Selected Papers from the PATAT'02, LNCS 2740. Also available in the Proceedings of PATAT'02, 90-103. Aug, 2002

Knowledge Discovery in a Hyper-Heuristic for Course Timetabling Using Case-Based Reasoning

E.K. Burke[1], B.L. MacCarthy[2], S. Petrovic[1], R. Qu[1]

[1] School of Computer Science and Information Technology, Jubilee Campus, University of Nottingham, Nottingham, NG8 1BB, U.K.
{ekb, sxp, rxq}@cs.nott.ac.uk
[2] School of Mechanical, Materials, Manufacturing Engineering and Management, University of Nottingham, University Park, Nottingham, NG7 2RD, U.K.
bart.maccarthy@nottingham.ac.uk

Abstract. This paper presents a new hyper-heuristic method using Case-Based Reasoning (CBR) for solving course timetabling problems. The term Hyper-heuristics has recently been employed to refer to "heuristics that choose heuristics" rather than heuristics that operate directly on given problems. One of the overriding motivations of hyper-heuristic methods is the attempt to develop techniques that can operate with greater generality than is currently possible. The basic idea behind this is that we maintain a case base of information about the most successful heuristics for a range of previous timetabling problems to predict the best heuristic for the new problem in hand using the previous knowledge. Knowledge discovery techniques are used to carry out the training on the CBR system to improve the system performance on the prediction. Initial results presented in this paper are good and we conclude by discussing the considerable promise for future work in this area.

1 Introduction

1.1 Case-Based Reasoning

What is Case-Based Reasoning? Many techniques from Artificial Intelligence (AI) and Operational Research (OR) solve timetabling problems directly by employing heuristics, meta-heuristics and hybrids on the problem in hand [1, 2, 3]. Case-Based Reasoning (CBR) [4] is a Knowledge-Based technique that solves problems by employing the knowledge and experience from previous similar cases. Solutions or problem solving strategies that were used in solving earlier problems (*cases*) are maintained in a store (*case base*) for reuse. Adaptation usually needs to be carried out for the new problem employs domain knowledge of some kind. The solved new problems may be retained and the case base is thus updated. Leake [5] described CBR as follows:

"In CBR, new solutions are generated not by chaining, but by retrieving the most relevant cases from memory and adapting them to fit new situations."

A case usually consists of two major parts: the problem itself represented in a certain form to describe the conditions under which it should be retrieved; and the solution of the problem or the lessons it will teach. Throughout this paper, the term *source case* is used to denote the cases in the case base and the term *target case* is used to denote the new problem to be solved.

A *similarity measure* is usually defined by a formula to calculate the similarity between source cases and the target case. The most similar source cases are retrieved for the target case. The development of this similarity measure for large real-world problems such as those encountered in course timetabling presents one of the major research challenges in this area.

Case-Based Reasoning in Scheduling and Optimization Problems. Timetabling has been studied extensively over the years [1, 2, 3, 6, 7, 8, 9]. It can be thought of as a special type of scheduling problem. The potential for CBR has been discussed for different scheduling problems [10, 11]. A brief survey of CBR in scheduling was presented in [10] where three Case-Based Scheduling systems, SMARTplan [12], CBR-1 [13] and CABINS [14], were reviewed. The authors claimed that CBR is a very good approach in expert scheduling systems and emphasized potential research in dynamic scheduling. Other studies in case-based scheduling concerned a variety of scheduling problems and issues, i.e. optimization [15], nurse rostering [16] and educational timetabling problems [17, 18].

1.2 Course Timetabling and Hyper-heuristics in Scheduling

Timetabling Problems. A general timetabling problem involves assigning a set of events (meetings, matches, exams, courses, etc) into a limited number of timeslots subject to a set of constraints. Constraints are usually classified into two particular types: *hard constraints* and *soft constraints*. Hard constraints should under no circumstances be violated. *Soft constraints* are desirable but can be relaxed if necessary.

Over the last 40 years, there has been a considerable amount of research on timetabling problems [1, 2, 3, 19]. In the early days of educational timetabling research, graph coloring [20] and integer linear programming techniques were widely used [21]. Some of the latest approaches can be seen in [6, 7].

This paper concentrates on educational course timetabling problems. Modern heuristic techniques have been successfully applied to course timetabling [6, 8, 9]. Tabu Search (e.g. [22]) and Simulated Annealing (e.g. [23]) have been successfully applied. Evolutionary Algorithms/Genetic Algorithms (GAs) (e.g. [24]) and Memetic Algorithms (that hybridizes GAs with local search techniques) (e.g. [25]) have also been extensively studied. Constrained-Based techniques have also been widely employed (e.g. [26]).

Hyper-heuristics in Scheduling. Hyper-heuristics can be defined to be "heuristics that choose heuristics" or as "algorithms to pick the right algorithm for the right situation" [27]. The main reason for using the term *hyper-heuristics* rather than the widely used tern *meta-heuristics* is that hyper-heuristics represent a method of from a variety of different heuristics (that may include meta-heuristics).

Some research in scheduling has investigated this approach although it does not always use the term "hyper-heuristics". An approach was presented in [28] on open shop scheduling problems using GAs to search a space of abstractions of solutions to "evolve the heuristic choice". GAs have been employed to construct a schedule builder that chooses the optimal combinations of heuristics [29]. Another approach in [30] used a GA to select the heuristic to order the exam in a sequential approach for exam timetabling problems. A hybrid GA investigated for vehicle routing problems has demonstrated promising results [31, 32].

Some other research on hyper-heuristics has also been carried out on a variety of scheduling problems. Guided local search was used to select from a set of heuristics and also different parameters for the traveling salesman problems [33]. In [27] a hyper-heuristic approach was used to select from a set of lower level heuristics according to the characteristics of the current search space in a sales summit scheduling problem.

2 Case-Based Heuristic Selection for Course Timetabling

2.1 Knowledge Discovery for Course Timetabling

The overall goal of our approach is to investigate CBR as a selector to choose (predict) the best (or a reasonably good) heuristic for the problem in hand according to the knowledge obtained from solving previous similar problems. The goal is to avoid a large amount of computation time and effort on the comparison and choosing of different heuristics. A large number of approaches and techniques in AI and OR have been studied to solve a wide range of timetabling problems successfully over the years. Comparisons have been carried out in some papers on using different approaches in solving a specific range of problems. Thus the development of heuristics for timetabling is very well established and a reasonable amount of knowledge does exist on which specific heuristic works well on what specific range of timetabling problems. This provides a large number of cases that can be collected, studied and stored in the case base, providing a good starting point for solving new course timetabling problems.

In knowledge engineering, techniques in knowledge discovery and machine learning have been employed with success in a number of ill-structured domains. Knowledge discovery is the process of studying and investigating a collection of datasets to discover information such as rules, regularities, or structures in the problem domain. It was defined in [34] as a "non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data". A key step in the knowledge discovery process is data mining that may employ a wide range of techniques from AI, machine learning, knowledge acquisition and statistics. Knowledge discovery is usually carried out on databases and the application areas include medicine, finance, law and engineering [35].

In our CBR system the previous most similar cases provide information that facilitates the prediction of the best heuristic for the target case. The retrieval in CBR is a similarity-driven process that is carried out on cases described in specific forms. Thus the key issues are the case representation (that should be in a proper form to describe the relevant context within the timetabling problem) and how it influences the similarity between cases which is what drives the retrieval to provide an accurate prediction on heuristic selection.

Knowledge discovery techniques are employed to extract knowledge of meaningful relationships within the case-based heuristic selector via iterative training processes on cases of course timetabling problems. There are two iterative training stages used in the process. The first stage tries to discover the representation of cases with a proper set of features and weights. The second stage trains the case base so that it contains the proper collection of source cases. Both of the processes are carried out iteratively. The overall objective is to obtain the highest accuracy on retrievals for predictions of heuristics for target cases.

2.2 Knowledge Discovery Process on Case-Based Heuristic Selection

Getting Started. In most knowledge discovery approaches, the development starts from the data preparation. Cases in the system are represented by a list of feature-value pairs. A set of features is used to describe the relevant characteristics of the timetabling problems, and a value is given for each of these features in each case. The current CBR system examines the source cases and target case that are produced artificially with specific characteristics as their *problem part*. These include problems with different size, different timeslots, different rooms, etc. Some heuristics will work well on some problems and less well on others. This means that the system has many types of problems that are studied and collected. Appendix A presents a description of the problem specifications. For every source case and target case, 5 heuristics (described in Appendix B) are used to solve the problem beforehand. By checking the penalties of the timetables produced, these heuristics are stored with each case in an ascending order as its *solution part*.

The retrieval is a similarity-driven process that searches through the case base to find the most similar source cases. The similarity measure employs a nearest-neighbor method that calculates a weighted sum of the similarities between each pair of individual features between cases. Formula (1) presents the similarity measure between the source case C_s and the target case C_t in the system:

$$S(C_s, C_t) = \frac{1}{\sqrt{\sum_{i=0}^{j} w_i * (fs_i - ft_i)^2 + 1}} \qquad (1)$$

The notation is described as follows:
j is the number of features in the case representation
w_i is the weight of the *i*th feature reflecting the relevance on the prediction
fs_i, ft_i are the values of the *i*th feature in source case C_s and target case C_t respectively
The possible values of the features describing timetabling problems are all integers (see Appendix C). So the higher the value of $S(C_s, C_t)$, the more similar the two cases are.

The performance of the system is tested on different sets of target cases. The training on the system is targeted at a reasonably high accuracy on all of the (quick) retrievals for the target cases. Within each retrieval, the best two heuristics of the retrieved case are compared with the best heuristic of the target case. If the best heuristic of the target case maps onto any of the best two heuristics of the retrieved case, the retrieval is concluded as successful. Actually, in the training processes, we found that sometimes the penalties of the timetables produced by different heuristics are close or equal to each other. We choose the best 2 heuristics to be stored with each source case so that we have the best heuristic stored and still retain some randomness.

Training on the Case Representation. An initial case base is built up which contains a set of different source cases with artificially selected specific constraints and requirements from Appendix A. An initial list of features is first randomly selected to represent cases. Each of the features is initially assigned with the same normalized weights. There are 11 features (details of the which are given in Appendix C) in the initial case representation.

Our knowledge discovery on the case representation to train the features and their weights in the system adopts the iterative methodology presented in [36]. In every iteration, we:

a) Analyze the retrieval failures.
b) Propose new features to address retrieval failures.
c) Select a discriminating set of features for the new case representation.
d) Evaluate the competence of this representation.

In our CBR system, the training for case representation is a recursive failure-driven process carried out to refine the initial features and their weights. A schematic diagram of the knowledge discovery on case representation is given in Fig. 1. The knowledge discovery process in the system includes the following steps:

Adjusting feature weights. The best two heuristics of the retrieved case are compared with the best one of the target case to see if the retrieval is successful (the best heuristic of the target case mapped onto one of the best two of the retrieved case). Adjustments on feature weights are iterative error-driven processes: the weights of the features that result in the failures of the retrieval are penalized (decreased) and those that can contribute successful retrievals are rewarded (increased) to discriminate the source cases that should be retrieved from the others that should not be retrieved.

Removing irrelevant features. After certain rounds of iterative adjustments, the weights of some of the features may be small enough to be removed from the feature list. This means that these features are either irrelevant or less important and thus are not needed in the case representation. Retaining the irrelevant features may confuse the retrieval process, as the similarities between cases may be too close to each other, thus reducing the number of the successful retrievals and decreasing the system performance.

Introducing new features. When the adjustment of feature weights does not result in a successful retrieval for a target case, new relevant features are added. New features are proposed by studying if they can distinguish the correct source case from the others, if they can give a prediction of success, or if they can express the specific characteristics in a particular case.

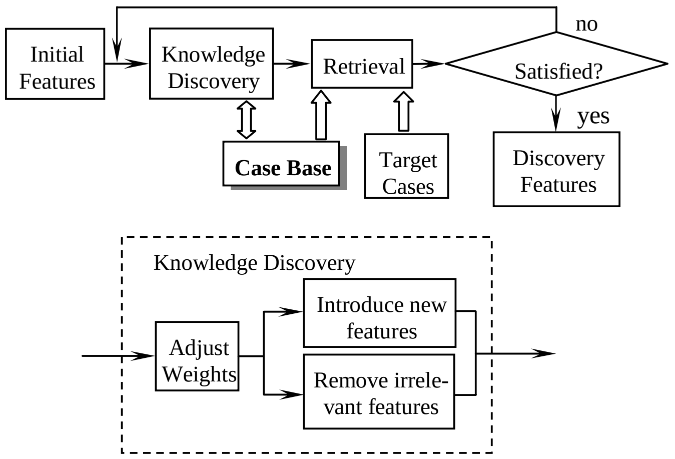

Fig. 1 Schematic Diagram of Knowledge Discovery on Featrues and Their Weights

Due to the complexity of the problem, at the beginning we do not know what features are relevant to the similarity-driven retrieval and which should be used to represent cases. Also we do not know their weights as we do not know how important they are to properly calculate the similarity that influences the heuristic selection. By using the recursive knowledge discovery process presented above, irrelevant and less important features are removed from the initial feature list. The feature vector that gives the highest accuracy on retrievals for all of the target cases will be employed as the basis for the second stage of knowledge discovery. The trained case representation (with 6 features left) after the first stage of training is presented in Appendix D.

Training on the Case Base. Case selection is a particularly important issue in building up a case base. Sometimes, keeping irrelevant source cases can decrease the system performance and increase the space and time requirements of the system. The objective of the second stage training is to select a collection of relevant cases without redundancy for the case base.

Firstly we build up two initial case bases with source cases of 9 different sizes with 10, 15, … to 50 courses in them:

a) "OneSet" – For each size, 5 source cases are produced, each has one of the 5 heuristics listed in Appendix A as its best heuristic. We name this case base "OneSet" as it contains one set of the 5 heuristics for cases with different sizes (thus in "OneSet" there are 9 * 5 = 45 source cases).
b) "TwoSet" – For each size, 10 source cases are produced, each two have one of the 5 heuristics listed in Appendix A as their best heuristic. It is named "TwoSet" and in total there are 9 * 5 * 2 = 90 source cases in "TwoSet".

The target cases are produced with the size of 10, 20, 30, … to 100 courses, for each size with 10 instances. Thus there are 10 * 10 = 100 target cases to be tested on the two initial case bases. The best heuristics for each of them is obtained beforehand to evaluate the retrieval.

A database is built up containing these two case bases and the target case set. The training process on these two initial case bases is carried out recursively using the "Leave-One-Out" strategy: Each time a source case is removed from the case base we test to see if the number of successful retrievals on the case base for all of the target cases is increased. If removing a source case decreases the number of successful retrievals, it will be restored back to the case base as it may contribute to successful retrievals for certain types of cases. Otherwise, if the number of successful retrievals increases or does not change, it will be removed from the case base as a redundant case. The process stops when the highest number of retrievals is obtained on all the target cases.

Finally after the second stage of training, there are 14 and 15 source cases left in the original two case bases, respectively. To test the system performance, an experiment is carried out on both the initial and trained case bases for another set of target cases that are, of course, not the same as those of the training set. The accuracies of the system performance on these case bases are shown in Table 1.

Table 1 Accuracies of System Performance on Initial and Trained Case Bases

Case Base	Retrieval Accuracy
"OneSet" (45 cases)	42%
"TwoSet" (90 cases)	60%
Trained Case Base from "OneSet" (15 cases)	70%
Trained Case Base from "TwoSet" (14 cases)	71%

We can observe that the initial “TwoSet” provides better performance than that of “OneSet”. By storing more source cases, the system is equipped with more knowledge and thus is capable of providing better performance during the retrieval. We can also see that the second training process removes quite a lot of source cases that are redundant or that are harmful to the performance of the CBR system. With a smaller number of more relevant source cases retained in the case bases, the system performance is improved to provide higher accuracies of predictions of suitable heuristics. To obtain better system performance, a higher number of relevant source cases need to be selected in the case base.

3 Conclusion and Future Work

This paper presents the first step of our work in developing a hyper-heuristic method using CBR for heuristic selection on course timetabling problems. Knowledge discovery techniques employ relatively simple methods and just a few training processes are carried out. The results are good and indicate the possible advantages of employing knowledge discovery techniques in the course timetabling domain. We believe better results may be obtained after further training processes are carried out which employ a range of knowledge discovery techniques.

There are many more complex and elaborate techniques that can be investigated and integrated into the CBR system to improve its performance. For example, for the case representation we currently employs a simple technique that is manually carried out to choose the features and adjust their weights. This can be seen as a feature selection task, which is the problem of selecting a set of features to be used as the data format in the system to achieve high accuracy of prediction. Feature selection is an important issue in machine learning [37] for which a variety of traditional techniques exist. Some recent work employing AI methods such as evolutionary algorithms [38] to optimize the feature selection also provide a wider range of possible research directions. For complex timetabling problems, these more efficient algorithms can be employed to carry out the searching on features more effectively when dealing with larger data sets. Our future work will study and compare these different techniques to optimize the case representation to improve the system performance on a wider range of larger timetabling problems. New features are being studied and introduced into the system. For example, some refined features such as the number of rooms with a range of capacities, the number of courses with more than a certain number of constraints, etc can be introduced to give a more specific description of problems. Other issues relating to knowledge discovery in the CBR system may include how to deal with the incomplete data in case bases and how to involve domain knowledge in the system. User interaction in knowledge discovery is also important on tasks like judgment and decision-making, in which humans usually perform better than a machine.

The current system uses 5 simple heuristics to implement the analysis and testing on the case-based heuristic selection. Future work will study more heuristics in the system. Also the testing cases are artificially produced to give a systematic analysis on as many types of problem as possible. After the initial study of using CBR as a heuristic selector we have increased our understanding of the area. Real-world benchmark timetabling data (such as that presented in [39]) will be collected and stored in the case base for solving real-world problems. Adaptation may also need to be conducted to utilize domain knowledge on some of the heuristics retrieved for the new problem.

4 References

1. Carter, M.W., Laporte, G.: Recent developments in practical examination timetabling. In: [6]. (1995) 3 – 21
2. Carter, M.W., Laporte, G.: Recent developments in practical course timetabling. In: [9]. (1997) 3 – 19
3. Schaef, A.: A Survey of Automated Timetabling. Artificial Intelligence Review. **13** (1999) 87 – 127
4. Kolodner, J.L. Case-Based Reasoning. Morgan Kaufmann, 1993.
5. Leake, D. (ed.): Case-Based Reasoning: Experiences, Lessons and Future Directions. AAAI Press, Menlo Park, CA. 1996.
6. Burke, E.K., Erben, W. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers from the Third International Conference. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin. 2000.
7. Burke, E., Petrovic, S.: Recent research directions in automated timetabling. European Journal of Operational Research. 140(2) (2002) 266 – 280
8. Burke, E.K., Ross, P. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin. 1995
9. Burke, E.K., Carter, M. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference. Lecture Notes In Computer Science 1408. Springer-Verlag, Berlin. 1997.
10. MacCarthy, B.L., Jou, P.: Case-based reasoning in scheduling. In: Khan MK, Wright CS (eds.). Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques (AMPST96), (MEP Publications Ltd, 1996), (1996) 211 – 218
11. Schmidt, G.: Case-based reasoning for production scheduling. International Journal of Production Economics, **56-57** (1998) 537 – 546
12. Koton, P.: SMARTlan: A case-based resource allocation and scheduling system. Proceedings of Workshop on Case-Based Reasoning (DARPA). (1989) 285 – 289
13. Bezirgan, A.: A case-based approach to scheduling constraints. In: Dorn, J. and Froeschl, K.A. (eds.), Scheduling of Production Processes. Ellis Horwood Limited. (1993) 48 – 60
14. Miyashita, K., Sycara, K.: CABINS: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. Artificial Intelligence, **76** (1995) 377 – 426
15. Cunningham, P., Smyth, B.: Case-based reasoning in scheduling: reusing solution components. The International Journal of Production Research. **35** (1997) 2947 – 2961.
16. Scott, S., Simpson, R., Ward, R.: Combining case-based reasoning and constraint logic programming techniques for packaged nurse rostering systems. Proceedings of the Third UK Case-Based Reasoning Workshop. 1997.

17. Burke, E.K., MacCarthy, B. Petrovic, S., Qu, R.: Structured cases in CBR – re-using and adapting cases for timetabling problems. Journal of Knowledge-based System, **13**(2-3) (2000) 159 – 165
18. Burke, E.K., MacCarthy, B.L. Petrovic, S., Qu, R.: Case-based reasoning in course timetabling: an attribute graph approach. In: Aha, D.W. and Watson, I. (eds.): Case-Based Reasoning Research and Development. Proceedings of 4th International Conference on Case-Based Reasoning (ICCBR2001). Lecture Notes in Artificial Intelligence 2080. Vancouver, Canada. (2000) 90 – 104
19. Burke, E.K., Jackson, K.S., Kingston, J.H., Weare, R.F.: Automated timetabling: the sate of the art. The Computer Journal, **40**(9) (1997) 565 – 571
20. Werra, D.: Graphs, hypergraphs and timetabling. Methods of Operations Research (Germany F.R.), **49** (1985) 201 – 213
21. Carter, M.W.: A lagrangian relaxation approach to the classroom assignment problem. IFOR. **27**(2) (1986) 230 – 246
22. Costa, D.: A Tabu Search Algorithm for Computing an Operational Timetable. EJOR, **76** (1994) 98 – 110
23. Abramson, D.: Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms. Management Science, **37**(1) (1991) 98 – 113
24. Corne, D., Ross, P.: Peckish Initialisation Strategies for Evolutionary Timetabling. In: [6], (1995) 227 – 240
25. Carrasco, A.P., Pato, M.V.: A Multiobjective Genetic Algorithm for the Class/Teacher Timetabling Problem. In: [19]. (2000) 3 – 17
26. Zervoudakis, K., Stamatopoulos, P.: A Generic Object-Oriented Constraint-Based Model for University Course Timetabling. In: [19], (2000) 28 – 47
27. Burke E., Kendall, G, Newall, J. Hart, E., Ross, P and Schulenberg S: Hyper-heuristic: An Emerging Direction in Modern Search Technology. In: Glover and Kochenberg (eds.) Handbook of Meta-Heuristics. Kluwer 2003, 457 – 474
28. Fang, H.L., Ross, P., Corne, D.: A Promising Hybrid GA/Heuristic Approach for Open-Shop Scheduling Problems. The 11th European Conference on Artificial Intelligence (ECAI'94). John Wiley & Sons, Ltd, 1994.
29. Hart, E., Ross, P., Nelson, J.: Solving a Real-world Problem Using an Evolving Heuristically Driven Schedule. Evolutionary Computation, **6**: (1998) 61 – 80
30. Terashima-Marin, H., Ross, P., Valenzuela-Rendon, M.: Evolution of Constraint Satisfaction Strategies in Examination Timetabling. Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99), Morgan Kaufmann, (1999) 635 – 642
31. Shaw, P.: Using Constraint Programming and Local Search Methods to Solve Vehicle Routing Problems. Proceedings of CP-98. (1998) 417 – 431
32. Berger, J., Sassi, M., Salois, S.: 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), Morgan Kaufmann, (1999) 44 – 51
33. Voudouris, C., Tsang, E.P.K.: Guided Local Search and Its Application to the Travelling Salesman Problem. European Journal of Operational Research, Anbar Publishing 113(2) (1999) 469 – 499
34. Fayyad, U., Piatetsky-Shapiro, G., Smyth, P.: From Data Mining to Knowledge Discovery in Databases. In: Fayyad, U., Piatetsky-Shapiro, G., Smyth, Uthurusamy, R. (eds.): Advances in Knowledge Discovery and Data Mining. AAAI Press, Melo Park, CA, (1996) 1 – 34
35. Piatetsky-Shapiro, G.: Knowledge Discovery in Databases. AAAI Press, 1991.
36. Cunningham, P., Bonzano, A.: Knowledge Engineering Issues in Developing a Case-Based Reasoning Application. Knowledge-Based Systems, **12** (1999) 371 – 379

37. Hall, M.A., Smith, L.: Practical Feature Subset Selection Machine Learning. Proceedings of the Australian Computer Science Conference. 1996.
38. Freitas, A: A Survey of Evolutionary Algorithms for Data Mining and Knowledge Discovery. To appear in: Ghosh, A., Tsutsui, S. (eds.): Advances in Evolutionary Computation. Springer, 2002.
39. Carter, W.M., Laporte, G.: Examination Timetabling: Algorithmic Strategies and Applications, Journal of the Operational Research Society, **74** (1996) 373 – 383
40. Burke, E., Newall, J. and Weare, R.: A Simple Heuristically Guided Search for the Timetabling Problem. Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98), (1998) 574 – 579

Appendix A: Course Timetabling Problems Specification

Hard constraints:
1. Two courses cannot be scheduled into the same timeslot
2. A course should be carried out n times a week
3. Each course has a specific room requirement with type and capacity
4. There is a specified number of periods for each course timetabling problem

Soft constraints:
1. One course should be scheduled before or after another
2. Inclusive/exclusive - a course should/should not be scheduled into a fixed timeslot
3. Consecutive - a course should/should not be scheduled into a timeslot consecutive to that of another

Appendix B: Heuristics Used in the System

1. LD – Largest degree first.
   All the courses not yet scheduled are inserted into an “unscheduled list” in descending order according to the number of *conflicts* the course has with the other courses. This heuristic tries to schedule the most difficult courses first.
2. LDT – Largest degree first with tournament selection.
   This heuristic is presented in [40]. It is similar to LD except that a course employing tournament selection is selected from a subset of the “unscheduled list”. Here, a probability value of 30% is used to get a subset from the list. This heuristic tries to schedule the most difficult courses first but also give some randomness.
3. HC – Hill climbing.
   An initial timetable is constructed randomly then is improved by hill climbing.
4. CD – Color degree.
   Courses in the “unscheduled list” are ordered by the number of *conflicts* they have with those courses that are already scheduled in the timetable. Usually those courses with a large number of such *conflicts* are harder to schedule than courses with a smaller number of conflicts.
5. SD – Saturation degree.

Courses in the "unscheduled list" are ordered by the number of periods left in the timetable for them to be scheduled validly. This heuristic gives higher priority to courses with fewer periods available.

Appendix C: Initial Features and Their Weights for Cases

f_0: number of hard constraints / number of events
f_1: number of soft constraints / number of events
f_2: number of constraints / number of events
f_3: number of periods / number of events
f_4: number of rooms / number of events
f_5: number of not consecutive courses / number of constraints
f_6: number of consecutive courses / number of constraints
f_7: number of hard constraints / number of constraints
f_8: number of soft constraints / number of constraints
f_9: number of hard constraints / number of periods
f_{10}: number of soft constraints / number of periods
normalized weight $w_i = \text{factor}_i * 1$ / sum of weights of all the features
initial $\text{factor}_i = \{1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1\}$

Appendix D: Trained Features and Their Weights

f_0: number of exclusive courses / number of events
f_1: number of inclusive courses / number of events
f_2: number of constraints / number of events
f_3: number of rooms / number of events
f_4: number of hard constraints / number of periods
f_5: number of not consecutive courses / number of constraints
normalized weight $w_i = \text{factor}_i * 1$ / sum of weights of all the features
$\text{factor}_i = \{45, 10, 10, 15, 30, 6\}$