Università degli Studi di Bologna
DEIS

Exploring CP-IP based techniques for the bid evaluation in combinatorial auctions

Alessio Guerri Michela Milano

April 11, 2003

DEIS Technical Report no. DEIS-LIA-03-003 *LIA Series no. 64*

Exploring CP-IP based techniques for the bid evaluation in combinatorial auctions

Alessio Guerri [1] Michela Milano [1]

[1] *DEIS*
Università di Bologna
V.le Risorgimento, 2
40136 Bologna, Italy
{aguerri,mmilano}@deis.unibo.it

April 11, 2003

Abstract. Combinatorial auctions are now an important e-commerce application where bidders can bid on combination of items. The problem of selecting the best bids that cover all items, i.e., the winner determination problem (WDP), is NP-hard. The time constrained variant of this problem, considered in this paper, is the bid evaluation problem where temporal windows and precedence constraints are associated to each task in the bid. Many approaches have been proposed for the winner determination problem, coming mainly from the Integer Programming (IP) community and recently from the multi-agent community, while the bid evaluation problem received less attention.

Surprisingly, the Constraint Programming (CP) community has almost never considered neither of the problems, while we believe that CP solvers or hybrid CP-IP solvers can provide an important contribution to the field. In particular, as soon as temporal side constraints are introduced in the problem. In this paper, we propose different algorithms based on CP, IP and CP-IP. We show that even the simplest pure CP based approach outperforms existing approaches. Since none of the algorithms developed in this paper dominates all the others, they are good candidates for algorithm portfolio design. Therefore, we identified a set of instance-dependent structural features that allow to select the best class of algorithms to apply. An interesting result achieved is that we found a correspondence between the standard deviation of the clustering coefficient and the performances of IP or CP based algorithms. We believe this is the first step toward an automatic algorithm portfolio selection.

Contents

1 Introduction

Business to business e-commerce applications are becoming more and more popular. Among them, auctions are an important way of allocating items among autonomous and self-interested agents. Items are not limited to goods, but can represent also resources and services.

Traditionally, auctions are aimed at selling or buying a single item; the auctioneer tries to maximize his/her profit if selling an item or minimize his/her outcome if buying an item. Since bidders make bids on a single item, it is easy to choose the best bid, i.e., the one providing the highest revenue. This kind of auction follows the *sequential auction mechanism*. However, it is difficult to bid in these auctions when more than one item is needed since one bidder can have preferences on bunches of items. In this case, a bidder should make hypothesis on what the other bidders will bid.

To partially solve the problem, the *parallel auction mechanism* has been proposed, where bidders can bid on a set of items simultaneously. Again, it is easy to choose the best bid by simply selecting the best one. A problem in parallel auctions can arise: it can happen that no bidding should start since all bidders wait for other bids to perform the best offer.

Recently, a third kind of auction mechanism has been proposed, the so called *combinatorial auctions*, see [13] for an overview. Among M items, bidders can bid on combinations of items, and associate a price for each combination. The auctioneer should solve the winner determination problem, i.e., he should choose the best bids that cover all items[1] at a minimum cost or maximum revenue. Clearly, the problem becomes combinatorial and deciding the best bids is now difficult. Thus, for a long time combinatorial auctions have not been considered an alternative to single or parallel auctions. However, recently, solving methods have became effective and combinatorial auctions have been considered an alternative. Practical approaches used to solve the problem have been proposed mainly based on Integer Programming techniques [10], Dynamic Programming [12], incomplete methods [15] and search algorithms [13] [14] using preprocessing techniques that can be seen as a form of propagation.

A variant of this problem is the so called bid evaluation problem for coordinated tasks. When the auctioneer should, for example, buy a set of services, he should also consider temporal constraints. Therefore, items in the bid are associated to a temporal window, a duration and are linked by precedence constraints. In this case, beside the winner determination problem, the auctioneer should maintain feasibility of the temporal constraints. To our knowledge, the only system that tackles this problem is MAGNET (Multi-Agent Negotiation testbed) [2] and it is based on Integer programming and Simulated Annealing.

[1] Under the *free disposal* assumption some item could also be left uncovered.

Surprisingly, Constraint Programming (CP) has been very rarely used to solve these problems. To our knowledge, only one approach to the winner determination problem has been proposed [3], but we think CP can be widely used as an effective tool and language for modeling and solving combinatorial auctions. In particular, CP can be effective when additional constraints are introduced.

In this paper, we present and evaluate different approaches based both on CP and IP to the time constrained variant of the winner determination problem. The flexibility of CP allows the introduction of many other side constraints, like for example, constraints on competitor bidders, resource constraints, constraints limiting the risk and many other.

In this paper, we propose different algorithms: a pure CP algorithm (implemented with Depth First Search DFS and Limited Discrepancy Search LDS), two approaches based on pure IP and one hybrid approach merging CP and IP (again implemented both with DFS and LDS) to the bid evaluation problem. We show that even the simplest approach we developed based on pure CP outperforms the one presented in MAGNET (this software has been kindly provided by the authors J. Collins and M. Gini).

Since none of the algorithms dominates all the others (but those based on DFS), they are good candidates for algorithm portfolio design. In algorithm portfolios [6] different algorithms are run in parallel or in sequence. An additional possibility for algorithm portfolio is to have an automatic selection strategy for the best algorithm on a given instance. Therefore, we tried to select among the set of instance-dependent structural features proposed in [9] the ones that allow to select the best algorithm. An interesting result achieved is that the standard deviation of the clustering coefficient provides clear indication if to use an IP or a CP based algorithm. We believe this is the first step toward an automatic algorithm portfolio selection.

2 Problem Description

In combinatorial auctions, we have two major combinatorial optimization problems. The bid evaluation and the winner determination problem. In the winner determination problem the auctioneer has to find the set of winning bids according to one or more optimization criteria. In the bid evaluation problem, beside a winner determination problem, we have time windows and temporal constraints to be taken into account. We mainly consider an auction where the auctioneer buys a set of tasks (services) which are sequenced by temporal precedence constraints and are associated to temporal windows and durations.

2.1 Bid evaluation problem

We have different variants of combinatorial auction. In this paper, we consider **single unit reverse auctions** where the auctioneer wants to buy a set M of items (services) minimizing the cost. In single unit auctions items are distinguishable while in multi unit auctions there are several unit of each item. The auction we consider is called reverse since the auctioneer has to buy items, while in traditional auctions items are sold.

Each bidder j ($j = 1..n$) posts one or more bids. A bid is represented as $B_j = (S_j, Est_j, Lst_j, D_j, p_j)$ where a set $S_j \subseteq M$ of services is proposed to be sold at the price p_j. Est_j and Lst_j are lists of earliest and latest starting time of the services in S_j and D_j their duration.

The bid evaluation problem can be seen as the combination of two problems: the winner determination problem WDP and the Simple Temporal Problem [4] since the auctioneer has a set of temporal constraints (precedence constraints) that define the feasibility of the assignments computed by the winner determination problem.

2.1.1 Winner determination problem: ILP model

The integer linear model of the winner determination problem is the following: we have decision variables x_j that take the value 1 if the bid B_j is winning, 0 otherwise.

$$\min \sum_{j=1}^{n} p_j x_j \tag{1}$$

$$\text{subject to} \quad \sum_{j|i \in S_j} x_j = 1 \quad i = 1..m \tag{2}$$

$$x_j \in \{0, 1\} \tag{3}$$

The objective function minimizes the total cost which is computed as the sum of prices of winning bids. Constraints (2) state that the number of winning bids containing the same item should be equal to one. This means that all items should be covered and each item should be covered by exactly one bid. This model is the formulation of a set partitioning problem that is a structured, well known and widely studied problem in the Operations Research community [1].

An important assumption that can be done in combinatorial auctions is that of *free disposal*. In this case, not all items should be covered. Thus, if free disposal is assumed, symbols $=$ in constraints (2) in the above model are transformed in $\leq$. In this case, the problem is equivalent to a set packing problem, where not all items should be covered. The only constraint imposed on winning subsets of bids is that two subsets should have an empty intersection.

2.1.2 Winner determination problem: CP model

Auctions can be easily modelled in Constraint Programming. We have a set of m variables $X_1, \ldots, X_m$ representing the items to be bought. Each variable X_i ranges on a domain containing the bids mentioning item i. We have also redundant 0-1 variables representing bids, $Bid_1, \ldots, Bid_n$. Constraints are the following: we recall that $B_j = (S_j, Est_j, Lst_j, D_j, p_j)$. If an item is taken from the bid B_j, all other items should be taken from the same bid.

$$X_i = j \rightarrow X_k = j \;\; \forall k \in S_j$$

If a bid is not chosen for one item, then all items in the bid should be taken from other bids.

$$X_i \neq j \rightarrow X_k \neq j \;\; \forall k \in S_j$$

Another important constraint that can trigger an effective propagation is a variant of the global cardinality constraint [11] whose propagation can be explained as follows: if the bid B_j is chosen as winning, the number of variables X_i that take the value j is exactly the cardinality of the set S_j. Otherwise, if the bid B_j is not chosen as winning, that number is 0. We recall that the global cardinality constraint can be written as $gcc(Var, Val, LB, UB)$ where Var are variables, Val are values and LB and UB are the minimum and the maximum number of occurrences of each value in Val assigned to Var. The constraint holds iff the number of occurrences of each value in Val assigned to Var is indeed within LB and UB. In our case, the constraint could be specialized as follows: $gcc(X_i, j, 0, |S_j|)$, where X_i is the set of variable representing items contained in the bid B_j, and $|S_j|$ is the cardinality of the set of items contained in the bid B_j.

For combinatorial auctions, we need a variant of this constraint that, instead of having a lower and upper bound on the number of occurrences, has a set of alternative values, in our case these values are two: 0 if the bid is not a winning one or $|S_j|$ otherwise. Thus, instead of having a lower and upper bound, we need a domain variable whose domain contains two values: 0 and $|S_j|$.

We have also constraints on binary variables Bid_i. First, for each item i we collect all bid-variables containing the item i, i.e., $B_j^i = \{B_j | i \in S_j\}$. The number of bid-variables that take the value 0 should be $|B_j^i| - 1$.

In addition, we have constraints linking items and bids.

$$Bid_i = 1 \rightarrow X_k = i \;\; \forall X_k \in S_i$$
$$X_k = i \rightarrow Bid_i = 1$$
$$Bid_i = 0 \rightarrow X_k \neq i \;\; \forall X_k \in S_i$$
$$X_k \neq i \rightarrow Bid_i = 0$$

2.1.3 Introducing temporal constraints

A second problem in combinatorial auction is that of bid evaluation for coordinated tasks in a population of self-interested independent agents, obtained by adding to the winner determination problem a set of time windows and temporal constraints. We describe here the problem of bid evaluation [2] starting from a single unit reverse auction described before, but any other type of auction can be considered as well.

The auctioneer has to perform a job that involves a set of coordinated tasks within a limited time window. The resources needed to perform these tasks should be acquired from self-interested suppliers (the bidders). Thus, the auctioneer should decide for each item which is the time window associated with it. The larger the windows, the highest the probability that item could be provided within the window at a low price, but also the higher the risk that the items cannot be allocated respecting the precedence constraints. In general, the auctioneer leaves the bidder free of proposing the windows associated to each task, but he/she should check the feasibility of temporal constraints. Thus, each supplier provides a bid j where associated to each item $i \in S_j$ there is a temporal window $[Est_{ij}, Lst_{ij}]$ where the item should start and a duration D_{ij}.

The auctioneer should select suppliers that cover all tasks to be performed, ensuring temporal constraints, at the minimum cost.

Therefore in the models described before we should introduce temporal constraints. In the IP model we introduce variables $Start_{ij}$ associated to each item i in bid j. These variables range on the temporal windows $[Est_{ij}, Lst_{ij}]$ for item i taken from bid j. For each pair of items t_i and t_j linked by a precedence constraint, we find all pairs of bids b_i and b_j containing that items; if S_{b_i} and S_{b_j} have an empty intersection we compute $Est_{t_i b_i} + D_{t_i b_i} - Lst_{t_j b_j}$, where $D_{t_i b_i}$ is the duration of t_i in bid b_i. In case the result is positive, that is domains of $Start_{t_i b_i}$ and $Start_{t_j b_j}$ do not contain any pair of values that could satisfy the precedence relation, we introduce the constraint $x_{b_i} + x_{b_j} \leq 1$, which prevents both bids from appearing in the same solution; otherwise, if the result is zero or negative, we introduce the constraint

$$Start_{t_i b_i} + D_{t_i b_i} - Start_{t_j b_j} + M(x_{b_i} + x_{b_j}) < 2M$$

where M is a large number. The term $M(x_{b_i} + x_{b_j})$ makes the constraint satisfied in the case where either $x_{b_i} = 0$ or $x_{b_j} = 0$.

Instead, in the CP model we introduced temporal variables $Start_i$ associated to each item i. These variables range on the union of all temporal windows $[Est_{ij}, Lst_{ij}]$ for item i taken from all bids j mentioning i. In addition, if two items i and j are linked by a precedence constraint, then the constraint

$$Start_i + D_i \leq Start_j$$

is introduced, where D_i is a domain variable containing the durations of i in all bids mentioning i. Obviously, variables $Start$ and X are connected by constraints, in the sense that, if a value b is removed from the domain of a variable X_i, all values in the domain of $Start_i$ that are supplied by no other bid already contained in the domain of X_i are removed too.

An additional constraint which could be imposed when precedence relations are part of the problem is the Precedence Graph [7] that allows to represent and propagate temporal relations between pairs of activities as well as to dynamically compute the transitive closure of those relations. The role of the precedence graph is to incrementally deduce new edges given the ones already posted on the graph. This constraint has not been used in this paper.

2.2 Implemented algorithms

We implemented four algorithms (plus two variants) to solve the bid evaluation problem. Two algorithms are based on the IP model: the first (referred in tables to as IP) is a traditional complete solver implementing Branch and Bound based on linear relaxation, while the second (referred in tables to as LR+IP) is an incomplete approach that solves the linear relaxation of the problem, then ranks the variables according to their shadow price, and finally solves the IP problem considering only the first $p\%$ variables, where p is a parameter to be experimentally tuned.

One algorithm is based on the pure CP model presented above. Starting from the same model, one variant explores the search tree with Depth First Search CP-DFS and one variant with Limited Discrepancy Search CP-LDS. In both cases, the heuristic used to select the variable value is the bid-price divided by bid-size, that is $p_i/|S_i|$.

The last approach, referred to as LR+CP in tables, performs an indeed quite loose but effective integration. Starting from a CP model, we solve a linear relaxation at the root node and we order values based on shadow prices. Again we have two variants, one based on DFS (LR+CP-DFS) and one based on LDS (LR+CP-LDS).

3 Previous Approaches

Many approaches have been proposed for the pure winner determination problem.

A Dynamic Programming (DP) approach has been proposed in [12]. DP explores the space of exhaustive partitions by proceedings systematically from small sets to large ones. An interesting feature of this approach is that its complexity is independent from the number of submitted bids. Thus, if the number of items is small, DP can be the technique of choice.

In [10] IP is used to solve a winner determination problem. The paper provides an interesting insight on shadow prices, on special structured problems for which the LP solution is integral and on lower and upper bound computation.

Several works on search algorithms have been proposed. The one by Sandholm [13] is worth mentioning. Beside an effective algorithm, this paper provides an interesting overview on existing approaches. The algorithm proposed allows all combinations of items, produces an optimal solution relying heavily on the fact that the bids are sparse. During search a cumulated cost is maintained and always compared with the best solution found so far. In the search tree, every relevant partition of bids is represented exactly once by a path from the root node to a leaf. The algorithm uses two parallel search trees explored with a IDA* strategy and pre-processing techniques to cut some branches.

A similar approach to this one is that called CASS [5]. It is a branch and bound algorithm that uses a particular method for the search strategy, called *binning* method to improve the average case speed of child generation.

Maybe the most efficient algorithm for winner determination has been proposed by Sandholm et al. [14]. It is called CABOB - Combinatorial Auctions Branch On Bids. Its main structure is a depth-first branch and bound search that branches on bids. The algorithm exploits a data structure called bid graph and a number of pruning techniques based on the continuous relaxation of the problem and the dual problem. In particular, it exploits the dual problem to compute shadow prices, i.e., upper bounds on prices of single items.

Several incomplete approaches have been proposed. For instance a greedy algorithm has been developed in [10]. However, one of the most interesting algorithm is one exploiting Limited Discrepancy Search [15] which achieves solutions that achieve the 99% of the optimal solution in the 1% of computing time.

As for as the bid evaluation problem is concerned the only approach to our knowledge is that proposed by [2]. In that paper, the MAGNET system is described exploiting a ILP approach to the bid evaluation problem starting from a single unit reverse auction. In addition a Simulated Annealing (SA) variant of the algorithm is proposed achieving better results. Beside model constraints, other implied constraints are used in the preprocessing phase to restrict the search space. Note that these preprocessing are a simple form of constraint propagation which can be easily implemented in a CP solver, not only in preprocessing but during the whole search tree. The comparison between MAGNET and the pure CP approach we developed is given in section 4.1.

4 Experimental results

4.1 Comparing CP and MAGNET

To show the effectiveness of Constraint Programming for the bid evaluation problem, we firstly compared the pure CP algorithm we developed and MAGNET [2] on instances generated using MAGNET itself.

Tasks	Bids	Opt	Best SA	Time (ms)				Failures	
				M-IP	**SA**	**DFS**	**LDS**	**DFS**	**LDS**
5	15	9508	9508	70	30	30	30	19	17
5	15	11384	11384	60	50	10	10	2	2
5	15	8496	8496	60	40	30	30	16	3
5	15	9622	9622	71	30	10	10	6	8
5	15	10920	10920	141	311	10	10	1	1
5	15	12319	12319	60	10	10	10	3	3
5	15	12979	12979	70	30	11	10	6	6
5	15	10333	10333	90	40	10	10	3	8
5	15	10311	10311	70	10	10	10	4	7
5	15	6624	6624	80	10	10	10	2	3
10	35	17653	17653	160	2100	10	10	8	8
10	35	19115	22927 (83%)	140	1812	60	40	59	44
10	35	15318	15318	160	1532	20	10	5	5
10	35	17297	17297	90	831	10	10	4	4
10	35	15317	16532 (92%)	80	1933	10	10	0	0
10	35	19758	19758	100	1582	421	40	220	27
10	35	15865	17005 (93%)	150	1352	10	10	4	4
10	35	17795	18059 (98%)	100	2085	20	20	7	7
10	35	16492	16492	281	1993	10	10	4	4
10	35	15172	15172	100	1091	20	20	6	6
10	35	13815	19063 (72%)	-	2093	20	30	4	4
10	35	14107	16746 (84%)	-	4186	12458	490	99300	2543
10	35	17519	20643 (85%)	-	1131	6609	70	57486	398
10	35	14088	18037 (78%)	-	3265	761	51	16414	230
10	35	19468	22521 (86%)	-	1873	1372	40	11119	138
10	35	16065	20862 (77%)	-	2173	10	10	1	1
10	35	15274	17994 (85%)	-	3245	341	30	2148	76
10	35	12106	14031 (86%)	-	3175	20	10	20	17

Table 1. *Results on instances generated by MAGNET*

We ran our experiments on a 800 Mhz Intel Pentium 3 (with 256MB RAM). We considered four kinds of MAGNET instances: the first, very easy to solve, with 5 tasks and 15 bids on average; the second with 10 tasks and 35 bids; the third with 10 tasks and 100 bids; the last, very hard to solve, with 20 tasks and 400 bids. Results for the first three kinds of instances are shown in Table 1, while results for the last kind are shown in Table 2.

For each instance set we used MAGNET implementing Simulated Annealing (column SA) and, when possible, MAGNET using Integer Programming (column M-IP) and our CP algorithm implementing Depth First Search (column DFS) and Limited Discrepancy Search (column LDS) as search strategies. Search times are in milliseconds and for DFS and LDS we compare also the number of failures occurred. For MAGNET generated instances we keep all parameters constant except for the task-count and bid-count values, as done in [2].

In the first two sets of experiments (5 tasks and 15 bids and 10 tasks and 35 bids), the M-IP approach always finds the optimal solution, while in the third (10 tasks and 100 bids) it does not provide the optimal solution within 15 minutes. In the first set also SA provides the optimal solution, while, in the second set, it provides the optimal solution only in the 60% of the cases. In the third set of experiments it never provides the optimal solution. Our approach always finds the optimal solution in all instance sets.

In Table 1 the Best SA column reports the best solution found by SA and, if it is not optimal, we report the percentage w.r.t. the optimal solution.

Table 2 shows results for the fourth instance set: 20 tasks and 400 bids on average. These instances are very hard to solve so it is not possible to find the optimal solution with none of the approaches. All algorithms were stopped after 15 minutes. In this table, we compare SA, DFS and LDS since M-IP does not compute any solution in the time limit. The letter k in the failure column means 10^3 while M means 10^6. It is worth noting that our solutions are, on average, 30% better than those produced by MAGNET. Moreover, the time to produce the best solution is in general considerably lower than 15 minutes. In the Best solution columns the best obtained result is reported, while the other two are described as percentage w.r.t. the best solution found.

Best solution			**Search time (ms)**			**Failures**	
SA	**DFS**	**LDS**	**SA**	**DFS**	**LDS**	**DFS**	**LDS**
55%	88%	19490	10875	30	110	30	454
72%	97%	26859	11256	30	40	33	29
66%	99%	23887	10164	718814	60	2.5M	67
-	96%	28036	-	30	320061	39	1.1M
24467	98%	24467	8062	30	40	10	10
69%	75%	26833	6960	40	9193	64	28991
63%	98%	25997	7871	1272	739534	4157	1.3M
56%	24789	96%	9393	476295	113273	2.09M	249k
68%	24718	99%	3695	30	57753	28	152k

Table 2. *Results on instances of 20 tasks and 400 bids*

The relative quality of SA with respect to LDS and DFS is also depicted in Figure 1 where we show the trend of the solution quality for hard instances with SA, DFS and LDS.

4.2 Experimental results of CP, IP and hybrid approaches

In this section, we provide results on instances generated using both MAGNET and CATS, a suite for generating realistic auction instances, [8]. CATS instances

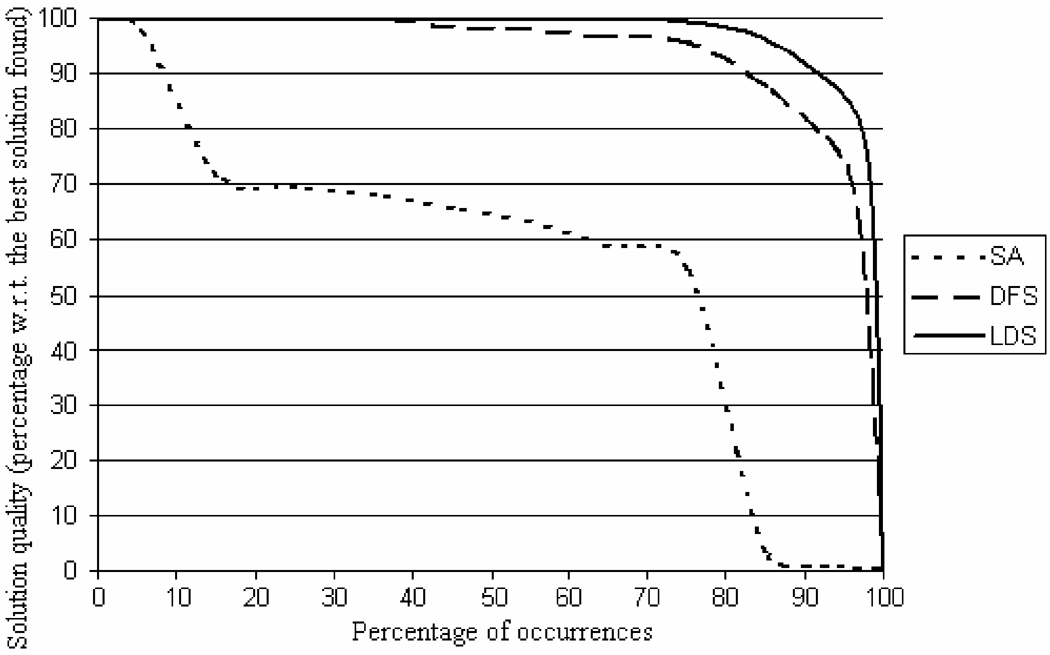

Figure 1. *Trend of the solution quality for instances of 20 tasks and 400 bids*

are more realistic, allowing to set an higher bid-size and bid-size variability. We generated problems with 10, 15, 20 and 30 tasks and with a number of bids growing from 40 up to 1000, using parameter settings as shown in [8].

We ran these experiments on a 2.4GHz Pentium 4 with 512Mb RAM.

In Table 3 we present results for all algorithms described in section 2.2 (the algorithm acronyms are used in Tables) except for CP-DFS since it is not able to find the optimal solution in 15 min. time limit. The column T/b shows the mean tasks-per-bid value. The last column represents the percentage of variables considered by the IP solver according to the shadow price rank. In these experiments we could not compare our algorithms with MAGNET because MAGNET was not able to provide any solution within the time limit.

We can observe that, depending on the instance features, some instances are best solved by CP-based approaches (namely CP-LDS, LR+CP-DFS and LR+CP-LDS), while the others by IP approaches (namely IP and LR+IP); in particular, the higher the number of tasks-per-bid feature, the best the performances of CP approach. This result will be best formalized in next subsection.

Table 4 shows results on very hard instances of 20 tasks and 400 bids in the first part of the table, and 1000 bids in the last. These instances have a low mean tasks-per-bid value, that is respectively 1.15 and 1.12. In the columns named Best solution

Tasks	Bids	T/b	Search time (ms)					CR%
			CP-LDS	LR+CP-DFS	LR+CP-LDS	IP	LR+IP	
15	500	2.59	8539	15151	11475	9244	1772	40
15	500	4.29	428	1866	613	16740	1765	20
15	500	7.57	874	8809	720	12022	9782	30
20	400	2.76	4118	6587	5898	4493	357	40
20	500	4.69	1794	1497	1694	-	56822	55
20	800	4.58	16453	18547	9334	359437	8624	40
20	1000	4.49	3085	9460	2082	-	9319	20

Table 3. *Comparison between algorithms*

Best solution				Search time (ms)				CR%
CP	LR+CP	IP	LR+IP	CP	LR+CP	IP	LR+IP	
-	32956 (81%)	26554	26554	-	15709	329	265	60
26708 (93%)	28497 (86%)	24789	24789	9781	6593	500	266	40
23897 (99%)	23897 (99%)	23887	23887	8609	765	406	282	60
25285 (98%)	25274 (98%)	24718	24718	15	15203	282	141	65
28036	28036	28036	28036	13688	17063	1610	281	65
-	-	32745	32745	-	-	687	360	95
24852 (98%)	24442 (99%)	24424	24424	16	31	312	125	40
19490	19490	19490	19490	31	329	259	234	45
27143 (99%)	26833	26833	26833	1688	3312	532	172	75
27592 (97%)	27458 (98%)	26859	26859	32	23875	188	109	1
29052 (79%)	29052 (79%)	23109	23109	47	47	92541	891	35
-	-	23007	23007	-	-	75412	35250	60
23685 (94%)	23685 (94%)	22363	22363	15016	16750	4766	532	25
25916 (99%)	25988 (99%)	25612	25612	7015	5000	2078	469	25
23982 (98%)	23560	23560	23560	31	2000	4031	297	1
-	-	22812	22812	-	-	95451	13922	70
23989 (94%)	22505	22505	22505	1218	47	641	156	5
-	26503 (97%)	25819	25819	-	31081	40000	640	50
24365 (97%)	24134 (98%)	23572	23572	19109	23937	63210	641	50
-	-	23847	23847	-	-	8019	1203	45

Table 4. *Comparison between algorithms on instances with 20 tasks and 1000 bids*

we report for each algorithm the best solution found and, eventually, the percentage w.r.t. the optimum. Both IP-based approaches always find the optimal solution. Since the LDS approach always dominates the DFS one, results obtained using DFS are not shown and in table 4 the CP-LDS is referred to as CP and LR+CP-LDS is referred to LR+CP. We can observe that, in the second set of instances, CP-based approaches provide a solution only 6 times over 10, and only twice it is the optimal one.

Finally, we ran experiments on hardest problems, with 30 tasks, 1000 bids and a growing tasks-per-bid value. We found that only the IP-based approach provided results, and that these results get worse at the rising of the tasks-per-bid value. Indeed, sometimes the CP-based approach provided a solution, but search time was so high and solution so far from the best that these values are not shown in the table. In Table 5 we show the mean search time for both complete and incomplete IP solver (i.e., IP and LR+IP). For the incomplete approach, the percentage of variables considered is reported in the last column. Only in the first 3 instance sets it was possible to prove optimality, while values in remaining rows refer to the time for providing the best solutions. All algorithms were stopped after 15 minutes.

Tasks for Bid	**Search time (ms)**		**CR%**
	IP	**LR+IP**	
1.40	36328	1235	70
2.49	242281	3500	45
3.34	9̃00000	6975	25
4.62	-	19681	25
6.52	-	25969	30

Table 5. *Comparison between IP algorithms on instances with 30 tasks and 1000 bids*

4.3 Problem structure analysis

In this section, we are interested in identifying a set of instance-dependent parameters that help in determining the best algorithm to solve the instance itself. We analyzed the structure of the instances trying to extract one or more features providing an indication on the best solving algorithm.

As mentioned in the previous section, tasks-per-bid values discern algorithms' quality, but those values depend from task and bid number, so we need to find a more accurate parameter. Starting from the notable classification in [9], we extracted from each instance the 25 features described in the paper. We refer to the bid graph, where each node represents a bid and each edge stands between two bids if there is one or more constraints containing that bids.

An interesting result achieved is that we noticed a correspondence between the standard deviation of the Clustering Coefficient in the bid graph and the experimental results. The Clustering Coefficient is a measure of the *local cliqueness*. Typically, in our instances this value ranges from 0.02 e 0.2, and each time it is greater than 0.09, the IP-based approach is preferable to the CP-based one. It is worth noting the fact that, if this value is close to 0.09 both approaches have satisfactory behaviors. For the instances considered it is a *systematic* result, but, unfortunately, for larger instances this feature extraction takes too much time.

Therefore, we looked for a similar but easy to compute parameter. We observed that there is a correspondence between this value and the Edge Density in the bid graph. The Edge Density is the ratio between the number of edges in the graph and the number of edges in a complete graph with the same number of nodes. This value can range from 0 to 1, and we observed that it is in inverse proportion with the standard deviation of the Clustering Coefficient.

We have identified three significant ranges for the Edge Density: when it is lower than 0.5, the Clustering Coefficient is always greater than 0.09 (thus the IP-based approach is preferable); when it is greater than 0.75, the Clustering Coefficient is always lower than 0.09 (thus the CP-based approach is preferable); in between we do not have a clear indication of the Clustering Coefficient and therefore on the preferable approach. In this case, we recompute the edge density allowing multiple edges between the nodes (when more than one constraint is present among them). If the new edge density is significantly greater than the previous one, the CP-based approach is to be preferred. If the new value remains quite unchanged the best approach is IP.

Once chosen the best algorithm we can further refine our decision: if CP is the selected approach, we can use two different value selection criterion: one based on the price CP-LDS (or CP-DFS) and one based on shadow prices LR+CP-LDS (or LR+CP-DFS). We did not identify any systematic way to decide the value selection criterion, but we noticed that, when CP approach is preferred to IP, search times for these two algorithms are often similar.

If the LR+IP approach is the technique of choice, we have to tune the percentage of variables to be considered. We did not find a systematic correspondence between those choices and any feature we calculated, but from the L_1 and L_∞ norms of the integer slack vector we can often find out a good superior boundary to the percentage of variables to be considered using IP approach. The integer slack vector starts from the solution vector of the linear relaxation of the problem, and replaces each component x_i with $min(|x_i|, |1 - x_i|)$. These norms are in a way a measure of how fractional is the linear relaxation solution; we noticed that, in general, the higher the norms, the higher the percentage of variables to include to obtain the best solution. Values in the columns named $CR\%$ in experimental results presented.

5 Conclusions

In this paper we have proposed a set of CP and IP based algorithms for the bid evaluation problem for coordinated tasks. We have described the underlying models and we have shown that even the simplest CP based algorithm developed outperforms existing Integer Programming based approaches. Since none of the algorithm developed dominates all the others on all instances we believe they are good candidates for algorithm portfolio. Therefore, we provided a preliminary instance structure analysis to identify easy-to-calculate features for guiding algorithm selection.

Acknowledgement

This work was supported by the SOCS project, funded by the CEC, contract IST-2001-32530. The information provided is the sole responsibility of the authors and does not reflect the Community's opinion. The Community is not responsible for any use that might be made of data appearing in this publication.

We would like to thank Andrea Lodi and Andrea Roli for useful discussion. In addition, we warmly thank Maria Gini and John Collins for providing the MAGNET software and for their assistance in using it.

References

[1] R. Borndorfer, Aspects of Set Packing, Partitioning, and Covering, Shaker Verlag, Aachen, Germany, 1998.

[2] J. Collins and M. Gini, An integer programming formulation of the bid evaluation problem for coordinated tasks, *Mathematics of E-Commerce*, Springer-Verlag, 2001.

[3] R. Menke and R. Dechter, An implementation of the Combinatorial Auction Problem in ECLIPSE, Technical Report, Irvine University of California, 2000.

[4] R.Dechter, I.Meiri, J.Pearl, Temporal Constraint Networks, *Artificial Intelligence*, Vol. 49, (1991), pp. 61-95.

[5] Y. Fujishima, K. Leyton-Brown and Y. Shoham, Taming the Computational Complexity of Combinatorial Cuctions: Optimal and Approximate Approaches, *Proc. IJCAI99*, 1999.

[6] C. Gomes, B. Selman, Algorithm portfolio design: theory vs. practice, *Proceedings of the Workshop on Uncertainty in AI*, UAI97, 1997.

[7] P. Laborie. Algorithms for propagating resource constraints in A.I. planning and scheduling: Existing approaches and new results. *Artificial Intelligence*, Vol 143(2):151-188, 2003.

[8] K. Leyton-Brown, M. Pearson and Y. Shoham, Towards an Universal Test Suite for Combinatorial Auction Algorithms, *Proc. EC00*, 2000.

[9] K. Leyton-Brown, E. Nudelman and Y. Shoham, Learning the Empirical Hardness of Optimization Problems: The Case of Combinaorial Auctions, *Proc CP02*, 2002.

[10] N. Nisan, Bidding and allocation in combinatorial auction, *Proc. EC00*, 2000.

[11] J. C. Regin, Generalized Arc Consistency for Global Cardinality constraint, *in Proc. AAAI96*, 1996.

[12] M.H. Rothkopf, A. Pekec and R.M.Harstad, Computatonally managable combinatorial auctions, *Management Sci.* 44(8), 1998.

[13] T. Sandholm, Algorithm for optimal winner determination in combinatorial auction, *Artificial Intelligence*, Vol 135, 2002.

[14] T. Sandholm, S. Suri, A. Gilpin and D. Levine, CABOB: a fast optimal algorithm for combinatorial auction, *Proc. IJCAI2001*, Morgan Kaufmann, 2001.

[15] Y. Sakurai, M. Yokoo, K. Kamei, An efficient approximate algorithm for Winner Determination in Combinatorial Auctions, *Proc. EC00*, 2000.