# Learning techniques for Automatic Algorithm Portfolio Selection

**Alessio Guerri** and **Michela Milano** [1]

**Abstract.** The purpose of this paper is to show that a well known machine learning technique based on Decision Trees can be effectively used to select the best approach (in terms of efficiency) in an algorithm portfolio for a particular case study: the Bid Evaluation Problem (BEP) in Combinatorial Auctions. In particular, we are interested in deciding when to use a Constraint Programming (CP) approach and when an Integer Programming (IP) approach, on the basis of the structure of the instance considered. Different instances of the same problem present a different structure, and one aspect (e.g. feasibility or optimality) can prevail on the other. We have extracted from a set of BEP instances, a number of parameters representing the instance structure. Some of them (few indeed) precisely identify the best strategy and its corresponding tuning to be used to face that instance. We will show that this approach is very promising, since it identifies the most efficient algorithm in the 90% of the cases.

## 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], [5], [12], [13], [14]) since it is considered an important aspect for explaining the algorithm behavior. The instance structure can be represented in different ways: (*i*) through the constraint graph where nodes are variables and arcs are constraints; (*ii*) through a set of features (pairs attribute-value) extracted from the constraint graph; (*iii*) 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.

To test our conjecture we used a combinatorial optimization problem as a case study, the Bid Evaluation Problem (BEP), 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. This problem has an interesting structure: it contains as subproblem 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 is successfully faced by Mathematical Programming techniques and not by Constraint Programming techniques (see for example [6]). 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.

The aim of this paper is to use machine learning techniques to identify a connection between the instance structure and the algorithm performance. For this purpose, we have built a *training set* by extracting, for each instance, a set of 25 features (as suggested by [8]) 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 [11].

For the case study considered we obtain very promising results. By considering only very few features from an instance, 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.

Two related work with the current one are [8] and [9]. The authors are interested in determining the intrinsic complexity of a problem with respect to a given algorithm (CPLEX in the paper). The authors use regression to predict the empirical hardness of an instance and apply the method to the Winner Determination Problems in combinatorial auctions (which is a sub-problem of the BEP where temporal constraints are relaxed). In [9], again using regression, the authors propose how to build an algorithm portfolio in an effective way exploiting an *hardness model* and perform experimental results on three

[1] DEIS, University of Bologna, Viale Risorgimento 2, 40136 Bologna, Italy. email:{aguerri,mmilano}@deis.unibo.it

well known algorithms for the winner determination problem.

The rest of the paper is organized as follows. In Section 2 we briefly outline the BEP models and the strategies, based on CP and IP, and their variants, that we consider for solving the BEP. In Section 3 we describe decision trees and how they are built and tested. In Section 4 we describe the generation of problem instances. We present experimental results in Section 5. Finally, we conclude and highlight our future plans in Section 6.

## 2 Case Study the Bid Evaluation in Combinatorial Auctions: models and algorithms

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 winner determination problem, time windows and temporal constraints are stated among bids. We mainly consider an auction where the auctioneer buys a set of items (in our case tasks or 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, minimizing the cost.

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 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 $i$ ($i = 1 \dots m$) 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 $t_p \prec t_s$.

We introduce a decision variable $x_j$ for each bid $j$, that takes the value 1 if the bid $B_j$ is a winning one, 0 otherwise. We also introduce a variable $Start_{ij}$ for each item $i$ taken from bid $j$. These variables range on the temporal windows $[Est_{ij}, Lst_{ij}]$ for item $i$ taken from bid $j$.

The objective function is $min \sum_{j=1}^{n} p_j x_j$, that minimizes the total cost.

In the BEP we have two kinds of constraints: covering constraints and precedence constraints. Covering constraints have the form $\sum_{j|i \in S_j} x_j = 1$, and state all items should be covered and each item should be covered by exactly one bid.

To represent precedence constraints we consider each pair of items $t_p$ and $t_s$ such that $t_p \prec t_s$, and 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 $Est_{b_p t_p} + D_{b_p t_p} - Lst_{b_s t_s}$, where $D_{b_p t_p}$ is the duration of $t_p$ in bid $b_p$. In case the result is positive, that is, domains of $Start_{t_p b_p}$ and $Start_{t_s b_s}$ do not contain any pair of values that could 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 $Start_{t_p b_p} + D_{t_p b_p} - Start_{t_s b_s} + M(x_{b_p} + x_{b_s}) \leq 2M$, 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 $x_{b_s} = 0$.

### 2.2 Bid Evaluation Problem: CP model

Auctions can be easily modelled in Constraint Programming. We recall that $B_j = (S_j, Est_j, Lst_j, D_j, p_j)$.

We introduce four sets of domain variables: $X$, that is an array of $m$ variables representing the items to be bought. Each variable $X_i$ ranges on a domain containing the bids mentioning item $i$. $Cost$, that is an array of $n$ variables 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. $Duration$ and $Start$, that are arrays of $m$ variables. $Duration_i$ ranges on the union of all duration $D_{ij}$ 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 $i$.

The objective function is $min \sum_{j=1}^{n} Cost_j$.

Each time a variable $X_i$ is assigned to a value $j$, that is item $i$ is assigned to bid $j$, we propagate the fact that all other items in $S_j$ should be assigned to the same bid, setting to $j$ variables $X_k$, $\forall k \in S_j$. Similarly, we set variables $Start_k$ and $Duration_k$, $\forall k \in S_j$ to values proposed by bid $B_j$. We also set to $p_j$ the variable $Cost_j$.

To consider precedence among tasks, we introduce the constraint $Start_p + Duration_p \leq Start_s$ for each couple of tasks $t_p$ and $t_s$ such that $t_p \prec t_s$.

Finally, we use $Distribute(X, J, 0, |S|)$, that can trigger an effective propagation. It is a variant of the global cardinality constraint [10], whose semantics 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. Parameters in $Distribute(X, J, 0, |S|)$ have the following meaning: $X$ is the array of variables representing items to be sold, $J$ is an array containing numbers tidily from 1 to $n$, where $n$ 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 $|S_j|$.

### 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. A 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) [3] 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 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 both CP-based and hybrid approaches. 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 [4] 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.

## 3 Decision trees

The Decision Tree Learning method analyzes attribute-value tuples of a training set of cases, i.e., well-classified instances, and deduces a tree, called decision tree, where each leaf is a class and each node specifies a test to be carried out on an attribute value, 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.

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. In this way, we can establish which is the error rate and which is the accuracy of the decision tree.

One important aspect of decision trees is their ability to generalize the results. Therefore, not all attributes should be used in the decision tree but only those that enable to partition instances in homogeneous groups as done in this paper.

## 4 Instance generation

To generate instances for our experiments we used two systems, the Multi-AGent NEgotiation Testbed (MAGNET [2]) and the Combinatorial Auction Test Suite (CATS [7]).

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 (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 faster than other algorithms). We considered a data set containing 200 instances. They are classified in the 2 classes HCP and IP described in Section 2.3. 53% of instances belongs to the HCP class, that is they are best solved by the Hybrid approach, while the remaining 47% belongs to the IP class.

## 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 [11], a Decision Tree Learning system.

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 10 times the data set in a training set with 130 cases, and a test set with the remaining 70 cases. All results presented in this paper represent the mean over these analysis. c4.5 produces decision trees that perform test only on those parameters that minimize the entropy and ends the analysis as soon as all cases belong to the same class (or a looser termination condition is achieved). In our paper, c4.5 is able to detect only a small subset of the input parameters in the decision trees to produce homogeneous classes.

### 5.1 Considered features

Our study is strongly based on a notable paper [8], 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 covering or temporal 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.

Starting from these graphs, the authors extracted 25 features representing the instance structure. We computed all 25 parameters for each instance in the data set and we used these values to create a set of attribute value tuples to be provided as input to c4.5.

Fortunately, once the tree is built, we can observe that not all parameters are significant to determine the most efficient algorithm. We explain here the meaning of those which are considered significant for our analysis. They are all 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 parameter 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.

**Clustering Coefficient:** The Clustering Coefficient (CC) is a measure of the *local cliqueness* of the graph. For each node in the graph we compute the number of edges connecting two neighbors of the node, than we divide this number by $k(k-1)/2$, where $k$ is the number of neighbors. We compute this value for every nodes in the 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.

### 5.2 Results

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. Hence, the importance of a rule is measured by this label.

The production rule involving the Clustering Coefficient (CC) 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 for large instances even the extraction time for CC only is too high.

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 [8], 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 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%. We whole decision tree has a prediction rate of 89%.

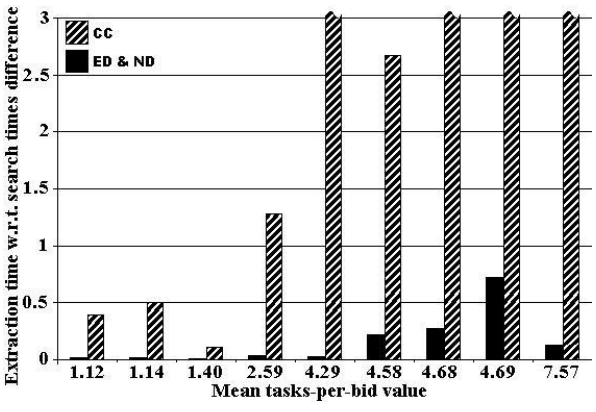


**Figure 1:** Extraction time for the attributes considered

Furthermore, we decided to see if we could have an indication of the cost of computing the three relevant feature values by using the tasks-per-bid (TxB) as an indicator. Figure 1

plots, for groups of instances with similar TxB values, the ratio between the feature values extraction times 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 the feature since the time used in selecting the best algorithm plus the time to solve the instance using it is greater than the time used by the worst algorithm to solve the instance. In Figure 1 we notice that the higher the TxB 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 TxB value contains 10 instances, and the x-axis value is the mean over these instances. Figure 1 shows that the maximum TxB value for which CC extraction time is convenient is about 2.

Summarizing, to select the best algorithm from the algorithm portfolio, if the tasks-per-bid value is lower than 2 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 used ED and ND.

The decision trees for the BEP are depicted in Figure 2. The values on which to perform the test at the nodes are those values that minimize the total entropy of the training set.

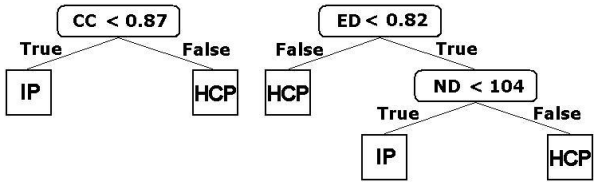


**Figure 2:** Decision trees for the BEP

In Section 1 we stated that the WDP is best solved by Integer Programming, but when side constraints as temporal constraints are added to the problem the CP approach becomes competitive. Therefore, we can conclude that temporal information is the only knowledge we need to select the best algorithm. We computed all parameters starting from a Bid Graph where only temporal constraints are considered (an edge between two nodes is added only if a precedence constraint exists between the bids associated to the nodes). We obtained almost the same decision trees, with different test threshold values, but a slightly lower (85%) prediction rate.

We can conclude that, even if temporal constraints are quite effective for selecting the best algorithm, the information gained by the WDP sub-problem structure can improve the accuracy of the classification. In fact, the density of the WDP Bid-Graph gives an information on the dimension, and hence the hardness, of the IP model.

## 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.
- we will use a machine learning method where attributed are weighted so as to take into account in the definition of the tree not only the entropy decrease, but also the computational effort to extract it.

## ACKNOWLEDGEMENTS

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 work.

We would like to thank Andrea Lodi and Andrea Roli for useful discussions. 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. Béjar, A. Cabiscol, C. Fernàndez, C.P. Gomes and F. Manyà, Capturing Structure with Satisfiability, *Proc. CP-2001*, pp. 137-152, 2001.

[2] J. Collins and M. Gini, An integer programming formulation of the bid evaluation problem for coordinated tasks, *Mathematics of the Internet: E-Auction and Markets; Vol.127*, pp. 59–74, 2001.

[3] M. L. Ginsberg and W. D. Harvey. Limited Discrepancy Search. *Proc. IJCAI-95; Vol. 1*, pp. 607-615, 1995.

[4] C.P. Gomes and B. Selman, Algorithm portfolio design: theory vs. practice, *Proc. UAI-97*, pp. 190-197, 1997.

[5] C.P. Gomes and B. Selman, Problem Structure in the Presence of Perturbations, *Proc. AAAI-97*, pp. 221-226, 1997.

[6] A. Guerri and M. Milano, CP-IP techniques for the Bid Evaluation in Combinatorial Auctions, *Proc. CP-2003*, pp. 863-867, 2003.

[7] K. Leyton-Brown, M. Pearson and Y. Shoham, Towards an Universal Test Suite for Combinatorial Auction Algorithms, *Proc. EC-00*, pp. 66-76, 2000.

[8] K. Leyton-Brown, E. Nudelman and Y. Shoham, Learning the Empirical Hardness of Optimization Problems: The Case of Combinaorial Auctions, *Proc CP-2002*, pp. 556-572, 2002.

[9] K. Leyton-Brown, E. Nudelman, G. Andrew, J. McFadden, Y. Shoham, Boosting as a Metaphor for Algorithm Design, *Proc. CP-2003*, pp. 899-903, 2003.

[10] J. C. Regin, Generalized Arc Consistency for Global Cardinality constraint, *in Proc. AAAI-96; Vol. 1*, pp. 209-215, 1996.

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

[12] T. Walsh, Search in a Small World, *Proc. IJCAI-99*, pp. 1172-1177, 1999.

[13] T. Walsh, Search on High Degree Graphs, *Proc. IJCAI-2001*, pp. 226-274, 2001.

[14] C. Williams and T. Hogg, Exploiting the deep structure of constraint problems, *Artificial Intelligence*, Vol. 70, pp. 72-117, 1994