PROCEEDINGS OF THE SEAL'02, 277-281. 18-22, NOV, 2002, ORCHID COUNTRY CLUB, SINGAPORE

CASE BASED HEURISTIC SELECTION FOR EXAMINATION TIMETABLING

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

ABSTRACT

The work presented in this paper could be thought of as a case based hyper-heuristic approach for examination timetabling problems. A hyper-heuristic can be taken to be an automated approach to choose heuristics. Heuristics and meta-heuristics are employed in this capacity in [1] and [2]. In this paper the case-based paradigm is explored as a heuristic selector for examination timetabling problems. We suggest heuristics during the problem solving process by employing heuristics which worked well on previous similar situations and that are memorized in a case base. The suggestions for heuristics are made according to the knowledge of modeling the partial solutions within particular periods of problem solving with specific heuristics, which are discovered by tabu search. Experimental results show that this approach performs better than individual heuristics and they indicate much potential future work to be carried out in this area.

1. INTRODUCTION

Timetabling problems are a special type of scheduling problem that have been well studied over the last 4 decades [3], [4], [5], [6] and [7]. To solve a general timetabling problem, a certain number of events (exams, courses, meetings, etc) needs to be assigned to a limited number of time periods while satisfying (as much as possible) the required constraints. Constraints can be grouped into two types: *hard* and *soft constraints*. Hard constraints cannot be violated under any circumstances. For example, two events with common resources (such as students) cannot be assigned simultaneously. Soft constraints are desirable but not essential. Examples of soft constraints are that two events should not be scheduled consecutively or an event should be scheduled into a specific room if possible. Recent research on timetabling employs a variety of algorithms. These include tabu search (e.g. [8] and [9]), simulated annealing (e.g. [10]) and evolutionary algorithms (e.g. [11], [12] and [13]). Other approaches include graph heuristics (e.g. [14]), constraint logic programming (e.g. [15]) and knowledge-based techniques (e.g. [16] and [17]). A large number of recent papers on a variety of timetabling problems can be found in [18], [19], [20], [21] and [22].

Hyper-heuristics have recently been studied for some scheduling problems [7], [23], [24] and [25]. These papers choose heuristics to solve problems often in attempt to raise the level of generality [1], [2] and [7]. Most of the previous approaches studied in timetabling employ a variety of pre-defined heuristics to be operated directly on the problem. However, a specific heuristic developed to work well on a particular type of problem may not work well on other problems. Indeed, this is often the case. A hyper-heuristic approach solves problems by selecting heuristics to be operated during the problem solving. The aim is that such approaches would be much more flexible and capable of solving a wider range of problems by adaptable learning according to the current particular situations.

Knowledge discovery techniques are employed in our approach to obtain the knowledge within the heuristic selectors on modeling problems, comparing cases and choosing heuristics. Fayyad et al. [26] defined Knowledge discovery to be the “non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data”. Applications of Knowledge discovery are usually within ill-structured domains, which of course, is exactly what timetabling problems are. Knowledge discovery is usually carried out on databases. Case-Based Reasoning (CBR) [27] in our approach works on a case base which stores the knowledge discovered and provides heuristics for the heuristic selector to solve the timetabling problem in hand.

2. THE CASE BASED HYPER-HEURISTIC

In Artificial Intelligence, problems are often solved by searching for possible solutions in the search space and using a variety of heuristics that guide the search towards a promising region. However, experience/knowledge of employing different heuristics during the problem solving is usually discarded afterwards. The objective of the proposed case based hyper-heuristic approach is to collect and model this experience/knowledge and reuse it when solving new problems. The overall system and the knowledge discovery process on specific heuristics for different timetabling problems are discussed in the following sub-sections. Throughout the paper, *source cases* will be used to denote the cases stored in the case base.

2.1. The overall CBR system

Figure 1 presents the CBR system that we have developed.

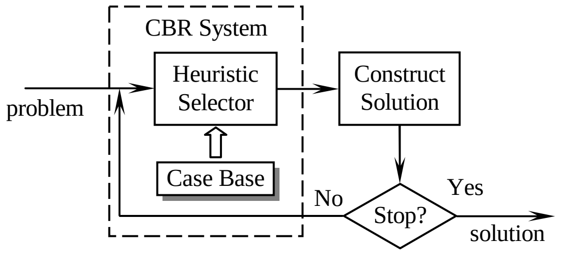

Figure 1 The case based hyper-heuristic system

To solve an examination timetabling problem input into the system, the Heuristic Selector finds different good heuristics from the case base to construct the solution step by step. In each step of the construction, the Heuristic Selector retrieves the most similar source case by using a similarity measure (presented later in formula (a)) to compare the current partial solution being constructed with source cases, which were partial solutions obtained during the problem solving of the previous problems. The best heuristic stored with this retrieved case is suggested and employed in the next step of the construction of the current solution. By retrieving specific heuristics from the case base according to the partial solutions obtained, the construction process is carried out step by step and is terminated when the stopping condition is met, which is when all of the exams are scheduled. The aim of the approach is that, by employing specific good heuristics in particular situations, good schedules can be made to produce good quality solutions.

In the CBR system developed:

- The Case Base is a collection of cases describing different possible partial solutions during the problem solving, with the suggested good heuristics that will be employed next to construct the solutions.
- Cases are all represented by a list of feature-value pairs, in which features are characteristics that describe the properties of different partial solutions obtained during the problem solving.
- Heuristics in the current system are four well-studied sequential methods in the timetabling literature that order the exams to be scheduled one by one by using heuristics. They are: Largest Degree, Largest Degree with tournament selection, Color Degree and Saturation Degree.
- The similarity measure calculates the sum of differences of the values between each pair of features in the cases being compared. This is the nearest neighborhood approach that is widely used with feature-value pair representations [29]. It is shown in formula (a):

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

where

- j is the number of features representing the cases;
- fs_i, ft_i are values of the ith feature in the source case C_s and new case C_t, which are the possible partial solutions during the problem solving.

The higher the similarity measure $S(C_s, C_t)$ is, the more similar the two cases are. The best two heuristics of the source case with the highest similarity are retrieved and suggested as good heuristics for the new case.

- The penalty function evaluates the quality of solutions and is presented in formula (b):

$$P = 10 \times S_1 + 5 \times S_2 \qquad \text{(b)}$$

where

- S_1 is the number of violations of exams being scheduled in consecutive time periods;
- S_2 is the number of violations of exams being scheduled in two time periods with only one time period in between.

The values 10 and 5 are weights that reflect the relevant importance of the constraints S_1 and S_2. They are chosen subjectively by utilizing our experience in timetabling.

2.2. Knowledge discovering in CBR

The basic assumption in CBR is that similar problems have similar solutions [27]. The retrieval process in CBR is a similarity-driven process that assesses cases by a similarity measure that compares the cases, which are represented by a list of features in the system. Thus in this hyper-heuristic approach, proper features need to be chosen in such a way that cases which they represent can be compared to make good suggestions for heuristics to use. That is, only those features of problems that can affect and contribute to the good suggestions of heuristics need to be employed to describe cases. Knowledge discovery uses the techniques that were investigated in our previous work on course timetabling [28] to discover features that model partial solutions and should be used in the Heuristic Selector. The knowledge discovery process is described below.

2.2.1. Preparation for knowledge discovery

To carry out knowledge discovery on the Heuristic Selector in the CBR system, a case base is built up which consists of a set of cases with their best heuristics. All of the heuristics being studied are implemented to construct solutions step by step for a set of examination timetabling problems. According to their performance at each step during the problem solving, these heuristics are ranked by evaluating the particular partial solutions they generated at that time. The heuristic that performs the best (makes the lowest penalty schedule) is given the highest rank and is seen as the best heuristic for the corresponding partial solution. These partial solutions, modeled by a list of possible features, and their best heuristics are then stored as source cases. It is possible that on some partial solutions, all or most of the heuristics perform the same. In this situation these solutions will not be stored as they may not be good representatives for a class of specific situations and thus may not contribute to good suggestions of heuristics.

A set of training cases is also produced with the expected heuristics that the Heuristic Selector should suggest. They are produced using the same process as that presented above. For each case (partial solution in each step of the solution construction), the best heuristic that makes the lowest penalty schedule is obtained. This heuristic will be the expected one that the Heuristic Selector should suggest. Knowledge discovery is reinforced to find the features used in the Heuristic Selector to suggest the expected heuristic in the training cases.

2.2.2. Knowledge discovery on the heuristic selector

Knowledge discovery is carried out to choose proper feature lists in case representation on the case base built up and a set of training cases whose best heuristics are obtained beforehand. The objective is to find a feature list to be used in the Heuristic Selector so that good heuristics can be selected to make the least penalty schedules in the next step of the construction of the partial solutions.

For each of the training cases, the Heuristic Selector finds the most similar source case. If the best heuristic of the training case obtained beforehand is within the best two heuristics of this retrieved source case, the suggestion is concluded as *successful* using this particular feature list within the Heuristic Selector. All of the training cases are input into the system to check the suggestions that the Heuristic Selector makes upon the feature list used. The *system performance* is defined as the percentage of the successful suggestions on all of the training cases.

Knowledge discovery is a process of adjusting the features selected to improve the system performance on all of the training cases. In the previous work, this was carried out by repeatedly investigating the reasons for failures in suggestions and changing the feature list accordingly.

In this work the knowledge discovery is seen as a search process on all of the possible combinations of features. Tabu search is used to carry out the search in the search space of different feature lists of different lengths [28]. An initial feature list is first selected randomly from a set of possible features to model all of the cases. Possible moves include changing one of the features, removing irrelevant features and introducing new features describing the characteristics of cases in the CBR system. The fitness function employed in the tabu search is the system performance obtained on all of the training cases. The feature list that gives the highest system performance on all of the training cases will be used in the Heuristic Selector within the case based hyper-heuristic approach to solve problems by employing the heuristics suggested during the problem solving.

3. EXPERIMENTS AND RESULTS

All of the timetabling problems in our system are generated by constructing a conflict matrix that defines the hard constraints between exams. Each element marked as ‘1’ in the conflict matrix denotes the conflict between the corresponding exams indicated by row and column. The density, which is the number of ‘1’s to the number of overall elements in the matrix, is from 0.65 to 0.85. The size of the problems ranges from 100 to 300 exams. All of the problem data used in this section is available online at http://www.cs.nott.ac.uk/~rxq/publications.htm.

3.1. Knowledge discovering on the heuristic selector

By carrying out all of the heuristics on a set of timetabling problems generated using the conflict matrix, 95 source cases are produced to build the case base and 95 training cases are produced and used to carry out the knowledge discovery. Tabu search is implemented to discover the feature list that gives the highest system performance on all of the training cases. Possible features include 11 different characteristics of the partial solution and the ratios between each pair of them. They are also available online at http://www.cs.nott.ac.uk/~rxq/publications.htm.

After the Knowledge discovery on feature lists, the case base is then refined by the "Leave-One-Out" strategy: Each time a source case is removed to see if the system performance is improved. Those that contribute to better system performance (which is decreased if it is removed) are retained.

Another set of 195 testing cases was generated to test the Heuristic Selector with the feature lists discovered by the tabu search presented above. The lengths of the feature lists, the number of cases in the refined case base and the system performance on both the training cases and testing cases are presented in Table 1.

Table 1 System performance on different lengths of feature lists found by tabu search

No. of features	System performance on training cases	No. of source cases	System performance on testing cases
2	97%	95	87%
3	100%	93	91%
4	100%	93	91%
5	100%	93	91%
6	92%	89	89%
7	100%	93	89%
8	87%	95	86%
9	87%	95	85%
10	85%	93	84%

From the system performance on both the training cases and testing cases we can observe that a higher number of features in the case representation does not yield better results. The system performs the best (where almost 9 out of 10 testing cases obtain the expected heuristics) with a relatively smaller number of features in the Heuristic Selector, namely from 3 to 7. This happens because when more less-relevant features are accessed in the Heuristic Selector, the similarities are too close to each other to find good suggestive source cases for the specific situation.

3.2. Problem solving using the hyper-heuristic

A set of 100 examination timetabling problems of different sizes is generated with densities ranging from 0.65 to 0.85 in the conflict matrix. These problems are solved using the hyper-heuristic approach on the CBR system with the feature lists discovered by tabu search. The average penalties of solutions obtained by individual sequential heuristics and the average penalties of solutions obtained by the case based hyper-heuristic on the feature lists of different lengths discovered are presented in Table 2 and Table 3, respectively.

Table 2 Average penalties of solutions by sequential heuristics

Sequential heuristics	CD	SD	LD	LDT
Penalty	328	202	245	252

Table 3 Average penalties of solutions by using case based hyper-heuristic with different features

No. of features	2	3	4	5	6	7	8	9	10
Penalty	182	199	193	195	196	203	197	200	203

The results shown in Table 3 indicate that, our approach provides solutions that have better average penalties than when using individual sequential heuristics (except in 2 cases – 7 and 10 features). By comparing the results with those presented in Table 1, we can see that roughly the higher the accuracy of the good suggestions on heuristics, the better the case based hyper-heuristic performs. Using the Heuristic Selector, good heuristics are suggested and thus better quality solutions are produced.

4. CONCLUDING REMARKS

In this paper we present a case based hyper-heuristic approach that chooses simple sequential heuristics using knowledge gained from solving examination timetabling problems. Based on this knowledge, the Heuristic Selector recommends good heuristics to construct solutions, aiming at minimizing the penalty of the schedule at each step. It has the ability to use heuristics adaptably during the problem solving. The aim is that the case based hyper-heuristic approach should be more flexible for solving a wider range of problems.

Of course, one of the main motivations behind exploring this heuristic selection approach is to attempt to develop systems that can operate at a higher level of generality than problem specific timetabling approaches. In addition, the approach also indicates that at least for the evaluation function used here, there is significant promise for also finding better quality solutions.

5. ACKNOWLEDGEMENTS

This work was supported by EPSRC grant number GR/N36837/01.

6. REFERENCES

[1] 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.

[2] H. T-Marin, P. Ross and M.V-Rendon, "Evolution of Constraint Satisfaction Strategies in Examination Timetabling", In Proceedings of the Genetic and Evolutionary Computation Conference, 635 – 642. 1999.

[3] M.W. Carter and G. Laporte, "Recent Developments in Practical Exam Timetabling", In: [17], 3 – 21. 1996.

[4] M.W. Carter and G. Laporte, "Recent Developments in Practical Course Timetabling", In: [19], 3 – 19. 1998.

[5] E.K. Burke, K.S. Jackson, J.H. Kingston, and R.F. Weare, "Automated Timetabling: The Sate of The Art". *The Computer Journal*, **40**(9): 565 – 571. 1997.

[6] A. Schaef, "A Survey of Automated Timetabling". *Artificial Intelligence Review*, **13:** 87 – 127. 1999.

[7] E.K. Burke, S. Petrovic. "Recent Research Directions in Automated Timetabling". *EJOR*, **140**(2): 266 – 280. 2002.

[8] D. Costa, "A Tabu Search Algorithm for Computing an Operational Timetable", *EJOR*, **76**: 98 – 110. 1994.

[9] L.D. Gaspero and A. Schaerf, "Tabu Search Techniques for Examination Timetabling", In: [20], 104 – 117. 2000.

[10] K.A. Dowsland, "Off the Peg or Made to Measure", In: [19], 37 – 52, 1998.

[11] P. Ross, E. Hart and D. Corne, "Some Observations about GA Based Timetabling", In: [19], 115 – 229. 1998.

[12] W. Erben, "A Grouping Genetic Algorithm for Graph Coloring and Exam Timetabling", In: [20], 132 – 158. 2000.

[13] M.P. Carrasco and M.V. Pato, "A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem", In: [20], 3 – 17. 2000.

[14] E.K. Burke, J. Newall, and R. Weare, "A Simple Heuristically Guided Search for the Timetabling Problem". In: Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), 574 – 579. 1998.

[15] P. Chan and G. Weil, "Cyclic Staff Scheduling Using Constraint Logic Programming", In: [20], 159 – 175. 2000.

[16] E.K. Burke, B.L. MacCarthy, S. Petrovic, and R. Qu, "Case-Based Reasoning in Course Timetabling: an Attribute Graph Approach". In: Proceedings of 4th International Conference on Case-Based Reasoning. 90 – 104. Lecture Notes in Artificial Intelligence 2080. Springer-Verlag. 2000.

[17] S. Petrovic and R. Qu, "Case-Based Reasoning as a Heuristic Selector in a Hyper-Heuristic for Course Timetabling Problems". To appear in proceedings of 6th International Conference on Knowledge-Based Intelligent Information & Engineering Systems 2000.

[18] 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.

[19] 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, 1998.

[20] 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.

[21] E.K. Burke and P. de Causmaecker (eds.), *The Forth International Conference on the Practice and Theory of Automated Timetabling*. To appear.

[22] E.K. Burke and S. Petrovic (guest editors), F*eature Issue of EJOR on Timetabling and Rostering*. To appear.

[23] E. Hart, P. Ross and J. Nelson, "Solving a Real-World Problem Using an Evolving Heuristically Driven Schedule". *Evolutionary Computation* **6**: 61 – 80, 1998.

[24] 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.

[25] J. Berger, M. Sassi and S. Salois, "A Hybrid Genetic Algorithm for the Vehicle Routing Problem with Windows and Itinerary Constraints", Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99), 44 – 51. 1999.

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

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

[28] E.K. Burke, B. MacCarthy, S. Petrovic and R. Qu, "Knowledge Discovery in Hyper-heuristic Using Case-Based Reasoning on Course Timetabling", To be published in [21].

[29] J. Kolodner, *Case-Based Reasoning*, Morgan Kaufmann Publishers, 1993.