Making Choices Using Structure at the Instance Level exploiting Case Based Reasoning*

C. Gebruers(1) and A. Guerri(2) and B. Hnich(1) and M. Milano(2)

(1) Cork Constraint Computation Centre,
University College Cork,
Cork, Ireland.
{c.gebruers,b.hnich}@4c.ucc.ie
(2) DEIS, University of Bologna, Viale Risorgimento 2,
40136 Bologna, Italy.
{aguerri,mmilano}@deis.unibo.it

1 Introduction

Constraint programming (CP) and Integer Linear Programming (IP) are both highly successful *technologies* for solving a wide variety of combinatorial optimization problems. When modelling a combinatorial optimization problem, there is often a choice about what technology to use to solve that problem. For instance in the Bid Evaluation problem (BEP) –a combinatorial optimization problem arising in combinatorial auctions– both CP and IP can successfully be used [3]. While there exist domains where one can easily predict what technology will excel, in many domains it is not clear which will be more effective. The BEP falls into the latter case.

How do we choose among technologies when all instances share the same *problem structure*? We are currently investigating machine learning methodologies to explore *structure at the instance level* as a means to distinguish whether to use CP or IP to solve instances of the BEP.

In [4] a technique based on Decision Trees is explored and used to select the best algorithm for a BEP instance. Here we investigate another approach based on Case Based Reasoning (CBR), a technique arising within Artificial Intelligence, as a framework within which to carry out this exploration. CBR utilises similarity between problems to determine when to reuse past experiences to solve new problems. An experience in this context is what technology to use to solve an instance of the BEP. Using a representation that distinguishes problems at the instance level, and a similarity function to compare problems expressed in this representation, enables us to determine what technology to use on a new problem instance.

* This work has received support from Science Foundation Ireland under Grant 00/PI.1/C075. This work was partially supported by the SOCS project, funded by the CEC, contract IST-2001-32530.

2 Bid Evaluation Problem

Combinatorial auctions are an important e-commerce application where bidders can bid on combination of items. The *Winner Determination Problem WDP* is the task of choosing, from among $\mathcal{M}$ bids, the best bids that cover all items at a minimum cost or maximum revenue. The winner determination problem is NP-hard. The *Bid Evaluation Problem BEP*, is a time constrained version of the *WDP* and consequently involves temporal and precedence constraints. Items in a bid are associated with a time window (and hence duration) and are interconnected by precedence constraints. A solution to the *BEP* involves solving the *WDP* and additionally satisfying the temporal and precedence constraints. We consider *BEP*s in the context of the *single unit reverse auction*, a variant of combinatorial auctions where the auctioneer wants to buy a set $\mathcal{M}$ of distinguishable items, minimising the cost. In [3], different approaches based on pure CP, pure IP and hybrid approaches mixing the two have been developed and tested. Furthermore, variants of these technologies are considered involving different parameter settings for each technology. In this work we concern ourselves with a sub-problem of deciding whether to use CP or IP as a solution technology for the BEP. Due to lack of space, we refer the reader to [3] for detailed information about the CP and IP models developed for the BEP.

3 What Technology?

As a consequence of dominant structure apparent in a problem domain, there are situations where clear predictions about whether to use CP or IP can be made. For instance, when side constraints complicate the problem, CP can accommodate them gracefully and indeed take advantage of them. When the problem is highly structured, polyhedral analysis can be highly effective. When the problem has a loose continuous relaxation, constraint propagation methods, and thus CP, can overcome this weakness. If, instead, relaxations are tight and linear constraints tidily represent the problem, IP should most probably be used. However deciding whether to use IP or CP to solve a particular combinatorial optimisation problem is often an onerous task. Experimental results evaluating the performance of the two strategies considered for the BEP, show that neither CP nor IP is a clear winner [3]. These results further indicate that there isn't any simplistic structure or problem feature, that correlates with choice of suitable technology. Hence we propose to explore whether structure at the instance level can be used to discriminate between CP and IP for the BEP.

4 CBR Framework

CBR enables past problem solving experiences to be reused to solve new problems [5]. CBR has been successfully used in the context of e.g. diagnosis and decision support [6], design and configuration [1], etc. Experiences are stored along with the problems they solve as *cases*. A case is a representative example

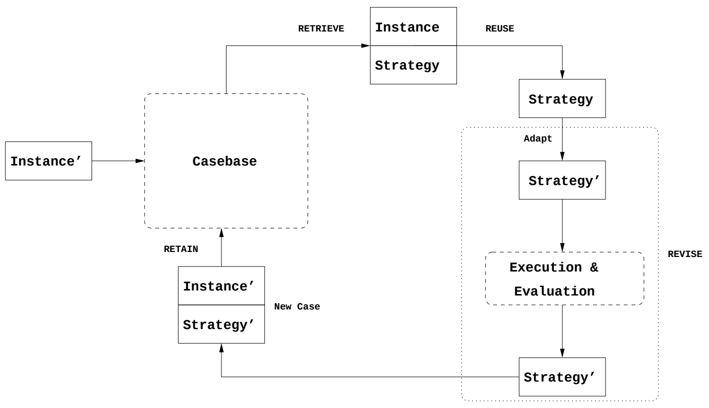

Fig. 1. 3 Methodology Overview

of a cluster of instances (a cluster can contain from 1 to n instances) that are similar to one another, but different from other clusters of instances. A particular technology (CP or IP in this work), is associated with one or more clusters of instances i.e. cases.

A CBR system consists of a four step cycle; *retrieve*, *reuse*, *revise*, and *retain.* To solve a new problem, we *retrieve* a case from the casebase, whose problem part is most similar to the new problem. We then *reuse* the experience part of the case to solve the new problem. The casebase may be *revised* in light of what has been learned during this most recent problem solving episode and if necessary the retrieved experience, and the new problem, may be *retained* as a new case. Every time a new problem instance is presented to the system, this cycle enables a CBR system to both learn new experiences and to maintain or improve the quality of the cases.

We now describe a CBR system called SELECTOR that enables us explore the use of structure at the instance level to predict whether to use CP or IP for BEP instances. Firstly, consider a case in SELECTOR. For now we will simply refer to the problem part of a case in abstract terms. We do this because the choice of how to represent a problem instance is one aspect to be explored. The experience part of a case corresponds to the appropriate technology for that instance (determined by experimentation) i.e. CP or IP. A case is thus a tuple composed of an abstract representation of an instance, and an appropriate technology (either CP or IP) that efficiently solves that instance. The final element of the system is a function f_{sim} to compute the similarity between two problem instances. The choice of f_{sim} is inextricably linked to the problem representation and hence is the other aspect we wish to explore in this work.

SELECTOR has two modes of operation; a training mode and a testing mode. A dataset is randomly divided into two sub-sets for training and testing purposes. Initially, the casebase is seeded with a random instance. In training mode, a casebase is assembled using the training problems. We expect that the instances retained in the casebase constitute examples of when to use the appropriate technology. The training mode consists of the following activities:

Retrieval: The current training instance is compared with every case in the casebase and the most similar case (established using an f_{sim}) is returned;
Reuse: The current training instance is solved using the technology identified by the case retrieved during the retrieval step;
Training Evaluation: The current training instance is solved using the other technologies that could have been chosen. The results of this step and the reuse step are recorded in preparation for the next step.
Revise & Retain: If the retrieved case has predicted incorrectly, then a new case is assembled consisting of the current training instance and the most appropriate technology (rather than the retrieved technology). This case is then saved to the casebase.

Once the casebase is non-empty, we can enter testing mode. For each testing problem, test mode does the following:

Retrieval: As for training mode;
Reuse: As for training mode;
Testing Evaluation: if the system correctly predicts the appropriate technology, we increment the good prediction score. Otherwise we increment the failure score. No new cases are added to the casebase in testing mode.

The training and testing phases can be intertwined at an interval n of the user's choosing. The system continues in training mode until n new cases have been added, then switches to testing mode. Once the testing set has been exhausted, the system reverts to training mode until a further n cases have been added... and so on until there are no instances left in the training set. This approach of intertwining training and testing enables us to identify learning behavior as casebases grow, and to consider the impact of an individual or group of cases on casebase performance. These factors are important for examining the impact of structure at the instance level.

This methodology is based on the intuition that if two instances are similar, then it follows that the same technology should be appropriate for both problems. Whether this approach works or not depends on two critical factors; how to represent problem instances and how we decide they are similar. In the next section, we give an example of representations and similarity measures.

5 Case study

5.1 Problem representations and similarity measures

Feature based approaches. We consider representations based on four features of the BEP that fully describe the problem; the number of bids (b), the number of tasks (t), the number of includes constraints (i), and the number of precedence constraints (p). All these features can be determined from the problem instance in polynomial time.

We explore different similarity measures between two *BEP* instances p1 and p2 with feature vectors $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$, respectively:

Weighted Block City: If $\boldsymbol{p_1} = \langle b_1, t_1, i_1, p_1 \rangle$, $\boldsymbol{p_2} = \langle b_2, t_2, i_2, p_2 \rangle$, then we refer to following family of block city similarity measures as *weighted similarity functions*:

$$sim(\boldsymbol{p_1}, \boldsymbol{p_2}) = w_b(\frac{|b_1 - b_2|}{b_{max} - b_{min}}) \bowtie w_t(\frac{|t_1 - t_2|}{t_{max} - t_{min}}) \bowtie w_i(\frac{|i_1 - i_2|}{i_{max} - i_{min}}) \bowtie w_p(\frac{|p_1 - p_2|}{p_{max} - p_{min}})$$

where $\bowtie \in \{+, *\}$, and f_{min} and f_{max} are the minimum and the maximum values for feature f, respectively. Note that we use normalization to remove the effects of different problem sizes.

Euclidean Distance: The similarity measure between two feature vectors is defined as the Euclidean distance between these two vectors.

Structural approach. Should the feature based representations prove insufficient to distinguish between IP and CP, we also propose to examine structural representations such as graphs. The principle graph representation we adopt for this purpose is a minor natural extension of the bid-good graph[1], where we add precedent edges between task vertices to represent the precedence constraints that exist between tasks. We compute similarity between problem instances using a graph matching technology based on an incomplete branch and bound approach. However, due to space limitations, the details of the algorithms are not shown.

5.2 Preliminary results

We generated a large variety of instances, spacing from easiest with 5 tasks and 15 bids to hardest with 30 tasks and 1000 bids, and with variable tasks-per-bid values and precedence graph structures. In this work, we applied CBR only to harder instances, where the difference between search times of the algorithms becomes considerable. This considered data set is composed of 90 instances, the easiest of which has 15 tasks, about 400 bids and a mean tasks-per-bid value little higher than 1.

The 90 problems were randomly divided 10 times into pairs of training and testing sets. The ratio of training to testing problems was $\frac{2}{3}$ training and $\frac{1}{3}$ testing. Each experiment was repeated 10 times using the 10 pairs of training and testing sets and results averaged over these 10 trials.

This work is at an early stage. The representations and similarity measures considered thus far may appear quite basic, however it is informative to see how effective relatively simple measures can be [2]. So far, all features weights

[1] a common way to represent the BEP, where vertices represent bids and tasks, and edges between bids and tasks represent the inclusion of tasks in bids. See [7]

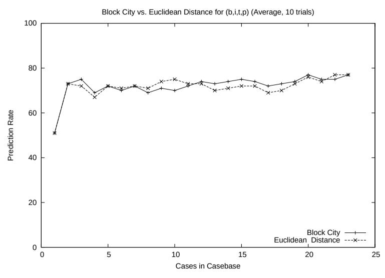

Fig. 2. Block City vs. Euclidean Distance as Similarity Measure (b,i,t,p)

are equal. No significant difference in prediction ability was observed between using a similarity measure based on Block-City or Euclidean Distance. Slight differences arise from different cases occasionally being chosen, but the overall effect is negligible as is apparent from Figure 2 which shows a typical result from testing a representation using all 4 parameters (b,t,i,p) in both a Euclidean Distance and Block City similarity measure. We observe it is possible to achieve an average prediction rate across 10 trials of 80% by using these very simple feature vectors and similarity measures. However, it is very misleading to draw any further conclusions based on such elementary experiments.

6 Conclusion

In this paper, we propose an investigation of instance structure as a discriminating factor among solution technologies for the BEP within a CBR framework. In our future work, we plan to perform extensive exploration of both feature based and more complex structure based representations.

References

1. Susan Craw, Nirmalie Wiratunga, and Ray Rowe. Case-based design for tablet formulation. In Proc. 4th European Workshop on CBR, pages 358–369, 1998. Springer.
2. Very Simple Classification Rules Perform Well on Most Commonly Used Datasets. Robert C. Holte (1993). Machine Learning, vol. 3, pp. 63-91.
3. A. Guerri and M. Milano, IP-CP techniques for the Bid Evaluation in Combinatorial Auctions, in Proc. CP2003, 2003.
4. A. Guerri and M. Milano, Learning techniques for Automatic Algorithm Portfolio Selection, Submitted.
5. J. Kolodner, Case-Based Reasoning, Morgan Kaufmann, 1993.
6. Mario Lenz, Hans-Dieter Burkhard, Petra Pirk, Eric Auriol, Michel Manago: CBR for Diagnosis and Decision Support. AI Commun. 9(3): 138-146 (1996)
7. K. Leyton-Brown, E. Nudelman and Y. Shoham, Learning the Empirical Hardness of Optimization Problems: The Case of Combinaorial Auctions, *Proc CP02*, 2002.