Selezione basata sulla struttura dell'istanza in un Algorithm Portfolio

Instance structure-based Portfolio Selection

Alessio Guerri & Michela Milano

SOMMARIO/*ABSTRACT*

Molti problemi combinatori hanno una struttura complessa che impedisce di scegliere un singolo algoritmo in grado di dominare gli altri su tutte le istanze. In questi casi é quindi necessario progettare un portfolio di algoritmi che contenga piú approcci per lo stesso problema. Come possiamo scegliere tra gli algoritmi quando tutte le istanze condividono la stessa struttura? Questo articolo rappresenta un primo tentativo di risposta a questa domanda usando tecniche di apprendimento automatico, nello specifico gli alberi decisionali, per selezionare il miglior approccio data l'istanza da risolvere. Mostreremo come le caratteristiche strutturali dell'istanza forniscano una chiara indicazione del miglior algoritmo da usare. Abbiamo approfonditamente testato la nostra congettura nel contesto delle aste combinatorie e in particolare sul Bid Evaluation Problem. Mostreremo come, per il problema analizzato, l'albero decisionale ottenuto identifichi il miglior approccio nel 90% delle istanze usando soltanto un insieme ridotto di caratteristiche dell'istanza stessa. Crediamo che questo risultato possa essere esteso a tutta una serie di problemi combinatori.

Many large scale combinatorial problems have a complex structure that prevents a single algorithm and a single technology to outperform all the others on all instances. Therefore algorithms portfolios can be defined embedding more that one approach to the same problem. How do we choose among algorithms when all instances share the same problem structure? This paper is a first attempt to answer this question by using a machine learning technique, namely decision trees, for selecting the best approach given the instance to be solved. We will show that structural instance features provide a clear indication on the best approach to be used. We have extensively tested our conjecture on a case study: the Bid Evaluation Problem in combinatorial auctions, a well known e-commerce application. We will show that, for the problem considered, the computed decision tree identifies the best approach for the 90% *of the instances by using a small set of instance features. We believe this result can be extended to a large variety of combinatorial problems.*

Keywords: Combinatorial Auctions, Constraint Programming, Integer Programming, Decision Trees, Instance Structure

1 Introduction

Large scale combinatorial optimization problems do not have a clear structure, may present many side constraints, and may include subproblems. Different instances of the same problem can have different characteristics and structure. In these cases, it is unlikely that a single algorithm can deal with them adequately. Thus, we can build an algorithm portfolio and try to select the best strategy given the instance to be solved. **Our conjecture is that the selection of the best strategy should be based on the instance structure.**

The structure of an instance has been recently widely studied (see for example [1], [6], [15], [16], [17]) since it is considered an important parameter for explaining the algorithm behavior. The instance structure can be represented in different ways:

- through the constraint graph where nodes are variables and arcs are constraints;
- through a set of features (pairs attribute-value) extracted from the constraint graph;
- through some geometrical properties, e.g., the polyhedral structure.

In this paper, we consider the instance structure based on a set of pairs attribute-value derived by the constraint graph. This representation is indeed more concise (and therefore less expressive) than the one based on the whole graph, but, as we will show, it is representative of the instance structure as well.

To test our conjecture we used a combinatorial optimization problem as a case study, the Bid Evaluation Problem (BEP) in combinatorial reverse auctions where a set of tasks (say, services) should be bought by an auctioneer from a set of self interested agents performing bids containing more than one item. The auctioneer should cover all tasks at a minimum cost. Each task has an associated time window and temporal constraints with other tasks. Note that the auctioneer does not communicate temporal constraints to bidders, but a posteriori checks that the temporal windows provided by bidders are consistent with constraints.

We describe here a simple example of a BEP, where the auctioneer wants to buy 3 tasks, t_1, t_2 and t_3, minimizing the total cost. A precedence constraint is imposed, stating that t_3 must be executed after the execution of both t_1 and t_2 is completed. Table 1 shows some bids that can be received for this auction. Some qualitative considerations follow:

- Each bidder gives a single price for a bundle of tasks.
- Each bidder provides an early start (**eStart**), a late start (**lStart**) and a duration (**Dur**) for each task individually.
- Bid b_1 must be a winning one because it only proposes tasks t_1 and t_3.
- Bids b_2 and b_3 cannot both be accepted because they both ask for task t_2. Each task must be covered by exactly one bid.
- Bids b_1 and b_2 cannot both be accepted, because the precedence relation $t_2 \prec t_3$ would be violated. This because the earliest time b_2 could complete t_2 is 280, while the latest time b_1 could start t_3 is 150.
- In the best solution for this problem the winning bids are b_1 and b_3. Task t_1 starts at 15, ends at 125 and is executed by b_1, task t_2 starts at 110, ends at 230 and is executed by b_2, task t_3 starts at 230, ends at 325 and is executed by b_1. t_3 starts at 230 and not at 225 (the early start time proposed by b_1) because the execution of t_2 ends at 230 and t_3 must start after the end of t_2. The total cost is 590.

Bid	**Cost**	**Tasks**	**eStart**	**lStart**	**Dur.**
b_1	300	t_1	15	30	110
		t_3	225	250	95
b_2	150	t_2	140	160	140
b_3	290	t_2	110	135	120

Table 1: Example of a BEP

The BEP has an interesting structure: it contains as sub-problem the well known Winner Determination Problem (WDP), where no temporal windows and constraints are considered. The WDP is a set partitioning problem, a well known and structured problem which has been successfully faced by Mathematical Programming techniques and not by Constraint Programming techniques (see for example [7]). However, as soon as temporal constraints are introduced, the structure is lost and there is not a single technique that best solves all instances, but depending on the *instance structure* one approach dominates the other.

For the BEP, in fact, both Constraint Programming (CP) and Integer Programming (IP) can successfully be used. Furthermore, once the basic strategy has been chosen, that strategy can often be fine-tuned through parameter setting as in IP, or through different search heuristics as in CP for example. The difference in performance between strategies and their variants can be significant from one problem instance to another.

In the example in Table 1, if we don't consider the temporal constraints, we obtain a WDP; in this case, the solution found for the BEP is still a feasible solution because the WDP is a sub-problem of the BEP, but now b_1 and b_2 is the optimal solution.

The purpose of this paper is to find an automatic algorithm portfolio selection based on the instance structure. Statistical methods have been applied for the WDP, [10] and [11]. The authors are interested in determining the intrinsic complexity of a problem with respect to a given algorithm (the IP CPLEX in the paper) by using regression to predict the empirical hardness of an instance. In [11], again using regression, the authors propose how to build an algorithm portfolio in an effective way exploiting an *hardness model* and perform experiments on three well known algorithms for the winner determination problem.

Here we apply a machine learning technique, decision trees, for selecting the best algorithm. The *training set* has been built by extracting, for each instance, a set of 25 features (as suggested by [10]) and the corresponding best algorithm. The training set is provided as input to a well known and widely used machine learning algorithm that builds decision trees, c4.5 [14].

For the case study considered we obtain very promising results. By extracting only very few features from the instance to be solved, we are able in

fact to select the best algorithm in the 90% of the test set. We believe this approach could be extended for a large variety of combinatorial optimization problems.

2 Case Study: Bid Evaluation in Combinatorial Auctions

We now describe the case study used to test our approach: combinatorial auctions. In combinatorial auctions bidders can bid on combination of items. In this context, we have two major combinatorial optimization problems. The Bid Evaluation Problem (BEP) and the Winner Determination Problem (WDP). In the WDP the auctioneer has to find the set of winning bids at a minimum cost or maximum revenue. The winner determination problem is NP-hard. For the WDP, Integer Programming approaches represent the technique of choice.

In the BEP, beside a WDP, time windows and temporal constraints are stated among bids. 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. We consider BEP in the context of the *single unit reverse auction*, a variant of combinatorial auctions where the auctioneer wants to buy a set of distinguishable items, minimising the cost.

To solve combinatorial optimization problems, we use two different approaches: one based on Integer Linear Programming (IP), and one based on Constraint Programming (CP). These approaches differ both in the modeling and solving side, have orthogonal characteristics and strengths. For this reason IP and CP have been recently integrated and combined (see [12] for a survey). Briefly, IP models are composed by integer variables, linked by linear constraints. The objective function is also linear. Often, as in this case, variables can assume only values 0 and 1. The solution strategy used in general for IP is branch and bound, where a relaxation of the problem is solved and sub-trees which do not provide optimal solutions are pruned. In CP instead variables take their value in a finite domain of values which can be integer and are linked by mathematical and symbolic constraints (not necessarily linear). The solution strategy adopted in CP is propagation and search. Propagation is a set of inference rules which remove provably infeasible domain values. Since propagation is incomplete, i.e., values left in the domains can be infeasible, search is needed to explore the rest of the problem. In the following, we describe the models used for the BEP: an IP model and a CP model.

2.1 Bid Evaluation Problem: IP model

Each bidder j ($j = 1 \dots n$) posts one bid.[1] 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 where $M = \{1 \dots m\}$ 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. A precedence constraint between two tasks t_p and t_s is represented as $t_p \prec t_s$.

The integer linear model of the WDP 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 (1) 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 [2].

To model temporal constraints, we introduce variables $Start_{ij}$ associated to each item $i = 1 \dots m$ taken from bid $j = 1 \dots n$. These variables range on the temporal windows $[Est_{ij}, Lst_{ij}]$ for item i taken from bid j. For each pair of items t_p and t_s linked by a precedence constraint, where t_s must be executed after the end of t_p, we find all pairs of bids b_p and b_s containing that items; if S_{b_p} and S_{b_s} have an empty intersection we compute $Est_{t_p b_p} + D_{t_p b_p} - Lst_{t_s b_s}$, where $D_{t_p b_p}$ is the duration of t_p in bid b_p. In case the result is positive, i.e., domains of $Start_{t_p b_p}$ and $Start_{t_s b_s}$ do not contain any pair of values that satisfy the precedence relation, we introduce the constraint $x_{b_p} + x_{b_s} \leq 1$, which prevents both bids from appearing in the same solution; otherwise, if the result is zero or negative, we introduce the constraint

$$Start_{t_p b_p} + D_{t_p b_p} - Start_{t_s b_s} + M(x_{b_p} + x_{b_s}) < 2M$$

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

Therefore, the complete IP model for the BEP is the following:

[1] Without loss of generality, if a bidder posts n bids we consider the bidder as n different bidders

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

$$Start_{ij} \in \{Est_{ij}..Lst_{ij}\}\ ,\ \forall i \in S_j$$

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

$$\sum_{j|i \in S_j} x_j = 1\ ,\ i = 1 \dots m$$

$$\forall\ i, i', j, j' | t_i \prec t_{i'}, S_j \cap S_{j'} = \emptyset, t_i \in S_j, t_{i'} \in S_{j'}$$

$$\begin{cases} x_j + x_{j'} \leq 1 & \text{if } Est_{t_i b_j} + D_{t_i b_j} - Lst_{t_{i'} b_{j'}} > 0 \\ Start_{t_i b_j} + D_{t_i b_j} - Start_{t_{i'} b_{j'}} + M(x_{b_j} + x_{b_{j'}}) < & \\ 2M & \text{if } Est_{t_i b_j} + D_{t_i b_j} - Lst_{t_{i'} b_{j'}} \leq 0 \end{cases}$$

The IP model for the example in Table 1 is the following:

$$x_1, x_2, x_3 \in \{0,1\}$$

$$Start_{11} \in \{15..30\}$$

$$Start_{31} \in \{225..250\}$$

$$Start_{22} \in \{140..160\}$$

$$Start_{23} \in \{110..135\}$$

$$minimize\ (300x_1 + 150x_2 + 290x_3)$$

$$x_1 = 1$$

$$x_2 + x_3 = 1$$

$$x_1 + x_2 \leq 1$$

$$Start_{11} + 110 - Start_{31} + M(x_1 + x_1) < 2M$$

$$Start_{22} + 140 - Start_{31} + M(x_1 + x_3) < 2M$$

$$Start_{23} + 120 - Start_{31} + M(x_1 + x_3) < 2M$$

2.2 Bid Evaluation Problem: CP model

Auctions can be easily modelled in Constraint Programming. We have a set of m variables $X_1, \dots, X_m$ representing the items to be bought. Each variable X_i ranges on a domain containing the bids mentioning item i. We use also a set of n variables $Cost_1, \dots, Cost_n$ representing the cost of the bid in the solution. Each variable $Cost_j$ can assume only values 0, if bid j is a losing bid, and p_j, if it is a winning one. Introducing these variables we can compute the objective function simply as $min \sum_{j=1}^{n} Cost_j$.

Constraints are the following: if an item is taken from the bid B_j, all other items in S_j should be taken from the same bid and the cost of the bid in the solution should be p_j.

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

$$X_i = j \rightarrow Cost_j = p_j$$

Another important constraint that can trigger an effective propagation is a variant of the global cardinality constraint [13], 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 is the following: $gcc(Var, Val, LB, UB)$ where Var is an array of variables, Val an array of values, LB and UB are two arrays holding 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 the respective LB and UB. In our case, we need a specialized global cardinality constraint, the so called *Distribute*. The constraint $Distribute(X, J, 0, |S|)$ works on the following parameters: X is the array of variables representing items to be sold, J is an array containing consecutive numbers from 1 to n, n is the number of bids, and $|S|$ is an array where each element $|S_j|$ is the cardinality of the set of items contained in the bid j. This constraint holds iff the number of occurrences of each value $j \in J$ assigned to X is exactly either 0 or $|S_j|$.

To model temporal constraints, we introduce domain variables $Duration_i$ and $Start_i$, associated to each item i, to deal with temporal constraints. $Duration_i$ ranges on the union of all durations D_{ij} for item i taken from all bids j mentioning i. $Start_i$ ranges 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 k are linked by a precedence constraint, then the constraint

$$Start_i + Duration_i \leq Start_k$$

is introduced. Obviously, variables *Start*, *Duration* and X are connected by constraints, in the sense that, if a value j is assigned to a variable X_i, the domain of $Start_i$ should be set to $[Est_{ij} \dots Lst_{ij}]$, and $Duration_i$ should be set to D_{ij}.

Therefore, the complete CP model for BEP is the following:

$$X_i :: \{j | i \in S_j\}$$

$$Cost_j :: [0, p_j]$$

$$Start_i :: \Big\{\{Est_{ij} \dots Lst_{ij}\} | i \in S_j\Big\}$$

$$Duration_i :: \Big\{D_{ij} | i \in S_j\Big\}$$

$$X_i = j \rightarrow X_k = j \quad \forall k \in S_j$$
$$X_i = j \rightarrow Cost_j = p_j$$
$$X_i = j \rightarrow Start_i :: [Est_{ij}, Lst_{ij}]$$
$$X_i = j \rightarrow Duration_i = D_{ij}$$
$$Start_i + Duration_i \leq Start_{i'} \quad \forall i, i' | i \prec i'$$
$$Distribute(X, J, 0, |S|)$$

$$min \sum_{j=1}^{n} Cost_j$$

An additional redundant constraint which could be imposed when precedence relations are part of the problem is the Precedence Graph [8] that allows to effectively 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.

The CP model for the example in Table 1 is the following:

$$X_1 :: [1] \ , \ X_2 :: [2, 3] \ , \ X_3 :: [1]$$
$$Cost_1 :: [0, 300] \ , \ Cost_2 :: [0, 150] \ , \ Cost_3 :: [0, 290]$$
$$Start_1 :: [15..30] \ , \ Duration_1 :: [110]$$
$$Start_2 :: [110..135, 140..160] \ , \ Duration_2 :: [120, 140]$$
$$Start_3 :: [225..250] \ , \ Duration_3 :: [95]$$

$$minimize \ (Cost_1 + Cost_2 + Cost_3)$$

$$Start_1 + Duration_1 \leq Start_3$$
$$Start_2 + Duration_2 \leq Start_3$$
$$Distribute([X_1, X_2, X_3], [1, 2, 3], [0, 0, 0], [2, 1, 1])$$

2.3 Implemented algorithms

We implemented the following algorithms:

IP based algorithm 1: The first IP-based algorithm is a traditional complete Branch and Bound based on linear relaxation.

IP based algorithm 2: The second IP-based algorithm is an incomplete approach based on shadow prices[2]. The Linear Relaxation (LR) of the problem without temporal constraints is solved. Temporal constraints have not been considered in LR for efficiency reasons. Once LR is optimally solved, the algorithm ranks the variables according to their shadow prices, and finally solves the IP problem considering only the most convenient $p\%$ variables, where p is a parameter to be experimentally tuned, and fixing the remaining to zero.

CP based algorithm: The CP-based algorithm uses Limited Discrepancy Search (LDS) [4] with a variable selection heuristic based on the First Fail Principle. The value selection heuristic first chooses the value representing the bid j with the lowest $p_j/|S_j|$ value, i.e., the minimum price-for-task value.

Hybrid algorithm: The hybrid algorithm is based on the CP model and is quite similar to the previous one. The only difference concerns the value selection heuristic. The hybrid algorithm performs an indeed quite loose but effective integration. The algorithm solves the LR of the IP problem without temporal constraints as described for the IP-based algorithm 2; for each bid the minimum between $p_j/|S_j|$ and the shadow price of the associated IP variable, both normalized w.r.t. the respective maximum value, is used to rank the values of the CP variables X.

To define the components of an algorithm portfolio, we should select among these algorithms those that obtain the best performance in at least one instance. Some qualitative considerations follow: when the CP-based approach outperforms the hybrid one, both IP-based approaches outperform CP and hybrid one. Therefore, we can remove the CP-based approach from the set of strategies considered and maintain only the hybrid one. Concerning the IP-based approaches, they have similar behavior w.r.t. the other techniques (either both are better or both worst than the other two), therefore as far as the portfolio is concerned, they are equivalent. Their selection depends on the accuracy of the solution needed. Therefore, we consider here a single IP approach representing both.

We can therefore build an algorithm portfolio [5] where only the IP-based and the hybrid approach are considered as candidates. The first is referred to as IP while the second is referred to as HCP (indeed it is strongly based on CP).

In general, in an algorithm portfolio all algorithms run together until the fastest one finds the solution. On the contrary, we analyze the instance in order to decide a-priori which algorithm will probably be the best, when applied to that particular instance of the problem.

[2]Shadow prices are derived from the dual variables and represent a bound on the price of each single item.

D	Bid	Con	Algo
L	100	2500	HCP
L	200	2900	IP
H	100	2000	IP
H	200	3000	HCP
M	150	1000	HCP

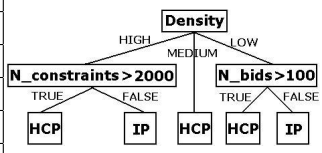

Figure 1: Example of training set and decision tree

3 Decision trees

The Decision Tree Learning method analyzes attribute-value pairs of a training set of cases, that is a set of well-classified instances, and deduces a tree where each leaf is a class and each node specifies a test to be carried out on an attribute, with a branch, and consequently a subtree, for each possible outcome of the test.

At each node, the method recursively selects an attribute and divides the cases in subsets, one for each branch of the test until all cases in each subset belong to the same class. At each node, the attribute selected for the test is the one that minimizes the entropy of the subsets generated after the test. Intuitively, the entropy of a collection of cases is the information about the non-uniformity of the collection itself. The entropy is minimum when all cases belong to the same class, otherwise the entropy is maximum when cases are uniformly classified in all possible classes.

Once created a learning base, the method can classify new instances, improving the learning base while achieving new results.

Consider a simple and small example to help the intuition of the proposed method. Suppose we want to select the best approach among HCP and IP on a given instance. We consider a training set composed in the example by 5 instances and for each instance we report values for three attributes. In Table 1 we have

Density: a discrete attribute that can assume three values: low, medium and high, representing the density of constraints between bids, w.r.t. a complete auction where all pairs of bids appear in one or more constraints.

N_bid: a continuous[3] attribute representing the number of bids in the auction.

N_constraints: a continuous attribute representing the number of temporal constraints in the model.

Each instance belongs to one of the classes HCP and IP, representing the best approach to solve the considered instance.

[3]Note that we consider as continuous those parameters that are not enumerated.

In Figure 1 we depict the decision tree for the training set in the Table, using the heuristic based on the above mentioned entropy definition. Note that in this simple example, all features appear in the decision tree. In general, however, only the most informative features are selected and tested in the decision trees. In fact, the learning algorithm stops as soon as all sets are homogeneous even before all features have been tested.

Once the decision tree has been computed, it is important to test its accuracy in classifying new instances. For this purpose, a test set is defined. A test set has the same structure of the training set: a set of feature-value pair and a class. The decision tree is used to classify the instances of the test set. In this way, we can establish which is the error rate and which is the accuracy of the decision tree.

4 Instance generation

To generate instances for our experiments we used two systems, the Multi-AGent NEgotiation Testbed system (MAGNET [3]) and the Combinatorial Auction Test Suite (CATS [9]).

MAGNET is a multi agent system designed to support the negotiation of coordinated tasks among self-interested agents. MAGNET generates instances by defining the bids, the tasks set, and the precedence graph among them. In MAGNET the user can tune a large set of parameters to modify some instance features.

Beside the largely used parameters as the number of tasks and bids, the user can define the types of task he would like to consider and specify the features for each type, as the average length or the probability of inclusion in the bids. In addition, the user can group some tasks into the same type, can tune the precedence graph structure and the bidding strategy. Concerning the bidding strategy, parameter tuning affects the mean bid cost and its variability, but poorly affects the bid size. The user can indeed specify the bid size on average, but the auction instance generator does not take strictly into consideration this parameter. Typically, the mean bid size for MAGNET generated instances is very low (it ranges between 1 and 2).

To overcome this limitation and generate instances with an higher bid size value, we used CATS, a system able to generate realistic combinatorial auction instances. Unfortunately, CATS generates WDP instances without temporal windows and constraints. So we generated WDP instances using CATS. Then, we generated instances with the same number of tasks using MAGNET. Finally, we produce a BEP by simply merging CATS bids and MAGNET temporal information. This kind of instances are more differentiable

and we generated instances with a mean bid size up to 8.

We generated a large variety of instances: the easiest have 5 tasks and 15 bids; the hardest have 30 tasks and 1000 bids. Instances have different tasks-per-bid values and precedence graph structure.

For the definition of the training set and the test set, we used only hard instances, since in that case the difference between the computational time of different algorithms becomes considerable (the best algorithm runs at least twice or three times, but often order of magnitude, faster than other algorithms). We considered a data set containing 200 instances.

5 Experimental Results

Our aim is to classify the BEP instances in two classes: IP, if the IP-based approach (either 1 or 2) described in section 2.3 is the best algorithm, and HCP, if the hybrid one is the best.

To perform our analysis we used c4.5 [14], a Decision Tree Learning system.

5.1 Considered features

Our study is strongly based on a notable paper [10], where the authors defined two graphs representing an auction: the Bid Graph and the Bid-Good Graph. The Bid Graph is a graph with a node for each bid and an edge between each couple of bids appearing together in one or more constraints. Therefore this graph represents conflicts among bids. The Bid-Good graph is a graph where each node represents either a bid or a good (a task in our case) and an edge exists between a bid node and a good node if the bid proposes the good. Figure 2 depicts the Bid Graph (left side) and the Bid-Good Graph (right side) for the example in Table 1. In the Bid Graph, arcs labelled as `Precedence` refers to couple of bids involved in precedence constraints, that is one bid contains a task that must be executed after the end of a task contained in the other bid; arcs labelled `Coverage` refers to couple of bids involved in coverage constraints, that is the bids have one or more tasks in common.

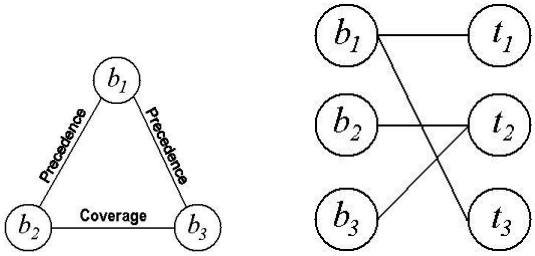

Figure 2: Bid and Bid-Good Graphs for the example in Table 1

Starting from these graphs, as done in [10] we extract 25 features from each instance in the data set and we used these values to create a set of tuples to be provided as input to c4.5.

To perform experiments, we split our data set in two parts: the *training set* and the *test set*. We build the decision tree on the basis of the training set, and we verify the quality of the resulting classification, using instances of the test set. We check the percentage of cases that are classified in the right class. We repeated this analysis randomly splitting for 10 times the data set in a training set with 130 cases, and a test set with the remaining 70 cases. For each attempt the constructed tree is the same: it has the same shape and in each branch the same feature is used to partition examples. All results presented in this paper represent the mean over these analysis.

c4.5 generates decision trees using only those parameters that lead to a minimum of the entropy of the training set, therefore, not all instance parameters are in general used in the decision tree. In our case, not all parameters contain useful information to select the best algorithm. For example, the Standard Deviation of the prices among all bids (the 23^{th} parameter of the list in [10]), is completely useless because it can not be informative of the structure of the CP and IP models.

We explain here the meaning of the parameters which are considered significant in the decision trees generated by our analysis. These parameters are extracted from the Bid Graph.

Edge Density: The Edge Density (ED) 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.

Standard Deviation of the Node Degree: The node degree is the number of edges starting from a node. Once collected in a vector this value for all bids, the Standard Deviation of the Node Degree (ND) is the standard deviation of the vector. Given a set of numbers, their Standard Deviation gives a valuation of how scattered they are around their mean value. The Standard Deviation of a vector can range from 0 to 1.

Clustering Coefficient: The Clustering Coefficient (CC) is a measure of the *local cliqueness* of the graph. For each node in the Bid Graph we compute the number of edges connecting two neighbours of the node, than we divide this number by $k(k-1)/2$, where k is the number of neighbours. We compute this value for every nodes in the Bid Graph and we put them in a vector. CC is the average of the values in the vector. CC is therefore the ratio between the number of edges connecting nodes in a neighborhood and the number of

edges in a complete graph with the same number of nodes. This parameter can range from 0 to 1.

A complete description of the 25 parameters can be found in [10].

5.2 Results

As introduced in the previous subsection, all results are the mean over the 10 analysis performed on the data set.

We run c4.5 using all the 25 parameters and we obtained a decision tree with a prediction error equal to 6%, using only 4 out of the 25 parameters. This result is very encouraging. However, the parameters considered significant are very expensive to compute. It means that given a new instance, we should extract very informative but costly parameters before being able to select the best algorithm.

Therefore, we tried to decrease the number or the cost of the parameters: c4.5, after building the decision tree, translates it in production rules. Each rule is then labelled with a number representing how many times it is triggered to classify the test set. Therefore, the importance of a rule is measured by this label.

The production rule involving the Clustering Coefficient is decisive in the 93% of cases, suggesting the right class in the 94% of these cases. Therefore, CC is supposed to be the most informative feature. So we run c4.5 using only CC. In this case the prediction error rises to 9% (still very good).

Using CC we obtained a good result, but unfortunately for large instances, even the extraction time for CC only is too high. Figure 3 shows, for groups of instances with similar tasks-per-bid values, the ratio between the CC extraction time and the difference between the search times of the best and the worst algorithm; when the ratio is greater than 1 it is not worth extracting CC since the time used in selecting the best algorithm and solve the instance using it is greater than the time used by the worst algorithm to solve the instance. In Figure 3 we can see that, the higher the tasks-per-bid value, the higher the CC extraction time. This is obvious because, by increasing the size of the task bundles proposed by bidders, a higher number of precedence constraints between bids (edges in the bid graph) is introduced. Each group of instances with similar tasks-per-bid value contains 10 instances, and the x-axis value is the mean over these instances.

We therefore need to find parameters whose extraction is fast, with a prediction rate almost equal to 91%, that is the CC prediction rate. We considered only the first 11 attributes described in [10], since their extraction time is very low. The decision tree prediction error is 11%. The parameters considered significant by c4.5 are only the edge and node density (ED and

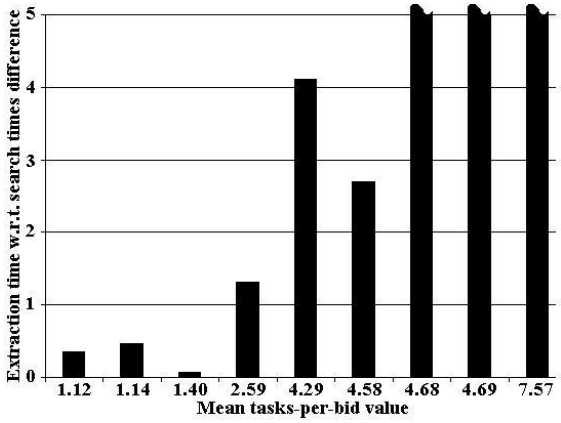

Figure 3: Extraction time for CC attribute

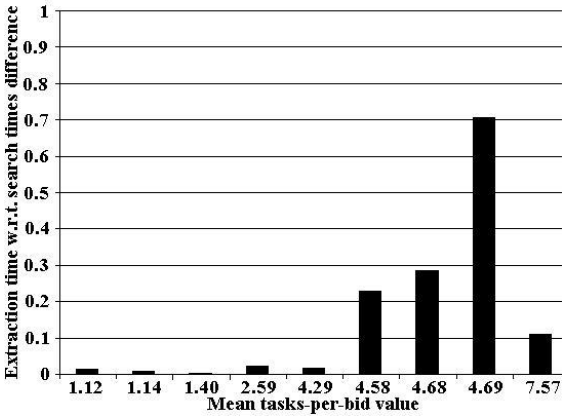

Figure 4: Extraction time for ED and ND attributes

ND). The production rule containing ED is used in the 43% of cases, with a prediction error of 1,5%. while the production rule containing ND is used in the 57% of cases, with a prediction error of 9%. ED and ND extraction is very fast; in Figure 4 is depicted the ratio between the extraction time, for both parameters, and the search times difference.

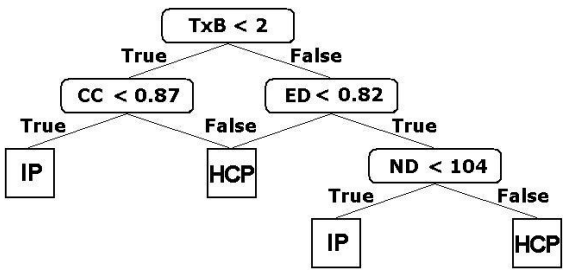

Figure 5: Decision tree for the BEP

Summarizing, to select the best algorithm from the algorithm portfolio, if the tasks-per-bid value is low we can use the CC parameter with a prediction rate of 91%, otherwise we can use the ED and ND parameters with a prediction rate of 89%. Clearly, if we accept

a lower accuracy, we can always use ED and ND. We experimentally found that the maximum tasks-per-bid value for which CC extraction time is convenient is about 2.

The whole decision tree for the BEP is depicted in Figure 5. The values on which to perform the test at the nodes are those values that minimize the total entropy of the training set. Obviously, each decision tree we obtained on a single analysis has different threshold values, but we noticed that the difference between these values over all trees is limited to non-significant decimal places. Therefore, the decision tree depicted in Figure 5 is equal to all other trees obtained in our analysis.

6 Discussion and Conclusions

We believe this paper is a promising first step toward the use of machine learning techniques in algorithm portfolio selection. Many steps should still be done:

- we are interested in the most appropriate representation that summarizes the structure of an instance. For this reason, we are investigating an approach based on Case Based Reasoning that enables to take into account the instance representation based both on features extracted from the graph and on the graph itself.
- we will extend this approach to other problems. In particular, we will consider problems with no clear structure but that contain side constraints that break the regularity of the structure itself.
- a long term aim is to use these results in the design of global constraints. When more than one filtering algorithm is defined, the user should choose if to enable the filtering and which algorithm to select. If the structure of the constraint is analyzed, we could at design level define a portfolio of filtering algorithms and select the most appropriate one. In this case, the requirement of efficient parameter extraction is even more important.

REFERENCES

[1] R. Béjar and A. Cabiscol and C. Fernàndez, F. Manyà, C.P. Gomes, Capturing Structure with Satisfiability, *Proceedings CP2001.*

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

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

[4] M. L. Ginsberg and W. D. Harvey. Limited Discrepancy Search. In C. S. Mellish, editor, *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence (IJCAI-95); Vol. 1*, pages 607–615, 1995.

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

[6] C.P. Gomes and B. Selman, Problem Structure in the Presence of Perturbations, *Proceedings of AAAI97.*

[7] A. Guerri and M. Milano, CP-IP techniques for the Bid Evaluation in Combinatorial Auctions, *Proc. CP2003*, 2003.

[8] 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.

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

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

[11] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, Y. Shoham, Boosting as a Metaphor for Algorithm Design, *Proc. CP2003*, 2003.

[12] M. Milano, Constraint and Integer Programming - Toward a closer integration, *Kluwer Academic Publisher*, 2004.

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

[14] J. Ross Quinlan, C4.5: programs for machine learning, Margan Kaufmann, 1993.

[15] T. Walsh, Search in a Small World, *Proceedings of IJCAI99.*

[16] T. Walsh, Search on High Degree Graphs, *Proceedings of IJCAI2001.*

[17] C. Williams and T. Hogg, Exploiting the deep structure of constraint problems, Artificial Intelligence, 70, pp. 72–117, 1994