Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies, Volume 82. Proceedings of KES'02, 336-340. Sep, 2002

Case-Based Reasoning as a Heuristic Selector in a Hyper-Heuristic for Course Timetabling Problems

S. Petrovic, R. Qu
School of Computer Science and Information Technology, Jubilee Campus, University of Nottingham, Nottingham, NG8 1BB, U.K.

Abstract. This paper studies Knowledge Discovery (KD) using Tabu Search and Hill Climbing within Case-Based Reasoning (CBR) as a hyper-heuristic method for course timetabling problems. The aim of the hyper-heuristic is to choose the best heuristic(s) for given timetabling problems according to the knowledge stored in the case base. KD in CBR is a 2-stage iterative process on both case representation and the case base. Experimental results are analysed and related research issues for future work are discussed.

1. Introduction

Case-Based Reasoning (CBR) is a knowledge-based technique that solves new problems based on knowledge and experiences from old problems that have been previously collected in the form of cases and stored in a case base (source cases) [1]. The overall idea is that by retrieving the most similar source case(s), CBR is capable of solving problems based on previous good solutions, avoiding solving problems from scratch and saving a lot of effort. Recently CBR has been employed with some success on some scheduling problems [2].

The term 'hyper-heuristic' is a relatively new term refers to a "heuristic which chooses heuristics" [3, 4, 5]. The main difference between hyper-heuristic and the widely studied meta-heuristic is that hyper-heuristic is a heuristic that chooses proper heuristics/meta-heuristics from a given set of heuristics, rather than operating directly on the problem. Meta-heuristics are usually (but not exclusively) pre-defined heuristics that operate directly on problems. A hyper-heuristic is thus a more flexible and more general approach that may be employed on a wider range of problems. Recently some research on scheduling has obtained promising results by employing hyper-heuristics, although sometimes the approaches do not use this name [6, 7, 8].

Timetabling problems have been heavily studied for the last 40 years [9, 10, 11, 12, 13, 14]. This paper studies course timetabling problems, which can be considered as multi-dimensional assignment problems that assign students and teacher into timeslots, and courses into rooms subject to certain constraints [10]. A large amount of work has been carried out using a variety of approaches. Some work has made comparisons between different heuristics on specific problems. This serves as a good starting point for using CBR as a large amount of data is available to be collected and studied. The aim of this paper is to investigate the employment of CBR as a heuristic selector in a hyper-heuristic to predict the best heuristic according to the knowledge discovered.

2. Knowledge Discovery in Case-Based Reasoning for Timetabling

The basic assumption that CBR works on is that similar problems will have similar solutions [1]. The retrieval is a similarity-driven process that finds the most similar source cases. The key issue of our work is to discover and represent the knowledge of problem solving in a way that allows us to find the most similar source cases that give good predictions of the best heuristics for the new problem.

Knowledge discovery (KD) was defined by Fayyad, Piatetsky-Shapiro and Smyth [15] as a "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". Some work has been carried out using evolutionary algorithms to fulfil this task [16]. Our work studies a 2-stage iterative process of KD on the case representation and the case base.

The first stage of KD on case representation. In the CBR system:

- Cases are represented by a list of feature-value pairs. Tabu Search and Hill Climbing will be used to search for the feature list from a number of given features. We introduce two sets of features – F_s that contains simple features and F_c that contains simple features and their combinations.

  $F_s = \{f_0, \dots f_I\}$ = {No. of courses; No. of slots; No. of constraints; No. of rooms; No. of hard constraints (which must not be violated, for example, two courses with common students are scheduled in the same slot); No. of soft constraints (all the other constraints that are desirable); No. of inclusive/exclusive courses (courses should/should not be scheduled to a fixed slot); No. of consecutive/non-consecutive courses (courses should/should not be scheduled to consecutive slot)}

  $F_c = \{f_0, f_1, \dots f_I, f_0/f_1, f_0/f_2, \dots f_I/f_{I-1}\}, f_i \in F_s, i = 0, \dots I$
- The case base is a database of a set of cases of timetabling problems with the best 2 heuristics from H for solving them.

  H = {Hill climbing; Largest degree first; Largest degree with tournament selection; Colour degree; Saturation degree} (see [17])
- The training cases are a set of cases in hand for the training in the knowledge discovery process. The best heuristic from H for solving them is obtained beforehand.
- The similarity measure given in formula (a) employs a nearest-neighbour approach that calculates the weighted sum of differences between each pair of features within two cases.

$$S(C_s, C_t) = 1 \Big/ \sqrt{\sum_{i=1}^{j} w_i \times (fs_i - ft_i)^2 + 1} \qquad \text{(a)}$$

  The notation in formula (a) is defined as follows:

  j is the number of features in the case representation

  w_i is the weight of the ith feature reflecting its relevance on the prediction

  fs_i, ft_i are the values of the ith feature in source case C_s and training case C_t, respectively
- Retrieval uses the similarity measure to compare each source case and the training case represented by a list of feature-value pairs. If the best heuristic of the training case is within one of the best 2 heuristics of the retrieved source cases, the retrieval will be seen as *sucessful* on the case representation with the features selected.

A schematic diagram of the discovering process on the CBR system described above is given in Figure 1. Tabu search and Hill climbing are used to discover features and their weights. The search space includes all the possible enumerations of features selected from F_c or F_s that describe the characteristics of the timetabling problems. An initial feature list is randomly selected for Tabu Search or Hill Climbing. During the search, feature lists are selected by moves that change one of the features and its weight. For each feature list selected, retrieval compares the heuristics proposed in the training cases and the source cases retrieved. The percentage of successful retrievals indicates the system performance on the feature vector selected. This process will be carried out iteratively until the stopping condition is met (here a pre-set time for searching). The feature vector that gives the highest accuracy of prediction will be used in the next stage of KD.

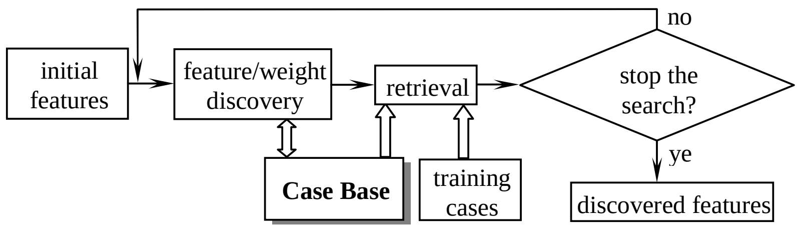

Figure 1 Schematic Diagram of Knowledge Discovery on Featrues and Their Weights

The second stage of KD on the case base. Source case selection is one of the key issues for the system performance of CBR. Irrelevant source cases may confuse the retrieval and decrease the system performance. In CBR, source cases are usually collected by studying the previous cases, or by gradually retaining solved cases during the lifetime of the system.

The knowledge discovery on the case base uses the "Leave-One-Out" strategy that removes one source case a time from a given case base. If the system performance is decreased, then the removed source case will be added back. Otherwise, another source case will be removed and this process will be carried out repeatedly until the best system performance is obtained. Then the case base is refined and only the relevant and representative cases are retained.

3. Experiments and Discussions on the Results

A set of experiments is carried out for the KD that is described in the last section.

Experiment 1. Knowledge discovery on features. Figure 2 shows the chart of the average system performance (percentage of successful retrievals for all the training cases) on 10 runs with the different number of features searched by Tabu Search and Hill Climbing.

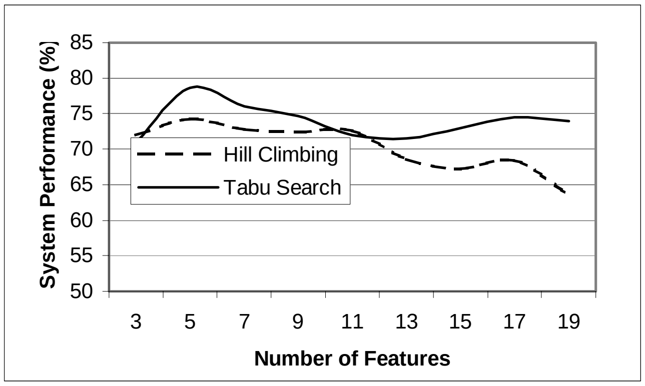

Figure 2 System Performance by Tabu Search and Hill Climbing on Different Number of Features from F_c

We can see that not surprisingly in general Tabu Search operates better than Hill Climbing. With 5 features in the case representation, the system has the highest performance. Using more features than needed decreases the system performance. We studied the similarity from different feature lists during the search and found that when there are more features, the similarities between the source cases and the training case are too close to each other and thus the retrievals are too confused to find good source cases for prediction.

Experiment 2. – Knowledge discovery on feature weights. This experiment was carried out because we found that when the features chosen are from F_c, the best performance is always obtained when their weights are the same. Table 1 shows the weights discovered for the best system performance when the features are selected from F_s that contains smaller number of simpler features.

Table 1 Weights Discovered when Features are Selected Form F_s

No. of Features	3	5	7	9
Tabu Search	{7, 8, 9}	{1, 1, 1, 1, 1}	{6, 10, 2, 5, 1, 2, 7}	{1, 9, 4, 3, 7, 5, 9, 4, 3}
Hill Climbing	{6, 7, 7}	{1, 1, 1, 1, 1}	{1, 2, 1, 3, 3, 7, 1}	{2, 2, 3, 3, 4, 5, 9, 1, 9}

We can see that when there are a lower number of simpler features to choose from, the system performs best on features with different weights. This may be because with simpler features, the similarity measure cannot get enough information for comparison between the cases. Thus the weights of these features need to be adjusted to give a better retrieval. When there are more complex features, the similarity measure may have enough information for comparison so adjusting the weights of features may not provide much improvement on the system performance. Another explanation may be that the features are much more important than their weights in the similarity measure. The feature list that gives the highest system performance in the first stage will be employed in the CBR system.

Experiment 3. – Knowledge discovery on the case base. We build up 2 case bases with a different number of source cases with different heuristics as their solutions. They are:

"CB1" – a database of 45 source cases with 10, 15, … 50 courses (9 different sizes) in the problem. For each size there is a set of 5 cases each solved with one heuristic from H.

"CB2" – a database of 90 source cases with 9 different sizes, each has a set of 10 cases, each two of them solved with one heuristic in H.

The "Leave-One-Out" strategy is used to refine the case bases. The system performance before and after this process is presented from the second column to the fourth column in Table 2. We can see that the sizes of the refined "CB1" and "CB2" are vastly reduced but higher system performance is obtained with more relevant source cases. Also we can observe that the refined "CB2" has a better performance as more source cases are trained in "CB2".

Table 2 System Performance Before and After the Second-Stage Process on Case Bases

Case Base	CB1	CB2	Refined CB1	Refined CB2	Learnt CB1	Learnt CB2
No. of Source Cases	45	90	6	8	12	13
System Performance	40%	56%	60%	66%	68%	70%

Experiment 4. – Case-based learning. We tested the refined case bases on another set of new cases. The results are slightly worse than those on the training cases used in the KD process. This means that new knowledge needs to be gradually discovered for solving more new problems. We know that CBR has the ability of learning by retaining useful cases during the problem solving. Solving new problems improves the system performance and that is presented in the last two columns in Table 2. Our study on knowledge discovery in the CBR system uses a limited number of cases. After the discovering process, the CBR system will need to learn to solve a wider range of problems that are not trained during that process.

4. Remarks and Future Work

In this work we studied knowledge discovery on CBR in a hyper-heuristic method to predict the best heuristic for timetabling problems. The observations obtained from the reasonably good experimental results may provide indication for further work on using CBR within a hyper-heuristic framework.

Experimental results show that the feature selection is more important than their weights in the retrieval if the features are elaborated enough. This gives us an indication that current features we are using to select from are reasonably good for case representation. More relevant features will be studied for a more precise case representation to improve the system performance. As irrelevant features in the case representation confuse the retrieval, refining the features selected is also an important task.

Source case selection is one of the key issues in CBR. Knowledge and problem solving experiences in timetabling need to be continuously discovered and stored in the case base. Our work investigated knowledge discovery on a limited number of artificially produced problems. Higher performance may be obtained by studying a large number of cases for the case base. Training on a higher number of real-world data needs to be carried out to obtain a wider range of knowledge in the timetabling domain.

The knowledge discovery techniques we studied in choosing heuristics may also be employed to discover knowledge in the search space that may guide the search towards a more promising region in problem solving using a variety of Artificial Intelligence methods. Our future work will also investigate more complicated hyper-heuristic methods using CBR for timetabling problems.

References

[1] Leake, D. (ed.): Case-based Reasoning: Experiences, Lessons and Future Directions. AAAI Press, Menlo Park, CA. 1996.

[2] B.L. MacCarthy and P. Jou, Case-based reasoning in scheduling. In: M.K. Khan and C.S. Wright (eds.), Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST96), (MEP Publications Ltd, 1996), 211-218, 1996.

[3] P. Cowling, G. Kendall, and E. Soubeiga, A parameter-free hyperheuristic for scheduling a sales summit. In: Proceedings of the 4th International Metaheuristic Conference (MIC2001), 127-131, 2001.

[4] E.K. Burke, S. Petrovic, Recent research directions in automated timetabling, To appear in *European Journal of Operational Research* 2002.

[5] E.K. Burke, B.L. MacCarthy, S. Petrovic, R. Qu, Knowledge discovery in hyper-heuristic using case-based reasoning on course timetabling, Accepted in the 4th Interanational Conference on the Practice and Theory of Automated Timetabling (PATAT 2002), 2002.

[6] E. Hart, P. Ross and J. Nelson, Solving a real-world problem using an evolving heuristically driven schedule, *Evolutionary Computation* **6** (1998) 61 – 80.

[7] P. Shaw, Using constraint programming and local search methods to solve vehicle routing problems, Proceedings of Principles and Practice of Constraint Programming, 417-431, 1998.

[8] H.T-Marin, P. Ross, M.V-Rendon, Evolution of constraint satisfaction strategies in examination timetabling, In Proceedings of the Genetic and Evolutionary Computation Conference, 635-642, 1999.

[9] M.W. Carter, and G. Laporte, Recent developments in practical exam timetabling. In: [11], 3-21, 1995.

[10] M.W. Carter, and G. Laporte, Recent developments in practical course timetabling. In: [12] 3–19, 1997.

[11] E.K. Burke, and P. Ross, (eds.), The First International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 1153, Springer-Verlag, 1995.

[12] E.K. Burke, and M. Carter, (eds.), The Second International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes In Computer Science 1408. Springer-Verlag, 1997.

[13] A. Schaef, A survey of automated timetabling. *Artificial Intelligence Review.* **13** (1999) 87-127.

[14] E.K. Burke, and W. Erben, (eds.), The Third International Conference on the Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science 2079, Springer-Verlag, 2000.

[15] U. Fayyad, G. Piatetsky-Shapiro, Smyth and R. Uthurusamy, (eds.) Advances in Knowledge Discovery and Data Mining, AAAI Press, Melo Park, CA, 1996.

[16] A. Freitas, A survey of evolutionary algorithms for data mining and knowledge discovery. To appear in: A. Ghosh, and S. Tsutsui, (eds.), Advances in Evolutionary Computation. Springer, 2002.

[17] E.K. Burke, J. Newall, and R. Weare, A simple heuristically guided search for the timetabling problem. Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), 574-579, 1998.