Empirical Hardness for Mixed Auctions*

Pablo Almajano[1], Jesús Cerquides[2], and Juan A. Rodriguez-Aguilar[1]

[1] IIIA, Artificial Intelligence Research Institute
CSIC, Spanish National Research Council
{palmajano,jar}@iiia.csic.es
[2] UB, WAI, Dep. Matemàtica Aplicada i Anàlisi
Universitat de Barcelona
cerquide@maia.ub.es

Abstract. Mixed Multi-Unit Combinatorial Auctions (MMUCAs) offer a high potential to be employed for the automated assembly of supply chains of agents. However, little is known about the factors making a winner determination problem (WDP) instance hard to solve. In this paper we empirically study the hardness of MMUCAs: (i) to build a model that predicts the time required to solve a WDP instance (because time can be an important constraint during an auction-based negotiation); and (ii) to assess the factors that make a WDP instance hard to solve.

Keywords: mixed multi-unit combinatorial auction, machine learning.

1 Introduction

In [1] we introduced the so-called *mixed multi-unit combinatorial auctions* (henceforth *MMUCA* for short) and discussed the issues of bidding and winner determination. Mixed auctions are a generalisation of the standard model of combinatorial auctions (CAs) [2]. Thus, rather than negotiating over goods, in mixed auctions the auctioneer and the bidders can negotiate over *supply chain operations* (henceforth *transformations* for short), each one characterised by a set of input goods and a set of output goods. A bidder offering a transformation is willing to produce its output goods after having received its input goods along with the payment specified in the bid. While in standard CAs, a solution to the winner determination problem (WDP) is a set of atomic bids to accept that maximises the auctioneer's revenue, in mixed auctions, the *order* in which the auctioneer "uses" the accepted transformations matters. Thus, a *solution* to the WDP is a *sequence of operations*. For instance, if bidder *Joe* offers to make dough if provided with butter and eggs, and bidder *Lou* offers to bake a cake if provided with enough dough, the auctioneer can accept both bids whenever he uses Joe's operation before Lou's to obtain baked cakes from butter and eggs. Since the existence of a solution is not guaranteed in the case of MMUCAs (unlike classical CAs), attention is focused not only on the winner determination

* Funded by projects IEA (TIN2006-15662-C02-01), AT (CSD2007-0022), EVE (TIN2009-14702-C02-01, TIN2009-14702-C02-02).

P. Meseguer, L. Mandow, and R.M. Gasca (Eds.): CAEPIA 2009, LNAI 5988, pp. 161–170, 2010.

problem, but also on the feasibility problem of deciding whether a given instance admits a solution at all. In fact, an important and peculiar source of complexity for MMUCAs lays hidden in this latter problem as noticed in [3].

The WDP for MMUCAs is a complex computational problem. In fact, one of the fundamental issues limiting the applicability of MMUCAs to real-world scenarios is the computational complexity associated to the WDP. In particular, it is proved in [1] that the WDP for MMUCAs is $\mathcal{NP}$-complete. And yet little is known about its hardness, namely about what makes a WDP instance hard to solve. Hence, on the one hand some WDP instances may unexpectedly take longer to solve than required (time is important in auction-based negotiations). On the other hand, lack of knowledge about the hardness of MMUCA prevents the development of specialised winner determination algorithms. Unlike classical CAs, little is known about whether polynomial-time solvable classes of MMUCAs can be singled out based on the structural and topological properties of the instances at hand. Thus, in this paper we try to make headway in the understanding of the hardness of MMUCAs by applying the methodology described in [4] that Lleyton-Brown et al successfully apply to CAs. The results in this paper must be regarded as the counterpart of the results about empirical hardness obtained in [4] for CAs.

The paper is organised as follows. In section 2 we outline an integer program introduced in [5] to efficiently solve the WDP for MMUCAs. In section 3 we outline the methodology introduced in [4] to subsequently employ it to build a model that predicts the time required by the integer program to solve a WDP instance. Next section 5 further exploits the methodology to assess the factors that make a WDP instance hard to solve. Finally, section 6 draws some conclusions and sets paths to future research.

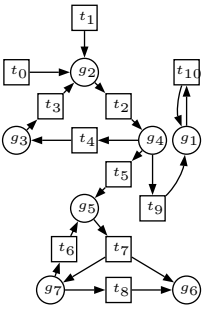
2 CCIP: A Topology-Based Solver

In this section we summarise CCIP, a mapping of the MMUCA WDP into an integer program (IP) that drastically reduces the number of decision variables required by the original IP described in [1] by exploiting the topological features of the WDP. CCIP will be employed in forthcoming sections to analyse the empirical hardness of the MMUCA WDP. Notice that hereafter we limit to outlining the intuitions underlying CCIP. We refer the reader to [5] for a detailed IP formulation.

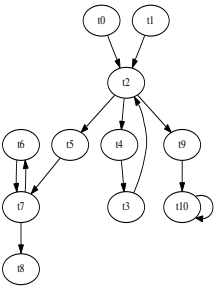
Consider that after receiving a bunch of bids, we draw the relationships among goods and transformations, as shown in Figure 1 (a). There, we represent goods at trade as circles, transformations as squares, a transformation input goods as incoming arrows and its output goods as outgoing arrows. Thus, for instance, transformation t_0 offers one unit of good g_2 and transformation t_2 transforms one unit of g_2 into one unit of g_4. Say that the auctioneer requires a multiset of goods $U_{out} = \{g_2, g_3\}$. Row 1 in table 1 stands for a valid solution sequence. Indeed, it stands for a valid solution sequence because at each position, enough input goods are available to perform the following transformation. Notice too

Table 1. Partial sequences of transformations

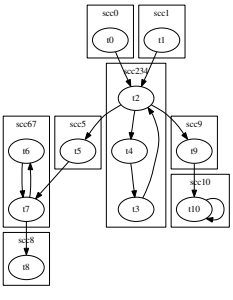
Position	1	2	3	4	5	6	7	8	9	10	11
Sequence 1	t_0	t_2			t_1		t_4				
Sequence 2	t_0	t_1	t_2	t_4							
Sequence 3	t_2	t_1	t_0	t_4							
Solution template	t_0	t_1	t_2 t_3 t_4	t_2 t_3 t_4	t_2 t_3 t_4	t_5	t_9	t_{10}	t_6 t_7	t_6 t_7	t_8



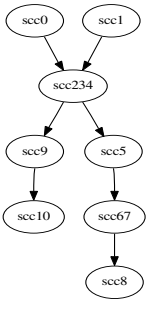
(a) A bid set



(b) TDG



(c) SCCs of the TDG



(d) The strict order

Fig. 1. An MMUCA bid set, the corresponding TDG, SCC, and Order Relation

that likewise row 1, row 2 also stands for a valid solution sequence because even though they differ in the ordering among transformations, both use exactly the same transformations, and both have enough goods available at each position. However, row 3 in table 1 is not a valid sequence, although it contains the same transformations, because t_2 lacks enough input goods (g_2) to be used.

In Figure 1 (a), it is clear that transformations that have no input goods can be used prior to any other transformation. Thus, transformations t_0 and t_1 can come first in the solution sequence. Moreover, we can *impose* that t_0 comes

before t_1 because swapping the two would yield an equivalent solution. If we now consider transformations t_2, t_3, t_4, we observe that: (i) they *depend* on the output goods of t_0 and t_1; and (ii) we cannot impose an arbitrary order among them because they form a cycle and then they can feed with input goods one another (they depend on one another). However, no permutation of the three can be discarded for the valid solution sequence. Furthermore, whatever their order, we can always use them before transformations t_5 and t_9 (since these depend on g_4) without losing solutions.

Assuming that the auctioneer does not care about the ordering of a solution sequence as long as enough goods are available for every transformation in the sequence, we can impose "a priori" constraints on the ordering of transformations without losing solutions. The way of imposing such constraints is via a *solution template*, a pattern that any candidate sequence must fulfill to be considered as a solution. For instance, row 4 in table 1 shows a sample of solution template. A solution sequence *fulfilling* that template must have transformations t_0 in position 1 and t_1 in position 2, whereas it is free to assign positions 3, 4, or 5, to the transformations in $\{t_2, t_3, t_4\}$. For instance, row 3 of table 1 does not fulfill the template in row 4, whereas rows 1 and 2 do.

Notice that the constraints in the solution template derive from our analysis of the dependence relationships among transformations. Hence, in order to build a solution template, we must firstly analyse the dependence relationships among transformations to subsequently use them to constrain the positions at which a transformation can be used.

At this aim, we can follow the sequential process illustrated in Figure 1:

1. Define the so-called transformation dependency graph (TDG), a graph where two transformations t and t' are connected by an edge if they have a good that is both output of t and input to t' (direct dependence). Figure 1 (b) depicts the TDG for the bids represented in Figure 1(a).
2. Assess the *strongly connected components* (SCC) of the TDG. Depending on the received bids, the TDG may or may not contain strongly connected components. In order to constrain the position of transformations, we transform the TDG in an acyclic graph where the nodes that form a strongly connected component are collapsed into a single node. The main idea is that the positions of transformations in a strongly connected component are drawn from the same set of positions, but we cannot impose any order regarding the position each transformation takes on. In Figure 1(c) we identify strongly connected components or SCCs in the graph. In figure 1(d) we can see the graph resulting from transforming (collapsing) each SCC into a node.

If there is a strict order among transformations (e.g. like the one depicted in Figure 1(d)), then we can always construct a solution template that restricts the positions that can be assigned to those transformations in a way that, if a solution sequence fulfills the solution template, the strict order is also fulfilled [6]. For instance, consider the solution template in row 4 in table 1 that we construct considering the strict order in Figure 1(d). Since the solution sequences in rows 1 and 2 of table 1 fulfill the solution template in row 4, they both fulfill the strict order.

Now we are ready to characterise valid solutions to the MMUCA WDP. Looking back at the solution sequences in rows 1 and 2 of table 1, we realise that both are *partial sequences*. A partial sequence is a sequence with "holes", meaning that there can be some positions in the sequence that are *empty*. Therefore, a valid solution to the MMUCA WDP can be encoded as a partial sequence of transformations that *fulfills* some solution template.

3 Empirical Hardness Models

In [4], Leyton-Brown et al. propose a general methodology to analyze the empirical hardness of $\mathcal{NP}$-hard problems. The purpose of the methodology is twofold. On the one hand, given an algorithm to solve some hard problem, it provides the guidelines to build (learn) a model that predicts the running time required by the algorithm to solve an instance of the problem. On the other hand, the methodology also discusses techniques to analyse the factors that determine how hard some distributions or individual instances are to solve (by the algorithm under study). Moreover, since the methodology is successfully applied to the particular case of CAs, it appears as an appropriate tool for analysing the empirical hardness of the WDP for MMUCAs. Our purpose will be to employ to: (i) build a model to predict the running time of the solver outlined in section 2; and (ii) to analyse the factors that make the WDP for MMUCAs hard. Before we apply the methodology (in forthcoming sections) to MMUCA, next we summarise its main steps[1]:

1. *Select an algorithm* as the objective of the empirical hardness analysis.
2. *Select an instance distribution*. To generate instances of the problem, it is eventually convenient to employ some artificial instance generator.
3. *Select features* of the problem instances. The values these features take on will be mapped to a predicted run-time. The features have to be good ones, avoiding uninformative features.
4. *Collect data*. Generate a good number of instances with the instance distribution selected at step 2. Then, solve the instances using the algorithm selected at step 1 and extract the features selected at step 3. Finally, divide the instances in three separate sets, namely one for training, one for validation, and one for testing.
5. *Build a model*. Choose a machine learning algorithm to learn to predict the running time required by the algorithm selected at step 1 to solve some problem instance characterised by the values of the features selected at step 3. Statistical regression techniques are the most convenient tool for this goal.

4 A Empirical Hardness Model for MMUCA

In this section we apply the methodology outlined in section 3 to build a model that predicts the running time of the solver outlined in section 2.

[1] We refer the interested reader to [4] for full details.

Step 1: Selecting an algorithm. As discussed in section 2, MMUCA WDPs can be solved via integer linear programming (ILP). We select an ILOG CPLEX implementation of the integer program outlined in section 2 (fully described in [5]), because it is the fastest algorithm reported in the literature, largely outperforming the solver in [1].

Step 2: Selecting an instance distribution. In order to generate instance distributions we resort to the artificial data set generator for MMUCA WDPs introduced in [7]. The algorithm takes inspiration on the structure of supply chains, whose formation has been identified as a very promising application domain for MMUCA [5,1]. A supply chain is composed of levels (or *tiers*). Each tier contains goods that are subsequently transformed into other goods within another tier in the supply chain. Within this setting, their generator allows to flexibly generate:

- *Supply chain structures with a varying number of levels.* Modelling from complex supply chains involving multiple levels (for example the making of a car) to simple supply chains involving a few parties.
- *Transformations of varying complexity.* Varying complexity on the goods involved in the input and output sides of the transformations in a supply chain.
- *Transformations representing different production structures.* The input and output goods from a transformation may come from different levels.
- *Bids per level.* Different bid distributions may appear at different levels, to control the degree of *competition* in the market.

Step 3: Selecting Features. In order to generate a list of features we start from the features described in [4] regarding the study of CAs. However, there are major differences between CAs and MMUCAs that lead to the list of features in table 2.

First of all, as formerly argued in [5] and [7], the *topological features* of the search space handled by the WDP matter. Indeed, while in [5] Giovannucci et al. observed that the order of the number of variables required by CCIP, the IP, is directly related to the size of the largest strongly connected component

Table 2. Groups of features

#Feature	Topological features
1-8	Node size statistics: average, maximum, minimum and standard deviation.
9-12	Node degree statistics: average, maximum, minimum and standard deviation.
13	Edge density: sum of all node degrees divided by the sum of the degrees in a complete graph.
14	Depth: The largest path in the SCC graph.
#Feature	**Problem size features**
15	Number of transformations.
16	Number of goods.
#Feature	**Price-based features**
17-20	Price statistics: average, max, min and standard deviation of prices.

(SCC), in [7] Vinyals et al. empirically observed that large SCCs (cycles) lead to high solving times. Therefore, we must consider the structural properties of the SCC graph (like the one in figure 1). Hence, we shall consider: inner features of SCCs (node size statistics), external features of SCCs (node degree statistics), general properties of the SCC graph (edge density and depth). Importantly, notice that these features were not considered when studying CAs in [4] because the topology of the search space for CAs is different.

Secondly, regarding problem size features, MMUCAs are again different from CAs because the former consider the notion of transformation. Therefore, we shall consider both the number of goods and transformations.

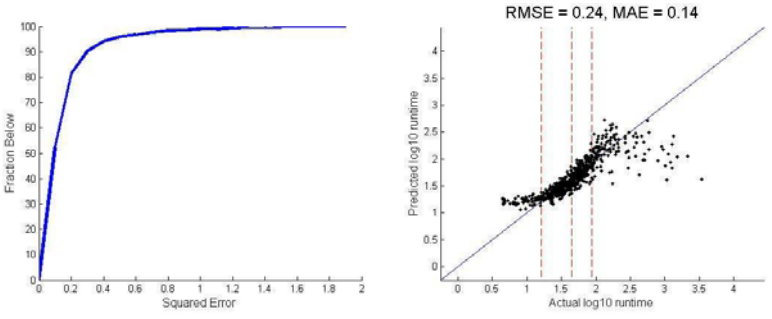
Finally, although Leyton-Brown et al. found in [4] that price-based features are not among the main factors driving the hardness of a WDP, we still consider them because although two WDPs may have similar SCC graphs, their optimal solutions might be different when considering different bid prices.

Step 4: Collecting Data. To collect data, we generated 2250 WDP instances with the generator in [7] after setting the probability of generating cycles ($p_b = 0.1$), the number of goods ($n_g = 20$), and the number of transformations ($n_t = 200$). Such parameters allow us to generate WDP instances whose solving times are acceptable. Moreover, the fact that we employ a probability distribution to generate cycles (parameterised by p_b) leads to WDP instances of varying complexity. This is because we observed in [7] that the solving time is sensitive to cycles in WDP instances. In other words, solving the WDP is costly when there are cycles. Such WDP instances may be output by the generator when the probability of generating cycles (p_b) is positive, as we set above[2].

Thereafter, once solved the WDP instances we solve them using CCIP, the integer program we have selected as a solver at step 1 above. Since eventually some WDP instances may take too long, we defined a maximum solving time (4500 seconds). If CCIP does not find any valid solution before the deadline, the instance is rejected and a new instance is generated and solved. In this way we avoid to deal with outliers in the empirical analysis. Once solved the problems, we extracted the values for the selected features described at the step 3 above. For each problems instance, the values of the features along with the solving time compose the data that will be subsequently employed to learn the model. Finally, we divided the data (features' values along with solving times) in three sets following the step 4 in the methodology described above: one for training, one for validation, and one for testing.

Step 5: Building Models. We decided to use linear regression (following the guidelines in [4]) to build a model that predicts the solving time of CCIP because: (i) it reduces the learning time; and (ii) it makes more intuitive to analyse and interpret the resulting models. We use as response variable an exponential model, applying the (base-10) logarithm to the CPLEX running times. As for error metrics, we do not apply the inverse transformation. In this way, we manage to

[2] Notice that some WDP instances may have no cycles, others may have several cycles, and cycles may encompass a varying number of transformations.



(a) squared errors (b) prediction

Fig. 2. Prediction scatter plot

uniformly penalize all the responses. We consider two error metrics: the squared-error metric (RMSE), because it is the most used in linear regression, and the mean absolute error (MAE), because it is more intuitive.

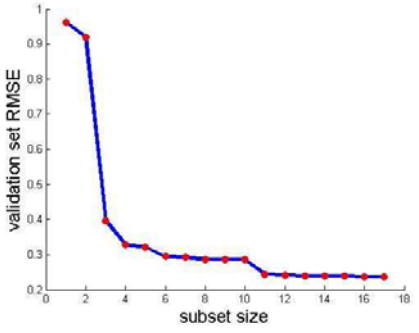
Figure 2(a) shows the cumulative distribution of squared errors on the test set. The horizontal axis represents the squared error, whereas the vertical axis corresponds to the cumulative distribution of squared errors on the test sets, namely the fraction of instances that were predicted with an error not exceeding the squared error in the x-value. Notice that the absolute error is greater than 1 (10 seconds) for only for 1% of the predictions.

Figure 2(b) shows a scatter plot of the predicted running time versus the actual running time in (base 10) logarithmic scales. In this case study, the RMSE is 0.24, whereas the MAE is 0.14. Because they indicate an average error of less than 1.8 seconds whenever the solving time ranges between 4.4 seconds and 4487 seconds, we conclude that the prediction model is acceptable. We also observe that most WDPs (actual run-times between 10 seconds and 100 seconds) are predicted with high accuracy, whereas predictions for WDPs with low solving times (easy WDPs) are pessimistic and predictions for WDPs with high solving times (hard WDPs) are optimistic. This occurs because the instance distribution generated at step 2 did not contain enough examples of easy and hard WDPs.

5 Analysing MMUCA Problem Hardness

In this section we analyse the MMUCA WDP hardness using the model produced in section 4 to assess the features that make the problem hard to solve.

As a first step, we assess the relationship between the *best* subset of features of a given size (ranging from 1 to 17) and the RMSE. In other words, a relationship between the features that minimise the RMSE (out of all the features in table 2) for each number of features. Figure 3(a) plots how the RMSE varies as the size of the subset of features increases. In order to obtain the best subset of features



Feature	Cost
average of all SCC node sizes	100.00
average of SCC node degree	28.26
SCC edge density	15.19
standard deviation of scc node size > 1	4.99
SCC depth	2.50

(a) Subset Size vs. RMSE (b) Cost of omission

Fig. 3. Linear Regression

for each subset size we employ a heuristic algorithm, the so-called forward-select, which begins with an empty set of features to progressively add one feature at each time. Forward-select adds as a new feature the one with the lowest RMSE when testing with the validation set. Based on the results in figure 3(a), we decided to analyse the model selecting five features because the RMSE slightly varies for more than five features.

We analyzed this five features' *cost of omission* in our model to evaluate the *importance* of each particular feature. The *cost of omission* for a particular feature is computed by analysing how its omission impacts on the RMSE: we train a model that omits the feature to subsequently compute the resulting increase on the RMSE. Figure 3(b) ranks the cost of omission, namely the impact of the hardness of the WDP, of the best five features. Several comments apply:

(1) The average of all SCC node sizes appears as the most important feature. Hence, the largest the SCCs, the harder the WDP. This result is in line with the theoretical results in [5].
(2) The second and the third places are the average of SCC node degrees and the SCC edge densities. Both features refer to an SCC node degree, namely to an SCC's number of children. Since the higher the number of children, the larger the number of potential solution sequences of the WDP, it is not surprising that features referring to SCC nodes' degrees have a high impact on the hardness of the WDP.
(3) The fourth position refers to SCCs whose size is greater than one, namely to SCCs containing cycles. In some sense, though less important than the top feature in the ranking, this feature is complementary to that one because it refers to the sizes of SCCs.
(4) The feature with the lowest cost of omission, the SCC graph depth, also influences the hardness of the problem because it indicates the maximum number of transformations that can compose a solution sequence, namely the maximum length of the solution sequence.

To summarise, complementary features 1 and 4 refer to the inner features of SCCs and refer to the impact of cycles on the hardness of the WDP. This is in line with the results in [7]. Furthermore, features 2 and 3 indicate that they are also important because they have a strong impact on the number of potential solution sequences to evaluate to solve the WDP. Finally, feature 5 is also relevant because it influences the lengths of those potential sequences. Therefore, unlike the analysis in [4] for CAs, our study shows that the features that most impact the hardness of the WDP for MMUCAs are all topological.

6 Conclusions and Future Work

In this paper we employed the methodology described in [4] to analyse the empirical hardness of CAs. With this methodology we obtained a model that successfully predicts the time to solve MMUCA winner determination problems. Therefore, we can effectively assess whether the WDP can be solved in time when there are constraints regarding the time to solve the WDP. Furthermore, we analyzed the hardness of the MMUCA WDP to learn that the topological features (of the SCC graph) are the ones that most impact the hardness of the problem. These results complement the theoretical results in [3]. We argue that specialised algorithms to solve the MMUCA WDP can benefit from this analysis. Regarding future work, we plan to complete our empirical study by analysing supply chains of varying sizes (in terms of number of goods and transformations).

References

1. Cerquides, J., Endriss, U., Giovannucci, A., Rodríguez-Aguilar, J.A.: Bidding languages and winner determination for mixed multi-unit combinatorial auctions. In: IJCAI, pp. 1221–1226 (2007)
2. Cramton, P., Shoham, Y., Steinberg, R. (eds.): Combinatorial Auctions. MIT Press, Cambridge (2006)
3. Valeria Fionda, G.G.: Charting the tractability frontier of mixed multi-unit combinatorial auctions. In: Proceedings of IJCAI 2009, pp. 134–139 (2009)
4. Leyton-Brown, K., Nudelman, E., Shoham, Y.: Empirical hardness models for combinatorial auctions. In: Cramton, et al. (eds.) [2], ch. 19, pp. 479–504
5. Giovannucci, A., Vinyals, M., Rodríguez-Aguilar, J.A., Cerquides, J.: Computationally-efficient winner determination for mixed multi-unit combinatorial auctions. In: Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-agent Systems, pp. 1071–1078 (2008)
6. Giovannucci, A., Cerquides, J., Rodriguez-Aguilar, J.A.: Proving the correctness of the CCIP solver for MMUCA, Tech. rep., IIIA-CSIC (2007)
7. Vinyals, M., Giovannucci, A., Cerquides, J., Meseguer, P., Rodriguez-Aguilar, J.A.: A test suite for the evaluation of mixed multi-unit combinatorial auctions. Journal of Algorithms 63, 130–150 (2008)