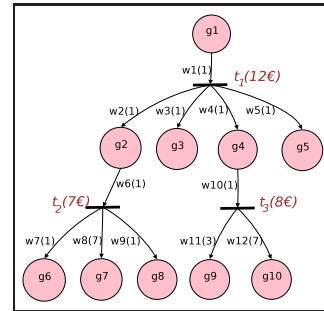


Figure 1. Example of a Transformability Network Structure.

formation 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 (MUCRA_{tR})) 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:

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

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

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* [3] 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 $q_k \in \{0, 1, \dots, \max_k\}, 1 \leq q_k \leq r$.

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 [4]. 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 is twofold. On the one hand, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider transformability relationships. On the other hand, we empirically identify the market conditions under which it is worth for the auctioneer/buyer to exploit transformability relationships. Thus, we provide rules of thumb for an auctioneer/buyer to help him decide when to run a MUCRA_{tR} instead of an MUCRA.

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*, as well as the market conditions where such savings occur. Thus, the WDP for an MUCRA will only consider the last two components of the data set, whereas the WDP for a MUCRA_{tR} will consider them all. In order to solve the WDP for an MUCRA we exploit its equivalence with the multi-dimensional knapsack problem [4]. The WDP for a MUCRA_{tR} 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 ([2]) 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 MUCRA_{tR} 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. We would like to fairly compare MUCRA with MUCRA_{tR}. With this in mind we consider that a very same benchmark — with identical bids — must be employed for both auction types. It comes out that is very difficult to provide a set of bid fair for both auction types.

As pointed out in section 1, when a buyer expresses a set of *t-relationships*, he is also implicitly stating some intrinsic relationships that hold among the goods involved in his production process. Consider a buyer requiring 100 units of good g_2 . Say that his TNS allows him to obtain 20 units of good g_2 transforming 1 unit of g_1 . Therefore, the buyer needs either 100 units of g_2 , or 5 units of g_1 (to transform it into g_2). Thus, providers of good g_1 should offer 20 times less units than providers of g_2 .

An example of offer that rewards a MUCRA_{tR} while penalising an MUCRA is: 5 units of good g_1 . In fact this offer cannot be included in the winning set of an MUCRA, since good g_1 is not even required by the buyer in his RFQ. Indeed, a MUCRA_{tR} can include this offer in the winning set, since the 5 units of g_1 are transformed via the TNS. On the other hand, consider that providers only submit offers for good g_2 . In this case a MUCRA_{tR} can not exploit transformations, and the outcomes of MUCRA and MUCRA_{tR} are identical, thus penalising MUCRA_{tR}. Therefore, we need to provide a fair enough bid set for both.

The solution we propose is to introduce the information about quantity relationships among goods directly into an RFQ. Hence, we enforce that the quantities required for different goods hold the quantity relationships induced by the TNS. Continuing the example above involving g_1 and g_2 , an RFQ respecting the quantity relationships would be requiring 5 units of g_1 and 20 units of g_2 . In the actual experiments, required quantities are then modulated by a gaussian distribution in a subsequent step, with the purpose of introducing some randomness.

Next, we detail how to artificially generate an RFQ. This is represented as a set $U = \{u_1, \dots, u_n\}$ where u_i stands for the number of required units of good g_i . Thus, generating an RFQ amounts to setting a value for each $u_i \in U$ as follows. First, we compute the number of required units of the root good: $u_{root} = units_{RFQ} \cdot \nu$. The $units_{RFQ}$ parameter stands for the average number of units required for the root good. Next, departing from the root good, we compute the number of required units u_i for each remaining good g_i of the TNS according to the following procedure. Let t be a transition such that g_i is one of its output goods, and $father(g_i)$ is its single input good¹. Then, we obtain u_i as follows:

$$u_i = \frac{u_{father(g_i)} \cdot M[g_i, t]}{|M[father(g_i), t]|} \cdot \nu \quad (3)$$

where $|M[father(g_i), t]|$ indicates the units of good $father(g_i)$ that are input to transition t ; ν is a value obtained from a normal distribution $N(1, \sigma_{RFQ})$; and $M[g_i, t]$ indicates the number of units of good g_i that are output by transition t .

Bids 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 $\langle p_j, [a_j^1, \dots, a_j^n] \rangle$ where p_j stands for the bid price and $[a_j^1, \dots, a_j^n]$ 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

¹ Recall that our method to construct acyclic TNSs ensures that there is a single input good per transition.

($p_{offered_units}, u_i$). 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 analyse, *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, which in turn are assessed as a variation of the buying agent's ones. 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 $\pi_{g_2,j} = \text{€}50$, his transformation cost (different from the buying agent's one) for t_2 is $\text{€}10$, and $w_6 = 1$. In such a case, the value to split down through t_2 towards g_6, g_7 , and g_8 would be $50 \cdot 1 + 10 = \text{€}60$. Say that g_7 is assigned 0.2 as share factor. Thus, $60 \cdot 0.2 = \text{€}12$ 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 $\text{€}1.7142 = \frac{12}{7}$. 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:

$$\pi_{g,j} = \frac{\pi_{father(g),k} \cdot |M[father(g),t]| + c(t) \cdot \nu}{M[g,t]} \omega_g \quad (4)$$

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 4. 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 4 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 a combinatorial auction with transformation data set are reported in table 1.

Table 1. Parameter settings of our empirical analysis.

Parameters	Possible Values			
n	8	10	12	14
r	2	3	4	5
$units_{RFQ}$	10	17	24	31
σ_{RFQ}	0.05	0.1	0.15	0.2
w_{min}, w_{max}	[2, 5]			
c_{min}, c_{max}	[10, 10]	[100, 100]	[20, 40]	[40, 80]
m	100	200	300	400
$p_{bid_density}$	0.2	0.3	0.4	0.5
$P_{offered_units}$	0.2	0.4	0.6	0.8
p_{root}	50	100	150	200
$\sigma_{root,price}$	0.05	0.1	0.15	0.2
$\mu_{production_cost}$	0.6	0.9	1.2	1.5
$\sigma_{production_cost}$	0.05	0.1	0.15	0.2

2.2 Experimental Settings and Results

Our goal is to determine under which market conditions MUCRAtr leads to savings when compared to MUCRA. At this aim, we empirically measure the differences in outcome cost between MUCRA and MUCRAtr. Thus, we define the *Savings Index (SI)* as: $SI = 100 \cdot (1 - \frac{C^{MUCRAtr}}{C^{MUCRA}})$; where C^{MUCRA} and $C^{MUCRAtr}$ are the costs associated to the optimal solutions found respectively by MUCRA and MUCRAtr WDAs.

With the aim of assessing the most sensitive parameters with respect to *SI*, we employ a fractional factorial experiment design [5]: we assign to each parameter four possible values, representing different grades in an increasing scale. Table 1 summarizes all the values that each parameter can take on. At each run of the experiment, we randomly assign one out of these four values to each parameter, generating a data set as detailed in section 2.1. Finally, we compute *SI* by assessing the optimal solutions for MUCRA and MUCRAtr.

Some remarks are in place as to how we assess *SI*: (1) we set a time deadline (800 sec.) to solve the problem: after such time is

elapsed, the partial solution obtained until that moment is considered; (2) since we know² that $SI \geq 0$, we will set it to 0 whenever it is negative; (3) we exclude the runs that did not find an optimal solution for MUCRA, since it makes no sense a savings comparative in this case.

We run 5756 instances of the experiments and for each run we sampled SI . In 150 cases (2.606%) the optimizer could not find an optimal solution within the time limit for MUCRA. In 289 cases (5.02%) the solver could not find an optimal solution within the time limit for MUCRA_{tR}. As explained above, the total number of samples that have been considered are $5756 - 150 = 5606$. Among these new samples, the optimizer could not find an optimal solution for MUCRA_{tR} for 191 (3.407%) tests.

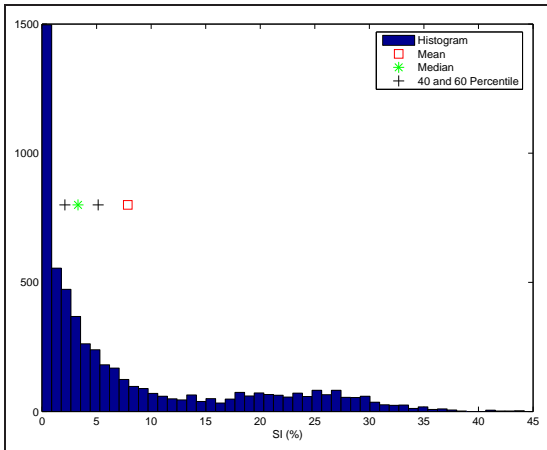


Figure 2. Histogram of SI .

General Observations. Figure 2 shows the histogram of SI . We observe its shape is highly skewed. From figure 2 and from the statistics in table 2 we observe that the mean and variance are not appropriate statistics to characterize the SI distribution. In fact, the high skewness makes the median a much better and stable descriptor. For all these reasons, we will employ the *median* and *percentile* statistics in what follows.

Table 2. Statistics for SI .

Statistic	SI
Mean	7.8574
Variance	93.1016
Range(min - max)	0 - 44.1959
Interquartile Range	11.06
Median	3.2987
70 - percentile	8.5971

In figure 2 and table 2 we observe that the savings of MUCRA_{tR} with respect to MUCRA go: (1) up to 44%; (2) beyond 3.29% in 50% of the cases; (3) beyond 8.59% in 30% of the cases. Next, we perform a sensitivity analysis in order to determine which parameters most affect an auction's outcome.

Sensitivity Analysis. Notice that the distribution depicted in figure 2 is bimodal: a mode around 0 and another one around 25. It is likely that there exists a parameter that causes such a behavior, a parameter that is heavily sensitive. In fact, figure 3 clearly shows that the highest mode corresponds to providers offering in average the 80% of the required units ($p_{offered_units} = 0.8$)³. The larger

² By definition $SI \geq 0$, since each solution of a MUCRA is also a solution to a MUCRA_{tR}, and thus $C^{MUCRA_{tR}} \leq C^{MUCRA}$.

³ Recall that $p_{offered_units}$ parameterises a binomial distribution describ-

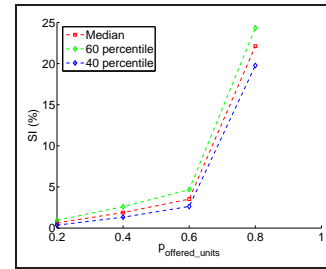


Figure 3. Variation of SI for different providers' capacities.

the percentage of offered units, the larger the number of unrequested goods that the MUCRA solution is likely to include; and hence, the more MUCRA_{tR} can take advantage over MUCRA by exploiting *t-relationships* to transform free-disposal goods into requested goods.

C1: *The higher the providers' capacities, the higher the expected savings when introducing t-relationships.*

For this reason, when analysing the behavior of SI with respect to the remaining parameters, we will differentiate two cases: (1) $p_{offered_units} < 0.8$; and (2) $p_{offered_units} = 0.8$.

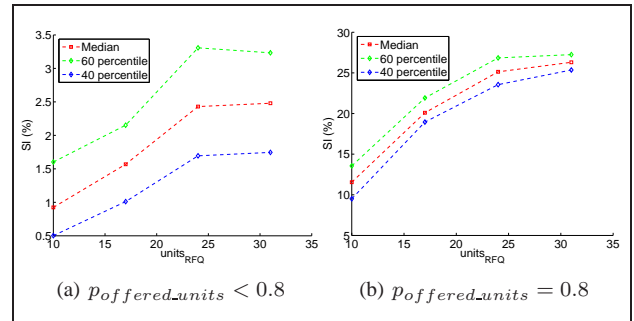


Figure 4. SI with respect to the number of required units.

Figure 4 shows the variation of SI with respect to the number of required units ($units_{RFQ}$). The high sensitivity of SI with respect to this parameter can be explained analysing the relationship between the number of required units and the arc weights. The larger $units_{RFQ}$, the larger the number of offered units (because of the binomial distribution parametrised by $(p_{offered_units}, u_i)$). After observing equation (2) we infer that the larger $units_{RFQ}$, the more fine-grained adjustments does a transformation allows because the number of output units of the transformation is smaller and smaller with respect to the number of required units. In other words, it is like trying to fill out bigger and bigger knapsacks using balls of the very same size every time. And hence the easier to redistribute free-disposal goods via transformations. For instance, say that a buyer associates to a MUCRA_{tR} the TNS in figure 1, with $w_1 = 5$, $w_2 = w_3 = 10$, and $w_4 = w_5 = 20$, and that his final requirements (U) are: 10 units of g_2 and g_3 , and 20 units of g_4 and g_5 . The activation of the transformation T_1 can happen at most 1 time, and an offer for g_1 has to be very convenient to activate it. If U is 100 units of g_2 and g_3 , and 200 units of g_4 and g_5 , the activation of T_1 is easier because it is more *fine-grained* with respect to the number of required units. Also notice that in figures 4(a) and 4(b) a saturation effect appears as $units_{RFQ}$ increases. We infer that although increasing $units_{RFQ}$ opens us more possibilities to redistribute offered goods via transformations, there is a limit because transformations carry a cost. Thus, our second conclusion is:

ing the number of units offered per good per provider. In other words, it characterises a provider's capacity in the market.

C2: The finer the granularity of the transformations, the higher the expected savings when introducing t -relationships.

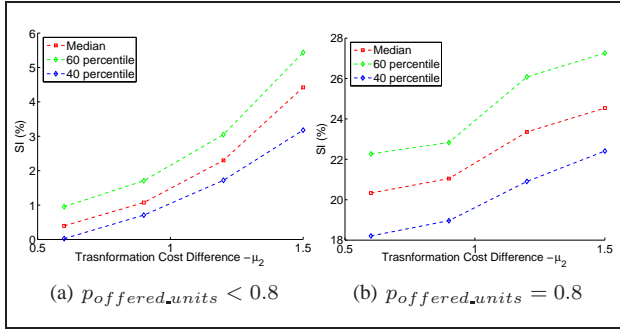


Figure 5. SI with respect to providers' transformation cost.

The third factor significantly affecting SI is the relationship between the transformation costs of a buying agent with the providers' ones ($\mu_{production_cost}$). If $\mu_{production_cost} > 1$ on average providers transform goods with a higher cost than the buying agent, and the other way around when $\mu_{production_cost} < 1$. Figures 5(a) and 5(b) show that SI increases as $\mu_{production_cost}$ increases. This behavior is motivated as follows: the increment of $\mu_{production_cost}$ models that in-house transformations are cheaper, therefore more likely to be exploited. a MUCRA tR saves with respect to a MUCRA as more in-house transformations are employed. In such a case the sets of winning bids of MUCRA and MUCRA tR largely differ among them. Figure 5 depicts the results when varying $\mu_{production_cost}$. Furthermore, we also observe that: (1) the variation of SI in both figures is around 4%. Thus, this increment is independent from the $Poffered_units$ parameter; (2) figure 5(b) is more irregular than figure 5(a), because of the wider spread of values when $Poffered_units = 0.8$; and (3) although $\mu_{production_cost}$ seemed intuitively to be the most sensitive parameter, our experiments shows that it is not.

A buyer can only obtain better savings with a MUCRA tR with respect to a MUCRA if there exists some possibility to perform in-house transformations. Taken to the extreme, if no in-house transformation is convenient, the outcome of a MUCRA tR is exactly the same as a MUCRA, and $SI = 0$. The third conclusion follows:

C3: The cheaper the in-house transformations with respect to the providers' ones, the higher the expected savings when introducing t -relationships.

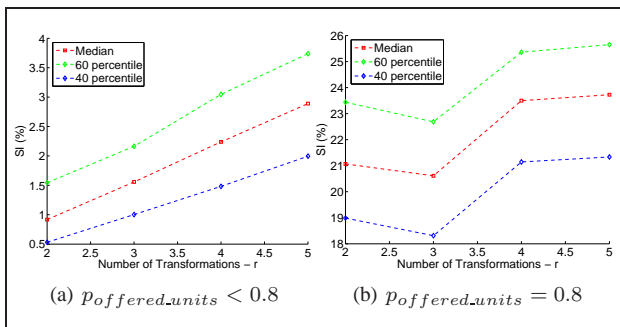


Figure 6. SI with respect to the number of transformations.

Figure 6 depicts SI when changing the number of transformations within the TNS (r). As expected, we notice that the more the transformations, the more likely the increment in savings. And yet, figure 6(b) is more irregular than figure 6(a) due to the same spread-effect described above.

C4: The more the number of transformations, the more the expected savings with respect to a MUCRA.

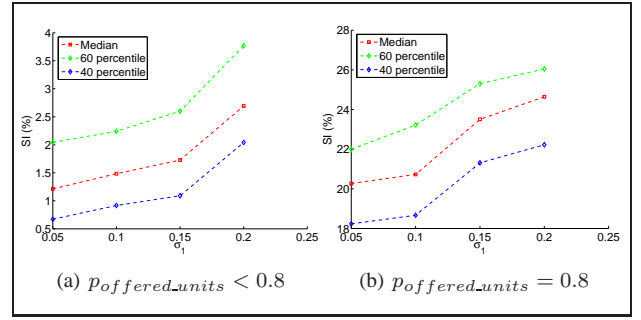


Figure 7. SI with respect to the variance of providers' prices.

Figure 7 shows the effect of varying the prices of goods among providers (σ_1). In other words, σ_1 controls price spread in the market. As the difference in prices grows (larger σ_1), does also SI grow. A reasonable explanation could be that MUCRA tR is more likely to benefit from a good price bid for an unrequested good, that the buyer can afterwards transform into a requested one. The fifth conclusion follows:

C5: The larger the market's prices spread, the higher the expected savings.

To summarise, we can indeed confirm, based on the observation above, that there are market conditions (identified by **C1**, **C2**, **C3**, **C4**, and **C5**) wherein it is worth using MUCRA tR instead of MUCRA.

3 Conclusions and Future Work

In this paper we have performed a set of experiments to quantitatively assess the potential savings in employing a MUCRA tR instead of a MUCRA. Furthermore, we have also identified the market conditions for which MUCRA tR is expected to lead to better auction outcomes to the auctioneer/buyer, namely: (1) markets with high-capacity providers; (2) auctions whose number of required units per good is large with respect to the units required by transformations (i.e. the likelihood of exploiting transformations is high); (3) auctions run by a buyer whose transformation (production) costs are cheaper than the providers' ones; and (4) markets where providers' competitiveness is not high (the more scattered the providers' competitiveness, the larger the expected savings). We envision several paths to future development. Firstly, a further extension to our experiments to carry out an analysis of the computational cost of introducing t -relationships (to complete the preliminary results presented in [1]). Secondly, the theoretical design of mechanisms that exploit t -relationships.

REFERENCES

- [1] Andrea Giovannucci, Juan A. Rodríguez-Aguilar, and Jesús Cerquides, 'Multi-unit combinatorial reverse auctions with transformability relationships among goods', in *LNCS*, volume 3828, Hong Kong, China, (December 2005). www.iiaa.csic.es/~andrea/papers/WINE318a.pdf.
- [2] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz, 'An algorithm for multi-unit combinatorial auctions', in *American Association for Artificial Intelligence (AAAI)*, (2000).
- [3] T. Murata, 'Petri nets: Properties, analysis and applications', in *IEEE*, volume 77, pp. 541–580, (1989).
- [4] T. Sandholm, S. Suri, A. Gilpin, and D. Levine, 'Winner determination in combinatorial auction generalizations', in *AAMAS 2002*, pp. 69–76, Bologna, Italy, (2002).
- [5] Box G.E.P. Hunter W.G. and Hunter J.S., *Statistics for experimenters An introduction to design, data analysis, and model building*, Wiley, 1978.