Journal of Scheduling, 9: 99-113, 2006.

CASE-BASED HEURISTIC SELECTION FOR TIMETABLING PROBLEMS

Edmund K. Burke, Sanja Petrovic, Rong Qu*

Automated Scheduling Optimisation and Planning (ASAP) Group, School of Computer Science and Information Technology, Jubilee Campus, University of Nottingham, Nottingham, NG8 1BB, U.K.

SUMMARY

This paper presents a case-based heuristic selection approach for automated university course and exam timetabling. The method described in this paper is motivated by the goal of developing timetabling systems that are fundamentally more general than the current state of the art. Heuristics that worked well in previous similar situations are memorized in a case base and are retrieved for solving the problem in hand. Knowledge discovery techniques are employed in two distinct scenarios. Firstly, we model the problem and the problem solving situations along with specific heuristics for those problems. Secondly, we refine the case base and discard cases which prove to be non-useful in solving new problems. Experimental results are presented and analyzed. It is shown that case based reasoning can act effectively as an intelligent approach to learn which heuristics work well for particular timetabling situations. We conclude by outlining and discussing potential research issues in this critical area of knowledge discovery for different difficult timetabling problems.

KEY WORDS: case based reasoning, course timetabling, exam timetabling, graph heuristics, knowledge discovery, meta-heuristics

*Correspondence to: Dr. Rong Qu, School of CSiT, Jubilee Campus, University of Nottingham, Nottingham, NG8 1BB, U.K. Telephone: ++44-115-8466503, Fax: ++44-115-8467591, e-mail: rxq@cs.nott.ac.uk

Journal of Scheduling, 9: 99-113, 2006.

1. INTRODUCTION

1.1 Case based reasoning

Case Based Reasoning (CBR) is a knowledge-based technique that solves new problems by employing previous experiences. It is described by Leake [44] 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."*

In CBR, previous problems and their solutions are modelled as *cases* in a certain way (such as a list of feature-value pairs). They are defined by Kolodner and Leake [42] as follows:

> *"A case is a contextualized piece of knowledge representing an experience that teaches a lesson fundamental to achieving the goals of the reasoner."*

To build a CBR system, previous problems and their good solutions (the *source* cases) are collected and stored in a *case base*. The retrieval process compares the source cases with the new case in hand (the problem to be solved) by using a similarity measure (usually defined by a formula). The overall idea is to solve a new problem by reusing the good solutions of the source case(s) which is the most similar to the new problem.

CBR has been applied successfully in a variety of research areas including diagnosis, legal advice, health and education (e.g. [46]). Such examples represent situations which are naturally easy to be modelled as cases using a list of features and which can be compared using a nearest neighbour approach as a similarity measure. Recent CBR research has been carried out with some success on more complex problem domains such as scheduling (e.g. [3, 8, 9, 14, 15, 16, 17, 32]). By remembering and reusing experience from previous similar problem solving, CBR is capable of providing reasonably good solutions within a limited amount of time, which is one of the most important requirements in dynamic scheduling. This has led to the situation where most applications studied in CBR for scheduling have been in the area of dynamic/reactive scheduling [33, 48, 56]. Other applications in scheduling, employing CBR, include travelling salesman

problems and single machine scheduling [32], nurse rostering [3, 57] and timetabling problems [14, 15, 16]. Most of this research is largely concerned with attempting to solve scheduling problems *directly*. However, the work described in this paper is concerned with choosing the right *heuristic* to be employed in an attempt to raise the level of generality at which timetabling systems can operate.

1.2 Timetabling problems

Timetabling problems arise in many real-world circumstances (e.g. nurse rostering [6], sports timetabling [35] and university timetabling problems [26, 27, 55]). A general timetabling problem includes scheduling a certain number of events (exams, courses, meetings, etc) into a limited number of time periods, while satisfying as many of the required constraints as possible. Constraints are usually grouped in the following way:

- HARD CONSTRAINTS cannot be violated under any circumstances. For example, two events with common resources (such as students) cannot be assigned simultaneously. A solution with no violations of hard constraints is often called a *feasible* solution.
- SOFT CONSTRAINTS are desirable but not essential. Examples are that two events should not be scheduled consecutively, or that an event should be scheduled into a specific room if possible.

Timetabling problems have been very well studied for more than 4 decades (e.g. see [12, 21, 49, 55]). Approaches include graph heuristics (see [7]), integer linear programming (e.g. [25]) and constraint logic programming (e.g. [29]) Recently a large amount of successful research has been carried out which has investigated meta-heuristic approaches for a variety of timetabling problems (e.g. see [22, 4, 10, 5, 23]). These include tabu search (e.g. [31, 39]), simulated annealing (e.g. [34]) and evolutionary algorithms (e.g. [19, 24, 36, 53]). Extensive work has also been carried out to study and compare different heuristics on specific timetabling problems (e.g. [30, 34]). In addition, fuzzy methodologies have recently been explored for both course and

exam timetabling (see ([1, 2, 50]). The high level of recent research activity serves as a good starting point for employing knowledge-based techniques for timetabling problems [38, 43] as more and more experiences are obtained and collected. This also provided us with the motivation for exploring CBR methodologies for directly solving university timetabling problems (e.g. [14, 15, 16]) as a large amount of data and problem solving experiences are available to be studied. Note that in this paper we are not attempting to directly solve timetabling problems. Instead we are concerned with underpinning the development of general timetabling systems by investigating CBR as a timetabling heuristic selector.

1.3 Case-based heuristic selection for timetabling problems

In artificial intelligence and operational research, problems are often solved by employing a variety of heuristics to guide the problem solving towards high quality solutions in promising regions of the search space. However, experience/knowledge for problem solving that has been gained by employing different heuristics is usually discarded afterwards. This paper investigates a case based heuristic selection approach that collects and reuses previous experience of the heuristics that were employed successfully within particular situations onto current 'similar' situations in timetabling. The aim is to develop intelligent systems that can 'guess' at which heuristic will work well on which problem, and thus is capable of dealing with any problem efficiently. Success in this area would underpin the development of general systems that do not need to be tailored to the particular problem in hand. Such a system would be significantly cheaper to implement than current 'problem specific' systems because they could be applied to a far wider range of problems. The motivation behind CBR is that humans often solve new problems by re-employing knowledge that has been collected from previous experience.

Most of the timetabling research approaches employ specifically tailored heuristics which are operated directly on the problem (e.g. see [22, 4, 10, 5, 23]). However, it is often the case that a specific heuristic that worked well on a particular type of problem may not work well on other problems. In our case-based heuristic selection approach, timetabling problems are solved

by adaptively employing the best heuristics predicted for particular situations by utilising past experience. The idea is similar to that of a hyper-heuristic which refers to a "*heuristic to choose heuristics*" (e.g. see [11, 52]). Some recent research on hyper-heuristics for timetabling has been undertaken (e.g. [8, 13, 18, 40, 41, 47, 54]) in an attempt to raise the level of generality of timetabling systems.

In our proposed case based heuristic selection approach, a list of feature-value pairs is employed to model the problems into cases. A knowledge discovery process on the features to represent the cases and on the case base in the system is presented in Section 2. We are concerned with both university course timetabling and university exam timetabling, which are usually very different problems with different possible constraints and problem characteristics. In total, we collected four different sets of potential features, simple features, and a combination of simple and complex features for both the course and exam timetabling problems. We study them separately in two different heuristic prediction approaches, but within a unified case based heuristic selection framework. They are presented in Section 3, for course timetabling, and Section 4, for exam timetabling, respectively. Finally Section 5 presents some discussions on research issues concerning data preparation and source case selection on this case based heuristic selection approach together with some concluding comments and future work.

2. KNOWLEDGE DISCOVERY IN CASE-BASED HEURISTIC SELECTION

The basic assumption behind CBR is that "similar problems will have similar solutions" [44]. The retrieval is a similarity-driven process that finds the solution by assessing the similarities between source cases and new cases. The key issue in tackling the development of a case based heuristic selector for timetabling is how to model the knowledge of which specific heuristics work well for particular problems and problem solving situations. A successful method should allow us to find the most similar source cases that give good predictions of the best heuristics for the new cases. This may be obtained in two ways:

1) Proper features need to be employed to model the problems into cases so that they can be compared to make good predictions of heuristics. That is, only those features that can affect and contribute to good suggestions of heuristics are needed to be used to represent cases;

2) Source cases need to be selected carefully so that the 'right' one can be retrieved for suggesting the appropriate heuristic for a specific range of new cases.

Our work studies a two-stage knowledge discovery process on the case representation and the source cases selection. Knowledge discovery is a process of "identifying valid, novel, potentially useful, and ultimately understandable patterns in data" [37]. It is usually carried out within ill-structured domains, which is exactly what timetabling problems are. In our approach, knowledge discovery techniques and strategies are employed to obtain the knowledge of modeling problems, comparing cases to choose heuristics and refining the case base. CBR serves as the base that stores the discovered knowledge and provides good heuristics for the problem in hand.

Particular features of the CBR system that we have developed can be itemised as follows:

- Cases (either overall problems or partial situations during problem solving) are represented by a list of feature-value pairs. Knowledge discovery is carried out to search for the feature list to be employed in the system.

- The case base is a database of problems with the best two heuristics from the heuristic sets (presented in Section 3 and Section 4, respectively) for each problem. Knowledge discovery is carried out to refine the source cases that should be retained in the case base.

- Training cases are a set of cases whose best heuristics are obtained beforehand. They are used in the knowledge discovery process to carry out the search for features and selection of source cases.

- The similarity measure presented in formula (a) employs a nearest-neighbour approach that calculates the weighted sum of differences between each pair of features within the two cases being compared.

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

The notations in formula (a) are described as follows:

j is the number of features employed in the case representation

w_i is the weight of the ith feature indicating its relevance to the comparison

fs_i, ft_i are the values of the ith feature in the source case C_s and the new case C_t in hand, respectively

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

- Retrieval uses the similarity measure to compare each source case and the new case in hand. During the knowledge discovery process, a set of training cases is used to discover the feature list and source cases. If the best heuristic of the training case obtained beforehand is within one of the best two heuristics of the retrieved source case, the retrieval will be seen as a *successful* retrieval. This criterion is set this way as we found that sometimes the qualities of the timetables produced by different heuristics are very close or even equal to each other. To have the best heuristic stored while still having some randomness for exploitation, we choose to store the best two heuristics for each source case.

- System performance is evaluated during the knowledge discovery process to guide the training on the features selected and the source cases retained in the system. It is defined as the percentage of successful retrievals for all of the training cases.

- In total, we studied 5 heuristics in our case-based heuristic selection approach for course and exam timetabling problems. They are described below:

    1. h_1 = Largest degree first – All of the events that are not yet scheduled are scheduled one by one in the descending order of the number of constraints the event has with the others.

    2. h_2 = Largest degree with tournament selection – This heuristic tries to schedule the most constrained events first but still has an element of randomness [20]. Every time the most constrained event is selected from a subset (we use 30%) of the events that are not yet scheduled.

    3. h_3 = Colour degree – Events that are not yet scheduled are ordered by the number of constraints they have with those that are already scheduled in the timetable.

    4. h_4 = Saturation degree – Events that are not yet scheduled are ordered by the number of periods available for them in the timetable.

    5. h_5 = Hill climbing – The initial timetable is randomly constructed and improved by this method.

2.1 The first stage of knowledge discovery for case representation

A schematic diagram of the knowledge discovery process for case representation is presented in Figure 1. Firstly an initial feature list with the same weights is randomly chosen. Search methods (such as tabu search and hill climbing) are employed to search for the feature vector that provides the best system performance (highest number of successful retrievals for all the training cases). For each feature list selected, retrieval compares the best heuristics obtained beforehand for all of the training cases with those best heuristics for the source cases retrieved for them. The system performance upon the feature list selected will be fed back for the next step of knowledge discovery in case representation.

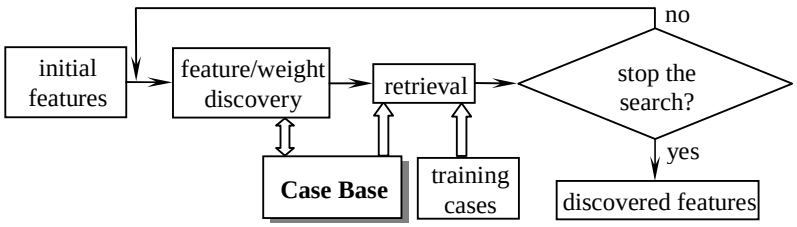

Figure 1 Schematic diagram of knowledge discovery process on features and their weights

In the searching process to discover the features and their weights, the search space includes all of the possible enumeration of features that describe the situations represented by timetabling problems. Possible moves include changing one of the features or their weights, removing irrelevant features and introducing new features. The fitness function is the system performance (percentage of successful retrievals on all of the training cases) obtained. Our current work aims to investigate the manipulation of 'knowledge' of heuristics for solving timetabling problems with the goal of raising the generality of the timetabling system. We thus employed tabu search and hill climbing which work relatively fast in obtaining reasonably good results. In tabu search, the length of the tabu list is set as 9 from a series of tests and the value recommended from the literature [51]. This process is carried out iteratively until the stopping condition is met (here a pre-set time for searching). The feature vector that gives the highest system performance will be employed in the next stage of knowledge discovery on the case base.

2.2 The second stage of knowledge discovery on the case base

Source case selection is one of the key elements that contribute to the good performance of a CBR system. Irrelevant source cases may contain wrong or redundant information, which confuses the retrieval process and decreases the system performance. Knowledge discovery on the case base uses the "Leave-One-Out" strategy that removes one source case at a time from the case base and checks its effect on the system performance. If the system performance is decreased (the number of successful retrievals for all of the training cases is decreased), the removed source case will then be

added back as it may contribute to a higher system performance on providing the best heuristics for certain types of timetabling cases. Otherwise, it will be removed permanently as either it contains wrong information in terms of predicting the best heuristics, or it is redundant in the system and thus is harmful to system performance. The aim is to obtain a case base of only the relevant and representative source cases.

3. CASE-BASED HEURISTIC SELECTION FOR OVERALL COURSE TIMETABLING PROBLEMS

We firstly investigate the case-based heuristic selection approach to predict the best heuristic for solving course timetabling problems. Two sets of features are collected to represent the problems and are employed within the knowledge discovery process. F_{cs} includes 10 simple features (presented in Table 1) and F_{cc} includes combinations of each pair of features in F_{cs}. H_c is a set which includes a selection of good timetabling heuristics. The three sets are listed below:

- $H_c = \{h_1, h_2, h_3, h_4, h_5\}$
- $F_{cs} = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9\}$
- $F_{cc} = \{f_i, f_m/f_n\}, f_i, f_m, f_n \in F_{cs}, i, m, n = 0, \ldots 9, m \neq n$

To guarantee the complete coverage of the heuristics in our set, all of the heuristics need to have at least one instance in the case base for knowledge discovery. We build up two case bases which consist of a different number of source cases with different best heuristics. 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 of which have one of the heuristics in H_c as its best heuristic. The second best heuristic of each case may be any of the other heuristics from H_c.

- *CB2* – a database of 90 source cases with 9 different sizes, each has a set of 10 cases. Each two of the 10 cases have one of the heuristics from H_c as their best heuristic. Compared with *CB1* it is more likely that more knowledge may be discovered from *CB2* for heuristic prediction.

Table 1 Possible features of course timetabling problems

feature ID	description
f_0	number of courses
f_1	number of time periods
f_2	number of constraints
f_3	number of rooms
f_4	number of hard constraints
f_5	number of soft constraints
f_6	number of courses that should be scheduled to a fixed time period
f_7	number of courses that should not be scheduled to a fixed time period
f_8	number of courses that should be scheduled to consecutive time periods
f_9	number of courses that should not be scheduled to consecutive time periods

We try to obtain the best feature set in the knowledge discovery process by comparing the heuristics that generate the best quality timetables with the least number of violations on all types of soft constraints. In the case based heuristic selection approach, our main concern is to compare and study the heuristics that generate the best schedule for a range of problems. In this phase of the research, for comparison purposes we start with feasible solutions where there is no violation of hard constraints during the training phase on the system. When an infeasible solution is generated, we either run the heuristics with random elements (h_2 and h_5) again to obtain a feasible solution, or simply discard the data tested.

3.1 Knowledge discovery on features and weights

We employ the relatively simple and fast search methods of tabu search and hill climbing, to search for the features from F_{cc} for case representation and to study the characteristics of different search

methods within the knowledge discovery process presented in Figure 1. The investigation of other search methods within the knowledge discovery process for feature selection will be a future research issue.

In total 100 training cases are used to carry out the knowledge discovery on features and their weight on the system with *CB1* as the case base. The best system performance (the highest number of successful retrievals on all of the training cases) on employing different numbers of features from feature set F_{cc} is presented in Figure 2. For each number of features, 10 runs of the system are carried out and the average system performance is obtained and presented.

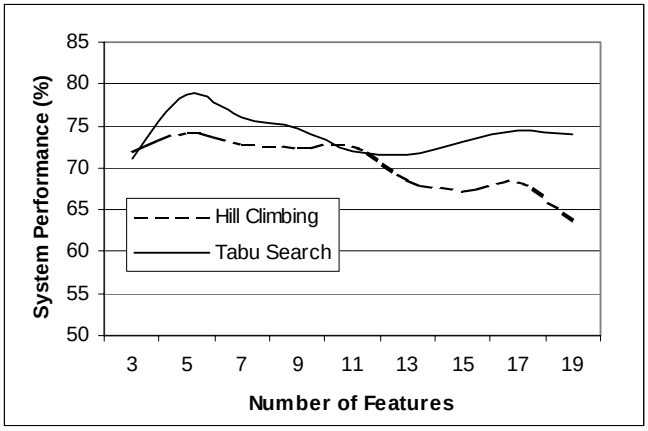

Figure 2 System performance on different numbers of features from F_{cc} by tabu search and hill climbing

We can see that (not surprisingly), in general, tabu search operates better than hill climbing. Both tabu search and hill climbing show that the system performs the best with 5 features in the case representation. With the 5 features obtained by tabu search, around 77 out of 100 training cases obtain the correct best heuristics from the retrieved source cases. We can also observe that the system performance is decreased when more than 5 features are employed in the case representation. We studied the similarities between the source cases and the training cases upon different numbers of features. It is observed that when there are more features being assessed in the

similarity measure, the similarities between many source cases and the training case are too close to each other. The similarity measure thus may not distinguish the most similar source cases for heuristic prediction as they may have just a slightly higher similarity value than the others and thus may not be the 'right' one to be retrieved. The retrieval is thus confused with these less-relevant features that make the similarities between cases too close. In general, employing more less-relevant features than needed decreases the system performance.

We also found that with the features chosen from a feature set (F_{cc}) that contains a higher number of complex features generated from simple features, the best system performance is always obtained when their weights are the same. This means that, moves that adjust the weights of the features in both tabu search and hill climbing do not contribute to better system performance. We thus test the knowledge discovery on the system with the features chosen from another feature set F_{cs} (presented above) which contains a smaller number of 10 simple features. The weights of the features are studied to detect the effect of different feature sets. Table 2 presents the simple features and their weights discovered when the highest system performance is obtained. The third and fourth row present the weights in brackets for the features presented in the second row discovered by tabu search and hill climbing, respectively.

Table 2 Weights discovered for features from F_{cs} by tabu search and hill climbing

no. of features	**3**	**5**	**7**	**9**
features	{f_1, f_4, f_6}	{f_3, f_4, f_7, f_8, f_9}	{f_0, f_1, f_3, f_4, f_5, f_6, f_7, f_8}	{f_0, f_1, f_3, f_4, f_5, f_6, f_7, f_8, f_9}
weights by 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}
weights by 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 with F_{cs}, where there are a lower number of simple features to choose from for the case representation, the weights for features discovered by tabu search and hill climbing are different except when 5 features are employed. This may be because, with simple features, the

similarity measure cannot get enough information to compare the cases, and thus the weights of these features need to be adjusted to obtain better assessments between cases during retrievals. With more complex features from F_{cc}, the similarity measure may have enough information for comparison and thus adjusting the weights of features does not make much contribution to better system performance. Another explanation may be that different complex features that are generated from simple features may contain the same simple features more than once in the similarity measure. For example, f_i/f_j and f_i/f_k in the feature list contribute to the weight of 2 for f_i in the similarity measure. So the similarity measure may in some way be affected by the importance of certain features.

The feature list that gives the highest system performance in the first stage will be employed in the CBR system for the second stage of knowledge discovery on the case base. It is shown below:

$F = \{f_2/f_1, f_9/f_1, f_2/f_7, f_1/f_0, f_2\}$

3.2 Knowledge discovery on the case base

Knowledge discovery on the case base is carried out on the two case bases *CB1* and *CB2* by using the "Leave-One-Out" strategy. The system performance before and after the "Leave-One-Out" strategy is presented in Table 3. The number of cases left in the case bases and the corresponding system performance after the "Leave-One-Out" strategy is presented in the column of *refined CB1* and *refined CB2*. We can see that the sizes of the *refined CB1* and *refined CB2* are vastly reduced but a higher system performance is obtained compared with that of case bases before the "Leave-One-Out" strategy (presented in the second column and the third column). This is obtained by retaining only the more relevant cases in the case base. The *refined CB2* provides a better system performance than the *refined CB1*, as more source cases are trained in *CB2* and thus more knowledge is discovered for solving more types of problems.

Table 3 System performance before and after the knowledge discovery on case bases

case base	***CB1***	***CB2***	***refined CB1***	***refined CB2***
no. of source cases	45	90	6	8
system performance	40%	56%	60%	66%

This section presented an approach using CBR for predicting the best heuristic for course timetabling problems. Experimental results show that the feature selection is more important than feature weights in the retrieval process to compare the cases if the features are good enough to give necessary information. This gives us an indication that the current features we are using to select from are reasonably good for case representation in timetabling problems. Employing more less-relevant features in case representation confuses the retrieval and thus decreases the system performance. Source case selection, which removes the redundant cases and retains only the representative cases, is also a key issue in this approach.

Knowledge discovery techniques in this approach employ relatively simple methods and just a few training processes lead to satisfying results. By providing the best or reasonably good heuristics without trying to tailor particular heuristics for specific course timetabling problems, this heuristic selection framework indicates the advantages of employing knowledge discovery techniques to develop a much more flexible approach for general timetabling problems.

4. CASE-BASED HEURISTIC SELECTION DURING PROBLEM SOLVING FOR EXAM TIMETABLING PROBLEMS

We also investigated the case-based heuristic selection approach to suggest the heuristics to be employed for constructing the exam timetables. The aim is that, by employing specific good heuristics in particular situations adaptively we can generate high quality schedules. Once again, we are motivated by the goal of developing more general timetabling systems. Cater and Lapore in 1996 [26] said (of examination timetabling), "*there have been hundreds of research papers on the subject and probably thousands of computer programs written (mostly by amateurs at*

each of the schools) to 'solve' their own particular variation on the theme". Our goal is to move away from this tendency to develop methods to solve one particular instance.

Two sets of possible features are collected. They are F_{es} that describes the characteristics of the problems and partial solutions (presented in Table 4), and F_{ec} that includes the combinations of features from F_{es}. They are listed below as:

- $F_{es} = \{f_0, f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8, f_9, f_{10}, f_{11}\}$
- $F_{ec} = \{f_i, f_m/f_n\}, f_i, f_m, f_n, \in F_{es}, i, m, n = 0, \ldots 11, m \neq n$

Table 4 Possible features of exam timetabling problems and partial solutions during problem solving

feature ID	description
f_0	number of exams
f_1	number of time periods
f_2	number of hard constraints
f_3	number of rooms
f_4	density of the conflict matrix
f_5	number of exams that are already scheduled in the partial solution
f_6	number of times that exams with common students are scheduled in consecutive time periods in the partial solution
f_7	number of times that exams with common students are scheduled one time periods apart in the partial solution
f_8	penalty of the partial solution
f_9	cost of scheduling the next exam into the partial solution
f_{10}	number of the most constrained exams
f_{11}	the number of constraints with the most constrained exams

4.1 The overall case based reasoning system

The CBR system developed for exam timetabling problems can be characterised by the following features:

- Possible heuristics investigated in the case-based heuristic selection approach include 4 well-studied sequential and well-established sequential methods for solving exam timetabling problems. They order the exams to be scheduled one by one into the draft timetable. The heuristic set H_e containing these 4 heuristics is presented below:

  $H_e = \{h_1, h_2, h_3, h_4\}$

- Source cases are different possible partial solutions obtained during the previous problem solving using sequential heuristics from H_e. For each of the partial solutions, the two heuristics that make the least violations in the next step schedule are stored and are the suggested "good" heuristics to be employed next.

- The penalty function is proposed in the same way as the fitness function that was employed in [28], where cost is given for exams that are scheduled too close to each other, aiming at spreading the exams out in the timetables. Note that, the lower the penalty of the timetable, the better is the quality of that timetable. The penalty function for the timetable t is presented in formula (b) to evaluate the quality of solutions produced by the heuristics.

$$P(t) = \sum_{s=1,2,3,4,5} w_s \times S_s \qquad \text{(b)}$$

The notations in formula (b) are described as follows:

S_s is the number of situations where students have exams scheduled s time periods apart;

w_s are the weights that reflect the importance of violations of different soft constraints, i.e. the number of times that exams are scheduled s time periods apart. It is set as $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_2 = 2$, $w_5 = 1$.

For the purpose of comparing the quality of solutions produced by different heuristics, we do not consider the infeasible solutions with violations of hard constraints during the training stage of the system.

Figure 3 represents an overview of the case based heuristic approach for exam timetabling problems. To solve a problem that has been input into the system, in each step (where one exam is scheduled to the draft partial timetable) the heuristic selector chooses the specific sequential heuristics from the case base to construct the partial solution obtained at that time. In the retrieval, the heuristic selector uses the similarity measure presented in formula (a) to compare the current partial solution being constructed with all of the source cases, which are partial solutions obtained during the solving of previous problems. The best heuristic stored with the most similar case is suggested and employed in the next step of the construction of the partial solution. This process is carried out step by step and is terminated when the stopping condition is met, namely when all of the exams are scheduled.

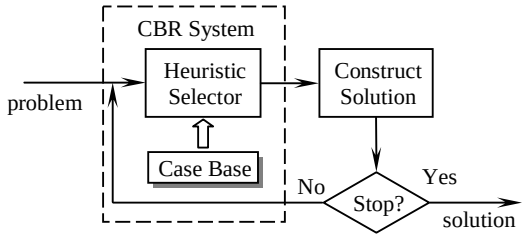

Figure 3 The case based heuristic selection approach

4.2 Data preparation for knowledge discovery

In this stage of system development, we want to evaluate the system performance on as many timetabling problems as possible to give a systematic analysis of this heuristic selection framework. There are just a few real-world benchmark timetabling problems that are widely available to the scientific community (see [28]). For this study, we require a large number of timetabling problems so we generated a large number of timetabling problems with as many different feature values as possible. Once we have a deeper understanding of the system performance on a variety of problems with good coverage of any possible features, more real-world timetabling data can be added into the system for further evaluation. These data sets are

publicly available to the scientific community at http://www.cs.nott.ac.uk/~rxq/publications.htm.

All exam timetabling problems are generated randomly by constructing a conflict matrix that defines the hard constraints between exams. Each element marked with '1' in the matrix denotes a conflict between the exams that are indicated in the corresponding row and column. The density (which is the number of '1's to the number of overall elements in the matrix) takes values from 0.65 to 0.85. The size of the problems ranges from 100 to 300 exams.

To carry out knowledge discovery, 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 the set of exam timetabling problems generated using the conflict matrix as an input. At each step (to schedule one exam) during the problem solving, the heuristic that makes the least number of violations of constraints on the partial solutions at that time is seen as the best heuristic for the corresponding partial solution. These particular partial solutions, modeled by a list of features, and their best heuristics are then stored as source cases. If on some partial solutions all or most of the heuristics perform the same, these cases will then not be stored in the case base as they are not good representatives for a class of specific problem solving situations.

Figure 4 presents an example of the heuristics that work the best during the solution process for an exam timetabling problem as a source case within the data preparation process. We found that, to construct the timetable, the best heuristic within a number of steps (usually to schedule more than 10 exams) is always the same. After a number of steps, the best heuristics chosen for the source cases are switched from one to another. To keep the case base within a certain small size, we only sample partial solutions (after each 10 steps of the timetable construction) as the source cases. We also store those cases that are near to the switching points of the heuristics (indicated by S in Figure 4). This is done to help the knowledge discovery process to detect the most important features in the case representation by distinguishing the key

information of the heuristics chosen within these cases (e.g. what differences in feature values generate which differences in choices of heuristic). By considering all of the heuristics on a set of timetabling problems, we obtained 95 potential source cases in total to build the case base.

A set of training cases (with their best heuristics) is also obtained using the same process as that used for producing the source cases. For each partial solution generated during the solution process for this set of training cases, the best heuristic that makes the least violations of the constraints is obtained. This heuristic will be the one that the heuristic selector should suggest during the knowledge discovery process. In total, 95 training cases are produced and used in the knowledge discovery process on our system.

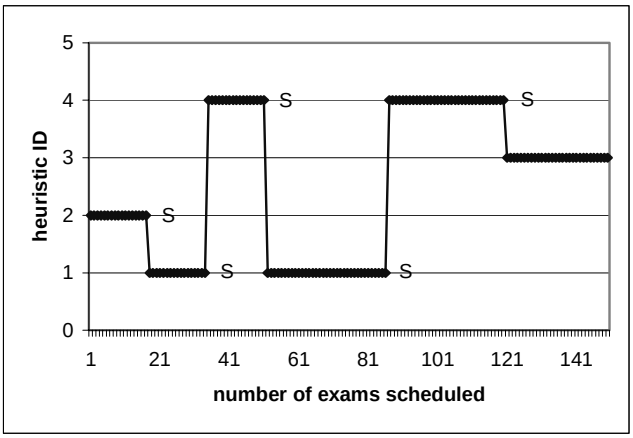

Figure 4 Best heuristics during the solution process of a source case

4.3 Knowledge discovery for the heuristic selector

Tabu search and hill climbing are employed (as they were for course timetabling) to discover the feature list that models the partial solutions to give the highest system performance on all of the training cases. The searching is reinforced to find a feature list within the system so that the best heuristics can be selected to make the least violations in the next steps of the solution construction.

For each of the training cases, the heuristic selector finds the most similar source case, whose best heuristic is then checked to see if it is a successful retrieval upon the particular feature list. All of the training cases are input into the system to check the suggested heuristics that the heuristic selector makes upon the feature list used. The feature list that gives the highest system performance on all of the training cases will be used in the heuristic selector to solve problems by employing the heuristics suggested during the problem solving. After the knowledge discovery on feature lists, the case base is then refined by using the "Leave-One-Out" strategy to remove redundant source cases.

Another set of 195 testing cases, which are different from the training cases used for the knowledge discovery process, is generated to test the heuristic selector with the feature lists discovered by the knowledge discovery process presented above. The lengths of the feature lists, the number of cases left in the refined case bases, the feature list discovered and the system performance on both the training cases and testing cases (percentage of successful retrievals on all of the training cases and testing cases) are presented in Table 5.

From the system performance on both the training cases and testing cases (presented in the second and fourth column) we can obtain similar observations to those that were drawn from Section 3. The system performs the best (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). In order to get good system performance, less relevant features should not be employed in the case representation.

Compared with the system performance for course timetabling problems (which is presented in Section 3), we can observe that higher system performance is obtained here (on both the training cases and testing cases). This is due to the fact that the exam timetabling problems studied in the system have fewer types of constraints than the course timetabling problems presented in Section 3. Knowledge discovery on the problems with a smaller number of possible characteristics is more likely to obtain higher system performance. Also we can see

that a relatively lower number of source cases are removed from the case base for exam timetabling problems (presented in the third column in Table 5) compared with that of course timetabling. The reason for this may be that there is a higher number of possible problem solving situations (for exam timetabling) than the possible problem characteristics (for course timetabling). And thus more source cases need to be retained in the approach presented in Figure 3 to provide the knowledge of heuristics which the new problem situations require.

Table 5 System performance on different feature lists for training and testing cases

no. of features	**system performance on training cases**	**feature list**	**no. of cases in the refined case base (originally 95)**	**system performance on testing cases**
2	97%	f_3/f_2, f_9	95	87%
3	100%	f_1, f_1/f_2, f_2/f_9	93	91%
4	100%	f_3/f_0, f_1/f_9, f_3/f_2, f_4/f_7	93	91%
5	100%	f_6/f_1, f_9/f_0, f_9/f_6, f_1, f_2/f_8	93	91%
6	92%	f_4/f_7, f_5, f_3/f_7, f_9, f_3/f_0, f_4/f_9	89	89%
7	100%	f_6/f_2, f_1/f_9, f_2/f_6, f_1/f_6, f_6/f_9, f_1/f_9, f_2	93	89%
8	87%	f_1/f_0, f_2/f_9, f_2/f_1, f_3/f_9, f_9/f_8, f_1/f_2, f_9/f_0, f_4/f_7	95	86%
9	87%	f_9/f_2, f_6, f_1/f_8, f_6/f_0, f_8/f_7, f_7, f_1/f_0, f_2/f_0, f_7/f_6	95	85%
10	85%	f_9/f_7, f_6, f_9, f_2/f_7, f_2, f_1/f_2, f_9/f_6, f_8/f_1, f_5/f_7, f_6/f_2	93	84%

4.4 Case-based heuristic selection

To test the case based heuristic selection approach, another 100 test exam timetabling problems with different sizes (100 to 300 exams) is generated by using conflict matrices with densities of values from 0.65 to 0.85. These problems are solved using the approach presented in Figure 3, employing the feature lists discovered on the refined case base. Due to the observation that the

heuristics switch from one to another after a certain number of scheduling steps, we set each step as scheduling 10 exams in the problem solving stage within the case-based heuristic selection approach presented in Figure 3. This may save a lot of retrieval time during the solution process while still ensuring that high quality solutions are obtained by employing the heuristics suggested from the case base. The average penalties of the solutions obtained by individual sequential heuristics, and by the case based heuristic selection with feature lists of different lengths discovered are presented in Table 6 and Table 7, respectively.

Table 6 Average penalties of solutions by individual sequential heuristics

sequential heuristics	**color degree**	**saturation degree**	**largest degree**	**largest degree with tournament selection**
penalty of solutions	328	202	245	252

Table 7 Average penalties of solutions by using case based heuristic selection with different features

no. of features	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
penalty of solutions	182	199	193	195	196	203	197	200	203

The results shown in Table 7 indicate that, our approach provides solutions that have better average penalties than those obtained by using only the individual sequential heuristics shown in Table 6. Just in two cases, namely with 7 and 10 features, slightly worse solutions are obtained by the case based heuristic selection approach comparing with the best solutions obtained by individual heuristics. By comparing the results with those presented in Table 5, we can see that roughly the higher the system performance during the knowledge discovery, the better the case based heuristic selection approach performs in the problem solving stage. By choosing good heuristics adaptively during the problem solving, better solutions can be obtained compared with those generated by only using individual heuristics.

In this section, a case based heuristic selection approach for exam timetabling problem solving has been developed and tested. The knowledge of selecting the best heuristics within different problem solving situations is stored within the source cases, which are refined to provide a guide during the problem solving to construct higher quality timetables. It is shown that, by employing this approach, better or competitive results are produced compared with the best results obtained from a set of specific heuristics on a range of exam timetabling problems.

This paper is aiming at investigating the learning ability of selecting different heuristics within a general problem solving framework for a variety of possible timetabling problems. The results obtained showed that it is capable of solving various problems by adaptively employing appropriate heuristics during the solution construction and thus naturally has higher generality than other specifically proposed approaches/heuristics for particular problems in the timetabling domain.

5. DISCUSSIONS AND FUTURE WORK

In this paper we investigated a case based heuristic selection approach within a general problem solving framework, aiming at producing reasonably good solutions quickly for a variety of timetabling problems by reusing previous experience of good heuristics in similar problem solving situations. We studied a knowledge discovery approach for case-based heuristic selection to predict the best heuristics for course timetabling problems, and the best sequential heuristics during the solution process for exam timetabling problems. The observations obtained from the experimental results showed that the knowledge of heuristics discovered helps in solving problems by using the best heuristics and guides the problem solving process to construct higher quality solutions. The heuristic selector in the approach for exam timetabling during the problem solving has the ability to use heuristics adaptively according to the problem solving situations, which is more flexible for solving a wider range of problems. Our aim is to develop a problem solving approach which can learn from the heuristics for problem solving and which is, thus much more flexible in solving a range of timetabling problems, and which can provide competitive results

over a range of specifically tailored heuristics for particular problems. Note that we are not aiming at competing with the best results reported in the literature by bespoke special purpose specific approaches. Our goal is to generate competitive results across a wider range of problems. We are attempting to underpin the next generation of timetabling systems which are more general and not specific to certain problems or problem instances.

The knowledge discovery process employs relatively simple techniques to carry out the training on case representation and the case base. Currently we are using tabu search and hill climbing to study the performance of the knowledge discovery within the case based heuristic selection framework. More other possible meta-heuristics, of course, can be investigated in a further investigation within this general framework. Experimental results showed that feature selection is a very important task and is more important than feature weights if the features are good enough to provide necessary information in the retrieval. 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 another important issue in this case base heuristic selection approach. Knowledge and problem solving experiences in timetabling need to be continuously discovered and stored. More comprehensive data analysis will be carried out and employed in our approach. Our current work investigated knowledge discovery on a number of artificially produced problems to study the performance of the system on as many problem situations as possible. Higher performance may be obtained by studying a larger number of cases for the case base. After we have obtained a deeper understanding of the system and approach on the data structure of a variety of types of problems, collection and training on a higher number of real-world data will be carried out to obtain a wider range of knowledge in the timetabling domain.

In the knowledge discovery process for exam timetabling problems, we found that a large amount of time is needed on the data preparation and analysis within the case based heuristic

selection approach. The potential source cases greatly affect the knowledge discovery process and problem solving, and thus should be carefully selected. Firstly, all of the cases need to be checked to remove duplication. Cases that are similar (with close values for each feature) and have the same best heuristics are also removed. Secondly, we found that during certain steps of solution construction, it is often the same best heuristic that is selected for the next step. In the current system, not all of the partial solutions during the data preparation are stored in the case base. Instead, we sample the cases after every 10 steps of scheduling to reduce the redundancy in the case base. Also, only those cases that are near to the point of switch between heuristics are chosen as the potential source cases for the knowledge discovery process.

As we know, knowledge discovery is a 'non-trivial process' and it takes significant time to process the raw data collected. Data analysis, which is a very important step in knowledge discovery, aims to obtain and organise the knowledge to allow us to select appropriate heuristics. Burke et al [9] investigated similarity measures for exam timetabling by running simulated annealing on a set of benchmark exam timetabling problems with different similarities and studied the data in detail. It is shown that after removing certain redundancies (e.g. students with just one enrolment, which do not affect the end result in terms of hard constraints), two problems which originally seem similar are actually quite different, or vice versa. It was seen that some enrolments are subsets of others and thus can be excluded as far as the similarity is concerned from the point of view of determining a purely feasible solution. More elaborate techniques, either manually or automatically, may need to be employed for source cases collection in data preparation in future work.

ACKNOWLEDGEMENTS

We would like to acknowledge the support of EPSRC for the research carried out in this paper (grant number GR/N36837/01).

Journal of Scheduling, 9: 99-113, 2006.

REFERENCES

1. Amintoosi M and Haddadina J (2005) Feature selection in a fuzzy student sectioning algorithm. In: [23], 148-161.

2. Asmuni H, Burke EK and Garibaldi JM (2005) Fuzzy multiple ordering criteria for examination timetabling. In: [23], 334-354.

3. Beddoe G., and Petrovic, S., Determining feature weights using a genetic algorithm in a case-based reasoning approach to personnel rostering. Accepted for publication in the *EJOR*, 2005.

4. Burke EK and Carter M (eds.) (1998), *The 2nd International Conference on the Practice and Theory of Automated Timetabling,* Lecture Notes in Computer Science 1408. Springer-Verlag.

5. Burke EK and de Causmaecker P (eds.) (2002), *The 4th International Conference on the Practice and Theory of Automated Timetabling,* Lecture Notes in Computer Science 2740. Springer-Verlag.

6. Burke EK, De Causmaecker P, Vanden Berghe G and Van Landeghem H (2004) The State of the Art of Nurse Rostering. *Journal of Scheduling*, **7**(6): 441-499.

7. Burke EK, De Werra D and Kingston J (2004) Applications to Timetabling. In: *Handbook of Graph Theory*. Gross J and Yellen J (eds.), pp. 445-474, Chapman Hall/CRC Press.

8. Burke EK, Dror M, Petrovic S and Qu R (2005) Hybrid Graph Heuristics within a Hyper-heuristic Approach to Exam Timetabling Problems. In: Golden BL, Raghavan S and Wasil EA (eds.). *The Next Wave in Computing, Optimization, and Decision Technologies.* Conference Volume of the 9th informs Computing Society Conference. pp. 79-91. Springer.

9. Burke EK, Eckersley A, McCollum B, Petrovic S, Qu R (2003) Similarity measures for exam timetabling problem. 1st Multidisciplinary Intl. Conf. on Scheduling: Theory and Applications, Vol. 1, pp. 120-136, Nottingham, UK, Aug 13-16, 2003.

10. Burke EK and Erben W (eds.) (2000) *The 3rd International Conference on the Practice and Theory of Automated Timetabling,* Lecture Notes in Computer Science 2079, Springer-Verlag.

Journal of Scheduling, 9: 99-113, 2006.

11. Burke EK, Hart E, Kendall G, Newall J, Ross P and Schulenburg S (2003) Hyper-heuristics: an emerging direction in modern search technology. In: Glover F and Kochenberger G (ed.), *Handbook of Meta-Heuristics*, Kluwer. pp. 457-474.

12. Burke EK, Jackson KS, Kingston JH and Weare RF (1997) Automated timetabling: the sate of the art, *The Computer Journal*, **40**(9): 565-571.

13. Burke EK, Kendall G, and Soubeiga E (2003) A Tabu-Search Hyperheuristic for Timetabling and Rostering. *Journal of Heuristics*, **9**(6): 451-470.

14. Burke EK, MacCarthy BL, Petrovic S and Qu R (2000) Structured Cases in Case-Based Reasoning - Re-using and Adapting Cases for Time-tabling Problems**.** *Knowledge-Based Systems*, **13**(2-3): 159-165.

15. Burke EK, MacCarthy BL, Petrovic S and Qu R (2001) Case-based reasoning in course timetabling: an attribute graph approach. In: Aha DW, and Watson I (eds.): *Case-Based Reasoning Research and Development*. Lecture Notes in Artificial Intelligence 2080. pp. 90-104.

16. Burke EK, MacCarthy B, Petrovic S and Qu R (2006) Multiple-retrieval case-based reasoning for course timetabling problems. to appear in *Journal of Operations Research Society*.

17. Burke EK, MacCarthy B, Petrovic S and Qu R (2002) Knowledge discovery in hyper-heuristic using case-based reasoning on course timetabling. In: [5], pp. 277-288.

18. Burke EK, McCollum B, Meisels A, Petrovic S and Qu R (2006) A Graph-Based Hyper Heuristic for Educational Timetabling Problems. To appear in the *EJOR*.

19. Burke EK and Newall JP (1999) A Multi-Stage Evolutionary Algorithm for the Timetable Problem. *IEEE Transactions on Evolutionary Computation*. **3**(1): 63-74.

20. Burke EK, Newall J, and Weare R (1998) A simple heuristically guided search for the timetabling problem. In: Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, pp. 574-579.

21. Burke EK and Petrovic S (2002) Recent research directions in automated timetabling. *EJOR*, **140**(2): 266-280.

Journal of Scheduling, 9: 99-113, 2006.

22. Burke EK and Ross P (eds.) (1996) *The 1st International Conference on the Practice and Theory of Automated Timetabling*, Lecture Notes in Computer Science 1153, Springer-Verlag.

23. Burke EK and Trick M (eds.) (2005) *The 5th International Conference on the Practice and Theory of Automated Timetabling*. Lecture Notes in Computer Science 3616, Springer-Verlag.

24. Carrasco MP and Pato MV (2002) A multiobjective genetic algorithm for the class/teacher timetabling problem, In: [5], pp. 179-192.

25. Carter MW (1986) A lagrangian relaxation approach to the classroom assignment problem. *IFOR*. **27**(2): 230 – 246, 1986.

26. Carter MW and Laporte G (1996) Recent developments in practical exam timetabling, In: [22], pp. 3-21.

27. Carter MW and Laporte G (1998) Recent developments in practical course timetabling, In: [4], pp. 3-19.

28. Carter MW and Laporte G (1996) Examination timetabling: Algorithmic Strategies and Applications, *Journal of Operations Research Society*, **74:** 373-383.

29. Chan P and Weil G (2000) Cyclic staff scheduling using constraint logic programming, In: [10], pp. 159-175.

30. Colorni A, Dorigo M, and Maniezzo V (1998) Metaheuristics for school timetabling. *Computational Optimisation and Applications*, **9**(3), 277-298.

31. Costa D (1994) A tabu search algorithm for computing an operational timetable, *EJOR*, **76**: 98-110.

32. Cunningham P and Smyth B (1997) Case-based reasoning in scheduling: reusing solution components. *The International Journal of Production Research*. **35**: 2947-2961.

33. Dorn J (1995) Case-based reactive scheduling. Technical Report CD-TR 95/75. Vienna University of Technology, Institut for Information Systems.

34. Dowsland KA (1998) Off the peg or made to measure, In: [4], pp. 37-52.

35. Easton K, Nemhauser G and Tick M (2004) Sports Scheduling. In: Leung JYT (ed.) *Handbook of Scheduling*. Chapter 52. CRC Press LLC.

Journal of Scheduling, 9: 99-113, 2006.

36. Erben W (2000) A grouping genetic algorithm for graph coloring and exam timetabling, In: [10], pp. 132-158.

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

38. Foulds LR and Johnson DG (2000) SlotManager: a microcomputer-based decision support system for university timetabling. *Decision Support Systems*, **27**: 307-381.

39. Gaspero LD and Schaerf A (2000) Tabu search techniques for examination timetabling, In: [10], pp. 104-117.

40. Han L, Kendall G (2003) Investigation of a Tabu Assisted Hyper-Heuristic Genetic Algorithm. Proceedings of Congress on Evolutionary Computation (CEC2003), pp. 2230-2237 (Vol. 3).

41. Kendall G and Hussin NM (2005) A Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at the MARA University of Technology. In: [23], 270-293.

42. Kolodner JL and Leake D (1996) A tutorial introduction to case-based reasoning. In: [44], 31-65.

43. Kong SC and Kwok LF (1999) A conceptual model of knowledge-based timetabling system. *Knowledge-Based Systems*, **12**: 81-93.

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

45. MacCarthy BL and Jou P (1996) Case-based reasoning in scheduling. In: Khan MK and Wright CS (eds.), Proceedings of the Symposium on Advanced Manufacturing Processes, Systems and Techniques, (MEP Publications Ltd, 1996), pp. 211-218.

46. Mantaras RL and Plaza E (1997) Case-based reasoning: an overview. *AI Communications*, **10**: 21-29.

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

48. Miyashita K and Sycara K (1995) CABINS: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *Artificial Intelligence*, **76**: 377-426.

Journal of Scheduling, 9: 99-113, 2006.

49. Petrovic S and Burke EK (2004). University Timetabling. In: Leung JYT (ed.) *Handbook of Scheduling*. Chapter 45. CRC Press LLC.

50. Petrovic S, Patel V and Yang Y (2005) Examination Timetabling with Fuzzy Constraints. In: [23], 313-333.

51. Reeves CR (1996) Modern heuristic techniques. In: R-Smith VJ, Osman IH, Reeves CR and Smith GD (eds.) *Modern Heuristic Search Methods*, pp. 1-25.

52. Ross P (2004) Hyper-heuristics. In: Burke EK and Kendall G (eds.). *Introductory Tutorials in Optimisation, Decision Support and Search Methodology*. Chapter 17. Kluwer.

53. Ross P, Hart E and Corne D (1998) Some observations about GA based timetabling, In: [4], pp. 115-229.

54. Rattadilok P, Gaw A and Kwan RSK (2005) Distributed Choice Function Hyper-heuristics for Timetabling and Scheduling. In: [23], 51-70.

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

56. Schmidt G (1998) Case-based reasoning for production scheduling. *International Journal of Production Economics*, **56-57**: 537-546.

57. Scott S, Simpson R and Ward R (1997) Combining case-based reasoning and constraint logic programming techniques for packaged nurse rostering systems. Proceedings of the 3rd UK Case-Based Reasoning Workshop, University of Manchester, U.K. 9 September 1997.

Journal of Scheduling, 9: 99-113, 2006.

Figure 1 Schematic diagram of knowledge discovery process on features and their weights

Figure 2 System performance on different numbers of features from F_{cc} by tabu search and hill climbing

Figure 3 The case based heuristic selection approach

Figure 4 Best heuristics during the solution process of a source case

Journal of Scheduling, 9: 99-113, 2006.