Composing Supply Chains through Multi-unit Combinatorial Reverse Auctions with Transformability Relationships among Goods

Andrea Giovannucci, Jesús Cerquides, and Juan A. Rodríguez-Aguilar

***Abstract*—In this paper we provide a decision support system (DSS) to auctioneers when assembling their supply chains. Sometimes an auctioneer/buyer has to decide whether to outsource some production processes or not (make-or-buy decision). For this purpose we add a new dimension to the goods at auction. A buyer can express transformability relationships (t-relationships) among goods: some goods can be transformed into others at some transformation cost. On the other hand, often another type of relationship among the goods at auction exists on the bidders' side, that is complementarity relationships. Combinatorial auctions are usually employed to deal with the problem of complementarities on the bidders' side. Then, in this paper we build upon combinatorial auctions to implement a DSS to solve make-or-buy decisions in markets characterized by combinatorial preferences.**

Firstly, we introduce the information about transformability relationships in the combinatorial auctions winner determination problem (WDP) so that an auctioneer/buyer can assess what goods to buy, to whom, and what transformations to apply to the acquired goods in order to obtain the required ones. In this way, an auctioneer can build his supply chain maximizing his revenue. We call the resulting model *Multi-Unit Combinatorial Reverse Auction with Transformability Relationships Among Goods* (MUCRAtR). This new auction type allows a buyer/auctioneer to express and communicate to bidders its internal production structure and its final requirements. Bidders can then formulate appropriate offers and send them back to the auctioneer. Upon receiving the offers, an auctioneer can determine, by means of a public selection rule, the cost minimizing combination of bids along with the internal operations leading to its final requirements. In order to solve the Winner Determination Problem. we map it to an optimization problem on Place/Transition Nets. Such mapping allows to efficiently solve the WDP for some problem classes, and provides a set of powerful formal tools for describing the underlying optimization problem. Finally, we quantitatively assess the potential benefits and drawbacks when considering transformability relationships.

***Index Terms*—Supply Chain, Combinatorial Auctions, Winner Determination Problem, Petri nets**

I. Introduction

A. Motivations

We are witnessing an important transformation of the firm organizational structure. Today's business world is experiencing a progressive disintegration of the traditional vertical integrity[1] of the enterprises' organizational structure. This is witnessed by a heavy increment in the use of outsourcing. Indeed, the structural integrity of organizations is breaking down; the traditional vertically integrated organizations, controlling as many of the production factors as possible, is being quickly replaced by better focused and more specialized organizations. An increased number of capable service providers, the pressure deriving from the hypercompetitivity, and the pervasive presence of technology impose a new strategic vision. As a result, new supply chain management [60] strategies are emerging, like strategic outsourcing [54], [31], [18] and collaborative supply chain network design [63]. This situation requires an increased automation across the supply chain. Indeed, static and vertically integrated supply chains are quickly giving way to more flexible value chains composed of partners that can be assembled in real time to meet unique requirements.

In this environment, it is critical the rapid, automatic and reliable selection of the right business partners. As a revealing phenomenon, one of the main objectives of current supply chain management [60] is to integrate as much as possible the *back-end* of the supply chain (its production and manufacturing portion) to the *front-end* (the final customer).

A spectrum of possible solutions is possibly needed by enterprises. On the one extreme, companies must make decisions about whether to outsource part of their production processes (buy/make decisions) in business environments characterized by myriads of possible partners (lower barriers caused an increment in competition). On the other extreme of the spectrum, virtual enterprises may need agile *decision support systems* (DSSs) that allow them to automatically form self-organizing supply chains. Indeed, we do believe that nowadays firms require DSSs that allow them to nimbly and automatically select strategic business partners. As a first step, those DSSs should allow firms to automate the process of partner selection, optimizing critical *make-or-buy* decisions across the supply chain (i.e. trading off decisions of internal vs external production) with multiple potential partners. With this aim, we deem absolutely necessary to tightly integrate the procurement and outsourcing strategies and decisions. Thus, the decision support should allow them to self-organize by allowing to:

- integrate and coordinate all the supply chain stake holders;

Andrea Giovannucci and Juan A. Rodríguez-Aguilar are with the Artificial Intelligence Research Institute, Spanish National Research Council, Bellaterra, Barcelona, 08193 SPAIN e-mail: {andrea,jar}@iiia.csic.es.

Jesús Cerquides is with the University of Barcelona. Gran Via de les Corts Catalanes, 585, 08007 Barcelona. Spain.

[1] In microeconomics and management the term *vertical integration* describes the degree to which a firm owns its upstream suppliers and its downstream buyers.

- include component suppliers into the supply chain design process;
- optimize the overall performance of the supply chain (i.e. not a local optimization);
- negotiate based on direct cost competition, that trades off decisions of internal vs external production. It is well known that one of the positive results of outsourcing is putting in competition the internal activities of an enterprise with the outside world.
- to express the partners' preferences in a flexible and efficient way.

With the above-mentioned requirements fulfilled, competitive companies could easily cope with the selection of optimal, tightly connected procurement, outsourcing, and collaboration strategies.

B. Requirements

In what follows we present a motivating example concerning the main issue we intend to face in this thesis: the problem of efficiently solving *make-or-buy* decisions across the supply chain when complementarities among goods hold on the bidders' side.

Example I.1. Consider a company, named *Grandma & co*, devoted to produce and sell apple pies. The internal production structure of the company, i.e. the way apple pies are prepared, is presented in figure 1. Each circle represents a raw, intermediate or manufactured good. Squares connecting goods represent manufacturing operations. An arc connecting a good to an operation indicates that the good is an *input* to the operation, whereas an arc connecting an operation to a good indicates that the good is an *output* of the operation. Then, *butter*, *sugar*, and *flour* are *input goods* to the *Make Dough* operation, whereas *dough* is an *output good* of the *Make Dough* operation. The labels on the arcs connecting *input goods* to operations, and the labels on the arcs connecting *output goods* to operations indicate the units required of each *input good* to perform an operation and the units generated per *output good* respectively. In our example, the preparation of two units of *dough* requires one unit of *butter*, three units of *sugar*, and two units of *flour*.

Each operation has an associated cost every time it is carried out. We label each operation with a cost. In our example, the *Make Dough* operation costs 5 €.

Consider that the marketing department at *Grandma & co* forecasts that two hundred apple pies will be sold within a month and that the stock of *Grandma & co* is composed of one hundred units of *flour* and two hundreds units of *sugar*. Therefore, the company starts an automated sourcing [32] process to acquire the basic ingredients needed for producing pies, namely *butter*, *sugar*, *flour*, *apples*, and *margarine*.

However, the production management staff decides to test a new sourcing process. Instead of limiting the procurement to basic ingredients, they decide to incorporate in the sourcing process intermediate and final goods as well, namely *dough*, *filling*, and *apple pies* in figure 1. More precisely, the production management wonders whether to *outsource* part of its production process. In fact, the executive staff noticed that more and more specialized enterprises are entering the organic food market. Since *Grandma & co* is a well-known brand for pies, it decides that in order to reduce costs, it could be suitable to negotiate and collaborate with those new brands.

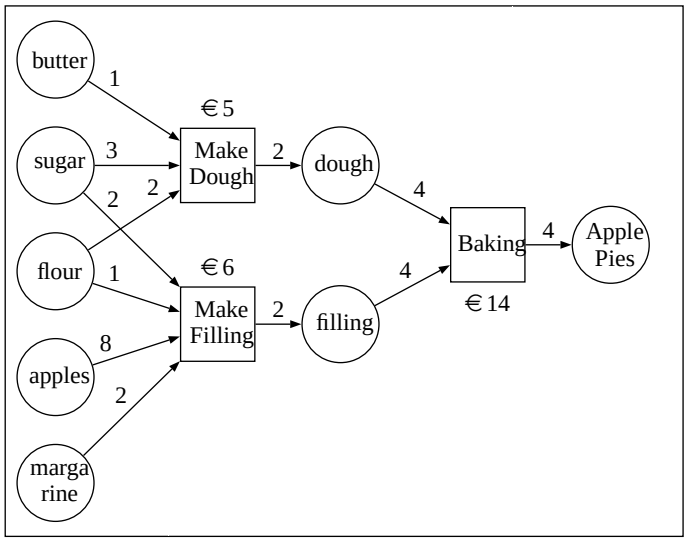


Fig. 1. Apple pie production flow.

As an additional constraint, the production management knows that strong complementarities among the negotiated goods exist on the supplier side. For instance, suppliers often sell margarine and butter as indivisible bundles. Thus, it is required that those complementarities are taken into account. □

Grandma & co realizes that it faces a decision problem: shall it buy the required ingredients and internally produce apple pies, or buy already-made apple pies (outsource all its production), or opt for a *mixed purchase* and buy some ingredients for internal production and some already-made apple pies? This concern is reasonable since the cost of ingredients plus preparation costs may eventually be higher than the cost of already-made apple pies. *Grandma & co* must take a decision among many possible mutually exclusive options:

- buy all the basic ingredients to internally produce all the pies;
- buy from suppliers all the pies and resell them under its name;
- buy already-made dough and filling from suppliers , and bake itself the cake;
- prepare part of the dough and part of the filling, and buy the rest from suppliers;
- buy part of the pies from suppliers and produce the rest itself;

Grandma & co is interested in quantitatively assessing what to buy and from whom, as well as what to produce in house. Such assessment depends on many factors:

(1) the market cost of the basic ingredients (butter, sugar, flour, apples, and margarine);
(2) the market cost of dough, filling, and pies;
(3) the stock goods at *Grandma & co*;

(4) the finally required goods (the sales forecast);
(5) the cost for performing at *Grandma & co* the operations *Make Dough*, *Make Filling*, and *Baking* (the internal cost structure);
(6) the number of units of each good either produced or required for each operation (the internal production structure); and
(7) the complementarity relationships among goods holding on the suppliers' side.

Hence, *Grandma & co* requires a complex decision support system along with a negotiation mechanism that helps it in detecting which is the revenue maximizing buying configuration and the internal operations to perform in order to obtain the finally required goods. It is easy to understand from the example that the procurement and outsourcing decisions are tightly linked. Notice that there is a mutual dependency among the outsourcing opportunity, the ingredients' market prices (as Dough, Apples,etc.) and other factors. This kind of dependencies must be absolutely captured by any proposed solution.

The literature on procurement has introduced combinatorial reverse auctions to deal with the problem of complementarities among goods on the bidders' side. In the following section we briefly recall some knowledge about electronic sourcing and combinatorial auctions.

1) The procurement phase: Although reverse auctions are certainly valuable to swiftly negotiate with providers, combinatorial (reverse) auctions may lead to more efficient allocations whenever complementarities among the goods at auction hold, as argued in [57], [16]. For instance, It can avoid phenomena like depressed bidding[2]. A combinatorial (reverse) auction [20] is an auction where bidders can sell (buy) entire bundles of goods in a single transaction. Although computationally very complex, selling (buying) items in bundles has the great advantage of eliminating the risk for a bidder of not being able to obtain (sell) complementary items at a reasonable cost (price) in a follow-up auction [3].

In particular, connected with the introduction of combinatorial auctions are bidding languages [49] and the winner determination problem (WPD) [40]. Winner determination is the problem, faced by the auctioneer, of choosing what goods to award to which bidder so as to maximize its revenue. The winner determination for combinatorial auctions is a complex computational problem. In particular, it has been shown that the WDP is NP-complete [56]. Bidding is the process of transmitting one's valuation function over the set of goods at offer to the auctioneer (or rather *some* valuation function — the bidders are of course not required to reveal their true valuation —).

Since *Grandma & co* aims at dealing with the case in which complementarities among goods hold at the bidder's side, combinatorial auctions is for sure among the most suitable sourcing methods. Then, we build upon combinatorial auctions. However, combinatorial auctions cannot be directly employed for the problem explained in example I.1 due to some intrinsic limitations. Then, in what follows, we analyze the requirements associated to the *make-or-buy* decision problem that are not fulfilled by combinatorial auctions, and we discuss the extensions required in order to deal with such decision problem.

2) Combinatorial Auction limitations: Say that *Grandma & co* opts for running a combinatorial reverse auction [59] with qualified providers for the procurement of all the required goods. Unfortunately, traditional combinatorial reverse auctions cannot be applied to solve such a problem for three reasons. Firstly, because of expressiveness limitations, namely an auctioneer (*Grandma & co*) cannot express:

- its internal manufacturing operations along with the producer/consumer relationships holding among them (for instance, in figure 1, the output of *Make Dough* is an input of *Baking*);
- the relationships between the manufacturing operations and the auctioned goods (for instance, in figure 1, the input to the *Make Dough* operation is three units of *sugar*, two units of *flour* and one unit of *butter*, whereas its output is two units of *dough*);
- the relationships between the received bids and the internal manufacturing operations;
- the requirements sent to bidders. This is clarified by observing that even though the final requirements of *Grandma & co* are two hundred apple pies, multiple request configurations fulfil such outcome, for instance:
  - two hundred already-made apple pies
  - the basic ingredients plus in-house production of two hundred apple pies

  How can *Grandma & co* formally describe its requirements? What should the requirements sent to bidders be? In fact, the optimal requirements depends on the received offers, and therefore cannot be stated a priori.
- the cost associated to performing each internal operation or a set of internal operations.

The second problem is that the outcome of a combinatorial auction only provides information about what goods to buy and from whom. However, the information about which internal manufacturing operations to perform and the order in which the auctioneer has to perform them (in the example of figure 1, the auctioneer cannot perform the *Baking* operation before *Make Dough* or *Make Filling*) is not provided.

Table I summarizes the requirements stemming from the *make-or-buy* decisions that are not supported by any state-of-the art solution.

Although combinatorial auctions help set the market price of each good, they do not incorporate the notion of internal manufacturing operation. This is why all the above-mentioned difficulties arise.

Summarizing, *Grandma & co* requires an extended combinatorial reverse auction that provides:

(1) a formal language to quantitatively express, analyze,

[2]Depressed bidding is a phenomenon associated to the fact that bidders may risk to obtain only a part of a set of complementary goods, and therefore bid less aggressively.

[3]Think of a combinatorial auction for a parking and house that are very close, as opposed to two consecutive single-item auctions for each of the individual items. One may risk to buy a parking in the first auction and missing the house in the second one. That is why they should be negotiated together.

TYPE	LIMITATION
Expressiveness	(1) internal manufacturing operations and the producer/consumer relationships among them (2) specification of an auctioneer's final requirements (3) relationships among the manufacturing operations, the auctioned goods, and the received bids (4) specification of an auctioneer's internal cost structure
WDP	(5) information about which in-house operations to perform and in which order

TABLE I
SUMMARY OF UNFULFILLED REQUIREMENTS.

and communicate its internal production structure and requirements; and

(2) an efficient cost minimizing winner determination solver that not only assesses which goods to buy and from whom, but also the sequence of internal manufacturing operations needed to obtain the finally required goods.

C. Contributions

In this paper we contribute with a generalization of combinatorial auctions providing support to the *make-or-buy* decision problems. In particular, we present an extension to combinatorial auctions that we shall refer to as *Multi-Unit Combinatorial Reverse Auction with Transformability Relationships Among Goods* (MUCRAtR). MUCRAtR automates *make-or-buy* decision problems in scenarios characterized by combinatorial preferences. This new auction type provides an auctioneer with a framework to optimize its outsourcing and procurement strategy. In particular, it allows an auctioneer:

- to formally express its internal production structure, i.e. the transformability relationships (*t-relationships*); and
- to automatically and efficiently assess which goods to buy and from whom, along with the *sequence* of internal operations to perform in order to obtain some required resources.

In order to provide a language to express the internal production structure of an auctioneer, we extend Petri Nets (refer to section II or to [48]), a well-known graphical and formal tool to analyze discrete dynamical systems. We call such extended model *Weighted Place Transition Nets* (WPTNs). The semantics of WPTNs naturally captures:

- the producer/consumer relationships holding among manufacturing operations; and
- the relationships among goods at auction, auctioneer's internal operations, and bids.

Next, in order to provide a formal definition to the auctioneer's decision problem, we define a new optimization problem on WPTNs, the *Constrained Maximum Weighted Occurrence Sequence Problem* (CMWOSP). The resulting optimization problem perfectly captures the nature of the auctioneer's decision problem. We anticipate that the newly introduced optimization framework allows to import a wide body of analysis methods from Petri Nets theory and apply them to our decision problem, thus providing methods and tools for its analysis. Subsequently, in order to practically solve the auctioneer's decision problem, we exploit analysis methods imported from Petri Nets theory and manage to provide an efficient Integer Linear Programming [33] formulation of the problem. However, this formulation only works whenever an auctioneer's internal production structure is acyclic. That is, there are no cycles in a production process.

Once solved the WDP, the buyer/auctioneer must be aware of the benefits and drawbacks of running an auction with *t-relationships*. In other words, the buyer must be aware of rules of thumb indicating when considering *t-relationships* is beneficial, and thus leads to more savings when composing his supply chain. Hence, as a second contribution, we empirically analyze the benefits and shortcomings, in terms of savings and computational costs, of introducing *t-relationships* with respect to a classic multi-unit combinatorial reverse auction by extending our former work in [26].

As to savings, our goal is twofold. Notice that to the best of our knowledge no benchmark or data set in the literature can be employed to perform our analysis, since: (1) none implements a *t-relationships* generator; and (2) *t-relationships* must be taken into account when generating bids. Thus, we had to implement our own auction scenario generator that adds some levels of complexity to the state-of-the-art CA data set generators. Our software generates artificial negotiation scenarios to compare in a fair way MUCRAs[4] with MUCRAtRs.

This paper is organized as follows. In section II we recall some background knowledge about Place Transition Nets that will be useful in order to understand what explained from section III on. In sections III we explain how to represent the search space associated to the decision problem. In section IV we introduce Weighted Place Transition Nets. In sections V and VI we map the optimization problem to CMWOSP. In section VII, we provide an explicit IP formulation of the WDP. In section VIII we explain how we generate a data set to assess the performances of MUCRAtR, and we provide an analysis of the empirical results. Next, in section IX we situate our work within the state-of-the-art. Finally, in section X we provide some concluding remarks and ideas for future developments.

All along the paper we make use of several abbreviations and symbols. In order to help the reader we summarize them in table II.

II. MATHEMATICAL BACKGROUND

In this section we introduce and carefully describe the Petri Nets formalism. Petri Nets are a powerful mathematical and graphical tool for the description of discrete distributed systems. Petri Nets (PNs) were firstly introduced in 1962 by Karl Adam Petri in his seminal dissertation ([53] in English and [53] in German). In particular PNs are suitable for describing systems in which parallelism, concurrency, and

[4]We recall that a MUCRA is simply a combinatorial reverse auction in which multiple copies of each item are auctioned [57]

Abbreviation	Meaning
WDP	Winner Determination Problem
t-relationships	Transformation Relationships
RFQ	Request For Quotes
CA	Combinatorial Auctions
MUCA	Multiunit Combinatorial Auctions
MUCRA	Multiunit Combinatorial Reverse Auctions
MUCRAtR	Multiunit Combinatorial Reverse Auctions with Transformability Relationships
CMWOSP	Constrained Maximum Weighted Occurrence Sequence Problem
PTN	Place Transition Nets
WPTN	Weighted Place Transition Nets
PTNS	Place Transition Net Structure
TNS	Transformability Network Structure
P	Set of Places
T	Set of Transitions
A	Set of Arcs
E	Arc Expression Function
C	Cost function for WPTN
$\mathcal{M}_0$	Initial Marking
J	Firing Occurrence Sequence
$\mathcal{K}_J$	Firing Count Vector
A	Incidence Matrix
u_k	firing vector
PTN_I	PTN representing internal manufacturing operations
PTN_E	PTN representing internal manufacturing operations plus received offers

TABLE II
SUMMARY OF NOTATION EMPLOYED IN THE PAPER.

synchronization play an important role. For a very good review on Petri nets, refer to [48].

Petri Nets can provide some distinctive advantages with respect to other approaches [55] like finite state machines. For instance, they can capture causal dependencies or independence among the different components of the system can be explicitly represented; they also allow to describe a system that is not inherently sequential, different levels of abstraction without having to change the description language, and so on.

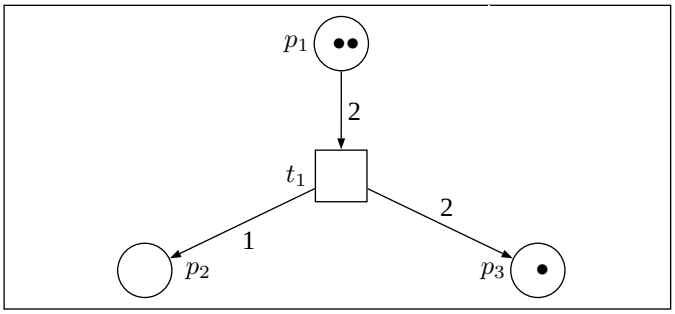


Fig. 2. Example of a Place Transition Net.

An example of Petri net is shown in figure 2. A PN is a bipartite graph: it has *place* nodes, *transition* nodes, and directed arcs connecting places to transitions and transitions to places. The places connected to a transition by means of input arcs are called the *input places* of the transition, and the ones connected by outgoing arcs from the transition are the *output places* of the transition. Places contain tokens. The graphical representation of a PTNS is composed of the following graphical elements: places are represented as circles, transitions are represented as rectangles, arcs connect places to transitions or transitions to places, and an *arc expression function* E labels arcs with values.

We will focus on a particular type of Petri net called Place Transition Net (PTN). Formally, following [48],

Definition II.1 (Place/Transition Net Structure)**.** A *Place/Transition Net Structure* (PTNS) is a tuple $N = (P, T, A, E)$ such that:

(1) P is a set of *places*;
(2) T is a finite set of *transitions* such that $P \cap T = \emptyset$;
(3) $A \subseteq (P \times T) \cup (T \times P)$ is a set of *arcs*;
(4) $E : A \to \mathbb{N}^+$ is an *arc expression* function (it represents the weights associated with the arcs, standing for the number of input/output tokens consumed/produced by the transition).

□

Furthermore, we have that $t^\bullet = \{p \in P \mid (t, p) \in A\}$ are the *output places* of t, and that ${}^\bullet t = \{p \in P \mid (p, t) \in A\}$ are the *input places* of t.

A distribution of tokens over the set of places is called a *marking*, and it stands for the state of the Petri net.

Definition II.2 (Marking)**.** A *marking* $\mathcal{M} : P \to \mathbb{N}$ of a PTNS is a multiset[5] over P. $\mathcal{M}(p) = k$ means that place $p \in P$ contains k tokens for marking $\mathcal{M}$.

□

A PTNS S with a given initial marking $\mathcal{M}_0$ is called a *Place/Transition Net* (PTN) and is noted $(S, \mathcal{M}_0)$.

Given a marking $\mathcal{M}$, we say that a transition is *enabled* if all its input places contain at least as many tokens as required by the the transition's input arcs. If the transition is enabled it can *fire* consuming tokens of the input places and producing

[5]A multiset is a set in which multiple appearances of the same element are allowed. The number of appearances of an element is called its multiplicity [9].

tokens in the output places. Intuitively, a transition is enabled if enough tokens are present in its input places. In what follows we state more formally the concepts of *enabled transition* and *firing of a transition*.

Definition II.3 (Enabled Transition)**.** Given a marking $\mathcal{M}$, a transition $t \in T$ is enabled iff:

$$\mathcal{M}(p) \geq E(p,t) \quad \forall p \in {}^\bullet t \tag{1}$$

□

An enabled transition may or may not fire. If it fires, it changes the current marking to a new marking by removing tokens from the input places and putting tokens into the output places. More formally

Definition II.4 (Firing of an enabled transition)**.** The *firing* of an enabled transition t removes $E(p_i,t)$ tokens from each input place p_i and adds $E(t,p_o)$ tokens to each output place p_o. The firing of a transition t changes marking $\mathcal{M}_{k-1}$ to a marking $\mathcal{M}_k$. The new marking can be computed employing the following equation[6]:

$$\mathcal{M}_k(p) = \mathcal{M}_{k-1}(p) + Z(t,p) \quad \forall p \in {}^\bullet t \cup t^\bullet \tag{2}$$

where $Z(t,p) = E(t,p) - E(p,t)$. In this case we write $\mathcal{M}_{k-1}[t > \mathcal{M}_k$ for denoting that the firing of transition t changes the $\mathcal{M}_{k-1}$ marking into the $\mathcal{M}_k$ marking.

□

A. Reachability

An important property we are interested in is whether we can reach a particular state of a PTN departing from a given initial state. This leads to the definition of *reachability*. In this section, we will introduce several concepts related to reachability. Intuitively, given an initial marking $\mathcal{M}_0$, and a final marking $\mathcal{M}_d$, the reachability problem consists in deciding if there exists a sequence of firings leading from $\mathcal{M}_0$ to $\mathcal{M}_d$.

The firing of an enabled transition changes the token distribution (marking) in a net according to the firing rule of definition II.4. Then, a sequence of firings will result in a sequence of markings.

Definition II.5 (Reachability)**.** A marking $\mathcal{M}_n$ is *reachable* from a marking $\mathcal{M}_0$ in a PTN structure S if there exists a sequence of firings that transforms $\mathcal{M}_0$ into $\mathcal{M}_n$. $\mathcal{M}_0$ is called the *start marking*, while $\mathcal{M}_n$ is called the *end marking*.

□

All the markings reachable from $\mathcal{M}_0$ in a PTN Structure S are noted as $R(S,\mathcal{M}_0)$, and are called the *reachable set* of a PTN.

Definition II.6. (Firing Sequence) Given a PTN structure S and a marking $\mathcal{M}_0$, a *firing* or *occurrence sequence* $J : \mathbb{N} \rightarrow T$ is a sequence of transitions:

$$J = \langle t_1,\ t_2, \ldots, t_n \rangle$$

[6]Henceforth, for simplicity, we implicitly assume that $E(p,t) = 0$ if $(p,t) \notin A$ and $E(t,p) = 0$ if $(t,p) \notin A$.

that changes the marking $\mathcal{M}_0$ into the marking $\mathcal{M}_n$. In this case we write $\mathcal{M}_0[J > \mathcal{M}_n$ as a shorthand to represent that the firing sequence J leads from $\mathcal{M}_0$ to $\mathcal{M}_n$.

□

Notice that in a *firing sequence* all the transitions must be enabled and fire with the order established by the very same sequence.

It can be shown that the start and end markings are related by the following equation:

$$\forall p \in P \qquad \mathcal{M}_n(p) = \mathcal{M}_0(p) + \sum_{t \in J} Z(t,p). \tag{3}$$

Definition II.7. The *firing count multi-set* associated with a *firing* or *occurrence sequence* J is a multiset $\mathcal{K}_J \in \mathbb{N}^T$ such that the multiplicity of each transition stands for the number of times it appears in the firing sequence. That is:

$$\mathcal{K}_J(t) = |J^{-1}(t)| \qquad \forall t \in T \tag{4}$$

where $|J^{-1}(t)|$ is the number of times transition t is fired in the firing sequence J.

□

B. The state equation

In this section we aim at providing an algebraic representation of Petri nets. Such representation will allow to compactly represent the reachability set in some cases.

For a Petri Net N with r transitions and n places, the *incidence matrix* $A = [a_{ij}]$ is an $r \times n$ matrix of integers. Each entry is given by $a_{ij} = a_{ij}^+ - a_{ij}^-$, where $a_{ij}^+ = E(t_i,p_j)$ stands for the weight of the arc connecting the t_i transition to its output place p_j, and $a_{ij}^- = E(p_j,t_i)$ stands for the weight of the incoming arc connecting place p_j to transition t_i.

It is straightforward that a_{ij}^+, a_{ij}^-, and a_{ij} represent the number of tokens added to, removed from, and changed in place j when transition i fires once.

Notice that in this new representation a transition t_i is enabled in a marking $\mathcal{M}$ iff

$$a_{ij}^- \leq \mathcal{M}(p_j) \quad j = 1,2,\ldots,n$$

In order to obtain an algebraic representation of a Petri net, we can represent a marking $\mathcal{M}_k$ as an $n \times 1$ column vector M_k such that the $j-th$ entry of M_k represents the number of tokens present in place p_j after the $k-th$ firing in some firing sequence ($M_k[j] = \mathcal{M}_k(p_j)$).

Finally, we define the *firing vector* u_k as an $r \times 1$ column vector of $r-1$ zeros and one nonzero entry. By setting a a 1 in the $i-th$ position ($u_k[i] = 1$), we indicate that transition t_i fires at the k-th firing. We can now express equation (2) in matrix form:

$$M_k = M_{k-1} + A^T u_k \quad k = 1,2,... \tag{5}$$

Say that $\mathcal{M}_n$ is reachable from $\mathcal{M}_0$ via the firing sequence $J = \langle t_1, t_2, \ldots, t_n \rangle$. We represent the transitions in J by

means of their firing vectors $\langle u_1, u_2, \ldots, u_n \rangle$. Then, by applying recursively equation (5), we obtain:

$$M_n = M_0 + \sum_{k=1}^{n} A^T \cdot u_k = M_0 + A^T \sum_{k=1}^{n} u_k = M_0 + A^T K_J \tag{6}$$

where K_J is an $r \times 1$ vector representing the firing count multiset $\mathcal{K}_J$, defined in equation (4), namely:

$$K_J[i] = \mathcal{K}_J(t_i) = |J^{-1}(t_i)| \qquad \forall i \in [1, r] \tag{7}$$

K_J is the *firing count vector* associated with the firing sequence J.

C. State equation and reachability

The following two results are taken from [48]. Say that $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$, then there exists a firing sequence $\langle u_1, u_2, ..., u_d \rangle$ bringing from $\mathcal{M}_0$ to $\mathcal{M}_d$. Therefore, a *necessary condition on reachability* can be expressed in terms of a matrix equation:

Theorem II.1. *If $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$, then the following equation has a non-negative integer solution $\mathbf{x}$:*

$$M_d = M_0 + A^T \mathbf{x} \tag{8}$$

where $\mathbf{x} = \sum_{k=1}^{d} u_k$ is the $r \times 1$ column vector of non-negative integers we called firing count vector.

□

Notice that the $i-th$ entry of vector $\mathbf{x}$ encodes the number of times a transition t_i must be fired to transform $\mathcal{M}_0$ into $\mathcal{M}_d$.

Equation (8) is called the *State Equation*, since it describes the states that a Petri net would reach if the transitions encoded in $\mathbf{x}$ were fired. However, notice that not all the states encoded by the state equation are actually reachable. That means that there may exist solutions to equation (8) that are not reachable states of a Petri net. However, it can be shown that sometimes all the states reachable by a Petri net are described by the state equation. In particular, this happens when the net is acyclic.

Before defining the concept of acyclicity, we have to explain what is a cycle. Since a Petri Net is a bipartite graph, a cycle in a Petri net is a sequence of

Definition II.8. A *directed cycle* in a Petri Net Structure (P, T, A, E) is a sequence of places and transitions $\langle p_1, t_1, p_2, t_2, \ldots, p_n, t_n, p_1 \rangle$ such that $\forall i \in [1, n]$ $(p_i, t_i) \in A$ and $(t_i, p_{i+1}) \in A$.

Definition II.9 (Acyclicity)**.** A PTNS is said to be acyclic if it does not contain any directed circuit.

□

In [48], it is shown that in an *acyclic* Petri Net, the condition expressed by theorem II.1 is not only necessary, but also sufficient.

Theorem II.2. *In an acyclic PTNS, $\mathcal{M}_d$ is reachable from $\mathcal{M}_0$ iff the following equation has a non-negative integer solution in $\mathbf{x}$:*

$$\mathcal{M}_d = \mathcal{M}_0 + A^T \mathbf{x} \tag{9}$$

□

That is, if there exists a solution to equation (9), a firing sequence reaching $\mathcal{M}_d$ from $\mathcal{M}_0$ is guaranteed to exist, and $\mathbf{x}$ represents its firing count vector.

Moreover, Murata [48] further extends the class of Petri nets for which the condition is still sufficient. These particular nets (trap-circuit and syphon-circuit nets) have special topologies with particular types of circuits. For such nets, the state equation represents all the reachable states if the initial marking $\mathcal{M}_0$ satisfies some constraints. Further efforts have been made for extending the validity of the state equation to more classes of Petri nets [61].

III. A first attempt: Place/Transition Nets

PTNs (see section II) are a very powerful tool to describe discrete dynamical systems, like for instance operating systems, work-flows, finite state machines, parallel activities, data-flow computation, producers-consumers systems with priority, and so on. The firing of a transition in PTNs represents a state change in a discrete system. Such a state change can only take place if some preconditions occur (i.e. the transition must be enabled). For instance, if we model manufacturing operations by means of transitions in a PTN, the execution of a manufacturing operation changes the state of the production system: some goods are consumed, while other goods are produced, whenever enough input goods are available.

In this section we try to model the problem of *Grandma & co* by means of PTNs. In section III-A we model via PTNs the internal production structure of an auctioneer, and in section III-B, we complement such PTN model by incorporating the offers received by the auctioneer.

Before getting into the detail, we recall the details about the example introduced in section I.

Example III.1. The data characterizing the *Grandma & co*'s decision problem are:

(1) The cost of its internal manufacturing operations:
   a) A *Make Dough* operation costs €5 each time it is carried out. It requires one unit of *butter*, three units of *sugar*, and two units of *flour* as inputs; and it produces two units of *dough* as output.
   b) A *Make Filling* operation costs €6 each time it is carried out It requires one unit of *flour*, eight units of *apple*, and two units of *margarine* as inputs; and it produces two units of *filling* as output.
   c) A *Baking* operation costs €14 each time it is carried out. It requires four units of *dough* and four units of *filling* as inputs; and it produces four units of *apple pie* as output.
(2) A sale forecast of 200 apple pies. This represents the final requirements of *Grandma & co*.
(3) A stock of one hundred units of *flour* and two hundreds units of *sugar*.

□

Then, if *Grandma & co* intends to run a combinatorial auction and to invite all its providers, it must be able to

- send them a request for quotes (RFQ) containing the number of required units for each good; and
- once received all bids, it must be able to determine which bids to accept and which internal manufacturing operations to perform in order to obtain the 200 apple pies.

However, some problems prevent the use of CAs. Firstly, it is not possible to a priori establish how many units of each good the auctioneer (*Grandma & co*) requires. In fact, this depends on the production plan, that can only be decided upon receiving the offers. Secondly, once received all bids, *Grandma & co* needs a winning rule for the optimal, efficient and automatic selection of the best set of bids and in-house operations.

A. Modelling the internal production structure

In this section we model an auctioneer internal production structure by means of PTN. Consider the following example.

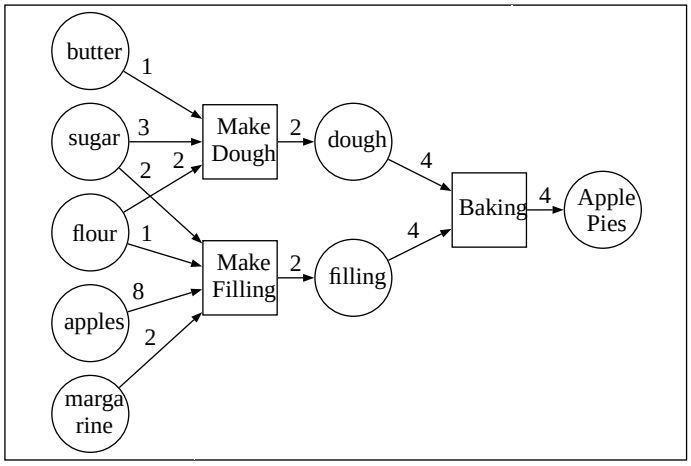


Fig. 3. PTNS associated to example III.1.

Example III.2. In figure 3, we associate a Place/Transition Net Structure (PTNS[7]) to the internal production structure of *Grandma & co*, characterized in example III.1. In doing this we associate *places* (P) to goods, *transitions* (T) to manufacturing operations, and input/output arcs (A) and their weights (E) to the quantity of goods consumed/produced by each manufacturing operation. Formally,

- The set of places is

$$P = \{butter, sugar, flour, apples, ...\}$$

- The set of transitions is

$$T = \{makedough, makefilling, baking\}$$

- The set of arcs is

$$A = \{(butter, makedough), (sugar, makedough), ...\}$$

.

[7]Refer to definition II.1.

- The arc weight function E is:

$$\begin{aligned} E(butter, makedough) &= 1 \\ E(sugar, makedough) &= 3 \\ E(flour, makedough) &= 2 \\ &..... \end{aligned}$$

Then, with this tool at hand, we can quantitatively represent the input resources needed and consumed by each manufacturing operation, the output resources produced, and the producer consumer relationships among the manufacturing operations.

We recall that a PTN is a PTNS with associated an initial marking $\mathcal{M}_0$ (see section II). The initial marking in a PTN usually represents the initial state of a discrete dynamic system. In the case of *Grandma & co* we can provide a similar semantics. The following example clarifies this point.

Example III.3. The initial marking $\mathcal{M}_0$ stands for the initial stock at *Grandma & co*. Indeed, the stock of a firm represents the "initial state" of its supply chain. The initial stock at *Grandma & co* is two hundreds units of sugar and a hundred units of flour (see example III.1). The multiset representation of the initial state would be:

$$\mathcal{M}_0 = 200'sugar + 100'flour$$

Thus, in figure 4, we graphically depict the initial marking of the PTN by means of numbers within places (circles). We call the resulting PTN PTN_I (I stands for Internal).

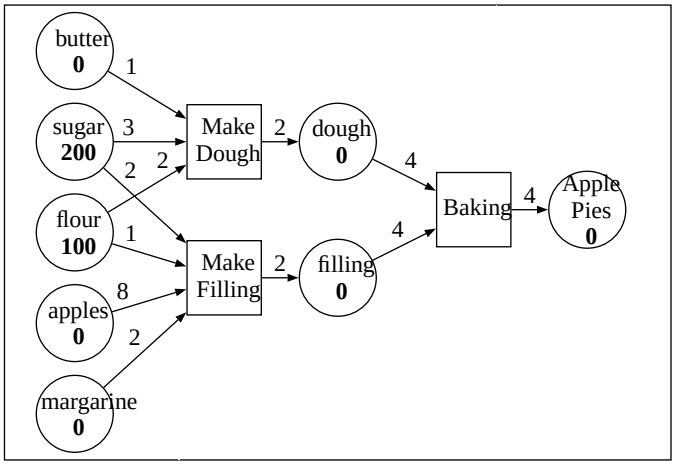


Fig. 4. PTN_I associated to example III.1.

Recall from section II that a transition in a PTN is enabled only if its input places contain enough tokens. For instance, in figure 4, transition *Make Dough* is *enabled* only if *at least* one unit of *butter*, three units of *sugar*, and two units of *flour* are within its input places. This is exactly what we require for a manufacturing operation to be *enabled*: it can not be performed unless the required goods are available. Moreover, looking at the *Baking* operation in figure 4, we observe that the producer/consumer relationships between *Make Dough* and *Baking* on one side, and between *Make Filling* and *Baking* on the other side, is quantitatively described by the PTN. Notice that the enabling condition guarantees that

a producer/consumer relationship is not only quantitatively represented, but also it is constrained to be implemented in its dynamics.

If a transition is enabled in a marking it can *fire* (see definition II.4). If a transition fires it consumes some input goods and produces some output goods. Once more, this is the semantics we require for a manufacturing operation: a manufacturing operation consumes a set of input resources and produces a set of output resources.

Example III.4. In table III we show what happens when the *Make Dough* transition fires. In the left image *Make Dough* is enabled. The execution of *Make Dough* provides some inputs to the *Baking* operation, as shown in the image on the right, thus perfectly describing the producer/consumer relationship among them.

□

What does it happen when there is a sequence of firings? As explained in section II-A, the PTN will pass through a succession of markings (states). In the case of *Grandma & co* a *marking* stands for the state of a production process, i.e. it describes the resources available at each state of the transformation process. In fact, it associates to each state the number of units of each good available to the auctioneer in that state. Accordingly, a manufacturing operation can be performed in a given state only if enough tokens are available in its input places in that state. The firing of a transition adds tokens into its output places likewise a manufacturing operation produces new available resources to the auctioneer.

If *markings* describe the level of resources currently available to an auctioneer, they naturally apply to describe the requirements of an auctioneer as well. An auctioneer aims at *reaching* a marking that fulfils its requirements (at least two hundreds tokens in the *applepie* place). This helps linking an auctioneer's requirements to its internal production structure.

In section II-A, we illustrated the problem of reachability, i.e. the problem of reaching a given marking $\mathcal{M}_d$ departing from an initial marking $\mathcal{M}_0$. We explained that it is a well studied problem in the PTN literature. The reader can imagine that the auctioneer is dealing with a similar problem: reaching a *marking* that fulfils its needs.

Summarizing, by means of the PTN representation we partially fulfill requirements (1) and (2) in table I. However, we still need to express:

- the relationships between the internal manufacturing operations and the received offers (limitation (3) in table I); and
- the information about the cost associated to bids' selection and to manufacturing operations' carrying out (limitation (4) in table I).

Then, in the next section we incorporate the description of the received offers into PTN_I.

B. Incorporating Bids

In this section we cope with limitation (3) in table I. That is, to establish a relationship among an auctioneer's internal production structure, the goods at auction, and the received offers. This entails relating the PTN description of section III (PTN_I) with the bidders' offers and the goods at auction.

Firstly, notice that the relation between the auctioned goods and the manufacturing operations is already accounted by PTN_I. It quantitatively specifies the goods required and produced by each manufacturing operation. Hence, it only remains linking the received combinatorial offers to the PTN_I. In fact, the utility of PTN_I is very limited if an auctioneer cannot link it to the received bids. For instance, the PTN (production process) described in figure 4 cannot work: there are not enough tokens (goods) to fire (run) any of the transitions (manufacturing operations). The problem is that the auctioneer (*Grandma & co*) needs to *buy* goods to feed its production process. Buying goods is equivalent to injecting tokens into the corresponding places. For instance, if *Grandma & co* decides to accept a bid offering 100 units of *butter*, this will inject 100 units into the *butter* place and will correspondingly increment the marking of the PTN. The counterpart of this operation would be putting a 100 into the *butter* place of figure 4.

As a consequence, incorporating bids into the PTN is quite natural. Indeed, they can be easily modelled by means of transitions as well. If a bid is selected, it must increase the amount of some available resources. Correspondingly, a transition adds tokens into its output places when fired. However, two features distinguish bids from manufacturing operations. Firstly, bids do not consume any resource. Secondly, bids can be run only once (it is not possible to accept a bid twice in our semantics). Therefore, each bid will be represented by a special type of transition, whose single input place will not be a good, but a sort of controller. Such a controller, named *bid place*, will enforce that a transition representing a bid is selected at most once. We will call this type of transitions *bid transitions*. In contrast, we will call the transitions corresponding to manufacturing operations *operation transitions*, and the places representing goods *good places*. We make clear the process of bid incorporation by means of an example.

Example III.5. Say that *Grandma & co* receives the combinatorial offers in equations (10) to (14) below from bidders. We represent an offer sent by a provider as a multiset $\mathcal{B} \in \mathbb{N}^G$, where G is the set of goods (in our case represented by places in figure 4), along with a cost. The multiplicity associated to each element of the multiset stands for the number of offered units for the element.

$$\mathcal{B}_1 \rightarrow 100'butter + 200'margarine \quad \text{at } €200 \quad (10)$$
$$\mathcal{B}_2 \rightarrow 200'flours + 300'sugar \quad \text{at } €100 \quad (11)$$
$$\mathcal{B}_3 \rightarrow 800'apples \quad \text{at } €200 \quad (12)$$
$$\mathcal{B}_4 \rightarrow 200'dough + 200'filling \quad \text{at } €1300 \quad (13)$$
$$\mathcal{B}_5 \rightarrow 200'apple\ pies \quad \text{at } €2400 \quad (14)$$

For instance, $\mathcal{B}_4 \rightarrow 200'dough + 200'filling$ at €1300 stands for a combinatorial bid offering two hundred units of *dough and* two hundred units of *filling* at €1300.

In figure 5 we intuitively show how to incorporate bids in equations (10) to (14) into the PTN_I on figure 4. PTN_I is shadowed, whereas the incorporated bids are in dark black.

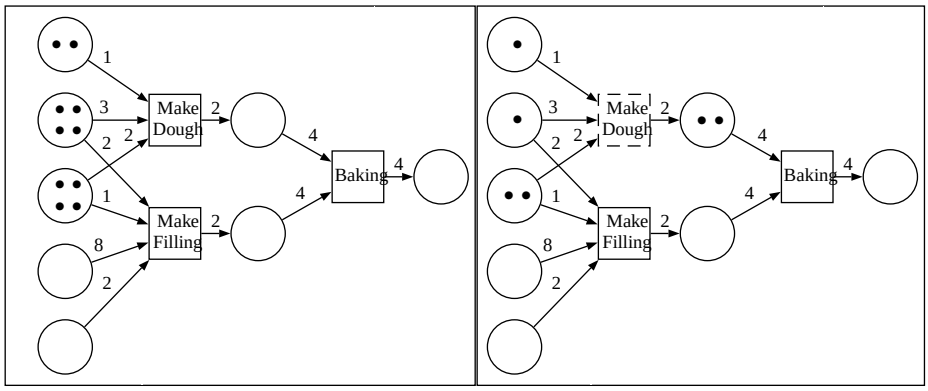


TABLE III
EXECUTION OF A MANUFACTURING OPERATION ON PTN_I.

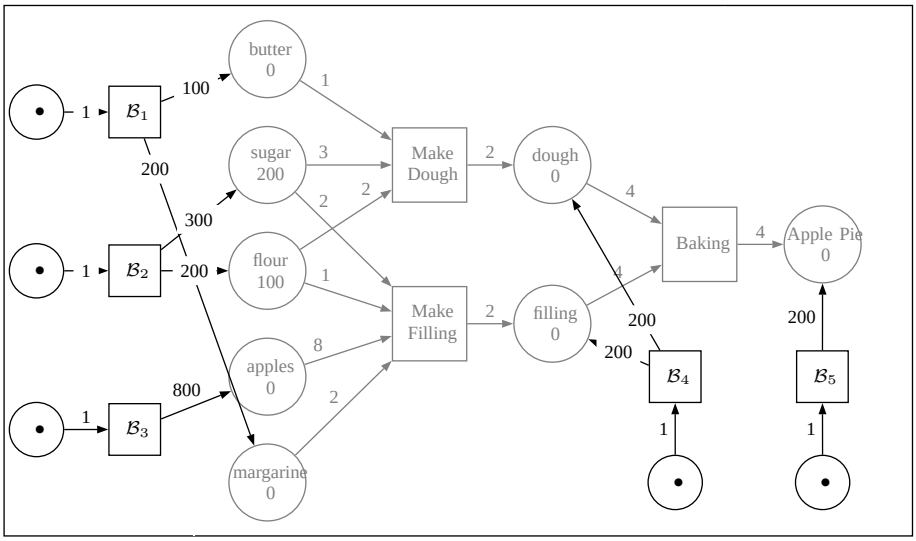


Fig. 5. PTN_E. Incorporating bids into the PTN_I of figure 4.

We will refer to the PTN in figure as the PTN_E (E from Extended). Notice that:

(1) The input places of *bid transitions* (transitions associated to bids and represented by $\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3, \mathcal{B}_4, \mathcal{B}_5$ in figure 5) only contain one token and their input arcs weigh one. Therefore, a bid transition can fire at most once.
(2) A *bid transition* does not have any other input place except from a *bid place*. Thus, it does not consume any resources.
(3) The output places of *bid transitions* are the goods offered in the corresponding bids, whereas the output arcs' weights are the number of offered units.Therefore, they increase the number of tokens present on the net if fired.

In table IV we graphically depict the evolution of the PTN in figure 4 when applying the firing sequence $J = \langle \mathcal{B}_1, makedough \rangle$. The upper picture shows the initial marking $\mathcal{M}_0 = 100'butter + 200'margarine$ (the stock at *Grandma & co*). The central picture shows the marking obtained after firing transition $\mathcal{B}_1$ (i.e. after accepting bid $\mathcal{B}_1$). Finally, the lower picture shows the marking obtained after firing $makedough$ (after performing the *Make Dough* operation). Notice that in both cases transitions $\mathcal{B}_1$ and *Make Dough* are enabled. Notice also that transition $\mathcal{B}_1$ cannot fire anymore, whereas *Make Dough* can.

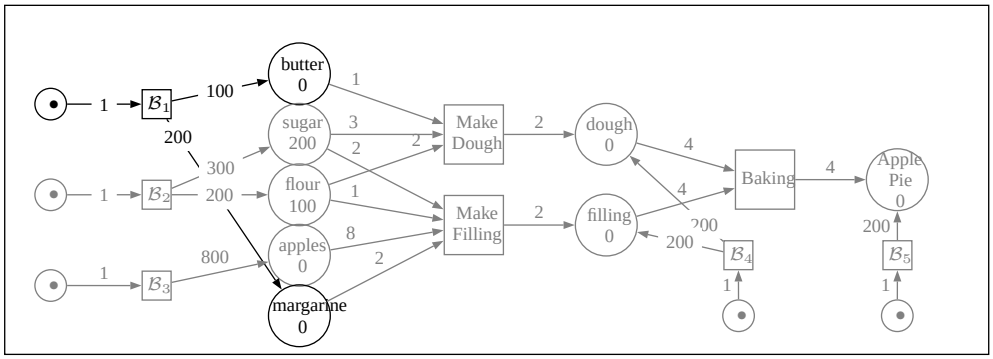
□

Summarizing, with the PTN in figure 5 *Grandma & co* can express:

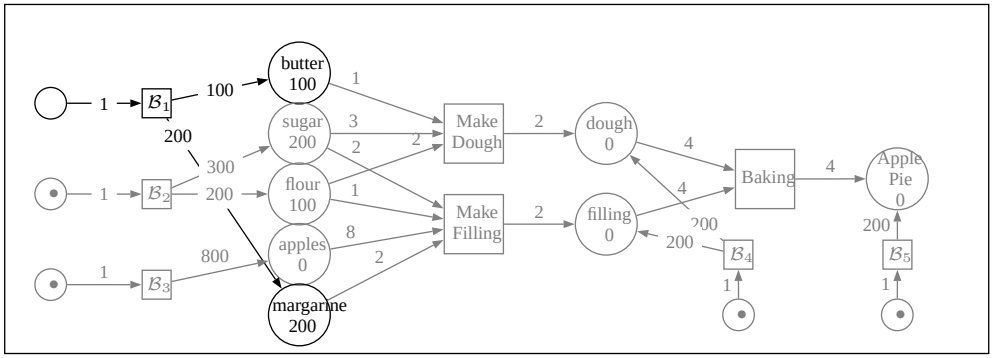
(1) its internal manufacturing operations along with the producer/consumer relationships among them (requirement (1) in table I);
(2) the relations among the auctioned goods, the received offers, and the manufacturing operations (requirement (2)); and
(3) its final requirements (requirement (3)).

Furthermore, it can obtain all the possible production states reachable by means of any legal combination of bids and internal operations. That is, it characterizes the combinatorial problem by providing a formalism to enumerate all the possible solutions. This can be achieved thanks to the dynamics of PTN (the firings). This is a crucial point: the PTN_E in figure 5 compactly represents all the possible decisions that *Grandma & co* can take.

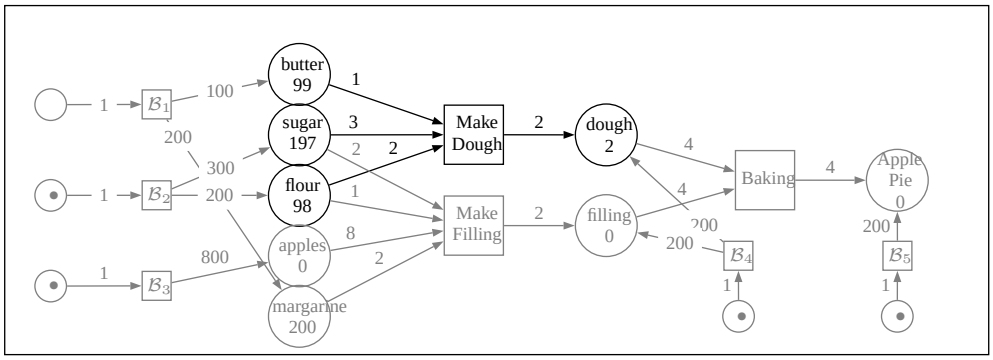
Unfortunately, *Grandma & co* is not interested in *simply*



(a) Initially.



(b) After selecting bid $\mathcal{B}_1$.



(c) After performing *Make Dough*.

TABLE IV
APPLYING THE FIRING SEQUENCE $J = \langle \mathcal{B}_1, makedough \rangle$.

reaching a state that fulfils its final requirements, it wants to *minimize* its costs as well. How can we quantify that performing manufacturing operations costs money? How can we quantify that buying goods costs money? It is under this point of view that PTNs lack of the necessary expressiveness and need to be extended. In the next section, we explain how to deal with such extension.

IV. WEIGHTED PLACE TRANSITION NETS

There is a feature of some discrete systems (in particular the one we consider) that, to the best of our knowledge, has never been considered so far in the PTN literature, and that we deem fundamental. A change in the state of a system may have an associated cost. For instance, in our case, a manufacturing operation has a cost associated to each time it is carried out. Thus, in order to model manufacturing operations, we need to extend Place Transition Nets to incorporate the notion of *transition cost*. Such extension will allow not only to represent the fact that a cost is associated to each transition firing, but also to easily compute the cost associated to a *firing sequence*.

One may think to associate a cost to a transition with alternative techniques. For instance, one may insert places that account for the quantity of money needed to perform an operation. Although operationally possible, this extension would constrain the applicability of our method:

- it would not allow expressing fractional quantities
- it would reduce the number of solutions by constraining the order of operation executions. In fact, it would enforce that money is actually present when the transformation takes place, whereas often the operation that injects money is the one at the end of the process.
- it would increase the complexity of the visualization by

including more places

Therefore, we present an extension, that we deem being the most natural. However, we do not exclude that one could find alternative solutions to the same problem.

The extension of PTN to incorporate the costs of operations and bids is quite natural and consistent with all the properties of PTN. If we aim at representing the fact that performing a manufacturing operation costs money, we simply have to associate a cost to the firing of an *operation transition*. Similarly, if we aim at representing that buying goods costs money, we have to associate a cost to the firing of each *bid transition*. In general, since both bids and manufacturing operations can be represented by means of PTNs, we have to associate a cost to each transition in a PTN.

A. WPTNSs and WPTNs

We extend the notion of Place Transition Net (see section II) by associating a *cost* to each transition. This leads us to the definition of *Weighted Place Transition Net Structure* (WPTNS) and *Weighted Place Transition Net* (WPTN).

Definition IV.1 (WPTNS)**.** A WPTNS is a a tuple (P, T, A, E, C) where:

- P, T, A, E are defined exactly like in a PTNS.
- $C : T \to \mathbb{R}$ is a cost function that associates a cost to each transition.

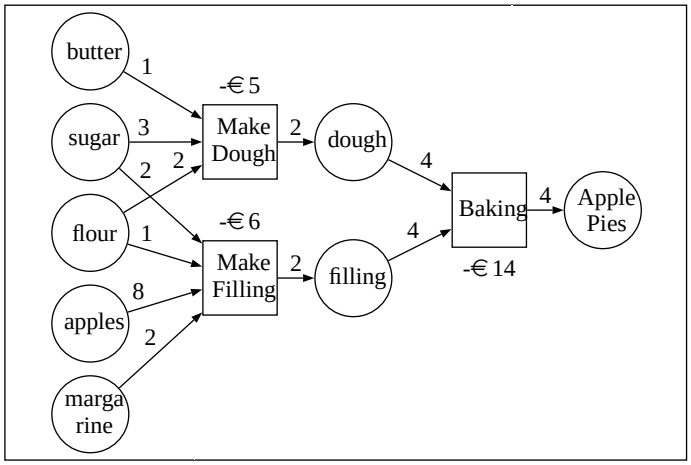


Fig. 6. WPTNS associated to example III.1.

Example IV.1. Let us associate a WPTNS to the internal production structure of *Grandma & co* specified in example III.1. At this aim we associate *places* (P), *transitions* (T), input/output arcs (A) and their weights (E) exactly like in example 3. The cost function (C) associates a cost to each manufacturing operation. A WPTNS employs the same graphical representation as a PTN (see section II), the only difference being that a cost labels each transition. We depict in figure 6 the resulting WPTNS, formally defined as in example 3 excepting the cost functions[8] C:

$$C(makedough) = -\text{€}\ 5$$
$$C(makefilling) = -\text{€}\ 6$$
$$C(baking) = -\text{€}\ 14$$

□

In figure 6, the values of C and the values of E label respectively transitions and arcs.

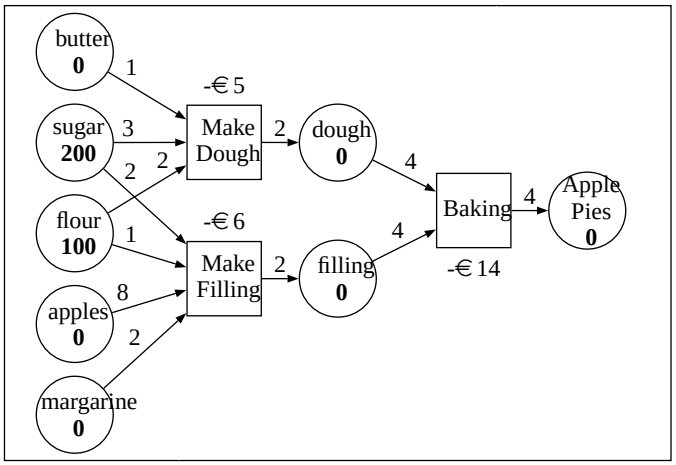


Fig. 7. WPTN associated to example III.1.

Analogously to a PTNS, we define a WPTN by associating to a WPTNS an initial marking $\mathcal{M}_0$.

Definition IV.2 (WPTN)**.** A WPTN is a pair $(N, \mathcal{M}_0)$, where N is s WPTNS, and $\mathcal{M}_0$is a multiset of places that stands for its initial marking.

The initial marking in a PTN represents the initial state of a discrete dynamic systems. The very same semantics is inherited by WPTNs.

Example IV.2. The initial marking $\mathcal{M}_0$ for the WPTNS in figure 6 *Grandma & co* is:

$$\mathcal{M}_0 = 200'sugar + 100'flour$$

In figure 7, we graphically depict the initial marking of the WPTNS in figure 6.

B. Dynamics of WPTNs

WPTNSs and WPTNs preserve all the properties of PTNSs and PTNs respectively, but allow the quantitative representation of the cost of a transition. Therefore, we can naturally extend to them all the concepts employed for PTNs. Those include the concepts of enabling of a transition, firing of a transition, marking, firing sequence, and so on (refer to section II).

In a PTN, if a transition is enabled in a marking it can *fire*. If a transition fires it consumes some input goods and produces some output goods. In a WPTN, something more happens. If

[8]The sign convention employed is negative values each time an auctioneer incurs in a cost.

a transition fires it carries out a cost, the cost associated to the fired transition.

Example IV.3. In table V we show what happens when the *Make Dough* transition fires. The transition generates a cost of €5. In the upper right corner we show the quantity of money spent by the auctioneer in the corresponding state.

□

What does it happen when there is a sequence of firings? Firstly, the WPTN will evolve through a succession of markings (states); and secondly, a cost will be associated to such a sequence of transitions (*firing sequence* in section II-A). Considering this, we can define the notion of *cost of a firing sequence* (C_{FS}) as:

Definition IV.3 (Cost of a firing sequence)**.** The cost C_{FS} associated to a firing sequence $J = \langle t_1, t_2, ..., t_d \rangle$ is the sum of all the costs of the transitions contained in the sequence:

$$C_{FS}(J) = \sum_{i=1}^{d} C(t_i) \tag{15}$$

If a transition fires more than once, say k times, then its cost will be added k times.

Example IV.4. In figure 8, analogously to figure 5, we incorporate into a WPTN the bids expressed in equations (10) to (14). Notice that the costs labelling *bid transitions* is the cost associated to the bids. Furthermore, in table VI, we repeat the firing sequence of table IV ($J = \{\mathcal{B}_1, makedough\}$) when a cost is associated to each transition. In this case, the cost associated to the firing sequence is $C_{FS}(J) = C(\mathcal{B}_1) + C(makedough)$ =-€200− €5 =- €205. In the upper right corner of each frame of table VI we highlight the cost associated to the corresponding firing.

V. Representing auction outcomes with WPTNs

In the previous section we introduced WPTNs and showed their powerful modelling features. The examples tried to give the intuitions behind the application of WPTN to our problem. In fact, we saw that the auctioneer faces a *make-or-buy* decision problem, and decides to solve it by means of combinatorial auctions. In this section, we aim at representing each of the outcomes of such auction given a description of the internal manufacturing operations, of the received bids, and of the auctioneer's final requirements . However, since an auctioneer is mostly interested in assessing the cost associated to each of such outcomes, we also associate an auctioneer's cost to each of the outcomes.

Then, firstly we introduce the *Transformability Network Structure* (TNS), a WPTN for modelling and communicating the internal manufacturing operations of an auctioneer. Secondly, we extend the TNS in order to incorporate the information regarding the received bids. This will result in the introduction of the *Auction Net*. This structure compactly expresses all the possible decisions an auctioneer may take, and quantifies the cost associated to each of such decisions. With those formal tools at hand, we can then define what a MUCRAtR is by providing an operational definition of valid auction outcome.

A. The Transformability Network Structure

In what follows we formally define the *Transformability Network Structure*. This corresponds to the net presented in figure 6. TNSs are useful for expressing the internal manufacturing operations of an auctioneer. This tool will have to quantitatively represent the input resources needed and consumed by each manufacturing operation, the output resources produced, the producer consumer relationships among the manufacturing operations, and the cost associated to each manufacturing operation. Summarizing, a TNS describes the different ways in which goods can be transformed and at which cost. More formally,

Definition V.1 (TNS)**.** A *transformability network structure* is a Weighted Place/Transition Net $N = (P, T, A, E, \mathcal{M}_0, C)$ such that we associate:

(1) the *places* in P to a set of goods G to negotiate upon[9].
(2) the *transitions* in T to a set of internal manufacturing operations;
(3) the *directed arcs* in A along with their weights E to the specification of the number of units of each good that are either consumed or produced by a manufacturing operation.
(4) the *initial marking* $\mathcal{M}_0$ to the quantity of each good initially available to the auctioneer (the stock). We indicate this particular initial marking with the multiset $\mathcal{U}_{in} \in \mathbb{N}^P$. Then, $\mathcal{M}_0 = \mathcal{U}_{in}$.
(5) a cost $C : T \to \mathbb{R}^+$ to each manufacturing operation.

In the next section we show how to incorporate the received bids into the TNS. The resulting WPTN is called *Auction Net*.

Example V.1. The WPTN introduced in example IV.2 is the TNS associated to the problem of *Grandma & co*, previously described in example III.1.

Notice that if an auctioneer communicates to bidders its TNS along with some constraints on the final marking (for instance, at least 200 tokens in the apple pie place), the bidders have all the information for composing meaningful offers. This completely fulfills the CAs expressiveness limitation in communicating to bidders an auctioneer's requirements (issue (2) in table I).

B. The Auction Net

In this section, we will thoroughly explain how to transform a TNS (figure 6) into an *Auction net* (figure 8). In the remaining of the chapter it is assumed that B is the set of received bids. Each bid is represented by a multiset $\mathcal{B} \in \mathbb{N}^P$ and has associated a cost encoded by the function $C_B : B \to \mathbb{R}^+ \cup \{0\}$.

Definition V.2 (Auction Net)**.** Given a set of bids B, and a TNS $N = (P, T, A, E, \mathcal{U}_{in}, C)$, an *Auction Net* is a WPTN

[9]Notice that a place represents a good. Thus, in what follows we will talk indifferently of *good places* and *goods*. That is, P and G are employed indifferently.

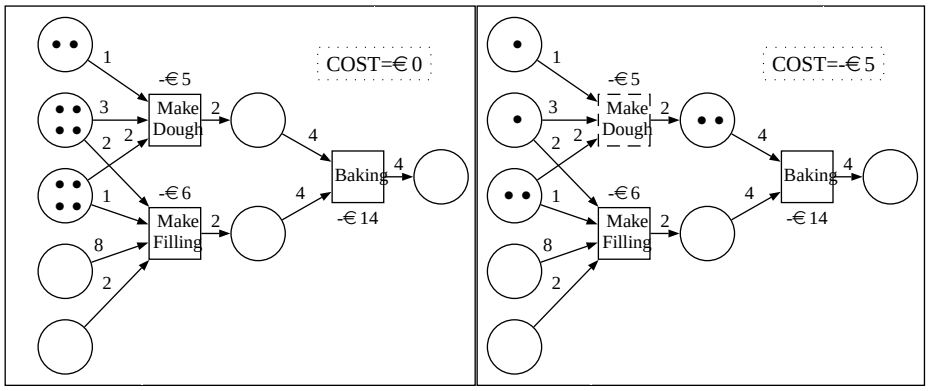


TABLE V
COST OF EXECUTING A MANUFACTURING OPERATION ON A WPTN.

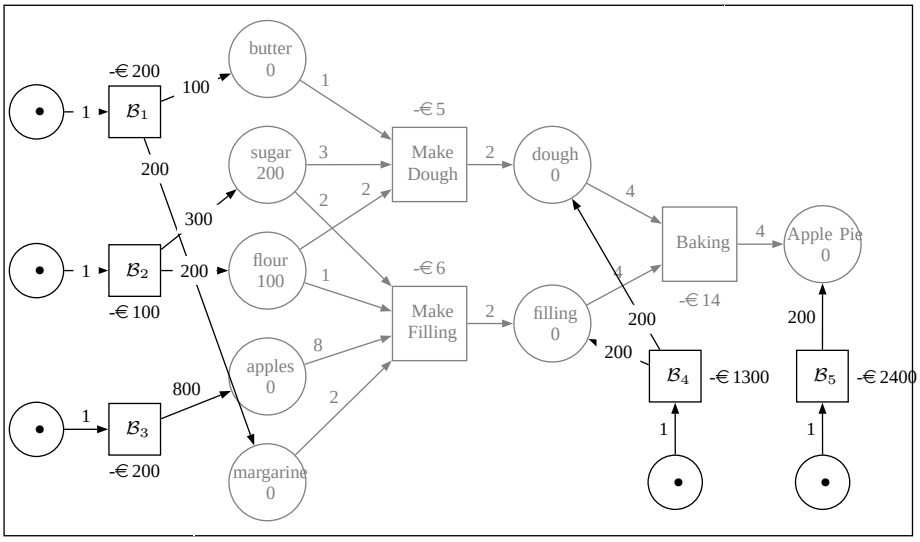


Fig. 8. Incorporating bids into the WPTN of figure 7.

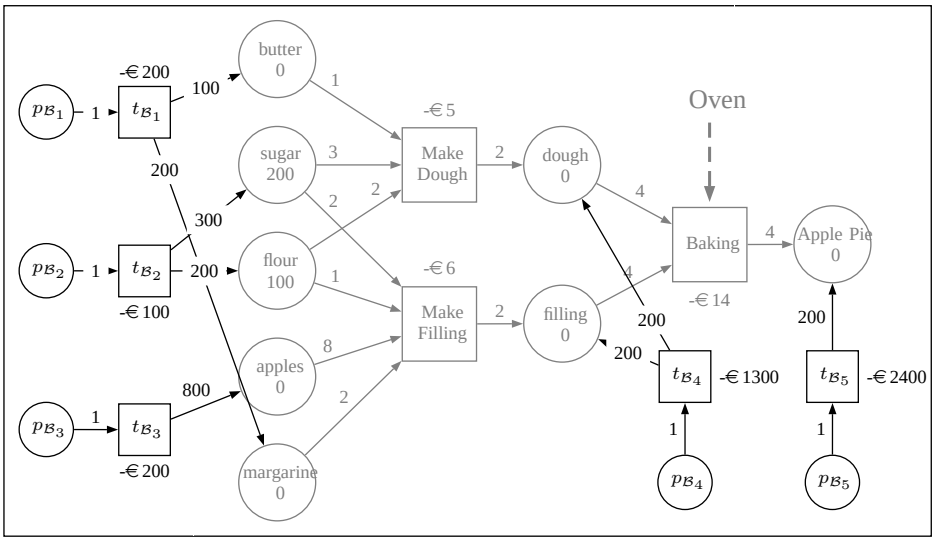
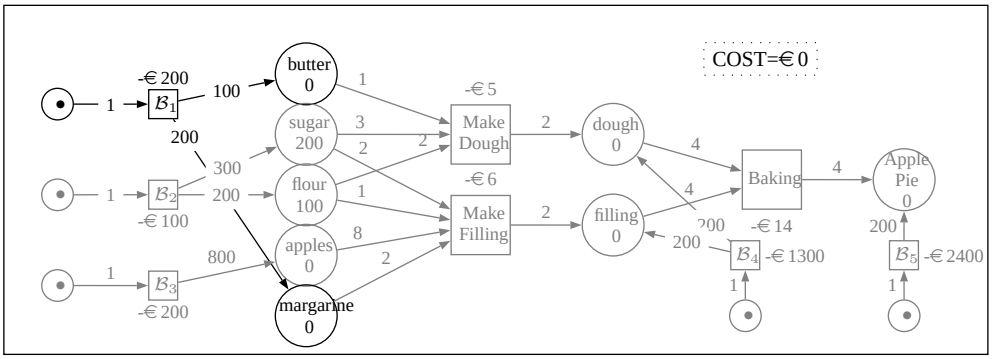


Fig. 9. Auction Net of the MUCRAtR in example III.1.

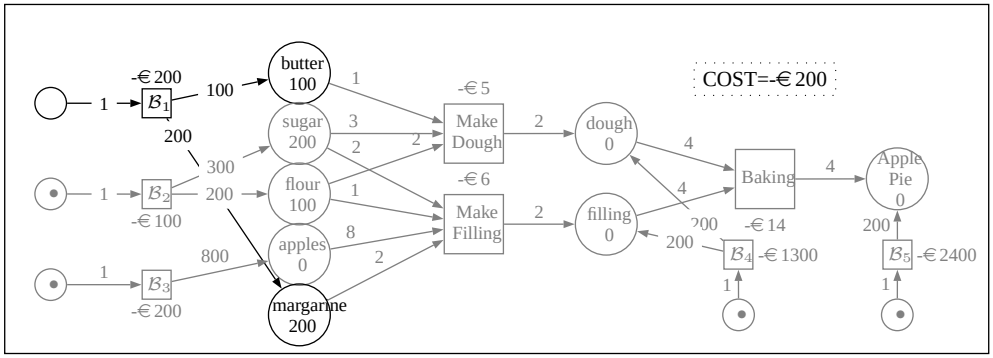
$S^* = (P^*, T^*, A^*, E^*, \mathcal{M}_0^*, C^*)$ where:

$$\begin{cases} P^* & = P \cup P_B \\ T^* & = T \cup T_B \\ A^* & = A \cup A_B \end{cases}$$

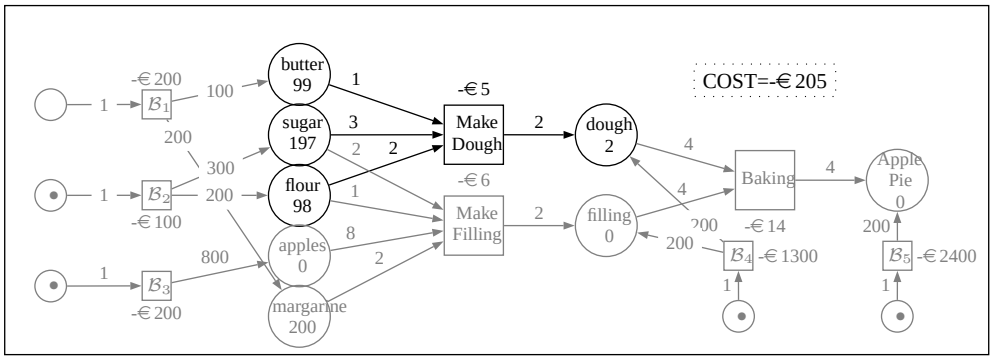
(1) P_B is the set of *bid places*. That is, for each bid $\mathcal{B} \in B$ add a place $p_{\mathcal{B}}$.
(2) T_B is the set of *bid transitions*. That is, for each bid



(a) Initially.



(b) After selecting bid $\mathcal{B}_1$.



(c) After performing *Make Dough*.

TABLE VI
APPLYING THE FIRING SEQUENCE $J = \langle \mathcal{B}_1, makedough \rangle$.

$\mathcal{B} \in B$ add a transition $t_{\mathcal{B}}$.

(3) A_B is the set of *bid arcs*. It is built as follows:

$$A_B = A_B^i \cup A_B^o$$

where

$$A_B^i = \{(p_{\mathcal{B}}, t_{\mathcal{B}}) \in P_B \times T_B \mid \forall \mathcal{B} \in B\} \quad (16)$$
$$A_B^o = \{(t_{\mathcal{B}}, p) \in T_B \times P \mid p \in \mathcal{B}\} \quad (17)$$

are the *input bid arcs* and *output bid arcs* respectively.

(4) The arc expression E^* function is built as follows:

$$E^*(x,y) = E(x,y) \qquad (x,y) \in A \quad (18)$$
$$E^*(t_{\mathcal{B}}, p) = \mathcal{B}(p) \qquad (t_{\mathcal{B}}, p) \in A_B^o \quad (19)$$
$$E^*(p_{\mathcal{B}}, t_{\mathcal{B}}) = 1 \qquad (p_{\mathcal{B}}, t_{\mathcal{B}}) \in A_B^i \quad (20)$$

(5) The cost function $C^* : T \cup T_B \to \mathbb{R}$ is built as follows:

$$C^*(t) = C(t) \qquad t \in T$$
$$C^*(t_{\mathcal{B}}) = C_B(\mathcal{B}) \qquad t_{\mathcal{B}} \in T_B$$

(6) The initial marking is defined as

$$\mathcal{M}_0^*(p) = \begin{cases} \mathcal{U}_{in}(p) & p \in P \\ 1 & p \in P_B \end{cases} \quad (21)$$

□

Example V.2. We extend the *TNS* of example IV.1 with the bids listed in equations (10) to (14). This gives raise to the *Auction Net* in figure 9. $(P, T, A, E, \mathcal{M}_0, C)$ have been defined in example IV.1. Then, $S^* = (P^*, T^*, A^*, E^*, \mathcal{M}_0^*, C^*)$ is defined as follows:

(1) $P^* = P \cup \{p_{\mathcal{B}_1}, p_{\mathcal{B}_2}, p_{\mathcal{B}_3}, p_{\mathcal{B}_4}, p_{\mathcal{B}_5}\}$

(2) $T^* = T \cup \{t_{\mathcal{B}_1}, t_{\mathcal{B}_2}, t_{\mathcal{B}_3}, t_{\mathcal{B}_4}, t_{\mathcal{B}_5}\}$
(3) $A^* = A \cup A_B^i \cup A_B^o$ where

$$A_B^i = \{(p_{\mathcal{B}_1}, t_{\mathcal{B}_1}), (p_{\mathcal{B}_2}, t_{\mathcal{B}_2}), (p_{\mathcal{B}_3}, t_{\mathcal{B}_3}), ...\}$$
$$A_B^o = \{(t_{\mathcal{B}_1}, butter), (t_{\mathcal{B}_1}, margarine), \ldots\}$$

(4) $E^*(x,y) = E(x,y)$ if $(x,y) \in A$. When $(x,y) \in A_B$ we have:

$$E^*(t_{\mathcal{B}_1}, butter) = 100 \quad E^*(t_{\mathcal{B}_1}, margarine) = 200$$
$$E^*(t_{\mathcal{B}_2}, sugar) = 300 \quad E^*(t_{\mathcal{B}_2}, flour) = 200$$
$$\ldots \qquad \ldots$$
$$E^*(p_{\mathcal{B}_1}, t_{\mathcal{B}_1}) = 1 \quad E^*(p_{\mathcal{B}_2}, t_{\mathcal{B}_2}) = 1$$
$$\ldots \qquad \ldots$$

(5) $C^*(t) = C(t)$ when $t \in T$. When $t \in T_B$ we have:

$$C^*(t_{\mathcal{B}_1}) = \text{-€}200 \quad C^*(t_{\mathcal{B}_2}) = \text{-€}100$$
$$C^*(t_{\mathcal{B}_3}) = \text{-€}200 \quad C^*(t_{\mathcal{B}_4}) = \text{-€}1300$$
$$C^*(t_{\mathcal{B}_5}) = \text{-€}2400$$

□

Recall that by means of the PTN defined in example III.5, an auctioneer was able to compactly represent all the possible outcomes associated to any of its decisions. However, he had the problem to assess the cost associated to each of such outcomes. Notice that by means of the auction net, the auctioneer can now express both the outcomes of its decisions and the cost associated to each of them.

In order to define the winner determination problem for MUCRAtR one further step is required. We have to define an optimization problem whose solution retrieves the optimal firing sequence to apply to the auction net in order to obtain a desired final marking (in the case of *Grandma & co* more than 200 tokens in the apple pie place). This is the purpose of the following section.

C. Constrained Maximum Weight Occurrence Sequence Problem

Since there is a cost associated to each transition, one may be interested in finding a maximum (minimum[10]) cost firing sequence leading from an initial marking to some final marking. More importantly, one may be interested in finding a maximum cost firing sequence leading from an initial marking $\mathcal{M}_0$ to a final marking $\mathcal{M}_d$ that *fulfils a set of inequality constraints*. For instance, we may want to impose that in a final marking $\mathcal{M}_d$ each place contains exactly one token ($\mathcal{M}_d(p) = 1, \forall p \in P$), or at least 200 tokens in a given place (for instance, the *Apple Pie* place in example III.1 $\mathcal{M}_d(applepie) \geq 200$). With this aim we define the *Constrained Maximum Weight Occurrence Sequence Problem* (CMWOSP).

Definition V.3 (CMWOSP)**.** Given a WPTN $N = (P, T, A, E, \mathcal{M}_0, C)$, a set of inequality/equality constraints that a final marking $\mathcal{M}_d$ must fulfil, expressed as:

$$\forall p \in P \quad \mathcal{M}_d(p) \Delta_p h_p \tag{22}$$

where $\Delta_p \in \{<, \leq, =, \geq, >\}$ and $h_p \in \mathbb{N} \cup \{0\}$, find an occurrence sequence $J_{opt} = \langle u_1, u_2, ..., u_d \rangle$ that brings the initial marking $\mathcal{M}_0$ to a final marking $\mathcal{M}_d$ such that: (1) $\mathcal{M}_d$ fulfils all the constraints in equation (22); and (2) J_{opt} maximizes the total cost C_{FS}.

We can express the inequations (22) in matrix form:

$$M_d \mathbf{\Delta h} \tag{23}$$

where M_d is a vector whose $i-th$ component represents the number of tokens in place i, $\mathbf{\Delta}$ is a vector whose $i-th$ element contains $\{<, >, \leq, \geq, =\}$, and $\mathbf{h}$ is a vector whose $i-th$ element contains h_p. We will call the constraints in equation (22) or (23) the *final marking constraints.*

Proposition V.1. *CMWOSP is at least EXPSPACE-hard.*

Proof: The reachability problem for PTN can be reduced to a CMWOSP. It has been proved that the reachability problem is EXPSPACE-hard [43]. ■

VI. The Winner Determination Problem

In this section, we formally define the *winner determination problem* for MUCRAtR.

Informally, given a TNS expressing the internal manufacturing operations of an auctioneer over a set of goods G, an auctioneer's final requirements $\mathcal{U}_{out} \in \mathbb{N}^G$, and a set of received bids B, the *winner determination problem* amounts to finding the set of bids and internal operations that minimize the auctioneer's cost and produce at least the required goods.

The formal definition of the WDP relies on the *Auction Net*.

Definition VI.1 (Winner Determination Problem)**.** Given an auction expressed as $\langle N, \mathcal{U}_{out}, B \rangle$, where $N = (P, T, A, E, \mathcal{M}_0)$ is a TNS, $\mathcal{U}_{out} \in \mathbb{N}^G$ expresses the auctioneer final requirements, and B is the set of received bids. Let $S^* = (P^*, T^*, A^*, E^*, \mathcal{M}_0^*, C^*)$ be the corresponding *Auction Net*. The *Winner Determination Problem* amounts to selecting the set of bids B^* and the sequence of internal operations J^* that both minimize the auctioneer's cost and satisfy the the following *final marking constraints* on the *Auction Net*:

$$\mathcal{M}_d(p) \geq \mathcal{U}_{out}(p) \qquad \forall p \in P \tag{24}$$
$$\mathcal{M}_d(p) \geq 0 \qquad \forall p \in P_B \tag{25}$$

Proposition VI.1. *The WDP for a MUCRAtR $\langle N, \mathcal{U}_{out}, B \rangle$ can be reduced to a CMWOSP on the corresponding auction net. Such a CMWOSP is characterized by the following* final marking constraints*:*

$$\mathcal{M}_d(p) \geq \mathcal{U}_{out}(p) \qquad \forall p \in P \tag{26}$$
$$\mathcal{M}_d(p) \geq 0 \qquad \forall p \in P_B \tag{27}$$

Proof: The proof is by construction:

(1) Solve the CMWOSP on the *Auction Net* N_B. We name the CMWOSP solution J^{min}.

[10]In any optimization problem maximizing and minimizing are two dual representations of the very same problem. We will talk about maximization in what follows, but all the results can be easily applied to a minimization.

(2) The set of winning bids B^* corresponds to the *bid transitions* contained in J^{min}:

$$B^* = \{\mathcal{B} \in B | t_{\mathcal{B}} \in J^{min}\} \tag{28}$$

(3) The sequence J^* of internal manufacturing operations that an auctioneer has to perform internally is obtained by removing from J^{min} all the transitions that are not *operation transitions*. We denote this as follows:

$$J^* = J^{min}_{|T} \tag{29}$$

■

Notice carefully that in a CMWOSP the sum of the weights associated to the overall transitions is maximized. However, since negative costs are associated to both bid transitions and operation transitions, maximizing the sum of the weights implies minimizing the auctioneer's costs.

Example VI.1. If *Grandma & co* receives the bids in equations (10) to (14), the decision minimizing its costs and allowing it to obtain the 200 apple pies is:

(1) to select bid $\mathcal{B}_4$ to obtain *dough* and *filling*; and
(2) to subsequently bake them at *Grandma & co* after running fifty times the *Baking* operation.

If we look at it on the WPTN, this corresponds to the firing sequence

$$J = \langle \mathcal{B}_4, \underbrace{Baking, Baking, Baking, \ldots, Baking}_{50 \text{ times}} \rangle \tag{30}$$

Then, the cost of this decision is assessed as follows:

$$cost(\mathcal{B}_4) + 50 \cdot cost(\textit{Baking}) = -€\,1300 - €\ 700 = -€\,2000. \tag{31}$$

The reader can check that this is the best possible option for the auctioneer: it exploits the initial stock, it brings to a marking that fulfils *Grandma & co* requirements, it minimizes the costs.

□

Finally, the optimization problem of the auctioneer is clearly stated, and there is a rule for selecting the winners. Thus, we have solved issue (4) in table I as well. Since we obtained this result by directly employing place transition nets, we can import all the techniques employed for them. As a first example, we show how to solve the winner determination problem by means of Integer Programming [33]. With this aim, we just show that some particular CMWOSP can be solved by means of Integer Programming.

VII. Solving the WDP by means of IP

In this section, firstly we show that the CMWOSP can be solved by means of Integer Programming under some special conditions. Then, we show that those conditions are fulfilled when the underlying PTN is acyclic. Finally, we explicitly state the IP solving the WDP.

A. Solving the CMWOSP by means of IP

In section II-B, we showed that under some hypothesis on a PTN, it is possible to express its overall reachability set by means of an equation, the state equation (see section II-C). The state equation describes all the states that an acyclic PTN can reach, and it is a linear equation. That is all we need to generate our integer program.

We recall also that, by means of the *state equation*, it is possible to represent in matrix form the firings and markings of a PTN (see section II-B):

- Let us associate to each place $p_i \in P$ a position i in a vector $M_k \in \mathbb{N}^{|P|}$. The integer contained in the $i-th$ position of the M_k vector corresponds to the number of tokens contained in a a place p_i after k firings in some sequence. Then, M_0 is the initial marking, M_1 is the marking obtained after the firing of some transition, and so on.
- Let us associate to each transition $t_j \in T$ a position j in a vector of integers $\mathbf{x} \in \mathbb{N}^{|T|}$. The integer contained in the $j-th$ position of $\mathbf{x}$ encodes the number of times transition t_j has been fired.

With this representation, the state equation can be written as:

$$M = M_0 + A^T \mathbf{x} \tag{32}$$

The very same formalism holds for WPTN. In fact, the only difference is that there is a cost associated to each transition. Then, can we represent in matrix form the cost of a sequence bringing from M_0 to M via the transitions encoded in $\mathbf{x}$ as well? The answer is quite easy. Notice that $\mathbf{x}$ in equation (32) stands for the number of times each transition is fired for transforming marking M_0 into marking M. Then, if we know the cost of each transition, according to definition in equation (15), we have to multiply the cost of each transition by the number of times it is fired. Then, we define a vector $C_T \in \mathbb{R}^{|T|}$ whose $j-th$ position represents the cost associated to transition t_j ($C_{FS}(t_j)$). Hence, the cost associated to the firing sequence represented by $\mathbf{x}$, noted as $J_{\mathbf{x}}$, is:

$$C_{FS}(J_{\mathbf{x}}) = \mathbf{x}^T C_T \tag{33}$$

The idea behind the mapping to IP is finding a set of linear equations that:

(1) constrains the decision variables associated to transitions to hold a value encoding a valid firing sequence;
(2) constrains the marking obtained by firing the selected transitions to fulfil a set of equality/inequality constraints; and
(3) maximizes the sum of the costs associated to the selected transitions.

Notice that point (1) can be easily fulfilled when the net is acyclic by means of the *state equation*. Since the *state equation* represents all the reachable states, it is enough to apply to it a set of inequality/equality constraints to fulfil point (2). Finally, since in a WPTN a cost is associated to each transition, maximizing the cost associated to the selected *firing sequence* we satisfy point (3) as well.

In what follows we go into the formal details of what we explained above. The following theorem states that if we can

represent all the reachable states of a PTN by means of the state equation, then the CMWOSP can be solved by means of IP.

Theorem VII.1. *Consider an WPTN* $(P, T, A, E, \mathcal{M}_0, C)$ *with incidence matrix*[11] $\mathbf{A}$. *If the state equation describes all the reachable states* M *of the WPTN, then all the non-negative integer solutions the following integer program:*

$$max \quad \mathbf{x}^T c_T \tag{34}$$

$$subject\ to \quad \mathcal{M}_0 + \mathbf{A}^T\mathbf{x}\ \mathbf{\Delta h} \tag{35}$$

represent the firing count vectors of all the optimal solutions to the CMWOSP defined by $\langle \sim, \mathbf{h} \rangle$

Proof: Notice that equation (35) simply imposes that the end marking fulfils the constraints defined by $\langle \sim, \mathbf{h} \rangle$ in equation (22). Equation (34) maximizes the cost $C_{FS}(J_{\mathbf{x}})$, associated to the firing sequence represented by $\mathbf{x}$ (see equation (33)). As a result, a solution $\mathbf{x}^*$ to the IP defined by equations (34) and (35) optimizes the sum of the costs associated to fired transitions, while ensuring that the final marking is reachable and fulfils the constraints defined by $\langle \sim, \mathbf{h} \rangle$. ■

According to the results stated in theorem II.2, it is possible to express the reachability set with the state equation when the PTN is acyclic. Then, we apply this result to our problem via the following corollary:

Corollary VII.1. *Provided that a WPTN is acyclic, every CMWOSP defined on it can be mapped into integer linear programming.*

Proof: . Since the WPTN is acyclic, in virtue of theorem II.2, all the reachable states M are the non-negative integer solutions of equation (9). Then, for theorem VII.1 the firing count vectors of all the solutions to the CMWOSP are the solutions to the IP in equations (34) and (35).

Hence, we solve the CMWOSP problem in two steps. First, we determine the optimal firing count vector $\mathbf{x}^{opt}$ by solving the Integer Linear Program (ILP) in equations (34) and (35). Then, we construct J_{opt} from $\mathbf{x}^{opt}$, for which each step is enabled. Since S is acyclic, we can establish a partial order among transitions so that $t_1 < t_2$ iff t_2 uses as input some output of t_1. We can construct an occurrence sequence J_{opt} by ordering the transitions in the firing count vector $\mathbf{x}_{J_{opt}}$ non-decreasingly according to our partial ordering. Every step in the so ordered occurrence sequence is guaranteed to be enabled. The occurrence sequence J_{opt} is consequently the solution to our CMWOSP. ■

Thus, we can also cope with requirement (5) in table I.

B. The IP Formulation in practise

We have shown that the CMWOSP can be solved by means of an ILP in the case that the underlying WPTN is acyclic in section VII-A. We showed in section VI that the winner determination problem for MUCRAtR is a CMWOSP. In this section we show that the WDP for MUCRAtR can be solved by means of IP when the auctioneer's TNS is acyclic. Furthermore we will explicitly write down the IP model.

[11]Refer to section II-C.

The first assumption is that no cycles are added when we extend a TNS into an Auction Net. This is very easy to show.

Proposition VII.1. *Given an acyclic TNS* $(P, T, A, E, \mathcal{M}_0, C)$, *the corresponding Auction Net* $(P \cup P_B, T \cup T_B, A \cup A_B, E_B, \mathcal{M}_0^{Bid}, C_{Bid})$ *will also be acyclic.*

Proof: Say that there is a *bid transition* $t_{\mathcal{B}}$ that includes a cycle that was not present in the TNS. The output places of bid transitions are always in P (see definition V.2). Then, in order to have a cycle, there should be a transition with input places in P that has the input place of $t_{\mathcal{B}}$ as an output place. However, this is impossible since, according to definition V.2, the input places of *bid transitions* have only output arcs. ■

Naturally, it follows that:

Corollary VII.2. *When the TNS is acyclic, the WDP can be solved by means of IP.*

Next, we explicitly express the IP model solving the *make-or-buy* decision problem, or equivalently solving the WDP for MUCRAtR.

The mathematical model is built according to the following rules:

(1) there are n goods, indexed with $i \in \{1, 2, \ldots, n\}$
(2) there are m internal manufacturing operations, indexed with $j \in \{1, 2, \ldots, m\}$
(3) there are l multi-unit combinatorial bids, indexed with $k \in \{1, 2, \ldots, l\}$
(4) a_{ij} is the difference between the weight of the arc connecting operation transition j to good i and the weight of the arc connecting good i to operation transition j in the auction net. Formally, in the WPTN language $a_{ij} = E(j, i) - E(i, j)$. Informally, this represents the flow of tokens in place i when transition j is fired.
(5) u_i^{in} is the quantity of good i initially available to the auctioneer (the stock).
(6) u_i^{out} is the quantity of good i finally required by the auctioneer (the sale forecast).
(7) b_{ki} is the weight of the arc connecting bid transition k to good i.
(8) bp_k is the weight of the arc connecting the bid place p to the bid transition k.
(9) bu_k^{in} is the quantity of tokens initially available in bid place of bid k[12].
(10) c_j is the cost associated to internal operation j.
(11) p_k is the price associated to bid k.
(12) $y_k \in \mathbb{N} \cup \{0\}$ is an integer decision variable (for each bid $k \in \{1, 2, \ldots, l\}$) taking on value w if bid k has been selected w times[13].
(13) $x_j \in \mathbb{N} \cup \{0\}$ is an integer decision variable (for each transition $j \in \{1, 2, \ldots, m\}$) taking on value w if transformation j is fired w times in the optimal firing sequence.

[12]Notice that we know that this is always one. However, for the sake of generality we consider it as a parameter.

[13]Notice carefully that we know that this variable can take only value 0 or 1. Then, it is a binary decision variable. However, in order to be formal, we hypothesize that is an integer variable for the moment.

With this in mind, the IP model is expressed with the following equations:

$$\max \quad \sum_k y_k \cdot p_k + \sum_j x_j \cdot c_j \tag{36}$$

$$u_i^{in} + \sum_k y_k \cdot b_{ki} + \sum_j x_j \cdot a_{ij} \geq u_i^{out} \qquad \forall i \tag{37}$$

$$bu_k^{in} - y_k \cdot bp_k \geq 0 \qquad \forall i \tag{38}$$

Equation (36) minimizes (recall the the costs are negative) the sum of the costs associated to bids plus the costs associated to internal manufacturing operations. Equations (38) and (37) correspond to equation (22) of the CMWOSP. We split it into two equations since they implement different inequalities. This is made clear if compared with equations (21), (26), and (27). Indeed, equation (37) implements equation (26), whereas equation (38) implements equation (27).

If we observe equation (38), and recall that $bu_k^{in} = 1$ for all k (see equation 21), and that $bp_k = 1$ for all k (see equation 20), the equation becomes:

$$1 - y_k \geq 0 \qquad \forall k \tag{39}$$

Considering that y_k is an integer decision variable, it turns out clear that it becomes a binary decision variable $y_k \in \{0, 1\}$. Hence, the whole optimization problem in equations (36) to (38) can be rewritten under this hypothesis:

$$\max \quad \sum_k y_k \cdot p_k + \sum_j x_j \cdot c_j \tag{40}$$

$$u_i^{in} + \sum_k y_k \cdot b_{ki} + \sum_j x_j \cdot a_{ij} \geq u_i^{out} \qquad \forall i \tag{41}$$

This ILP can be readily implemented with the aid of an optimization library. The number of decision variables needed to encode this problem is $|T| + |B|$, where T is the set of internal supply chain operations and B is the set of received bids. The number of required constraints is $|G|$, where G is the set of goods.

C. Comparison with a traditional MUCRA IP solver

In what follows we compare the IP formulation of the MUCRAtR WDP with the IP formulation of a traditional Multi-unit Combinatorial Reverse Auction (MUCRA) WDP. In order to solve the WDP for a MUCRA, as formalized in [59], we exploit the equivalence to the multi-dimensional knapsack problem pointed out in [35]. Sandholm et al. in [59] show how MUCRA can be solved by means of IP. In this case the problem is stated by means of the following parameters and variables:

(1) there are n goods, indexed with $i = \{1, 2, \ldots, n\}$
(2) there are l multi-unit combinatorial bids, indexed with $k = \{1, 2, \ldots, l\}$
(3) u_i^{out} is the quantity of good i finally required by the auctioneer.
(4) b_{ki} is the quantity of good i offered in bid k.
(5) p_k is the price associated to bid k.
(6) $y_k \in \{0, 1\}$ is a binary decision variable (for each bid $k \in \{1, 2, \ldots, l\}$) taking on value 1 if bid k has been selected and 0 otherwise.

Then, the problem of selecting the best offers can be expressed with the following IP model:

$$\max \quad \sum_k y_k \cdot p_k \tag{42}$$

$$\sum_k y_k \cdot b_{ki} \geq u_i^{out} \qquad \forall i \tag{43}$$

In this case the number of decision variables is $|B|$, and the number of constraints is $|G|$. Then, our formulation of the WDP can be clearly regarded as an extension of the ILP we must solve for a MUCRA (as formalized above). In fact, the second component of expression 40 changes the overall cost as transformations are applied, whereas the second component of expression 41 makes sure that the units of the selected bids fulfil a buyer's requirements taking into account the units consumed and produced by transformations.

Observe the analogy between the IP in equations (40) and (41), and the IP in equations (42) and (43).

The first terms of both IP are equivalent. In the MUCRAtR IP we add the contributions due to the firing of transformations. It seems a trivial extension. However, notice carefully that we showed that this cannot be done for every possible class of nets.

D. Exploiting the power of WPTNs

In order to provide a demonstration of the graphical and modelling support provided by our mapping to WPTNs, we incorporate the possibility for an auctioneer to express set-up costs associated to operations and bounds on the number of times operations can be performed. That is, each time a manufacturing operation is implemented, there is a set of fixed costs that should be taken into account into the optimization problem: like machinery, specialized personnel, and so on. We show how to graphically incorporate such costs and bounds into the WPTNs. Once included such constraints into the WPTNs in a suitable way, in virtue of the theoretical results exposed in section VII, we have available a WDP solver that incorporates the new constraints. In figure 10, we show how to augment the auction net in figure 9 to incorporate set-up costs and bounds. The set-up cost must be subtracted from the auctioneer revenue if a transformation is employed at least once. In the case of figure 10 we associate a fixed cost of 2000 euros to the *Make Dough* operation and we enforce that it can be executed at most 200 times.

It is very easy to see why this particular topology ensures that both constraints are fulfilled. In fact, in order to run the *Make Dough* operation, we have first to make sure that there is at least one token in each of its input places. However, in order to add tokens to the place on the top of the *Make Dough* transition, we have to fire the t_{FC} transition. Once this is fired, the 2000€ are subtracted from the total revenue and the 200 tokens are added to the *Make Dough* input place. This ensures that the *Make Dough* transition can be fired at most 200 times.

VIII. Empirical Evaluation

The main purpose of our experiments is to empirically assess the benefits and shortcomings provided by the introduction of *t-relationships* among goods with respect to a

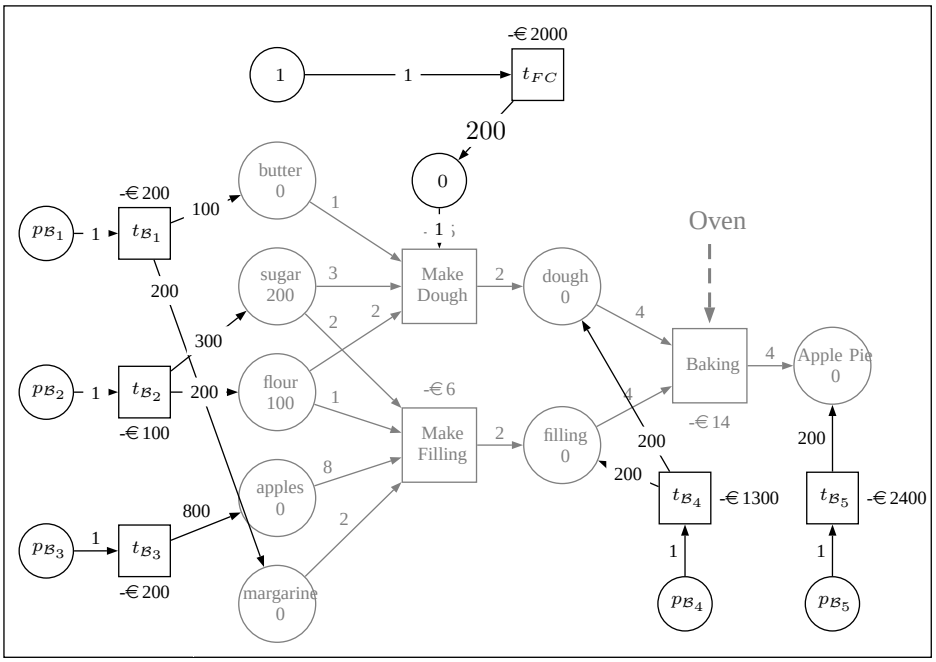


Fig. 10. Incorporating Fixed Costs.

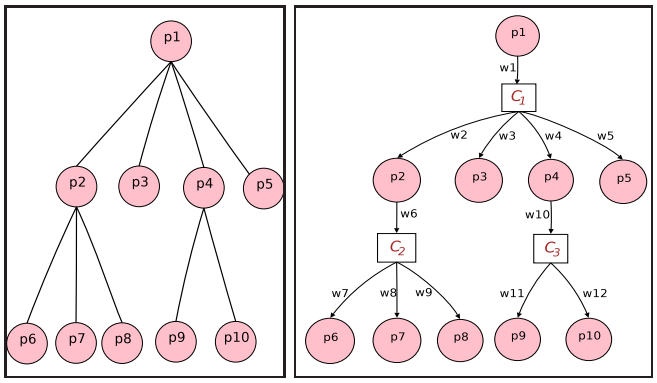
classical multi-unit combinatorial reverse auction. Our analysis is developed along two lines; (1) in terms of savings with respect to MUCRA; and (2) in terms of computational costs to assess the price to be paid for the savings. As to savings, on the one hand, we quantitatively assess the potential savings the auctioneer/buyer may obtain with respect to combinatorial reverse auctions that do not consider *t-relationships*; on the other hand, we empirically identify the market conditions under which it is worth for him to exploit *t-relationships*. Thus, we provide rules of thumb for an auctioneer/buyer to help him decide when to run a MUCRAtR instead of a MUCRA. As to computational cost, we identify the market conditions that most affect the computational cost of MUCRAtR, both in absolute terms and relatively to MUCRA.

Each auction problem is 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*.

A. Auction Problem Generation

To the best of our knowledge, and as already pointed out in [4], no real-world benchmarks of combinatorial bids do exist. Thus, with the purpose of comparing winner determination algorithms, we find two approaches in the CA literature to generate artificial data sets: (1) design a specific generator for the MUCA/MUCRA domain, as in [41]; or (2) given the equivalence of the WDP for an MUCA/MUCRA to the multi-dimensional knapsack problem (MDKP) [35], employ the very same data sets used for evaluating MDKP solvers (e.g. [17]). Unfortunately, we cannot benefit from any previous results in the literature since they do not take into account the novel notion of *t-relationship*, and thus the generated auction problems never reflect such relationships among goods.

In order to solve the WDP for a MUCRA, as formalized in [59], we exploit the equivalence to the multi-dimensional knapsack problem pointed out in [35]. In section VII-C we showed how MUCRA can be solved by means of IP. Each auction problem is composed of a TNS, an RFQ, and a set of combinatorial bids. The WDP for a MUCRA considers the last two components of the auction problem, whereas the WDP for a MUCRAtR considers them all.



(a) A tree (b) Corresponding TNS

Fig. 11. Extension of a tree to a TNS

1) TNS Generation: Firstly, we consider the creation of a TNS. As explained in section II-C, 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 auction problems. 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 11(b) depicts an example of a

TNS that satisfies such requirements. Our generator randomly constructs TNSs receiving as inputs: (1) the set of goods P; (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 $U[c_{min}, c_{max}]$).

Algorithm 1 BUILD-TNS$(P, r, w_{min}, w_{max}, c_{min}, c_{max})$

1: $\Sigma \subset P \times P$; {'*Father of*' relationship.}
2: **repeat**
3: $P_{added} \leftarrow \emptyset$; {Nodes already added to the tree.}
4: $P_{fathers} \leftarrow \emptyset$; {Nodes that have at least one child.}
5: $\Sigma \leftarrow \emptyset$;
6: pick randomly $p_\epsilon \in P$;
7: $P_{added} \leftarrow P_{added} \cup \{p_\epsilon\}$
8: **for each** $p_{son} \in P \setminus P_{added}$ **do**
9: **if** $|P_{fathers}| < r$ **then**
10: pick randomly $p_{father} \in P_{added}$;
11: **else**
12: pick randomly $p_{father} \in P_{fathers}$;
13: **end if**
14: $\Sigma = \Sigma \cup \{(p_{father}, p_{son})\}$;
15: $P_{added} \leftarrow P_{added} \cup \{p_{son}\}$;
16: $P_{fathers} \leftarrow P_{fathers} \cup \{p_{father}\}$;
17: **end for**
18: **until** $(|P_{fathers}| = r)$
19: $T \leftarrow \emptyset$;
20: $A \leftarrow \emptyset$;
21: $E : A \rightarrow \mathbb{R}_+$;
22: **for each** $(p_{father} \in P_{fathers})$ **do**
23: $T \leftarrow T \cup \{t\}$; {Create a new transition.}
24: $A = A \cup \{(p_{father}, t)\}$;
25: $E(p_{father}, t) \leftarrow$ GET-RANDOM-INTEGER(w_{min}, w_{max});
26: **for each** $(p_{father}, p_{son}) \in \Sigma$ **do**
27: $A \leftarrow A \cup \{(t, p_{son})\}$;
28: $E(t, p_{son}) \leftarrow$ GET-RANDOM-INTEGER(w_{min}, w_{max});
29: **end for**
30: **end for**
31: $C : T \rightarrow R_+ \cup \{0\}$;
32: **for each** $t \in T$ **do**
33: $C(t) \leftarrow$ GET-RANDOM-REAL(c_{min}, c_{max});
34: **end for**
35: **return** $\langle P, T, A, E, C \rangle$

Algorithm 1 constructs acyclic TNSs. It is composed of three sequential sub-algorithms. The first part (lines 1–18) creates a tree structure. The second part (lines 19–30) extends the tree to create a TNS by creating transformations and attaching weights to the edges connecting places with transformations (as illustrated in figure 11). The third part (lines 31–35) attaches a cost to each transformation. Since we aim at empirically assessing the potential savings when considering *t-relationships* independently of TNSs' shapes, distinguishing feature of our algorithm is that it is capable of constructing acyclic TNSs that may largely differ in their shapes (e.g. with different widths, depths, either symmetric or asymmetric, and so on).

Lines 1–18 of algorithm 1 constructs a tree of $|P|$ nodes, r of them having at least one child (i.e. a tree with r branching points). The $P_{fathers}$ set contains the nodes with at least one child. The tree is represented as a binary relationship Σ among goods, the *father of* relationship. Thus, $(p_1, p_2) \in \Sigma$ stands for p_1 *is the father of* p_2. Given such relationship, it is easy to build random trees. Departing from an empty set of added nodes (P_{added}): (1) choose randomly a root node p_ϵ and add it to P_{added} (lines 6–7); (2) assign a father to each node that has not already been added, (the father is chosen randomly within the already added nodes if the number of branching points is less than required, and within the nodes with at least a child otherwise (lines 8–17)). This constructive process first builds the root (p_ϵ), then assigns a child (p_2) to the root (p_ϵ), then assigns a child (p_3) to a random node within $\{p_\epsilon, p_2\}$, and so on. In order to construct a tree of exactly r branching points, the process iterates until such condition is satisfied.

Lines 19–30 of algorithm 1 use Σ to generate a TNS (as illustrated in figure 11) whose arc expressions are chosen in the integer interval $[w_{min}, w_{max}]$. Each father node is attached to a *t-relationship* as input good, while its children nodes are attached to the same *t-relationship* as output goods. Finally, the algorithm assigns to each arc in the created TNS a random integer in $[w_{min}, w_{max}]$ as arc expression.

Lines 31–35 attaches to each created *t-relationship* a transformation cost sampling from a uniform continuous distribution $U^*(c_{min}, c_{max})$.

The output of the algorithm is thus a TNS defined by: the set of *t-relationships* T, the set of arcs A, the arc weight function E, and the transformation cost C.

2) RFQ generation: Our aim is to fairly compare MUCRA with MUCRAtR. 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 bids which is fair for comparison purpose. As pointed out in section I, when a buyer expresses a set of *t-relationships* among a set of goods, he is also implicitly stating some intrinsic relationships that hold among goods. If we generate bids that do not take into account the relationships among goods enforced by *t-relationships*, the resulting bids are plausible for an auction type, but not for the other one. Consider the following example. A buyer requires 100 units of good p_2. Say that his *TNS* allows him to obtain 20 units of good p_2 after transforming 1 unit of p_1. Therefore, the buyer either needs 100 units of p_2, or 5 units of p_1 (to transform it into p_2). Thus, providers of good p_1 should offer 20 times less units than providers of p_2. An offer of 5 units of good p_1 would benefit a MUCRAtR but penalize a MUCRA. In fact this offer cannot be included in the winning set of a MUCRA since good p_1 is not even required by the buyer in his RFQ. Indeed, a MUCRAtR can include this offer in the winning set, since the 5 units of p_1 are transformed via the TNS. On the other hand, consider that providers only submit offers for good p_2. In this case a MUCRAtR can not exploit transformations, and the outcomes of MUCRA and MUCRAtR are identical, thus penalizing MUCRAtR. Therefore, we need to provide a fair enough bid set for both. As shown below the adequate generation of an RFQ can largely help in this task.

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 specified by the transformations in a TNS. Following the example above, a possible RFQ holding the quantity relationships among goods would require 5 units of p_1 *and* 100 units of p_2.

Next, we detail how to artificially generate an RFQ. Generating an RFQ amounts to setting for each place $p \in P$ a value

to $\mathcal{U}(p)$ as follows. First, we compute the number of required units of the root good: $\mathcal{U}(p_\epsilon) \simeq \mathbf{n}(units_{RFQ}, \sigma_{RFQ})$, where $\mathbf{n}(\mu, \sigma)$ is a normal distribution. 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 $\mathcal{U}(p)$ for each remaining good p of the TNS according to the following procedure. Let t be a transition such that p is one of its output goods, and p' is its single input good. More formally, consider $t \in T, \{p, p'\} \subseteq P$ such that $(t, p) \in A$ and $(p', t) \in A$[14]. Then, we obtain:

$$\mathcal{U}(p) \simeq \mathbf{n}(\frac{\mathcal{U}(p') \cdot E(t,p)}{E(p',t)}, \sigma_{RFQ}) \tag{44}$$

where $E(p', t)$ indicates the units of good p' that are input to transition t; and $E(t, p)$ indicates the number of units of good p that are output by transition t. In the current experiments, required quantities are modulated by a normal distribution with the purpose of introducing some randomness.

3) Bid Set Generation: Finally, we complete the artificial generation of an auction problem by generating a set of plausible bids. In order to generate a new bid $\mathcal{B}$ our generator firstly obtains a number of jointly offered goods from a binomial distribution $z \sim \mathbf{b}^*(p_{offered\_goods}, |P|)$ ([34]); and next it randomly selects a subset of goods $P' \subseteq P$ such that $|P'| = z$. For each $p \in P'$, the number of offered units is obtained from another binomial distribution $\mathcal{B}(p) \sim \mathbf{b}^*(p_{offered\_units}, \mathcal{U}(p))$. In both cases we employ binomial distributions because our aim is to maintain a proportionality relationships among: (1) the number of negotiated goods ($|P|$) and the density of bids ($|P'|$); and (2) the number of required units $\mathcal{U}(p)$, and the number of offered units $\mathcal{B}(p)$. In this way, when the number of required units increases, so does the number of offered units proportionally. Thus, we can analyze separately the effects of such parameters on savings, and avoid correlation effects. The binomial distribution allows to analyze, *ceteris paribus*, the effect of increasing the number of required units. Notice that we mark the binomial distributions with a star because we consider only non null samples drawn from such distributions.

4) Pricing policy: 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. We assume that all providing agents produce goods in a *similar* manner (they share similar TNSs). However, goods' prices and transformation costs slightly differ from provider to provider. In practice, our providing agents use the same TNS as the buyer, though each one has his own transformation costs, which in turn are assessed as a variation of the buyer's ones. Thus, for each bid 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 d_ε parameter, standing for the average unitary price of the root good (e.g. the root good in figure 11(b) is p_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, the unitary price for the root good is assessed as $\pi_{root,j} \simeq d_\varepsilon \cdot |\mathbf{n}(\mu_{root\_price}, \sigma_{root\_price})|$, where $\mathbf{n}$ is a normal distribution[15]. After that, our pricing algorithm recursively proceeds as follows. Given a bid $\mathcal{B} \in B$ and a good $p \in P$ 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 of p by the value labelling the arc connecting the input good to the transition, and adding a 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 11(b) and a provider P_j such that its unitary cost for p_2 is $\pi_{d_2,j} = 50€$, *his* transformation cost (different from the buyer's one) for t_2 is $10€$, and $w_6 = 1$. In such a case, the value to split down through t_2 towards p_6, p_7, and p_8 would be $50 \cdot 1 + 10 = 60€$. Say that p_7 is assigned 0.2 as share factor. Thus, $60 \cdot 0.2 = 12€$ would be allocated to p_7. Finally, that amount should be split further to obtain p_7 unitary price since $w_8 = 7$. Then, the final unitary price for p_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 submitting bid $\mathcal{B}$. Let $t \in T, \{p, p'\} \subseteq P$ such that $(t, p) \in A$ and $(p', t) \in A$. Then, we obtain $\pi_{p,\mathcal{B}}$, the unitary price for good p in bid $\mathcal{B}$ as follows:

$$\pi_{p,\mathcal{B}} \simeq \frac{\pi_{p',\mathcal{B}_k} \cdot E(p',t) + C(t) \cdot |\mathbf{n}(\mu_{pc}, \sigma_{pc})|}{E(t,p)} \, \omega_p \tag{45}$$

where $\pi_{p',\mathcal{B}_k}$ is the unitary price for good p' in bid $\mathcal{B}_k$ submitted by provider $P_k \neq P_j$; $E(p', t)$ stands for the units of good p' that are input to transition t; whereas $\mathbf{n}(\mu_{pc}, \sigma_{pc}) = \mathbf{n}(\mu_{production\_cost}, \sigma_{production\_cost})$ is a normal distribution that weighs transformation cost $C(t)$; $E(t, p)$ indicates the number of units of good p that are output by transformation t; and ω_p is the share factor for good p. Notice that after applying our pricing algorithm we obtain Π, an $|P| \times |B|$ matrix storing all unitary prices.

From equation (45) we can readily obtain the bid price for a bid $\mathcal{B} \in B$ as $d_\mathcal{B} = \sum_{p \in P} \mathcal{B}(p) \cdot \pi_{p,\mathcal{B}}$. Several remarks apply to equation (45). Firstly, for each *t-relationship* t, the share factors for output goods must satisfy $\sum_{(t,p) \in A} \omega_p = 1$. Secondly, it may surprise the reader to realize that the value to propagate down the TNS ($\pi_{p',\mathcal{B}_k}$) 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

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

[15]We consider the absolute value of the sample since the large tails of the normal distributions could occasionally bring about negative values.

regarded as noise that could eventually lead to diverting results.

After generating a complete auction problem, 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 buyer's *t-relationships* can be obtained. Therefore, the difference is that a MUCRAtR WDA does consider and exploit both the buyer's *t-relationships* along with the implicit transformation cost within each bid, while a MUCRA WDA does not.

To summaries, the parameters we must set to create an auction problem for a combinatorial auction with transformation are reported in table VIII-A4.

B. Experimental Settings

In what follows we describe the experimental design employed, the way we run and collected the data, as well as the hardware and software employed in the runs.

Recall that our aim is to determine the market conditions wherein: (1) MUCRAtR leads to savings when compared to MUCRA; and (2) MUCRAtR is computationally harder than MUCRA. At this aim, we introduce the following measures:

- *Savings Index* (SI). Difference in outcome cost between MUCRA and MUCRAtR, defined 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;
- *Cost Index* (CI). The ratio between the computational solving times of MUCRA (T^{MUCRA}) and MUCRAtR ($T^{MUCRAtR}$), defined as:
$$CI = \frac{T^{MUCRAtR}}{T^{MUCRA}}$$
.

With the aim of assessing the most sensitive parameters with respect to SI, CI, T^{MUCRA}, and $T^{MUCRAtR}$, we employ a fractional factorial experiment design [11]: we assign four possible values to each parameter, representing different grades in an increasing scale. Table VIII-A4 summaries 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 an auction problem as detailed in section VIII-A. Finally, we compute SI and CI after calculating the optimal solutions for the MUCRA and MUCRAtR WDPs.

Some remarks are in place as to how we assess SI and CI: (1) we set a time deadline (800 sec.) to solve each WDP: once such time elapses, the partial solution obtained until that moment is considered; (2) since we know[16] that $SI \geq 0$, we 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 to compare savings in this case.

[16]By definition $SI \geq 0$, since each solution of a MUCRA is also a solution to a MUCRAtR, and thus $C^{MUCRAtR} \leq C^{MUCRA}$.

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 MUCRAtR. As explained above, the total number of samples that have been considered are $5756 - 150 = 5606$. Among these samples, the optimizer could not find an optimal solution for MUCRAtR for 191 (3.407%) tests.

C. Experimental Results

The solvers for the MUCRATR and MUCRA WDP have been developed with the aid of ILOG's (www.ilog.com) CPLEX. The auction problems have been generated with the aid of MATLAB 7.0.4 (www.mathworks.com). We employed a Pentium IV with 3.1 Ghz and 1Gb RAM to run the experiments. In what follows we describe and discuss the results.

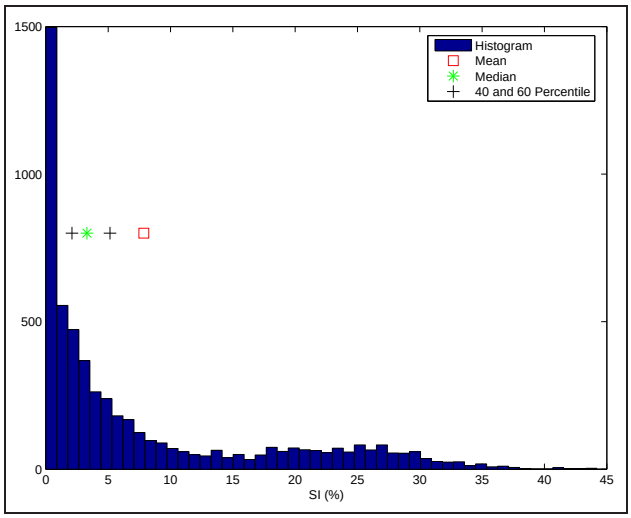


Fig. 12. Histogram of SI.

1) Analysis of Savings: **General Observations.** Figure 12 shows the histogram of SI. We observe that its shape is highly skewed. From figure 12 and from the statistics in the second column of table VIII we observe that the mean and variance are not appropriate to characterize the SI distribution. In fact, the high skewness makes the median a much better and stable descriptor. For all these reasons, we employ the *median* and *percentile* statistics in what follows. In figure 12 we observe that the savings of MUCRAtR 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.

Sensitivity Analysis. Notice that the distribution depicted in figure 12 is bimodal: a mode around 0 and another one around 25. Figure 13 clearly shows that the highest mode corresponds to providers offering the 80% of the required units in average ($p_{offered\_units} = 0.8$)[17]. The larger the percentage

[17]Recall that $p_{offered\_units}$ parameterises a binomial distribution describing the number of units offered per good per provider. In other words, it characterises a provider's capacity in the market.

Parameter	*Explanation*	*Values*			
n	Number of goods	8	10	12	14
r	Number of transitions	2	3	4	5
$units_{RFQ}$	Average units required for the root good	10	17	24	31
σ_{RFQ}	Variance of the required units for the root good	0.05	0.1	0.15	0.2
w_{min}, w_{max}	Minimum/Maximum arc weight	[2 , 5]			
c_{min}, c_{max}	Minimum/Maximum Transformation cost	10/10	100/100	20/40	40/80
m	Number of bids to generate	100	200	300	400
$p_{offered\_goods}$	Sets the number of goods jointly present in a bid	0.2	0.3	0.4	0.5
$p_{offered\_units}$	Sets the number of unit offered per good	0.2	0.3	0.4	0.5
d_ε	Average price of the *root* good	50	100	150	200
μ_{root_price}	Parameters of a Gaussian	1	1	1	1
σ_{root_price}	distribution weighing the *root* price d_ε	0.05	0.1	0.15	0.2
$\mu_{production\_cost}$	Gaussian distribution setting the production	0.6	0.9	1.2	1.5
$\sigma_{production\_cost}$	costs difference between buyer and providers	0.05	0.1	0.15	0.2

TABLE VII
PARAMETERS CHARACTERIZING OUR EXPERIMENTAL SCENARIO.

Statistic	SI	CI
$Mean$	7.8574	440.2153
$Variance$	93.1016	7.3912e+06
$Range(min - max)$	0 - 44.1959	0.004 - 5.6252e+04
$Interquartile\ Range$	11.06	43.7573
$Median$	3.2987	7.1792
$70 - percentile$	8.5971	30.5165

TABLE VIII
STATISTICS FOR SI AND CI.

of offered units, the larger the number of surplus goods that the MUCRA solution is likely to include (goods in a larger number than required); and hence, the more MUCRAtR can take advantage over MUCRA by exploiting *t-relationships* to transform surplus goods into requested goods. Hence, the following conclusion follows:

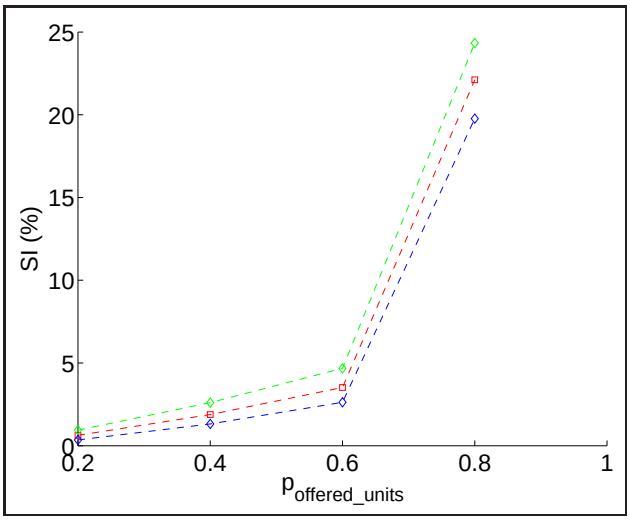


Fig. 13. Variation of SI for different providers' capacities.

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

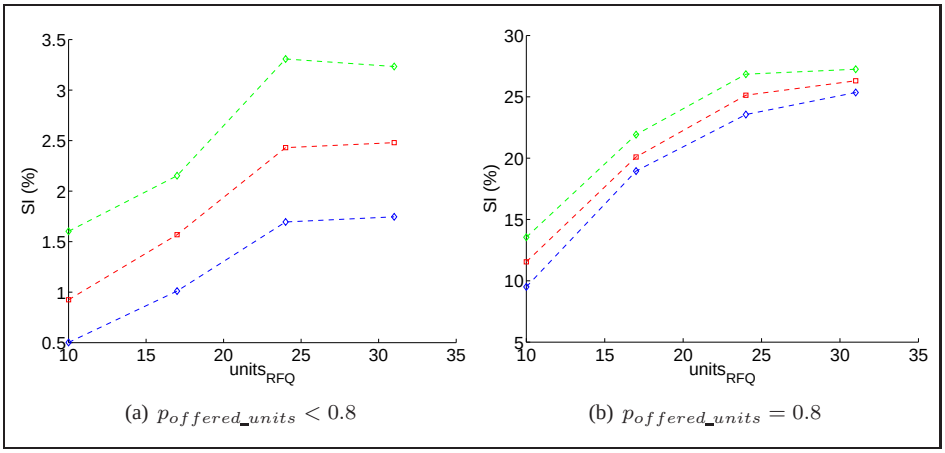
When analysing the behaviour of SI with respect to the remaining parameters, we will differentiate two cases: (1) $p_{offered\_units} < 0.8$; and (2) $p_{offered\_units} = 0.8$.

Figure 14 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 (41) we infer that the larger $units_{RFQ}$, the more fine-grained adjustments does a transformation allow because the number of output units of the transformation is smaller and smaller with respect to the number of required units. Also notice that in figures 14(a) and 14(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:

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

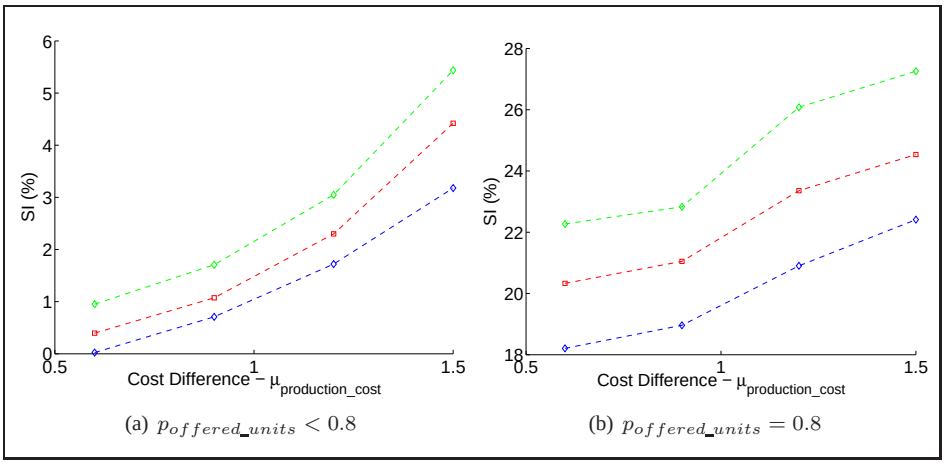
The third factor significantly affecting SI is the relationship between the transformation costs of a buyer and the providers' ones ($\mu_{production\_cost}$). If $\mu_{production\_cost}$ is larger than one, *on average* providers transform goods with a higher cost than the buyer, and the other way around when $\mu_{production\_cost} < 1$. Figure 15 depicts the results when varying $\mu_{production\_cost}$. Figures 15(a) and 15(b) show that SI increases as $\mu_{production\_cost}$ increases. This behaviour is explained as follows: the increment of $\mu_{production\_cost}$ models that in-house transformations are cheaper, therefore more likely to be exploited. A MUCRAtR 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 MUCRAtR largely differ among them. Furthermore, we also observe that: (1) the variation of SI in both figures is around 4%. Thus, this increment is independent from the $p_{offered\_units}$ parameter; (2) figure 15(b) is more irregular than figure 15(b), because of the wider spread of values when $p_{offered\_units} = 0.8$; and (3) although $\mu_{production\_cost}$ seemed intuitively to be the most sensitive parameter, our experiments show that it is not.

A buyer can only obtain better savings with a MUCRAtR with respect to a MUCRA if there exists some possibility to perform in-house transformations. Taken to the extreme, if



(a) $p_{offered\_units} < 0.8$ (b) $p_{offered\_units} = 0.8$

Fig. 14. SI with respect to the number of required units.



(a) $p_{offered\_units} < 0.8$ (b) $p_{offered\_units} = 0.8$

Fig. 15. SI with respect to providers' transformation cost.

no in-house transformation is convenient, the outcome of a MUCRAtR 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 16 depicts SI when changing the number of transformations within the TNS (r). As expected, we notice that the more the number of transformations, the more likely the increment in savings. And yet, figure 16(b) is more irregular than figure 16(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 17 shows the effect of varying the prices of goods among providers(σ_{root_price}). In other words, σ_{root_price} controls price spread in the market. As the difference in prices grows (larger σ_{root_price}), does also SI grow. A reasonable explanation could be that MUCRAtR 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 summarize, we can confirm, based on the observations above, that there are indeed market conditions (identified by **C1**, **C2**, **C3**, **C4**, and **C5**) wherein it is specially worth using MUCRAtR instead of MUCRA.

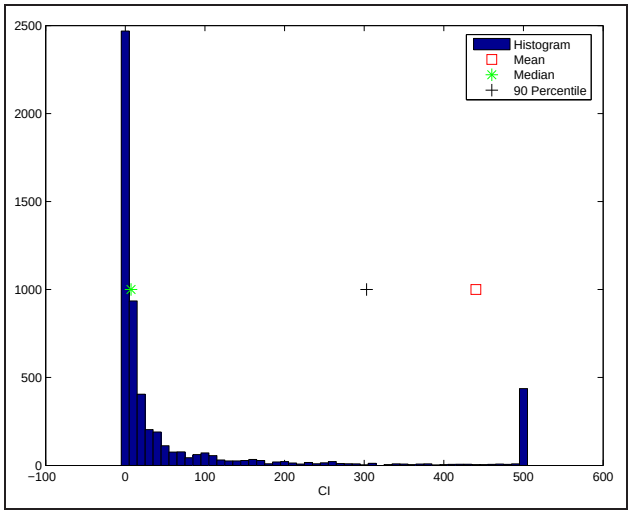
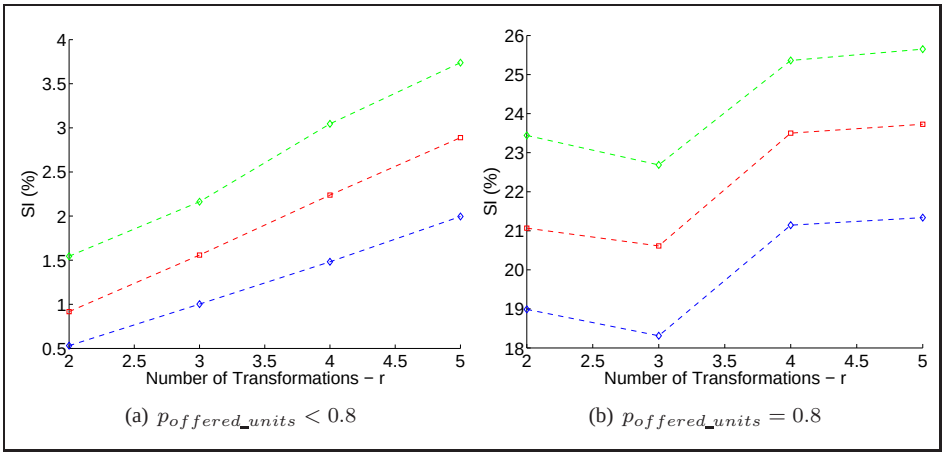
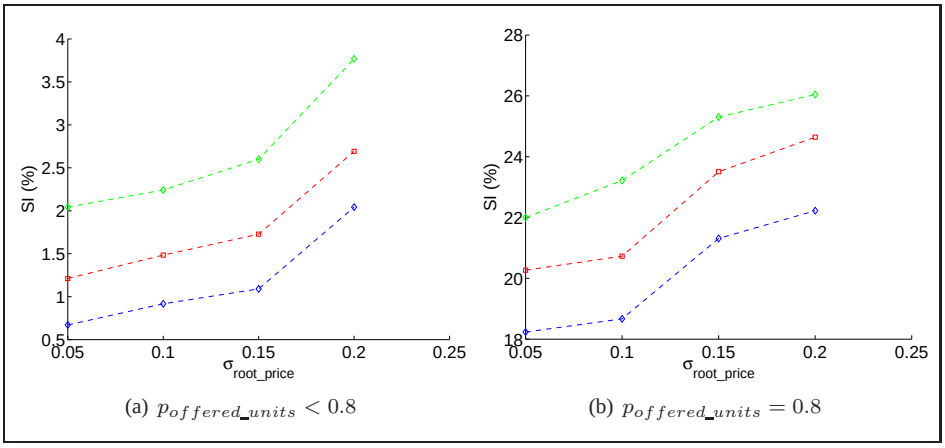


Fig. 18. Histogram of CI.



(a) $p_{offered\_units} < 0.8$ (b) $p_{offered\_units} = 0.8$

Fig. 16. SI with respect to the number of transformations.



(a) $p_{offered\_units} < 0.8$ (b) $p_{offered\_units} = 0.8$

Fig. 17. SI with respect to the variance of providers' prices.

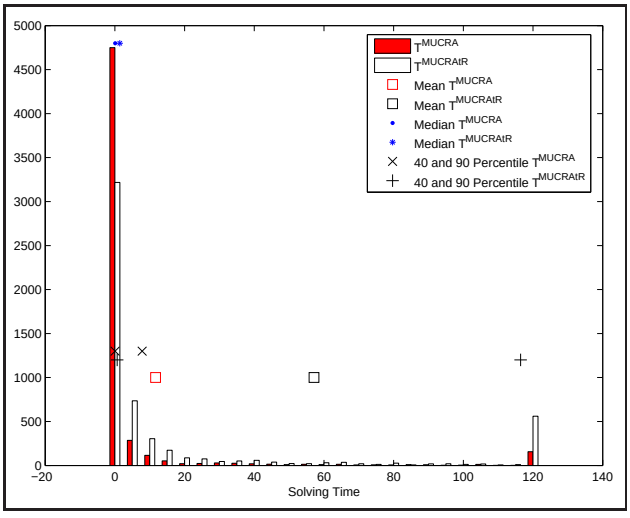
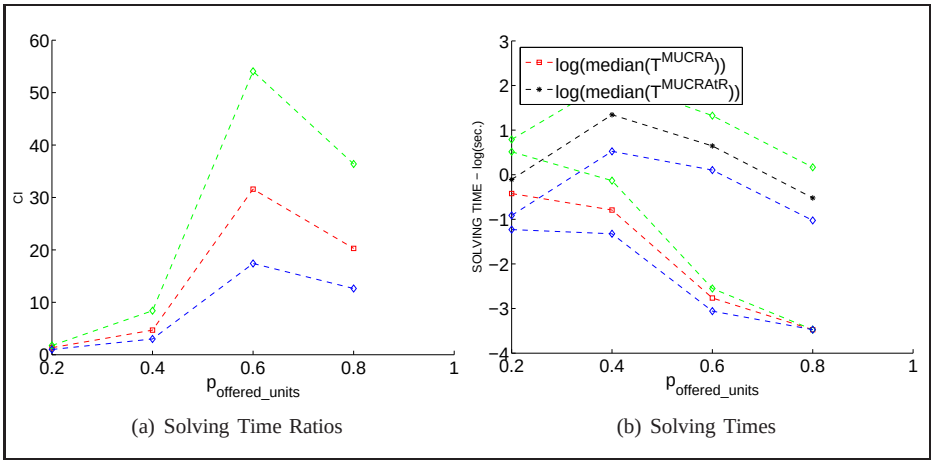


Fig. 19. CI histogram for MUCRA and MUCRAtR.

2) Analysis of Computational Costs: **General Observations.** Figure 18 shows the histogram of CI, and figure 19 shows the histograms of $T^{MUCRAtR}$ and T^{MUCRA}. We observe that their shapes are highly skewed, with very long tails. From figures 18 and 19, and from the statistics in the third column of table VIII, we observe that the mean and variance are not appropriate to characterize neither CI nor $T^{MUCRAtR}$ nor T^{MUCRA}. In fact, the high skewness makes the median a much better and stable descriptor. Furthermore, notice that the fact that we impose a time deadline biases the mean, whereas it does not affect the median. Hence, we employ the *median* and *percentile* in what follows. Figure 19 shows that the probability distribution of $T^{MUCRAtR}$ is much more skewed than T^{MUCRA}'s: their median is very similar, but their 90-percentiles are not. Thus, as expected, *t-relationships* introduce a sensitive increment in computational cost when solving the associated WDP. From table VIII we notice that in 50% of the cases $T^{MUCRAtR}$ is less than 7.1 times larger than T^{MUCRA}. In 70 % of the cases it is less than 30.5 times larger.

Sensitivity Analysis. Our goal is to identify the parameter settings for which the computational cost increases the most. With this aim, in what follows we perform a sensitivity analysis of CI, $T^{MUCRAtR}$ and T^{MUCRA} with respect to different parameters. For each of them we display two graphics, one picturing CI (for instance 20(a)) and another one picturing $T^{MUCRAtR}$ and T^{MUCRA} (for instance 20(b)). Notice that all figures picturing $T^{MUCRAtR}$ and T^{MUCRA}



(a) Solving Time Ratios (b) Solving Times

Fig. 20. CI variation with respect to the number of offered units.

plot the logarithms of their medians because the scales of the two curves may largely differ between them.

Figure 20 illustrates the computational costs when varying the number of offered units ($p_{offered\_units}$). As $p_{offered\_units}$ increases we observe that in a MUCRA less bids are needed to fulfil a buyer's requirements. Furthermore, it is more likely to generate surplus goods, since it is more difficult to perfectly match the buyer's requirements. Therefore, the number of possible bid combinations to explore by a MUCRA solver diminishes as $p_{offered\_units}$ increases, and hence its computational time. Figure 20(b) confirms such a trend for MUCRA. In the same figure we observe that MUCRAtR does not follow the same pattern. We hypothesize that, while in the MUCRA case surplus units are free-disposal, and thus paid for without making any use, in the MUCRAtR case *t-relationships* can transform them into further required goods. Thus, while many bid combinations leading to excessive surplus goods in a MUCRA are pruned from the search space, in a MUCRAtR they are not. Hence, the following conclusion follows:

C6: *The performance penalty is higher when dealing with high-capacity and medium-capacity providers.*

Conclusions **C1** and **C6**, shows that in markets with high-capacity providers both the expected savings and the computational cost are higher than in markets with medium-capacity providers.

Figure 21 analysis the computational costs when varying the number of required units ($units_{RFQ}$). We observe that T^{MUCRA} is not significantly affected by this parameter, whereas there is a positive correlation with $T^{MUCRAtR}$. 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 (41) we infer that the larger $units_{RFQ}$, the more fine-grained adjustments does a transformation allows, because the number of output units of transformations 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. Thus, the number of possibilities to redistribute surplus goods via transformations increases, and hence the computational cost of a MUCRAtR (recall the example justifying conclusion **C2**). From the discussion above, the following conclusion follows:

C7: *The finer the granularity of transformations, the higher the performance penalty.*

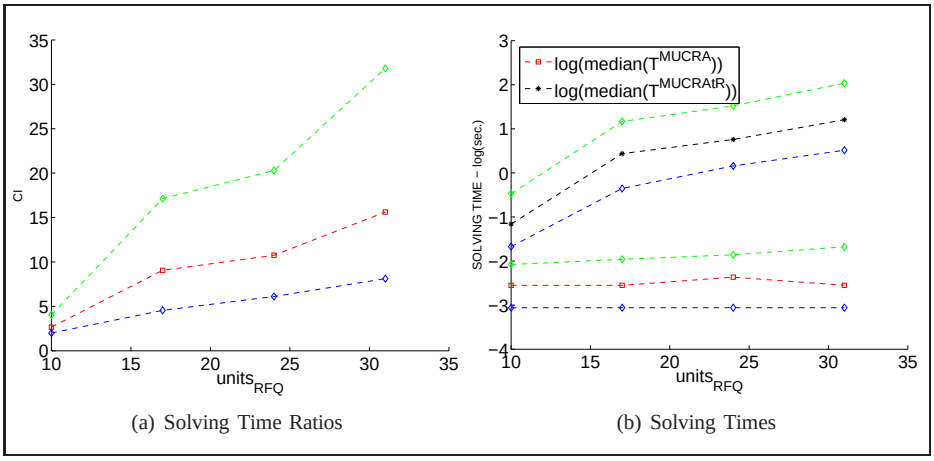
Considering conclusion **C7** along with conclusion **C2**, we infer that the higher the expected savings the more computational price we should be prepared to pay for them as the granularity of transformations is finer and finer.

Figure 22(b) illustrates the computational costs with respect to the variance of the prices among providers (σ_{root_price}). While for a MUCRA the computational time slightly diminishes when increasing σ_{root_price}, for MUCRAtR we notice a drastic decrement. Both decrements are due to a more efficient pruning of the search space. Larger σ_{root_price} increases the differences between *bargain bids* and *costly bids*, thus making easier to prune costly bids. The following example explains why such effect is more significant for MUCRAtR than for MUCRA. Say that a provider offers the root good of a buyer's TNS (for instance p_1 in figure 11(b)) at a very low price. Since the variance among prices is very high, the other offers' prices will be quite scattered. Therefore having found a good bid helps pruning a great quantity of further offers, as well as the transformations applicable to such offers. This effect is more significant for MUCRAtR than for MUCRA since not only offers are pruned, but also possible transformations associated to them. The MUCRAtR search space reduction is larger with respect to the MUCRA one, considering that both bids and possible associated transformations are pruned. Hence, the following conclusion follows:

C8: *The larger the market's prices spread, the lower the performance penalty.*

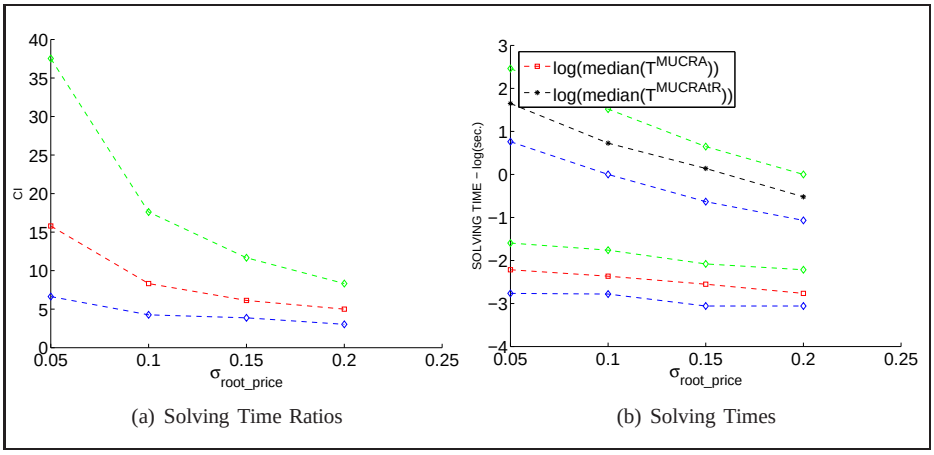
Considering conclusion **C8** along with **C5** we observe that, unlike the trend showed by conclusion **C6** and **C7**, the higher the expected savings the lower the computational price we pay for them as the market's prices spread increases.

Figures 23 and 24 show the solving times when varying the number of bids (m) and the number of required goods ($|P| = n$) respectively. In figures 23(b) and 24(b) we observe that the $T^{MUCRAtR}$ slope is steeper than the T^{MUCRA} one. We expect that for both auctions the WDP becomes harder



(a) Solving Time Ratios (b) Solving Times

Fig. 21. CI variation with respect to the number of required units.



(a) Solving Time Ratios (b) Solving Times

Fig. 22. CI variation with respect to the σ_{root_price} parameter.

when m and n increase. In fact larger values of m involve larger numbers of variables of the associated integer programs, whereas larger values of n correspond to larger numbers of constraints (see equations (40) and (41)). Besides, notice that the MUCRA WDP is a sub-problem of the MUCRAtR WDP: a solution to the former is also a solution to the latter, but not the other way around. A MUCRAtR WDP solver explores a larger search space than a MUCRA, since this does not take into account transformations. Indeed, the MUCRAtR problem is harder. Therefore, when m increases, not only the space of possible bid combinations, but also the space of possible transformations does increase. The search space increment due to transformations only affects MUCRAtR. This explains the slope of T^{MUCRA} with respect to $T^{MUCRAtR}$. Thus, the following conclusion follows:

C9: *The larger the market's offer, the higher the performance penalty.*

Notice that unlike the sensitive parameters identified so far, the number of bids did not appear to be sensitive in our empirical scenario when performing the analysis of savings.

The same argument holds for n. With more goods to combine, we expect an increment in the space of possible combinations of bids for both MUCRA and MUCRAtR. Nonetheless MUCRAtR is affected by a further increment in the search space due to transformations. And hence,

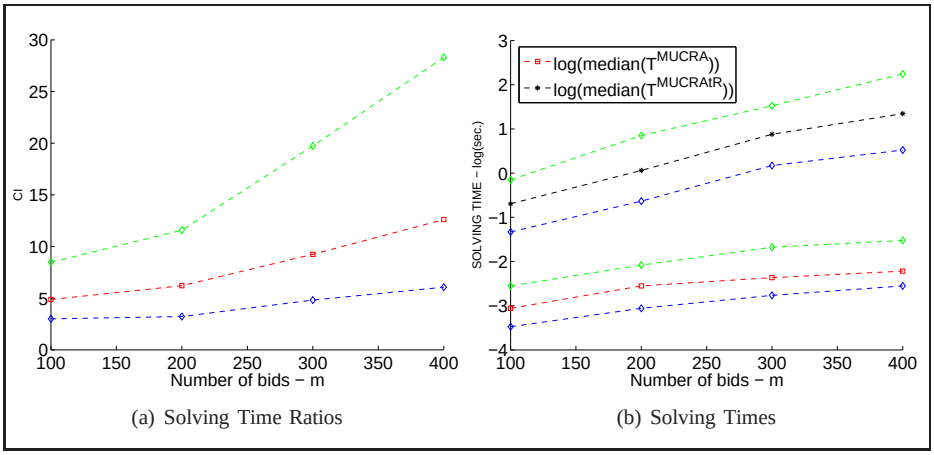
C10: *The larger the number of goods, the higher the performance penalty.*

We also observe that, as expected, the more the number of transformations (r), the larger the time to solve MUCRAtR, whereas the MUCRA solving times remain unchanged. Therefore,

C11: *The more the number of transformations, the higher the performance penalty.*

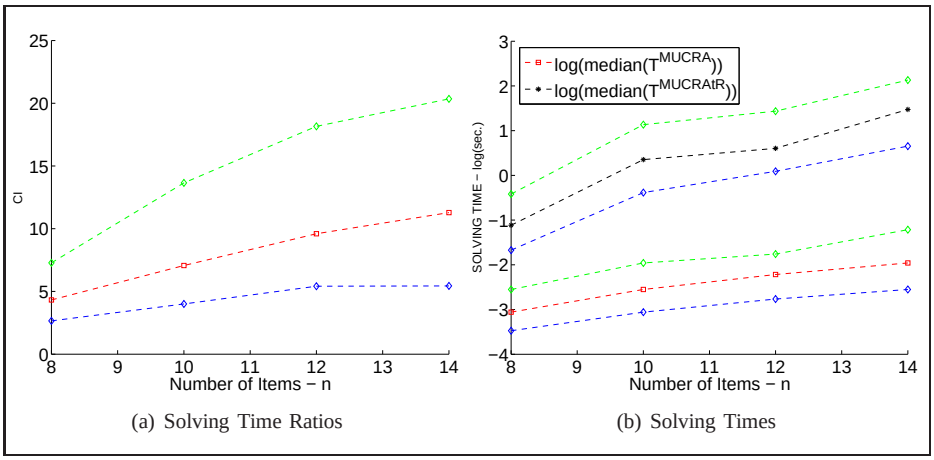
Considering conclusion **C10** along with **C4** we derive that the higher the expected savings the higher the computational price we pay for them as the number of transformations increases.

Finally, figure 25 illustrates the computational costs when varying the bid density ($p_{bid\_density}$). In figure 25(b) we notice that the $T^{MUCRAtR}$ slope is less steep than the T^{MUCRA} one. We expect larger solving times both for MUCRA and MUCRAtR when increasing the bid density, because it is harder to find feasible solutions. This effect is more important for a MUCRA than for a MUCRAtR because of the beneficial effects of transformations: with the help of transformations many bid combinations not leading to a MUCRA feasible solution can eventually lead to a feasible one for the MUCRAtR case. Thus, when the optimizer identifies good feasible



(a) Solving Time Ratios (b) Solving Times

Fig. 23. CI variation with respect to the number of bids.



(a) Solving Time Ratios (b) Solving Times

Fig. 24. CI variation with respect to the required goods.

solutions, it can more effectively prune the search space. From this, we derive the following conclusion:

C12: *The higher the direct competition, the lower the performance penalty.*

Here we borrow the notion of *direct competition* from economics, meaning the competition where goods that perform the same function compete against each other.

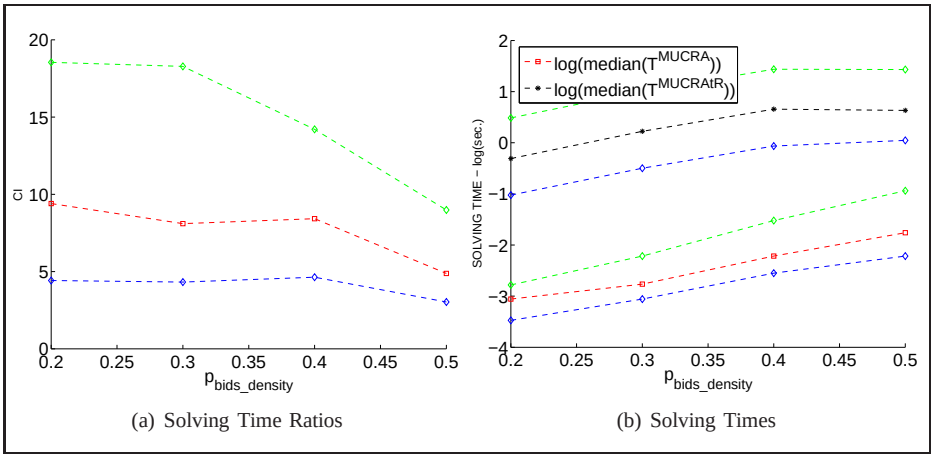
To summarize, notice that while conclusions **C6**, **C7**, **C11**, identify market situations wherein there is a direct correlation between the expected savings and the performance penalty, conclusion **C8** identifies a market situation wherein there is an inverse correlation between the expected savings and the performance penalty.

IX. Related Work

Combinatorial Auctions [20] are a particular type of auctions where bidders can submit offers over bundles of items. This is allowed since complementarities among items may hold on the bidders side, giving to bidders the possibility to express more properly their preferences. In this way more efficient allocations are expected. Furthermore, the risk of speculation on the bidders' side is reduced. For a detailed survey on CA refer to [20]. Although computationally very complex [59], the fact that bidders can express their preferences over bundles of goods may help an auctioneer obtain better deals via a combinatorial auction. Moreover, buying items in bundles has the great advantage of eliminating the risk for a bidder of not being able to sell complementary items at a reasonable price in a follow-up auction. The study of the mathematical, game-theoretical and algorithmic properties of combinatorial auctions has recently become a popular research topic. This is due not only to their relevance to important application areas such as electronic commerce or supply chain management, but also to the range of deep research questions raised by this auction model.

CAs have a high potential to be employed as an allocation mechanism in a wide variety of real-world domains. Thus, they have been proposed to be employed for allocating loads to trucks in the transportation market [14], routes to buses [13], goods/services to buyers/providers in industrial procurement scenarios [45], airport arrival and departure slots [46], and radio-frequency spectrum for wireless communications services [52]. Finally, Walsh in [65] employed them for supply chain formation.

In the last decade different topics related to CAs have been considered, namely the design of auction mechanisms, bidding languages, and algorithms for the WDP.



(a) Solving Time Ratios (b) Solving Times

Fig. 25. CI variation with respect to the bid density.

Auction theory studies the formal properties of auctions as shown in the surveys of [37] and [47]. Nonetheless CAs have recently attracted the attention of economists and game theorists. Associated to auction theory is also the design of auction mechanisms, devoted to study *how* to run an auction in order to guarantee some economic properties such as, for instance, efficiency, incentive compatibility, individual rationality, etc. For instance, [6], [51], [5], [19], [38], and [1] describe some mechanisms for CAs. Notice though that in this paper we do not focus on the design of any mechanism since it is out of our scope, and hence it is left out as future work.

The design of bidding languages has also attracted much attention in the literature on CAs. Bidding languages offer bidders different ways of expressing their preferences. The bidding languages more widely used are the OR and XOR bidding languages ([50] and [58]). While the XOR bidding language allows to express that a collection of bids are mutually exclusive, i.e only one out of the collection can be selected, the OR language instead allows for selecting any possible subset of the bids, being the price the sum of the selected bids' prices. [50] and [58] thoroughly study the problem of expressing preferences on the bidder side. Nonetheless, and to the best of our knowledge, there are no proposals in the literature regarding the possibility to express transformability relationships among goods on the auctioneer side. Our contribution in this paper enriches the bidding languages available to buyers/auctioneers by providing a language that allows to express transformability relationships among goods. In this paper we implicitly employ the OR bidding language.

One of the fundamental issues limiting the applicability of CAs to real-world scenarios is the computational complexity associated to the winner determination problem. In particular, it has been proved that the WDP is NP-complete [56]. In [56] Rothkopf gives an Integer Programming (IP) formulation for the CA WDP considering bids expressed in the OR bidding language, that we extend in this paper in order to include *t-relationships*. General IP solvers [4] and special purpose algorithms (e.g. [57], [24],and [41]) have been employed to solve the WDP, but it is well known that it does not exist a general solver that performs well in all situations. As to the model we present in this paper, we choose an IP formulation to ease its implementation and because there are many powerful IP optimization libraries available in the market.

According to [60], "In a typical supply chain, raw materials are procured and items are produced at one or more factories, shipped to warehouses, for intermediate storage, and then shipped to retailers and customers. [...] The supply chain, consists of suppliers, manufacturing centers, warehouses, distribution centers, and retail outlets.".

Supply chain management (SCM) "is a set of approaches utilized to efficiently integrate suppliers, manufacturers, warehouses, and stores, so that merchandise is produced and distributed at the right quantities, to the right locations, and at the right time, in order to minimize system-wide costs while satisfying service level requirements" [60]. One of the core objectives of the supply chain is to perform a global optimization across the supply chain. But many features of the way businesses are run today prevent this to happen: the uncertainty underlying the supply, the demand, the transportation time, the vehicles and the tools breakdowns. Furthermore the various stake-holders across the supply chain locally maximize their utility disregarding the performances of the other elements within the supply chain. In fact, the different components often have even conflicting objectives. Traditional SCM deals with all these problems acting on different aspects of control: distribution network configuration, supply contracts, distribution strategies, supply chain integration and strategic partnering, inventory control, outsourcing and procurement strategies, information technology and DSSs, etc. In particular, aspects relevant to our work are outsourcing and procurement strategies. Recently, many works have focused on multiagent based supply chain management [12], [44], [25], [21], [66], [36], [7], [28], [29], [30], [62].

In [65], Walsh introduces combinatorial auctions for supply chain formation. These represent an extension of combinatorial auctions in which a whole supply chain is negotiated via an auction. In such a context, askers, sellers and transformers participate and submit bids within the same auction. In order to cope with this new auction Walsh introduces the Task

Dependency Network (TDN), a network representing all the producer/consumer relationships among the bidders. Walsh's model can be sometimes adapted to solve problems similar to the ones we are presenting in our work. Indeed, Task Dependency Networks are limited to transformations relationships with a single output. Thus, any problem in which a good is split into parts cannot be expressed by means of a TDN (for instance deciding whether to sell a whole cow or splitting it and selling its parts). As a second difference, even if its formulation sometimes can be adapted to our case, he focuses specifically on a different problem, the problem of distributed supply chain formation. In particular, he is interested in mechanism design issues, and considers computational aspects as marginal. Instead, in our work we solely focus on computational aspects. In particular, as mentioned in section I, we are concerned with the formal analysis of the decision problem faced by an auctioneer when choosing whether to outsource or not a set of production processes.

The approaches in [15] and [27] build upon Walsh and extend the computational treatment of combinatorial auctions for the supply chain formation winner determination problem. In these works, no assumption is done on the consumer/producer network formed by bidders, thus allowing any type of transformation relationship. However, their focus is still on supply chain formation, and they do not analyze the decision problem we are dealing with.

As far as we know, nobody dealt directly with the *make-or-buy* decision problem employing reverse combinatorial auctions. On the one hand, combinatorial reverse auctions solve the problem of procurement when complementarities among goods exist on the supplier side. On the other hand, operations research has studied the best *make-or-buy* decisions based on past production information, sell forecast, providers' offers, etc [2][18]. However, nobody embedded the decision problem into the procurement problem when complementarities among goods hold, nobody analyzed the procurement decisions in conjunction with the outsourcing decisions in a combinatorial scenario.

Finally, no real world benchmarks of CAs have been reported in the literature. Many efforts have been done so far to generate plausible data sets to be employed to test WDP algorithms. Even some experiments have been run with human bidders ([8] and [39]). Nonetheless, as pointed out in [42], such data sets are not useful for assessing the WDP computational complexity. In the absence of test suites, it is common practice to artificially generate data sets. Some examples are [24], [10], and [22] for single-unit CAs, and [41] for multi-unit CAs. Multi-unit CAs have also been tested employing multidimensional knapsack problem benchmarks, borrowed from the operations research community. A more realistic approach to generate bids is presented in [42], where complementarity relationships among goods are made explicit at bid generation time. Another realistic approach is taken in [3], where the authors design bidding strategies that efficiently identify desirable bundles in the framework of the transportation industry domain (focusing therefore on single-round, first-price, sealed-bid forward CAs). Walsh in [64] simulates the behavior of bidders in a supply chain formation environment. Even though he introduces bids on transformation processes, we cannot exploit his results since they are only valid within the context of supply chain formation. Although a large amount of literature has been devoted to studying the generation of bids, we cannot exploit such results since none takes into account *t-relationships* in a context similar to ours. Thus, our contribution along this direction stems from the design and implementation of a generator of data sets that includes *t-relationships* among goods.

[18]For a general review on decision support to supply chain management refer to [23].

X. Conclusions and Future Work

A. Conclusions

Most of the currently studied and employed combinatorial auctions deal with the negotiation of goods, disregarding eventual production relationships holding among them. The information about such relationships helps improve the outcome of a negotiation. In order to fill this gap, we introduced a novel combinatorial auction extensions that help in determining the revenue-maximizing strategy for partner selection in supply chain network design and planning. The new auction type, called *Multi Unit Combinatorial Reverse Auctions with Transformability Relationships among Goods (MUCRAtR)*, copes with *make-or-buy* decisions.

In section I-B2, we thoroughly described the requirements that must be fulfilled in order to solve *make-or-buy* decision problems. Since we built upon combinatorial auctions, we also explained the limitations of CAs that hinder their application to our problem. In table IX we recall both the requirements and the corresponding CAs limitations associated with the *make-or-buy* decision problem. We observed that all the CAs limitations stem from the fact that they can neither express nor represent an auctioneer's internal manufacturing operations.

The first requirement hindering the application of CAs to our problem is that they can neither represent internal manufacturing operations nor the producer/consumer relationships among them. In order to apply CAs to solve the *make-or-buy* decision problem, we provided a formal framework to represent internal manufacturing operations. At this aim we decided to employ Place/Transition Nets (PTNs) [55] because:

(1) they naturally help capture the notion of manufacturing operation;
(2) they have a well-defined semantics that can naturally accommodate the notion of sequence of operations and consumer/producer relationships;
(3) they have an integrated description of both states and actions to characterize the search space where operations occur;
(4) they have a large number of formal analysis methods that allow the investigation of structural and behavioral (dynamic) properties of the net; and
(5) they have a graphical representation that is intuitively very appealing to study problems related to the topology of the supply chain.

	Requirements	CAs	MUCRAtR
1	express a request on bundles of goods	✓	✓
2	express an auctioneer's initial stock		✓
3	express producer/consumer relationships among internal operations		✓
4	specify an auctioneer's final requirements		✓
5	express relationships among manufacturing operations, auctioned goods, and received bids		✓
6	formally and graphical represent the search space associated to the auctioneer's decision problem		✓
7	specify the auctioneer's internal cost structure		✓
8	information about which in-house operations to perform and in which order		✓

TABLE IX
REQUIREMENTS OF TO THE *make-or-buy* PROBLEM.

Thus, we modelled the internal production structure of an auctioneer by means of a PTN, that we referred to as PTN_I. Not only does this formal representation allow to describe the quantity of resources either produced or consumed by a manufacturing operation, the producer/consumer relationships among operations, and the quantity of goods available to an auctioneer after each operation, but it also allow to express preconditions over a manufacturing operation by means of a *firing rule*. By the application of the firing rule, we impose that a manufacturing operation can *only* be run if its input goods are available. This property is critical for the correct representation of a production process: the implementation order of a production process is constrained by the availability of resources at each step.

Then, a PTN_I completely specifies an auctioneer's internal manufacturing operations and the producer/consumer relationships among them (requirement (5) in table IX). Moreover, a PTN_I allows an auctioneer to specify his requirements and communicate them to bidders (requirement (4) in table IX). This is obtained by specifying a configuration (marking) to end up with. If an auctioneer communicates to a set of bidders his PTN_I along with a description of the final state of such PTN describing his requirements, then the bidders can infer all the possible configurations of offers fulfilling such requirements.

Next, in order to express the relationships among internal manufacturing operations, auctioned goods, and received bids (requirement (3) in table IX), we incorporated the received bids into PTN_I. At this aim, we exploited the fact that a bid that offers goods can be regarded as a transition (*bid transition*) that injects tokens into PTN_I. Unlike transitions corresponding to manufacturing operations, bid transitions do not consume input resources and can be fired only once. The PTN_I augmented with bid transitions was called PTN_E (where E stands for *Extended*).

By means of a PTN_E, an auctioneer can compactly express all the possible outcomes of any of his possible decisions. By decision we mean the selection of bids together with a sequence of internal operations to perform. Thus, a PTN_E both formally and graphically represents the search space associated to the auctioneer's decision problem (requirement 6 in table IX). We successfully linked bids, manufacturing operations, and goods at auction, and we fully represent all the possible decisions an auctioneer may take in a unified representation.

However, the goal of the auctioneer is not only to find a feasible outcome, but also to find an outcome that minimizes his costs. Thus, an auctioneer needs to quantify the cost associated to each decision. With this aim, he has to associate a cost to the selection of a bid, and a cost to the performance of a manufacturing operation. Unfortunately, in their original definition, place transition nets do not incorporate the notion of cost associated to the firing of a transition. Then, we defined a new type of Place Transition Net, the so-called *Weighted Place Transition Nets* (WPTN), to express the notion of cost associated to transition firings or to the firing of sequences of transitions.

Then, we transformed both PTN_I and PTN_E into WPTNs by associating to each *operation transition* the cost of the corresponding manufacturing operation and to each *bid transition* the bid cost. The resulting WPTNs were called *Transformability Network Structure* (TNS) and *Auction Net* respectively: a TNS completely describes an auctioneer's internal manufacturing operations, whereas an Auction Net compactly represents the set of possible auctioneer's decisions along with the corresponding cost. Then, *Transformability Network Structure* and *Auction Net* allow the auctioneer to express his internal cost structure and to incorporate it into his decision problem (requirement (7) in table IX).

The auctioneer needs to select the set of offers along with the sequence of internal manufacturing operations to perform that minimize his costs and allow him to obtain his final requirements (point (8) in table IX). With this purpose, we defined the auctioneer's decision problem as an optimization problem on the *Auction Net*. Thus, we introduced a new type of reachability problem over WPTNs, and called this new optimization problem the *Constrained Maximum Weight Occurrence Sequence Problem* (CMWOSP). Intuitively, this optimization problem involves finding an optimal cost sequence of transitions on a WPTN that leads to a final state which fulfils some constraints.

Additionally, we provided an important result on the CMWOSP. We showed that the CMWOSP can be solved by

means of Integer Programming on acyclic WPTNs, namely on WPTNs that do not contain any directed cycle.

The CMWOSP perfectly captures the semantics of the auctioneer's decision problem in a MUCRAtR: to find the set of *bid and operation transitions* that minimizes an auctioneer's revenue. Thus, we formalized the auctioneer's decision problem in a MUCRAtR as a CMWOSP on the Auction Net. Two major benefits, and therefore contributions, stemmed from the formalization of the MUCRAtR WDP by means of a CMWOSP:

(1) the CMWOSP provides as a result both the set of bids to accept and the sequence of operations to perform in order to obtain the auctioneer's final requirements (requirement (8) in table IX);
(2) the *make-or-buy* decision problem can be solved by means of Integer Programming for a large class of supply chain network topologies (acyclic).

Summarizing, we provide the auctioneer with a formalism to express his requirements and to communicate them to bidders; and a rule for determining the optimal allocation, i.e. the set of winning bids and the sequence of internal operations to perform. In this way we provide a solution to all the requirements needed for extending combinatorial auctions for dealing with the *make-or-buy* decision problem.

The solution to the WDP that we provide can be employed as a decision support system in different settings:

- *Combinatorial auctions*. As a winner determination solver in a MUCRAtR.
- *Negotiation*. A buyer, after receiving a set of offers from his providers, can compute the best offers and eventually counter-offer.
- *What-if supply chain analysis*. A buyer, aware of the prices and capacities of his providers, can test different configurations of his supply chain.

Furthermore, we have performed a set of experiments to empirically evaluate the benefits and drawbacks of introducing *t-relationships*. Our experiments have shed light along two main directions. Firstly, they have helped identify the negotiation conditions under which MUCRAtR is expected to lead to savings with respect to MUCRA, namely negotiations: (1) with high-capacity providers; (2) for which the likelihood of exploiting transformations is high; (3) run by an auctioneer/buyer whose transformation costs are lower than the providers' ones; and (4) in markets where providers' offers are scattered. Secondly, our experiments also confirm that MUCRAtR is indeed computationally harder than MUCRA. In particular we notice that the most critical parameters affecting the MUCRAtR WDP solver performances are: (1) the number of offered units; (2) the number of required units; (3) the number of bids; and (4) the number of negotiated goods. From these observations, we conclude that it is worth employing MUCRAtR instead of MUCRA under a significant number of negotiation conditions. If so, we must keep in mind that, as a general trend, the more the expected savings the more the performance penalty. Furthermore, we can also conclude that the providers' capacities is the most sensitive parameter to help a buyer/auctioneer to get insights on the expected savings under some particular negotiation scenario.

B. Future Work

We envision several paths to future development. On the theoretical side, mechanism design and further analysis on families of petri nets (not just acyclic). On the computational side, local algorithms. On the practical side, assess the value of our approach in actual scenarios with real-world data (for instance in the automotive industry).

It is for sure a matter of discussion whether a buyer should communicate to bidders its TNS. This problem should be probably faced employing a game theoretic perspective. Yet, it is a matter of future study to understand which type of negotiation is a combinatorial auction without a public winning rule.

Finally, we believe that providing decision support for bidders in MUCRAtR is necessary to help them cope with the complexity of participating in a combinatorial negotiation scenario.

It seems quite natural also to consider a further extension. If an auctioneer can incorporate into the auction its internal operations, why not to incorporate information about the bidders' internal operations as well? That is, in a MUCRAtR an auctioneer decides whether to produce in-house or to buy as already made the goods he requires. However, there is a third possibility, a bidder may offer to *perform* an operation for the auctioneer. In such a case, the auctioneer would be able to outsource not only goods, but manufacturing operations or services as well.

ACKNOWLEDGEMENTS

This work was funded by projects IEA (TIN2006-15662-C02-01), Agreement Technologies (CONSOLIDER CSD2007-0022, INGENIO 2010), and the Jose Castillejo programme (JC2008-00337) of the SpanishMinistry of Science and Innovation. This works was also funded by the Juan de la Cierva programme supporting the first author (JCI-2008-03006) and by the EU funded Synthetic Forager project (ICT-217148-SF).

Thanks to the anonymous reviewers for their very valuable feedback.

REFERENCES

[1] Susan Powell Ailsa Land and Richard Steinberg. *PAUSE: A Computationally Tractable Combinatorial Auction*, chapter 6. Combinatorial Auctions. MIT Press, 2006.
[2] N. Aissaoui, M. Haouari, and E. Hassini. Supplier selection and order lot sizing modeling: A review. *Computers and Operations Research*, 34(12):3516–3540, 2007.
[3] N. An, W. Elmaghraby, and P. Keskinocak. Bidding strategies and their impact on revenues in combinatorial auctions. *Journal of Revenue and Pricing Management (to appear)*, 2005.
[4] A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Fourth International Conference on Multiagent Systems (ICMAS 2000)*, pages 39–46, Boston, MA, 2000.
[5] Lawrence M. Ausubel and Paul Milgrom. *Ascending Proxy Auctions*, chapter 3. Combinatorial Auctions. MIT Press, 2006.
[6] Lawrence M. Ausubel and Paul Milgrom. *The Lovely but Lonely Vickrey Auction*, chapter 1. Combinatorial Auctions. MIT Press, 2006.

[7] M. Babaioff and W.E. Walsh. Incentive Compatible Supply Chain Auctions. *Intelligence (SCI)*, 28:315–350, 2006.

[8] Jeffrey Banks, John O. Ledyard, and David P. Porter. Allocating uncertain and unresponsive resources, an experimental approach. *RAND Journal of Economics*, 1989.

[9] W.D. Blizard. Multiset theory. *Notre Dame J. Formal Logic*, 30(1):36–66, 1988.

[10] C. Boutilier, M. Goldszmidt, and B. Sabata. Sequential auctions for the allocation of resources with complementarities. In *IJCAI*, pages 1–23, 1999.

[11] G.E.P. Box, W.G. Hunter, and J.S. Hunter. *Statistics for experimenters. An introduction to design, data analysis, and model building*. Wiley, 1978.

[12] C. Brahim and J.P. Müller. *Multiagent-based Supply Chain Management*. Springer, 2006.

[13] Estelle Cantillon and Martin Pesendorfer. *Auctioning Bus Routes: The London Experience*, chapter 22. Combinatorial Auctions. MIT Press, 2006.

[14] C. Caplice and Y. Sheffi. *Combinatorial Auctions for Truckload Transportation*, chapter 21. Combinatorial Auctions. MIT Press, 2006.

[15] J. Cerquides, U. Endriss, A. Giovannucci, and J. A. Rodriguez-Aguilar. Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In *Proc. of the 20th intl. joint conferences on artif. intelligence (IJCAI)*, pages 1221–1226, Hyderabad, India, January 2007.

[16] J. Chen and He Huang. On-line multi-attributes procurement combinatorial auctions bidding strategies. *Lecture notes in computer science*.

[17] P.C. Chu and J.E. Beasley. A genetic algorithm for the multidimensional knapsack problem. *Journal of Heuristics*, 4:63–86, 1998.

[18] M.F. Corbett. *The Outsourcing Revolution: Why it Makes Sense and how to Do it Right*. Dearborn Trade Pub., 2004.

[19] Peter Cramton. *Simultaneous Ascending Auctions*, chapter 4. Combinatorial Auctions. MIT Press, 2006.

[20] Peter Cramton, Yoav Shoham, and Richard Steinberg, editors. *Combinatorial Auctions*. MIT Press, 2006.

[21] R.K. Dash, P. Vytelingum, A. Rogers, E. David, and N.R. Jennings. Market-Based Task Allocation Mechanisms for Limited-Capacity Suppliers. *Ieee Transactions On Systems Man And Cybernetics Part A Systems And Humans*, 37(3):391, 2007.

[22] Sven de Vries and Rakesh Vohra. Combinatorial auctions: A survey. *INFORMS Journal of Computing*, 15(3):284–309, 2003.

[23] SS Erenguc, NC Simpson, and AJ Vakharia. Integrated production/distribution planning in supply chains: An invited review. *European Journal of Operational Research*, 115(2):219–236, 1999.

[24] 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.

[25] N. Garg, D. Grosu, and V. Chaudhary. Antisocial Behavior of Agents in Scheduling Mechanisms. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 37(6):946–954, 2007.

[26] A. Giovannucci, J. Cerquides, and J. A. Rodríguez-Aguilar. Benefits of combinatorial auctions with transformability relationships. In *Proceedings of the 17th European Conference on Artificial Intelligence (ECAI 2006)*, pages 717–718, Riva di Garda, Italy, August 2006.

[27] A. Giovannucci, J.A. Rodríguez-Aguilar, J. Cerquides, and U. Endriss. Winner determination for mixed multi-unit combinatorial auctions via petri nets. In *Twentieth International Conference on Autonomous Agents and Multi Agent Systems (AAMAS 2007)*, Hawaai, USA, May 2007. To appear.

[28] A. Giovannucci, JA Rodriguez-Aguilar, A. Reyes, FX Noria, and J. Cerquides. Towards automated procurement via agent-aware negotiation support. In *Autonomous Agents and Multiagent Systems, 2004. AAMAS 2004. Proceedings of the Third International Joint Conference on*, pages 244–251, 2004.

[29] A. Giovannucci, JA Rodríguez-Aguilar, A. Reyes, FX Noria, and J. Cerquides. Enacting agent-based services for automated procurement. *Engineering Applications of Artificial Intelligence*, 21(2):183–199, 2008.

[30] A. Giovannucci, M. Vinyals, JA Rodriguez-Aguilar, and J. Cerquides. Computationally-efficient winner determination for mixed multi-unit combinatorial auctions. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems (AAMAS 08)*, volume 2, pages 1071–1078, 2008.

[31] M.F. Greaver. *Strategic Outsourcing: A Structured Approach to Outsourcing Decisions and Initiative*. AMACOM, 1999.

[32] Aberdeen Group. Making e-sourcing strategic: from tactical technology to core business strategy. Technical report, Aberdeen Group, 2002.

[33] F.S. Hillier and G.J. Lieberman. *Introduction to Operation Research*. McGraw-Hill, 1986.

[34] R. V. Hogg and J. Ledolter. *Engineering Statistics*. MacMillan Publishing Company, 1987.

[35] R. C. Holte. Combinatorial auctions, knapsack problems, and hill-climbing search. In *Lecture Notes in Computer Science*, volume 2056, pages 57 – 66. Springer-Verlag, Heidelberg.

[36] S. Hong, BN Nag, and D. Yao. Modeling Agent Auctions in a Supply Chain Environment. *International Journal of Intelligent Information Technologies*, 3(1):14–36, 2007.

[37] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[38] Peter Cramton Lawrence M. Ausubel and Paul Milgrom. *The Clock-Proxy Auction: A Practical Combinatorial Auction Design*, chapter 5. Combinatorial Auctions. MIT Press, 2006.

[39] John O. Ledyard, David P. Porter, and Antonio Rangel. Designing organizations for trading pollution rights. *Journal of Economic Behaviour and Organizations*, 1997.

[40] D. Lehmann, R. Mueller, and T. Milgrom Sandholm. *The winner determination problem*, chapter 12. Combinatorial Auctions. MIT Press, 2006.

[41] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of the American Association for Artificial Intelligence Conference (AAAI)*, pages 56–61, 2000.

[42] Kevin Leyton-Brown and Yoav Shoham. *Combinatorial Auctions*, chapter 18. A Test Suite for Combinatorial Auctions. MIT Press, 2006.

[43] R Lipton. The reachability problem requires exponential space. Technical Report 62, Yale University.

[44] N. Liu, M.A. Abdelrahman, and S. Ramaswamy. A Complete Multiagent Framework for Robust and Adaptable Dynamic Job Shop Scheduling. *Ieee Transactions On Systems Man And Cybernetics Part C Applications And Reviews*, 37(5):904, 2007.

[45] Gail Hohner Martin Bichler, Andrew Davenport and Jayant Kalagnanam. *Industrial Procurement Auctions*, chapter 23. Combinatorial Auctions. MIT Press, 2006.

[46] George L. Donohue Michael O. Ball and Karla Hoffman. *Auctions for the Safe, Efficient, and Equitable Allocation of Airspace System Resources*, chapter 20. Combinatorial Auctions. MIT Press, 2006.

[47] Paul Milgrom. *Putting Auction Theory to Work*. Cambridge University Press, 2004.

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

[49] Noam Nisan. *Bidding Languages for Combinatorial Auctions*, chapter 9. Combinatorial Auctions. MIT Press, 2006.

[50] Noam Nisan. *Bidding Languages for Combinatorial Auctions*, chapter 9. Combinatorial Auctions. MIT Press, 2006.

[51] David C. Parkes. *Iterative Combinatorial Auctions*, chapter 2. Combinatorial Auctions. MIT Press, 2006.

[52] Aleksandar Pekec and Michael H. Rothkopf. Combinatorial auction design. *Manage. Sci.*, 49(11):1485–1503, 2003.

[53] CA Petri. Kommunikation mit automaten. Schriften des iim nr. 2, Institut fur Instrumentelle Mathematic, 1962. Technical report, English translation: Technical Report RADC-TR-65-377, Griffiths Air Base, New York, 1966.

[54] J.B. Quinn and F.G. Hillmer. Strategic Outsourcing. *The McKinsey Quarterly*, (1), 1995.

[55] W. Reisig. *Petri nets: an introduction*. Springer-Verlag, New York, NY, USA, 1985.

[56] Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. Computationally manageable combinational auctions. *Management Science*, 44(8):1131–1147, 1998.

[57] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[58] T. Sandholm and C. Boutilier. *Preference Elicitation in Combinatorial Auctions*, chapter 10. Combinatorial Auctions. MIT Press, 2006.

[59] T. Sandholm, S. Suri, A. Gilpin, and D. Levine. Winner determination in combinatorial auction generalizations. In *AAMAS '02: Proceedings of the first international joint conference on Autonomous agents and multiagent systems*, pages 69–76, Bologna, Italy, 2002. ACM Press.

[60] D. Simchi-Levi, P. Kaminsky, and E. Simchi-Levi. *Designing and Managing the Supply Chain: Concepts, Strategies, and Case Studies*. Irwin/McGraw-Hill, 2000.

[61] A. Tarek and N. Lopez-Benitez. Optimal legal firing sequence of petri nets using linear programming. *Optimization and Engineering*, 5(1):25–43, March 2004.

[62] M. Vinyals, A. Giovannucci, J. Cerquides, P. Meseguer, and J.A. Rodriguez-Aguilar. A test suite for the evaluation of mixed multi-unit combinatorial auctions. *Journal of Algorithms*, 2008.
[63] N. Viswanadham. The past, present, and future of supply-chain automation. *IEEE Robotics & Automation Magazine*, 9(2):48–56, 2002.
[64] W. E. Walsh. *Market protocols for decentralized supply chain formation*. PhD thesis, 2001. Chair-Michael P. Wellman.
[65] W. E. Walsh, M. P. Wellman, and F. Ygge. Combinatorial auctions for supple chain formation. In *Proc. of the 2nd ACM Conference on Electronic Commerce*, 2000.
[66] B.P.C. Yen and OQ Wu. Internet scheduling environment with market-driven agents. *Systems, Man and Cybernetics, Part A, IEEE Transactions on*, 34(2):281–289, 2004.