Journal of Operations Research Society, 57(2): 148-162, 2006.

Multiple-Retrieval Case-Based Reasoning for Course Timetabling Problems

Edmund K. Burke[1], Bart L. MacCarthy[2], Sanja Petrovic[1], Rong Qu[1]

[1]Automated Scheduling Optimisation and Planning Research Group

School of Computer Science and Information Technology, The University of Nottingham

Nottingham, NG8 1BB, U.K

[2] Business School, The University of Nottingham, Nottingham, NG8 1BB, U.K

Abstract. The structured representation of cases by attribute graphs in a Case-Based Reasoning (CBR) system for course timetabling has been the subject of previous research by the authors. In that system, the case base is organised as a decision tree and the retrieval process chooses those cases which are sub attribute graph isomorphic to the new case. The drawback of that approach is that it is not suitable for solving large problems. This paper presents a multiple-retrieval approach that partitions a large problem into small solvable sub-problems by recursively inputting the unsolved part of the graph into the decision tree for retrieval. The adaptation combines the retrieved partial solutions of all the partitioned sub-problems and employs a graph heuristic method to construct the whole solution for the new case. We present a methodology which is not dependant upon problem specific information and which, as such, represents an approach which underpins the goal of building more general timetabling systems. We also explore the question of whether this multiple-retrieval CBR could be an effective initialisation method for local search methods such as Hill Climbing, Tabu Search and Simulated Annealing. Significant results are obtained from a wide range of experiments. An evaluation of the CBR system is presented and the impact of the approach on timetabling research is discussed. We see that the approach does indeed represent an effective initialisation method for these approaches.

Keywords: Timetabling problems, Case-Based Reasoning, Attribute graph, Scheduling problems, Hill climbing, Tabu search, Simulate annealing, Graph heuristic method

Introduction

Timetabling Problems

Timetabling problems arise in many contexts including transportation[1], sports events[2], employee rostering[3, 4], and educational timetabling[5, 6, 7]. They have been the subject of active recent research[8,9, 10, 11, 12]. This important research field continues to attract the attention of the scientific community as problems become more complex and as new breakthroughs provide better ways of solving them (for example see[13, 14, 15, 16]). Economic efficiency, costs and resource utilisation are also important drivers for improved timetable generation.

A general timetabling problem consists of assigning a number of events (exams, courses, meetings, etc) into a limited number of timeslots (periods of time) and venues, whilst minimising the violations of a given set of constraints. Associated constraints are usually classified into two types: *hard constraints* and *soft constraints*. Hard constraints should under no circumstances be violated (e.g. no person is assigned to two or more events simultaneously). Soft constraints are desirable but not essential to satisfy (e.g. two events should/should not be consecutive, one event should occur before another, etc).

Course timetabling is a multi-dimensional assignment problem where courses are assigned to classrooms and timeslots[7]. Early approaches to timetabling have included integer linear programming[17] and graph colouring techniques[18, 19]. More recently, various meta-heuristic techniques have been very successful in a wide range of timetabling problems. In course timetabling, Tabu Search was investigated[20] and employed in solving real-world problems[21] and the results were encouraging. Simulated Annealing has also been investigated for course timetabling problems[22, 23]. The Great Deluge algorithm was employed with some success[24]. Genetic Algorithms and Evolutionary Algorithms have been studied widely by researchers[25, 26, 27] in course/school timetabling and approaches that involve hybridising GAs with local search techniques, sometimes called Memetic Algorithms[28, 29] have shown promising results in general university timetabling. Constraint-based techniques have also been employed widely in timetabling[30, 31, 32]. Complexity issues in course timetabling have also been studied in some depth[33, 34]. A wide variety of research papers on different types of timetabling are also available[8, 9, 10, 11, 12]. In this paper we investigate CBR for course timetabling. In particular we present new retrieval and adaptation mechanisms and test their effectiveness in computational experiments based on our previous work[35, 36]. The research presented in this paper is partly motivated by the goal of developing timetabling approaches that are less dependent on problem specific information than the current state of art and that could, therefore, operate on a wide range of problems. Another motivation is the goal of developing effective initialisation methods for the more widely studied local search approaches.

Case-Based Reasoning

Case-based reasoning (CBR)[37, 38] is a knowledge-based paradigm where new problems are solved by using previous experience or knowledge. Previously solved problems and their good solutions are stored as *source cases* in a case base. New problems are solved by searching for the most similar source cases and reusing/adapting their solutions or problem solving strategies.

Aamodt and Plaza[39] presented a CBR framework where 4 "REs" describe the problem solving process that is represented by CBR. They are RETRIEVAL, REVISON, REUSE and RETAIN. We illustrate the CBR system scheme in Figure 1. The new problem to solve is input into the CBR system and compared with the source cases by using a particular similarity measure. The solutions of the most similar source case are *retrieved* and will usually be *revised* by employing rules or heuristics. The adapted solution is then *reused* for the problem in hand. More retrieval may need to be carried out if the adapted solution is unsuitable (for whatever reason). Some CBR systems *retain* the newly solved problems as new source cases thus the CBR system has the ability to learn new knowledge throughout its lifecycle. For a more detailed treatment of CBR see Leake (1996)[38].

FIGURE 1 ABOUT HERE.

In CBR, a similarity measure is used to assess the similarity between the problem in hand and the source cases. In most CBR systems, cases are usually represented by a list of feature-value pairs which represents the values of different features of the problem. The similarity measure can be defined as a nearest neighbourhood approach which sums the differences of values of the features[38]. It is noted that in complex problem domains, the issue of case representation is particularly important[40] and usually it leads to more complex similarity measures. The cases need to be described in such a way that the comparisons between them lead to retrieved source cases which are applicable for the new problems.

The intuitive motivation for exploring CBR for timetabling come about by observing that in the real world, newly generated timetables are often based on previous similar timetables. Indeed, altering "last year's timetable" is an approach favoured by many timetabling officers in schools and universities across the world. A paper which analyses the results of a questionnaire which was completed by administrators in 56 British universities clearly demonstrates that (at least in terms of timetabling) the practice of “re-using part of last year’s timetable” is employed by a significant number of the universities that responded[41]. Of course, there is no guarantee that last year’s timetable represents a high quality solution. Indeed, in many cases it will not. However, it is often the case that a significant amount of effort is expended in universities to produce a “good” timetable (where “good” is defined by the user’s view of what they require). In subsequent years they often “tweak” this “good” timetable because they know (or at least believe) that it is good and want to minimise effort. This work is motivated by situations where institutions have generated a high quality solution and re-use a similar solution each year.

In this particular paper, we work with specially constructed timetabling problems. We are moving towards real world problems. However, because of the nature of case based reasoning, we first need to work with data whose structure we understand (unlike the situation with real world problems) and this is what we are undertaking in this paper. Our goal is to understand how case based reasoning might work on large problems whose structure we understand. The idea is that by doing this, we are better placed to develop a system that can work well when we later apply it to real world data.

We believe that CBR is a valuable technique for timetabling problems that usually have complex constraint features as it indirectly puts emphasis on constraint-directed search (in that it is guided by solutions to “similar problems”). Current state of the art scheduling methodology tends to incorporate very specific information into the general meta-heuristic methods[42, 43, 44, 45]. The standard formulations of the basic meta-heuristic approaches are, of course, quite general in that they can be (and have been) applied to a wide range of problems. However, in order to be implemented on those problems, a considerable amount of research expertise and programming effort had to be employed. There is often a significant amount of problem specific information hard coded into the methods. Once these approaches are developed they cannot usually be applied to other problems without significant redesign. Indeed, in the case of examination timetabling, Carter and Laporte point out that many exam timetabling systems have been developed for different instances of the same problem i.e. they are specifically developed for the institution in which they were built[7].

The main point is that while the meta-heuristic methodologies are in themselves general, they require significant “tailoring” by experts to enable them to be applied to real problems. Once tailored, these approaches are often far too problem specific to be transferred to other problems. So if the constraints are altered then usually the method needs to be altered and this is often a very challenging and demanding task. However, case based reasoning approaches do not have such “specific” information hard coded into them. So a timetabling methodology which draws upon case based reasoning holds promise for being more generally applicable (without further re-design and programming effort) and this was another motivation for studying this approach. There are many real world cases where one institution’s “good” constraint is another institution’s “bad” constraint. Different institutions have very different ideas about what constitutes a good timetable. This point is clearly concluded (in the case of examination timetabling)[41]. It is possible to handle different constraints in meta-heuristic methods but by doing so we make the method more and more specific to those constraints. The more we do it, the less likely the method can be re-used in another situation. The point with CBR is that it is not reliant upon constraint specific information in the same way and so there is potential for a greater level of generality. A large amount of work on CBR has been conducted in a wide range of problem domains which are usually ill-structured including planning, design, advisory services, diagnosis and health[46]. There is a growing body of literature on methods that attempt to raise the level of generality of optimisation/search systems[47].

Another major motivation for our approach is to investigate whether or not CBR offers promise as an initialization method for local search approaches which have been widely applied in course timetabling (and indeed in a range of other scheduling problems). The overall question we are seeking to answer here is whether it is worthwhile to build a CBR system to provide us with a “good” solution which can be “fine tuned” by a local search approach to generate a “very good” solution.

As mentioned above, timetabling problems are, of course, a type of scheduling problem. Over the last decade or so there have been a few publications that specifically investigate CBR for some scheduling problems. The CBR scheduling systems that have been described in the literature include the SMARTplan system[48] that models the abstraction of problems of airlift management; the Clavier system[49] for a real-world autoclave management and loading problem; the CBR-1 system[50] where a case base was organised as a semantic net for dynamic job shop scheduling; and the CABINS system[51] that models the heuristic repair actions as cases to interactively repair schedules in job shop scheduling. Studies of some other approaches have also been carried out for steel production[52], traditional travelling salesman problems[53], single machine scheduling problems[53], nurse rostering[54, 55], dynamic shop floor problems[56] and production control problems[57]. Some work also addresses research issues in the applications of CBR systems to a wide range of scheduling environments and has presented general frameworks[58, 59]. To our knowledge, no other research has been reported that has investigated CBR specifically for educational timetabling problems except our own work[35, 36].

Problem Decomposition in Timetabling and CBR

Decomposition and partition techniques have been studied with some success in timetabling problems, which are usually very large and complex. The basic idea is to decompose the problem into a set of sub-problems that are small enough to be easily solved by using simple approaches. Then these (hopefully high quality) sub-solutions will be combined to provide a solution for the original problem. The difficulty, of course, is in "re-constructing" the sub-solutions to generate a "final" solution to the whole problem. The method has to avoid making assignments in "earlier" sub-problems which lead to situations where events in later sub-problems cannot be assigned to any timeslot without breaking hard constraints. Carter[60] presented an algorithm in course timetabling that decomposed the courses into relatively independent clusters, which can be solved more easily using reasonably simple approaches. Robert and Hertz[61] decomposed course timetabling problems into a series of easier assignment type sub-problems. Weare[62] also studied decomposing the timetabling data to produce shorter flexible length timetables. Burke and Newall[63] presented a multi-stage algorithm in an evolutionary approach that decomposed examination timetabling problems by using graph colouring heuristics, and the sub-problems were solved by using a memetic approach[28].

In CBR, decomposition techniques have been mostly employed successfully in design and planning domains where the cases were decomposed by sub-goals or abstraction and the case base was usually organised hierarchically[64, 65].

A Previously Presented Structured CBR Approach

In previous work the authors have shown that a structured CBR approach[35, 36] worked well in solving course timetabling problems but was incapable of providing good solutions for large problems. This is mainly because the case base storing the cases represented as attribute graphs grows significantly when the size of the cases increases. In this paper, we present an approach that partitions large timetabling problems into smaller solvable sub-problems whose solutions can be obtained by retrieving multiple cases from the case base. It draws upon the structured CBR approach[35, 36].

The next section presents a brief introduction to the structured CBR system. The new partitioning and adaptation approaches within this structured CBR system are then described. Computational experiments on the new approaches are reported and analysed. This is followed by concluding comments and directions for future work.

The Structured CBR Approach

Structured Cases in CBR

In many traditional CBR papers, a feature list is employed to represent cases[38]. The similarity between cases is obtained by the nearest-neighbour approach that calculates a weighted sum of the similarities between each pair of features in each case. However, Mantaras and Plaza[40] pointed out that the feature list representation is the most severe limitation of existing CBR systems for knowledge-rich applications with higher-order relations between features. Timetabling problems are constraint satisfaction problems that typically have a wide range of related constraints. Therefore the traditional case representation cannot capture all of the complex constraints which often significantly determine the solutions. Timetabling problems require much more complex case representations.

Structured representation has been successfully employed in CBR in some complex application areas. Borner et al. employed a structural similarity measure to assess the maximal common sub-graphs between cases[66] in a design task. In the literature, different approaches have been used to determine the required structure for complex cases. In particular, researchers have utilised semantic nets[67], graphs[68, 69] and trees[70]. The FABEL project[71] provided more information on structured CBR. Gebhardt[72] categorised the existing CBR systems employing structured cases into five groups: restricted geometric relationships, graphs, semantic nets, model-based similarities and hierarchically structured similarities.

In our approach, attribute graphs are employed to handle the level of knowledge required to tackle course timetabling problems[35, 36]. The constraints are represented by edges between each pair of courses, which are represented by vertices. The retrieval process described in this paper is based on the information about the constraints and attempts to find structurally similar cases for reuse. The over-riding motivation is that previous timetables with similar constraints will provide a sensible starting point for solving a new timetabling problem.

Attribute Graph Representation and Retrieval Process

Attribute graphs have been used by the authors to structurally represent the requirements in course timetabling problems[35]. Attributes associated with vertices and edges are shown in Table 1 and Table 2 respectively.

TABLE 1 AND TABLE 2 ABOUT HERE.

As an illustrative example, Figure 2 presents part of an attribute graph representing a course timetabling problem. The notation x:y denotes the label x of an attribute and its value y. *MathB* is labelled 0 meaning that it should be held just once a week. *LabA* and *MathA* (labelled 1 with values 2 and 3) will be held 2 and 3 times per week, respectively. *LabB* and *Physics* are labelled 2 and 3 respectively with values 2, which means that they should/should not be assigned into timeslot 2, respectively. The edge labelled 4 denotes that *LabA* should be held before *LabB* if possible. *MathB* and *Physics* should be held consecutively while *LabB* and *MathA* should not. Courses adjacent to edges labelled 7 should not be held simultaneously. In our approach, room constraints in course timetabling are considered separately in the adaptation phase after a set of potential candidate solutions are obtained from the CBR methodology.

FIGURE 2 ABOUT HERE.

The case base is built as a decision tree, storing cases represented by attribute graphs[36]. All the possible (partial) permutations of the courses in the cases are stored hierarchically by clustering the ones representing the similar (sub) attribute graphs under the same node in the tree. The goal of the retrieval process is to find cases that are structurally similar (have similar constraints) to the new case. In the retrieval, branch and bound is used to reduce the size of the search tree by cutting the branches storing graphs or sub-graphs that are not similar to those of the new case. The new case is classified in the decision tree to a set of nodes with similar graphs or sub-graphs, which are then reused in attempting to solve the new case. The similarity measure takes into account the cost of the adaptation of the retrieved case to meet the requirements of the new problem. Different levels of values of the costs are assigned empirically to the substitutions, deletions and insertions of vertices or edges in the new cases to make them the same as the retrieved cases. They are assigned to approximate the cost for the differences between sub-graphs and tested upon the difference they make for retrieving reusable and applicable sub-graphs. More details about the decision tree algorithm and the retrieval process are presented in our previous work[36].

Partitioning Large Timetabling Problems by Multiple-Retrieval

Multiple-Retrieval Approach

The previous retrieval process[36] retrieves those cases from the case base that are graph or sub-graph isomorphic, or that have similar graphs or sub-graphs with the new case. Good results on a large number of experiments provided clear evidence to indicate that this approach takes less effort to get high quality solutions based on the retrieved structurally similar cases. It worked well in solving problems that are smaller or almost the same size as the cases in the case base. However, in solving problems that are much larger, the small cases retrieved by employing this approach are incapable of providing much help in finding a good solution. The case base only stores relatively small cases, as the size of the decision tree that stores all the possible permutations of the attribute graphs increases significantly when the number or the size of the cases increases. With only limited help from a single retrieved case with a small matched part, the system may not be able to find good solutions for large timetabling problems.

These observations provide the motivation for developing the new multiple-retrieval approach presented in this paper. In the new approach, in each retrieval, cases that are similar to part of the un-matched new case are retrieved and the matched part of the new case is partitioned from it as a sub-problem. The partition is made by performing the retrieval process recursively. The recursive retrievals partition the problem into smaller solvable sub-problems based on the retrieval process employed in the previous CBR system[36]. A schematic diagram illustrating the process is presented in Figure 3.

FIGURE 3 ABOUT HERE.

A new graph is produced to represent the remaining part of the new case in each retrieval based on that of the last retrieval cycle. In each iteration, the following steps are performed on the attribute graph of the new problem:

(1) The matched part of the attribute graph of the new case in the last cycle is replaced by a *super vertex*. The attribute of the vertex is set as 0 (ordinary course).

(2) The super vertex in the new attribute graph keeps the edges that are originally adjacent to the matched vertices. When there is more than one edge between a vertex in an un-matched sub-graph and the matched vertices, then the attributes of the newly combined edges are decided by the following predefined priorities:

- In order to preserve the feasibility of the final solution, the label 7 that denotes conflict has the highest priority and overwrites the other label.
- In other cases, the new attribute of the new edge will be set as one of the original ones.

By setting conflict as the highest priority, we can guarantee that the combined final solutions (the combining process is shown in the next section) will always be feasible (i.e. satisfy the hard constraints). The newly generated attribute graphs structurally represent the relationships between the matched and un-matched part of the original graph. The attributes of the combined edges approximate (but do not exactly represent) the previous attributes. The possible violations of soft constraints will be fixed in the adaptation phase. Figure 4 illustrates how the new attribute graphs are generated. The vertices 1, 2 and 5 that match a case in the *i-1*th retrieval are combined into a super vertex S_i for the *i*th retrieval. All the edges adjacent to these matched vertices are now adjacent to S_i. In each retrieval, the matched part of the problem is partitioned as a sub-problem that may be solved by adapting the retrieved cases for it. The same process is carried out for the *i+1*th retrieval. This process stops when no more matched cases can be retrieved for a newly produced graph.

FIGURE 4 ABOUT HERE.

This multiple-retrieval approach is carried out on the same decision tree and partitions the problem by utilizing the case base rather than by employing fixed rules. It generates sub-problems automatically depending upon the cases in the case base. Usually more than one possible match can be found for each sub-problem that is partitioned. The most similar cases are used to generate a number of candidate timetables. The one with the lowest penalty (calculated by formula (2) below) is selected as the best solution for the new timetabling problem.

The similarity measure in the new multiple-retrieval approach is shown in formula (1). The individual similarity between each sub-problem and the retrieved cases for it is calculated in the same way as when using single retrieval[36], considering the costs of the substitutions, deletions and insertions of the vertices and edges. In our approach we assign costs by their effect on adaptation: substitution costs are lower than deletion and insertion costs; deletion costs are lower than insertion costs. The costs are set based upon experience. The sum of all the individual similarities is divided by the sum of the overall costs in all retrievals (P + A + D) and subtracted from 1, as shown below:

$$S(t_1,t_2)=1-\frac{\sum_{1}^{r}(\sum_{b=0}^{n} p_b+\sum_{i=0}^{m} a_i+\sum_{j=0}^{k} d_j)}{r(P+A+D)} \quad (1)$$

The notation used in formula (1) is described as follows:

r is the number of retrievals that need to be carried out on the new case until no more sub-problems can be partitioned from it;

p_b is the cost of substituting a vertex or edge of the new case t_1 with the corresponding vertex or edge in the retrieved case t_2 in every retrieval;

d_j and a_i are the costs of deleting and inserting a vertex or edge into or from the new case t_1;

n is the number of the matched vertices and edges in every retrieval;

m and k are the numbers of vertices and edges needed to be inserted into or deleted from the new case t_1, respectively;

P is the sum of the substitution cost of every possible pair of vertices or edges between new case t_1 and retrieved case t_2;

D and A are the sums of costs of inserting and deleting all of the vertices or edges into or from the new case, respectively.

Adaptation on Multiple Retrieved Cases

Generating sub-solutions. Before generating the whole solution we need to identify the sub-solutions based on each retrieved case. The sub-solution for each sub-problem is firstly obtained by substituting every matched course in the retrieved solution and deleting all the courses that are not matched. Then we will have a set of sub-solutions for all the sub-problems. After this we will have a set of partial sub-solutions for all the sub-problems. We can expect that the super vertices are either in the solutions of the sub-problems, or in the list of un-matched vertices.

Combining sub-solutions. Starting from the sub-solution of the last sub-problem, we substitute the super vertices in all of the sub-problems with their corresponding sub-solutions. This process is repeated until all of the sub-solutions are combined into a final solution (without any super vertices left) for the original new case. The combined solution is guaranteed to be feasible as we never release the constraints and all the sub-problems are feasible.

Figure 5 illustrates the combining process. Suppose we have obtained the *i*th and *j*th sub-solutions based upon the retrieved cases for the *i*th and *j*th sub-problems partitioned in Figure 3. We present the sub-solutions as lists of courses in timeslots, represented as boxes in Figure 4. These sub-solutions are combined by substituting the corresponding super vertex S_j by the *j*th sub-solution $\boxed{3\ 6\ 7}\boxed{S_i\ 4}$. Then we substitute S_i by the *i*th sub-solution $\boxed{2\ \ 5}\boxed{1}$. After substituting all the super vertices, a partial solution combining all the sub-solutions is generated for the original new case.

FIGURE 5 ABOUT HERE.

The combined partial solution is adapted to generate the final solution. The adaptation process uses the following basic timetabling method to allocate rooms and improve the CBR generated solution, taking into consideration the soft constraints presented in Table 1 and Table 2.

1. All the courses in the combined solution are assigned to the smallest feasible rooms available;
2. All the courses that cannot be assigned to rooms or violate the soft constraints are unscheduled and inserted into an unscheduled list. The courses that are not yet scheduled are also collected;
3. The courses in the unscheduled list are then rescheduled by a graph heuristic method with tournament selection considering the room constraints, which we will explain below.

Graph heuristic with tournament selection. The graph heuristic with tournament selection (GHT)[73] is used to schedule the courses in the unscheduled list one by one to the first timeslot and room with no violations of any constraints (penalty-free). Tournament selection is used to select the first course every time from a randomly chosen subset of courses of the unscheduled list sorted decreasingly by their importance (number of constraints with the other courses). Those courses that cannot be assigned to a penalty-free timeslot will be scheduled to the timeslots that lead to the lowest penalty after all the others have been scheduled. When a tie is met, the course is randomly assigned to an available timeslot. A course will be left as unscheduled if it cannot be scheduled without violating a hard constraint or no room is available.

The GHT approach forms the basis of the Optime examination timetabling system which has been commercially implemented in institutions in Australia, France, New Zealand, UK and the USA. Optime is being marketed by eventMap Ltd (a company which is a spin out of the Automated Scheduling, Optimisation and Planning Group at University of Nottingham). It quickly produces solutions that the users consider to be of high quality. Future work will enrich the case base with other “good” solutions but we must keep in mind that one institution’s good solution is another institution’s bad solution.

Penalty function. The penalty function given in formula (2) is used to evaluate every timetable generated in the experiments carried out in the next section.

$$\text{Penalty}(t) = 100\,U(t) + 5\,S(t) \qquad (2)$$

U(t) is the number of courses not scheduled. They are assigned a high cost of 100. Violations of soft constraints, indicated by S(t), are assigned a relatively low cost of 5.

Experiments and Results

We have carried out an extensive series of experiments on specially constructed data sets. At this stage, we need to analyse the behaviour of the multiple-retrieval approach on data that has been constructed in a systematic way. We are specifically not working with real data at this stage because we do not understand the structure of arbitrary large real world data sets. In order to understand how the CBR approach is working we need to specifically construct the data so that we understand the structure. By experimenting with data whose structure we understand we are better placed to develop a system that can work well when we later apply it to real world data. We have no say over the structure of the real world data but if data is causing a CBR system to act in a certain way, we need to be aware of the structure of that data in order to understand why. The point is that if we construct the data, we know what we have and have a better chance of understanding what features are playing a role. If we just take arbitrary real world data sets we have no idea about the specific structure of these problems and it will be so much more difficult to understand how they are affecting the system during the development stage.

A large number of experiments have been carried out to solve timetabling problems of different size on case bases with different types and sizes of cases. We define two types of cases in the case bases: simple and complex (of small or large size). In complex cases, every course has at most 4 (and at least 1) constraints. Courses in simple cases have at most 3 (and at least 1) constraints. Small cases have 6 to 10 courses and larger cases have 10 to 15 courses. Attributes of the courses are randomly generated. The solutions of these cases in the case bases are obtained by using Graph Heuristic with Tournament Selection (GHT)[73]. Recall that this method is currently implemented in a commercial system.

Nine sets of new cases are considered each with 20 different new cases of the same size. The first of these sets has 10 courses; the second has 15 courses and so on up to 50 courses. The GHT is used to solve these cases from scratch. These solutions are then compared with those from the multiple-retrieval CBR approach on different case bases.

The second major aspect of our experiments explores the employment of the multiple-retrieval CBR as the initialisation approach for Hill Climbing, Simulated Annealing and Tabu Search in order to determine whether CBR might provide solutions which are a good starting point for meta-heuristic methods.

Case Bases with Simple Cases

The first group of experiments is carried out on a set of case bases containing 5, 10 or 15 simple cases of small or large size (3 X 2 = 6 case bases in all). All the new cases are then input to these 6 case bases to be solved by using the multiple-retrieval approach with adaptation employing the GHT. These solutions are compared with those generated from scratch by the same GHT. Figure 6 presents two charts and a table displaying the average penalties of the timetables of 20 different new cases in each of the nine sets on the 6 case bases, and those generated by GHT alone. The curves in the chart are logarithmic trendlines that are drawn based on the results that are plotted in the chart. Rather than showing precise curves of the result points, they present the trends of the penalties of the timetables for cases over a range of sizes. The best average result for each new case type is highlighted in the table.

FIGURE 6 ABOUT HERE.

We can see that the multiple-retrieval CBR approach with GHT as the adaptation method produces lower penalty timetables than those obtained by using the GHT alone to generate the timetables from scratch. It is observed from the trendlines of the results from the charts that the penalties of the timetables obtained by using the CBR approach with different case bases are close to each other but, in general (7 out of 9), case bases with larger cases provide timetables with slightly higher penalties on average over the 20 different new cases.

Case bases with Complex Cases

Another set of experiments has been undertaken on the nine sets of new cases to investigate the use of case bases with complex cases. Figure 7 shows the average penalties of the timetables obtained from case bases with 5, 10 or 15 large and small complex cases. Again, in general, case bases with small cases provide better results than those with large cases (7 out of 9). In all of these cases, GHT on its own obtained solutions with a higher penalty value than the CBR approach that uses GHT as the adaptation method.

FIGURE 7 ABOUT HERE.

Comparison and Evaluation on Case Bases with Small Cases

From all the experiments carried out on different case bases, we can observe that case bases with both large and small cases provide better results than those obtained by the GHT without employing the CBR approach. CBR with case bases of smaller cases has better performance in terms of lower penalty timetables for the new cases of different size than CBR with large cases. Smaller sub-graphs in the retrieved multiple sub-solutions seem to provide a better basis for the adaptation to produce timetables of higher quality. Timetables combined from larger sub-solutions also have lower penalties than those obtained by the GHT method alone. However, the sub-solutions provided by retrieving larger cases are much more likely to be destroyed in the adaptation to fulfil the new constraints of the new cases and thus reusing smaller sub-solutions yields better results than when reusing larger sub-solutions upon solving the same problems.

The results of our experiments on case bases of small simple and complex cases are illustrated in Figure 8. We can see that CBR with case bases of complex cases provides better results than those produced by case bases of simple cases. Also, our previous tests[36] showed that complex cases in the case base provide more scheduling structures and lead to a higher proportion of successful retrievals than those from simple cases. So by building a case base of small complex cases, the multiple-retrieval CBR approach will perform the best in reusing previous small scheduling structures to provide a good basis for generating high quality timetables.

FIGURE 8 ABOUT HERE.

Comparison of Retrieval Time on Different Case Bases

The retrieval time of the multiple-retrieval CBR approach varies on different case bases for different new cases. We do not present the solution times of adaptation as they are just a few seconds in the worst case. The experiments are run on a Pentium III 800 Hz PC with 128MB memory. The overall retrieval times for new problems on the case bases with simple and complex cases are presented in Figure 9, showing that retrieval in case bases with small cases takes longer than with large cases. Retrieval in the case base with 5 small cases requires the longest time because the case base will provide small sub-solutions in every retrieval. Thus more retrievals on the case base are needed for the new case. Also the decision tree built from complex cases is much larger than that built from simple cases because a larger number of sub-graph structures is stored in the decision tree. Thus retrieval on the decision tree built from small complex cases takes much more time. With the limited number of scheduling structures that 5 simple cases can provide, more time is needed to find a match from the case base. Large cases provide larger sub-solutions for the new cases and thus less retrievals are needed so retrievals in case bases of large simple course cases need less time.

FIGURE 9 ABOUT HERE.

The retrieval time for case bases of complex cases shows a similar pattern to that of simple cases. The longest retrieval time is needed for the case base with 5 small complex cases. The case bases storing complex cases are much larger than those of simple cases, so the retrieval time is longer than that for the simple cases addressing the same new case.

Multiple-Retrieval CBR as the Initialisation Method for Local Search (Meta-)Heuristics

The results of our experiments led to a natural question: would the suggested CBR approach provide a good starting point for local search (meta-)heuristics such as Hill Climbing, Tabu Search and Simulated Annealing. The motivation here is that the CBR approach might be able to generate good solutions which local search could then "fine tune". With this question in mind, we carried out another set of experiments to investigate the possibility of employing the multiple-retrieval CBR with small complex cases as an initialisation method for local search methods. We compare the results from this method with results that employ initialisation by GHT alone. The table in Figure 10 presents the penalties of timetables generated by local search methods with the multiple-retrieval CBR and with GHT as the initialisation methods. The figures presented in parentheses give the number of new cases that cannot obtain feasible solutions by the specific methods. The best average results over all of the problems with all of the methods are presented in bold in the table. Due to the fact that not all of the new cases can obtain feasible solutions by Hill Climbing, we present only the results from Simulated Annealing and Tabu Search in the chart in Figure 10. We can observe that all the local search methods with multiple-retrieval CBR as the initialisation method significantly outperform local search methods with GHT as initialisation. Recall that GHT is a highly effective method which is commercially implemented in institutions around the world. The multiple-retrieval CBR does indeed provide a good starting point for the local search methods for these problems. It is particularly interesting that multiple-retrieval CBR with small complex cases as an initialisation method for Simulated Annealing provides the best results over all the other methods investigated in this paper.

FIGURE 10 ABOUT HERE.

Conclusions

This work demonstrates the value of investigating CBR for solving course timetabling problems. The knowledge implicitly embedded in previous high quality timetables is modelled and stored in a case base to help provide good quality timetables quickly and to avoid a large amount of computation and searching time. The multiple-retrieval CBR can be employed on timetabling problems of different sizes. Large timetabling problems are tackled by a partitioning process that is carried out recursively to automatically decompose the problems into smaller solvable sub-problems. The solutions of the partitioned sub-problem can be obtained by adapting high quality timetables from the retrieved problems that have similar constraints. High quality scheduling structures in the sub-solutions found by multiple retrievals are retained after the combination in the adaptation phase. These structures provide good scheduling blocks for the final solution of the new problem. By employing this approach, cases in the case base that are much smaller than the new problem to be solved can be reused repeatedly for solving parts of the new problem and thus the case base does not have to contain a large amount of large cases. This avoids the memory problem that plagues many structured CBR systems.

For every sub-problem that has been partitioned, there are always some retrieved cases (though with different similarities) for reuse. The differences between the retrieved cases and parts of the new problem are recorded and provide the adaptation information, leading to an efficient adaptation-guided retrieval. Thus the retrieved cases are guaranteed to be adaptable. A similarity measure takes into consideration how difficult it is to adapt these blocks in the retrieved cases according to the differences recorded to fulfil the constraints of the original problem.

One of the main motivations for the research described in this paper is the goal of developing a timetabling system that can operate at a higher level of generality than current technology can support. Such systems would, of course, be much less resource intensive to implement and would be applicable to a wider range of problems. The aim here is not (necessarily) to beat the well-studied meta-heuristic approaches that tend to be very problem specific. Rather, the aim is to develop systems that can deal with a wider range of problems and yet can still produce solutions that are comparable with the problem specific meta-heuristics. Meta-heuristic approaches to solve timetabling problems can be sensitive to initial parameter settings. The CBR approach, of course, has no such drawbacks, and as such, it may offer significant opportunities in the development of fundamentally more general timetabling/scheduling systems.

On the other hand, research into meta-heuristic methods has provided significant advances in timetabling technology. In addition, Burke, Newall and Weare[29] and Corne and Ross[74] have shown that appropriate initialisation strategies can improve the overall performance of timetabling meta-heuristics. Another potential impact that CBR could have on timetabling research is in its employment as an initialisation method for meta-heuristic methods. We have demonstrated in this paper that the multiple-retrieval CBR approach obtains good sub-solutions for the new problem and provides a good initialisation strategy for local search (meta-)heuristics such as Hill Climbing, Tabu Search and Simulated Annealing. Indeed, the CBR approach is able to employ past experience about solving “similar” problems to provide a high quality solution to the problem in hand. The local search (meta-)heuristics are then able to take such solutions and “fine-tune” them to provide further improvement.

Some Future Research Directions

A large number of experiments have been carried out in this paper. Future work will include testing our multiple-retrieval CBR system on sets of real-world benchmark course timetabling data. A range of benchmark course timetabling problems have been made available at http://www.idsia.ch/Files/ttcomp2002/ and http://iridia.ulb.ac.be/~msampels/ttmn.data/. These problems have been recently addressed in[75, 76, 77] and [24, 78]. We are currently putting together some more benchmark course timetabling problems. They are available at http://www.asap.cs.nott.ac.uk/themes/tt and the authors welcome further contributions from other timetabling researchers. Current research in course timetabling is trying to provide a CBR mechanism that can be easily adapted to solve a range of course timetabling problems. We also believe that, because of the general modelling method used, the basic mechanism of our structured multiple-retrieval CBR approach will be applicable in a range of problems (where the problems can be modelled as attribute graphs) like educational exam timetabling, and other types of constraint satisfaction problems.

Journal of Operations Research Society, 57(2): 148-162, 2006.

Of course, a major research question for future work is how to employ CBR for large real-world problems because the decision tree can grow exponentially. However, there is some promising recent research work which can help to deal with this problem. Some meta-heuristic methods including Memetic Algorithms[79, 80] that have been studied recently for graph matching may be potentially beneficial for the sub-graph matching in our CBR approach, but this hypothesis has to be tested as the subject of future work.

Acknowledgements

We would like to thank Dr. Jim Newall for his comments on this work. We would also like to acknowledge the invaluable comments from the anonymous referees. One referee, in particular, was extremely helpful in substantially improving this paper.

References

1. Wren A and Rousseau J-M (1995). Bus driver scheduling - an overview. In: Daduna JR, Branco I and Paix'ao JMP (eds.). *Computer-Aided Transit Scheduling*. Springer-Verlag, 173-187.
2. Easton K, Nemhauser G and Trick M (2004). Sports Scheduling. In: Leung J (ed.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*, Chapter 52. Published by CRC Press, USA.
3. Cheang B, Li H, Lim A and Rodrigues B (2003). Nurse rostering problems – a bibliographic survey. *European Journal of Operational Research*. **151**: 447-460.
4. Burke E, 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.
5. Carter M and Laporte G (1995). Recent developments in practical examination timetabling. In: Burke E and Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 3-21.
6. Bardadym V (1995). Computer-aided school and university timetabling: the new wave. In: Burke E and Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 22-45.
7. Carter M and Laporte G (1997). Recent developments in practical course timetabling. In: Burke E and Carter M (eds.): The Practice and Theory of Automated Timetabling: Selected papers from the 2nd International Conference. Lecture Notes in Computer Science 1408, Springer-Verlag: Berlin, pp 3-19.

8. Burke E and Ross P (1995). *The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference*. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin.
9. Burke E and Carter MW (1997). *The Practice and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference*. Lecture Notes in Computer Science 1408, Springer-Verlag: Berlin.
10. Burke E and Erben W (eds.) (2000). *The Practice and Theory of Automated Timetabling: Selected Papers from the 3rd International Conference*. Lecture Notes in Computer Science 2079. Springer-Verlag: Berlin.
11. Burke E and de Causmaecker P (2002). *The Practice and Theory of Automated Timetabling: Selected Papers from the 4th International Conference*. Lecture Notes in Computer Science 2740, Springer-Verlag: Berlin.
12. Burke E and Trick M (2004). *The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*. Pittsburgh, USA.
13. Wren A (1995). Scheduling, timetabling and rostering - a special relationship? In: Burke E and Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 46-75.
14. Burke E, Petrovic S (2002). Recent research directions in automated timetabling. *EJOR*, **140**(2): 266-280.
15. Petrovic S and Burke E (2004). University timetabling. Leung J (ed.) *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. Chapter 45. Published by CRC Press.
16. Schaerf A (1999). A survey of automated timetabling. *Artificial Intelligence Review* **13**: 87-127.
17. Carter M (1986). A lagrangian relaxation approach to the classroom assignment problem. *IFOR* **27**(2): 230-246.
18. de Werra D. (1985). Graphs, hypergraphs and timetabling. *Methods of Operations Research* (Germany F.R.). **49**: 201-213.
19. Burke E, Kingston J and de Werra D (2004). Applications to timetabling, In: Gross J and Yellen J (eds.), *Handbook of Graph Theory*, Chapman Hall/CRC Press. pp 445-474.
20. Schaerf A (1996). Tabu search techniques for large high-school timetabling problems. Proceedings of the Thirteenth National Conference on Artificial Intelligence, AAAI Press, Menlo Park, CA. pp 363-368.
21. Costa D (1994). A tabu search for computing an operational timetable. *EJOR*, **76**: 98-110.
22. Abramson D (1991). Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Man. Sc.* **37**: 98-113.
23. Elmohamed MAS, Coddington P and Fox G (1997). A comparison of annealing techniques for academic course scheduling. In: Burke E, Carter M (eds.) The Practice and Theory of Automated Timetabling: Selected Papers from the 2nd International Conference. Lecture Notes in Computer Science 1408, Springer-Verlag: Berlin, pp 92-112.

24. Burke E, Bykov Y, Newall J and Petrovic S. (2003). A time-predefined approach to course timetabling. *Yugoslav Journal of Operations Research*, **13**(2): 139-151.

25. Erben W and Keppler J (1995). A genetic algorithm solving a weekly course-timetabling problem. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 198-211.

26. Paechter B, Cumming A and Luchian H (1995). The use of local search suggestions lists for improving the solutions of timetable problems with evolutionary algorithms. In: Fogarty T (ed.) AISB Workshop on Evolutionary Computing. Lecture Notes in Computer Science 993, Springer-Verlag, pp 86-93.

27. Carrasco M and Pato M(2000). A multiobjective genetic algorithm for the class/teacher timetabling problem. In: Burke E, Erben W (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 3rd International Conference. Lecture Notes in Computer Science 2079, Springer-Verlag: Berlin, pp 3-17.

28. Burke E, Newall J and Weare R (1996). A memetic algorithm for university timetabling. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 241-250.

29. Burke E, Newall J and Weare R (1998). Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation Journal* (special issue on scheduling) **6**(1): 81-103.

30. Lajos G (1996). Complete university modular timetabling using constraint logic programming. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 146-161.

31. Deris SB, Omatu S, Ohta H and Samat PABD (1997). University timetabling by constraint-based reasoning: A case study. *JORS,* **48**(12): 1178-1190.

32. Nonobe K and Ibaraki T (1998). A tabu search approach to the constraint satisfaction problem as a general problem solver. *EJOR,* **106**: 599-623.

33. ten Eikelder HMM and Willemen RJ (2000). Some complexity aspects of secondary school timetabling problems. In: Burke E, Erben W (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 3rd International Conference. Lecture Notes in Computer Science 2079, Springer-Verlag: Berlin, pp 18-27.

34. de Werra D. (2002). Complexity of some special types of timetabling problems. *Journal of Scheduling.* **5**(2): 171-184.

35. Burke E, MacCarthy B, Petrovic S and Qu R (2000). Structured cases in CBR – re-using and adapting cases for timetabling problems. *Knowledge-based System.* **13**(2-3): 159-165.

36. Burke E, MacCarthy B, Petrovic S and Qu R (2001). Case-based reasoning in course timetabling: an attribute graph approach. In: Aha DW, Watson I (eds.) Case-Based Reasoning Research and Development. Proceedings

of 4th International Conference on Case-Based Reasoning (ICCBR-2001). Lecture Notes in Artificial Intelligence 2080, Springer-Verlag: Berlin, pp 90-104.

37. Kolodner J (1993). *Case Based Reasoning*. Morgan Kaufmann.

38. Leake D (ed.) (1996). *Case-Based Reasoning: Experiences, Lessons, & Future Directions*. The AAAI Press.

39. Aamodt A and Plaza E (1994). Case-based reasoning: foundational issues, methodological variations and system approaches. *AI Communications*, **7**(1): 39-59.

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

41. Burke E, Elliman D, Ford P and Weare R (1996). Examination timetabling in British universities - a survey. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 76-92.

42. Burke E, De Causmaecker P, Vanden Berghe G. (1998). A hybrid tabu search algorithm for the nurse rostering problem. In: McKay B, Yao X, Newton CS, Kim JH & Furuhashi T (eds.) Proceedings of the Second Asia-Pacific Conference on Simulated Evolution and Learning, Canberra, Australia. 187-194.

43. Watson JP, Rana S, Whitely LD and Howe AE (1999). The impact of approximate evaluation on the performance of search algorithms for warehouse scheduling. *Journal of Scheduling*, 2(2): 79-98.

44. Aickelin U and Dowsland K (2000). Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. *Journal of Scheduling*, John Wiley & Sons: Chichester, West Sussex UK. **3**(3):139-154.

45. Di Gaspero and Schaerf A (2000). Tabu Search Techniques for Examination Timetabling. In: Burke E, Erben W (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 3rd International Conference. Lecture Notes in Computer Science 2079, Springer-Verlag: Berlin, pp 104-117.

46. Marir F and Watson I (1994). Case-based reasoning: a categorised bibliography. *The Knowledge Engineering Review* **9**: 355-381.

47. Burke E, 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 (eds.) *Handbook of Meta-Heuristics*, Kluwer, Boston. pp 457-474.

48. Koton P (1989). SMARTplan: a case-based resource allocation and scheduling system. In: Hammond K (ed.) Proceedings of the Workshop on Case-based Reasoning (DARPA), San Francisco, California. Morgan Kaufmann. pp 285-289.

49. Hennessy D and Hinkle D (1992). Applying case-based reasoning to autoclave loading. *IEEE Expert* **7:** 21-26.

50. Bezirgan A (1993). A case-based approach to scheduling constraints. In: Dorn J, Froeschl KA (eds.): *Scheduling of Production Processes*. Ellis Horwood Limited: Chichester, UK, pp 48-60.

51. 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.

52. Dorn J (1995). Case-based reactive scheduling. In: R Kerr, E Szelke (eds.). *Artificial Intelligence in Reactive Scheduling*, London: Chapman & Hall, pp 32-50.

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

54. Beddoe G and Petrovic S (2003). A novel approach to finding feasible solutions to personnel rostering problems. In Proceedings of the 14th Annual Conference of the Production and Operations Management Society (POM), Savannah, Georgia, United States.

55. 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 Third UK Case-Based Reasoning Workshop, University of Manchester, U.K.

56. Szelke E and Markus G (1997). A learning reactive scheduler using CBR/L. *Computer Industry* **33**: 31-46.

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

58. MacCarthy B and Jou P (1995). A case-based expert system for scheduling problems with sequence dependent set up times. In: Adey RA, Rzevski G (eds.): *Applications of Artificial Intelligence*. Engineering X. Computational Machines Publications: Southampton, pp 89-96.

59. MacCarthy B and Jou P (1996). 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: Bradford, UK, pp 211-218.

60. Carter M (1983). A decomposition algorithm for practical timetabling problems. Working paper 83-06, Industrial Engineering, University of Toronto.

61. Robert V and Hertz A (1995). How to decompose constrained course scheduling problems into easier assignment type subproblems. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference, Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 364-373.

62. Weare RF (1995). Automated examination timetabling. PhD dissertation, University of Nottingham, Department of Computer Science.

63. Burke E and Newall J (1999). A multi-stage evolutionary algorithm for the timetabling problem. *The IEEE Transactions on Evolutionary Computation* **3**(1): 63-74.

64. Marir F and Watson I (1995). Representation and indexing building refurbishment cases for multiple retrieval of adaptable pieces of cases. In: Veloso M, Aamodt A (eds.): Case-Based Reasoning Research & Development: Proceedings of the 1st International Conference on Case-Based Reasoning (ICCBR-95), Springer-Verlag: Heidelberg, pp 55-66.

65. Smyth B, Cunningham P and Keane M (2001). Hierarchical case-based reasoning. *IEEE Transactions on Knowledge & Data Engineering*, **13**: 793-812.

66. Börner K, Coulon CH, Pippig E and Tammer EC (1996). Structural similarity and adaptation. In: Smith I, Faltings B (eds.): Advances in Case-based Reasoning: Proceedings of the 4th European Workshop (EWCBR'98), Springer-Verlag, Heidelberg, pp 58-75.

67. Andersen WA, Evett MP, Kettler B and Hendler J (1994). Massively parallel support for case-based planning. *IEEE Expert* **7**: 8-14.

68. Macedo L and Cardoso A (1998). Nested graph-structured representations for cases. In: Smyth B and Cunningham P (eds.) Advances in Case-Based Reasoning: Proceedings of the 4th European Workshop on Case-Based Reasoning, Lecture Notes on Artificial Intelligence 1488, Springer-Verlag: Heidelberg, pp 1-11.

69. Sanders KE, Kettler BP and Hendler JA (1997). The case for graph-structured representations. In: Leake D, Plaza E (eds.) Case-Based Reasoning Research and Development: Proceedings of the 2nd International Conference on Case-Based Reasoning (ICCBR-97), Springer-Verlag, Berlin, pp 245-254.

70. Ricci F and Senter L (1998). Structured cases, trees and efficient retrieval. Advances in Case-Based Reasoning: In: Smyth B, Cunningham P (eds.) Proceedings of the 4th European Workshop on Case-Based Reasoning. Lecture Notes on Artificial Intelligence 1488, Springer-Verlag: Heidelberg, pp 88-99.

71. Gebhardt F (1995). Methods and systems for case retrieval exploiting the case structure. FABEL-Report 39, GMD, Sankt Augustin.

72. Gebhardt F (1997). Survey on structure-based case retrieval. *The Knowledge Engineering Review* **12**:41-58.

73. Burke E, Newall J and Weare R (1998). A simple heuristically guided search for the timetable problem. In: Alpaydin E. (ed.), Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems, Tenerife, Spain, pp 574-579.

74. Corne DW and Ross PM (1996) Peckish initialisation strategies for evolutionary timetabling. In: Burke E, Ross P (eds.) The Practice and Theory of Automated Timetabling: Selected papers from the 1st International Conference. Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin, pp 227-240.

75. Rossi-Doria O, Blum C, Knowles J, Sampels M, Socha K and Paechter B (2002). A local search for the timetabling problem. In Proceedings of the 4th International Conference on the Practice And Theory of Automated Timetabling, PATAT 2002, pp. 124-127.

76. Socha K, Knowles J and Sampels M (2002). A max-min ant system for the university course timetabling problem. In Proceedings of the 3rd International Workshop on Ant Algorithms, ANTS 2002, Lecture Notes in Computer Science, Vol. 2463, Springer, pp. 1-13.

77. Burke E, Kendall G and Soubeiga E (2003). A tabu search hyperheuristic for timetabling and rostering. *Journal of Heuristics*. **9**(6): 451-470.

78. Kostuch P (2004). The university course timetabling problem with a 3-phase approach. In The Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling. Pittsburgh, USA.

79. Williams ML, Wilson RC and Hancock ER (1999) Deterministic search for relational graph matching. *Pattern Recognition* **32**(7): 1255-1271.

80. Cross ADJ, Myers R and Hancock ER (2000) Convergence of a hill-climbing genetic algorithm for graph matching. *Pattern Recognition* **33**: 1863-1880.

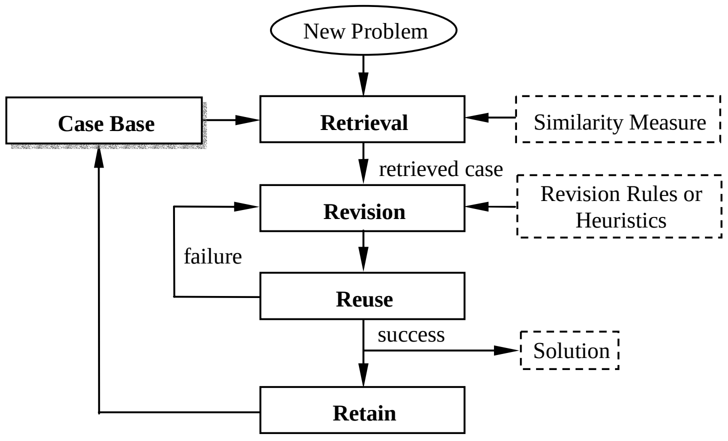


Figure 1 A Case-Based Reasoning Framework

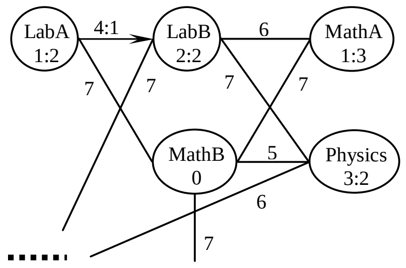


Figure 2 A Course Timetabling Problem Represented by the Attribute Graph

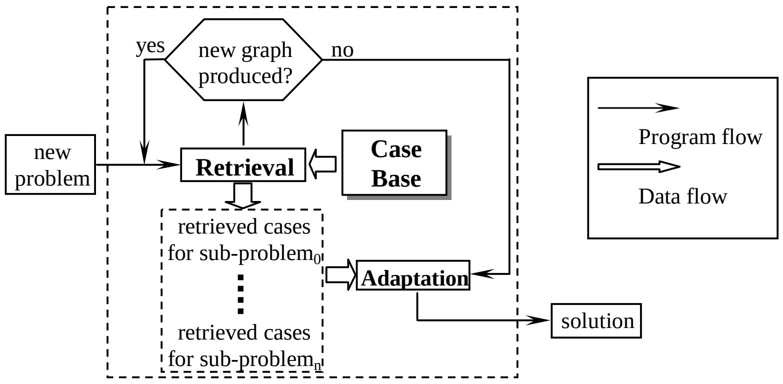


Figure 3 Schematic Diagram of the Multiple-Retrieval CBR System

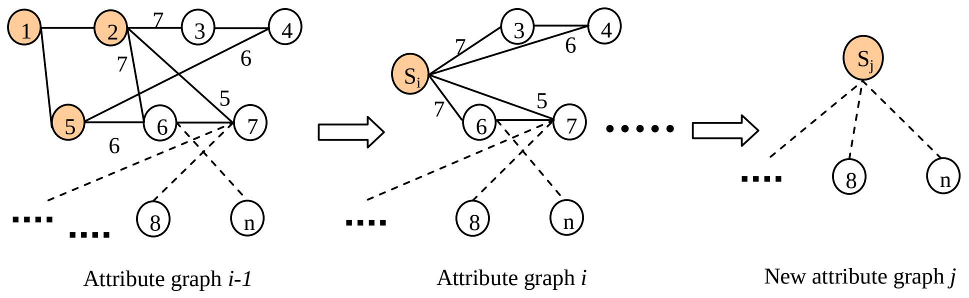


Figure 4 New Attribute Graph Generated after Each Retrieval

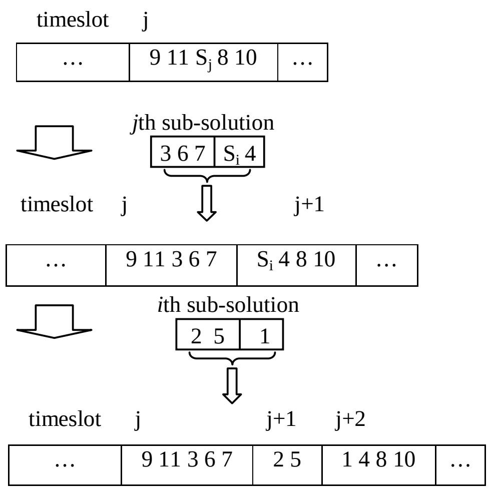
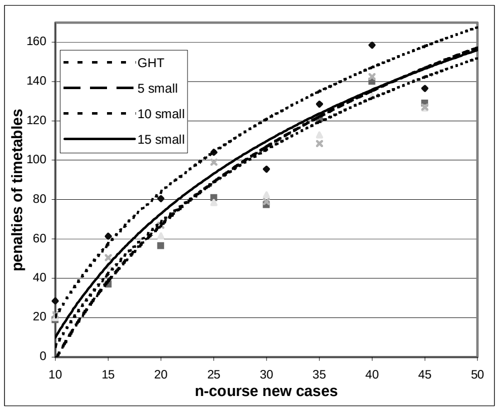
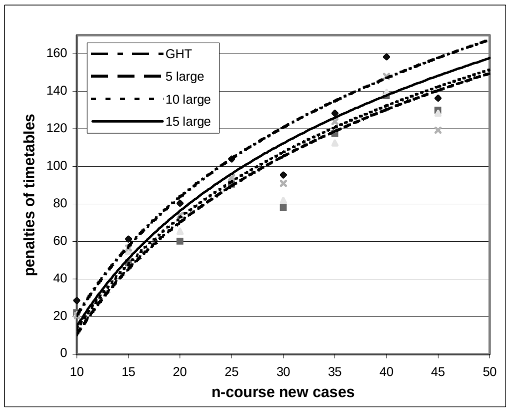


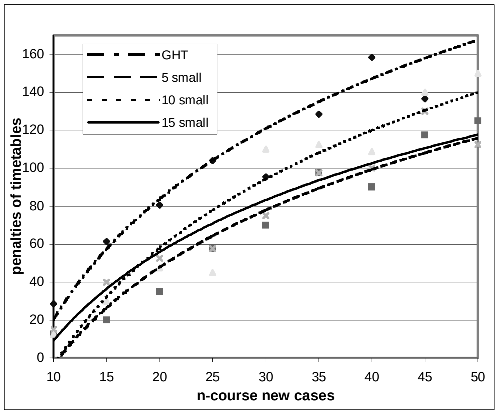
Figure 5 Combining the Solutions of the Sub-problems

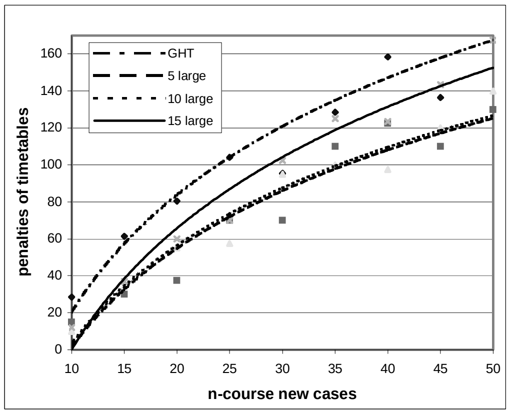




	GHT	5 small	10 small	15 small	5 large	10 large	15 large
10-course new case	28.5	**19**	20	21.5	22	21	20.5
15-course new case	61.4	**37**	46.5	50.5	48.5	54.5	56.5
20-course new case	80.5	**56.5**	61.5	67	60	65.5	74
25-course new case	104	81	**78.5**	99	90.5	94.5	94
30-course new case	95.5	**77.5**	82.5	79	78	82	91
35-course new case	128.5	121	113	**108.5**	117.5	112.5	124
40-course new case	158.5	140	**132.5**	142.5	137.5	139.5	148
45-course new case	136.5	129	126.5	127	130	128.5	**119.5**
50-course new case	200.5	200	193.9	199.5	**176**	182.5	193

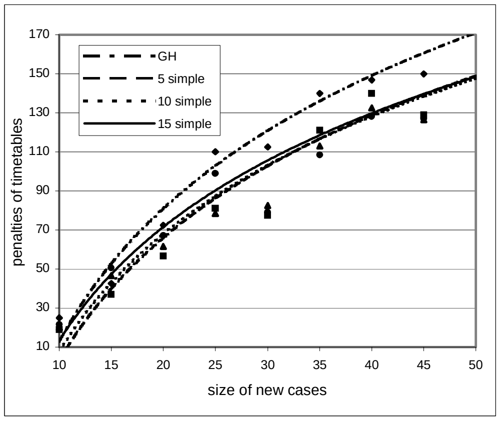
Figure 6 Penalties of Timetables by using GHT alone and CBR with (left: small, right: large) Simple Cases





	GHT	5 small	10 small	15 small	5 large	10 large	15 large
10-course new case	28.5	12.5	12.5	15	15	**10**	12
15-course new case	61.4	**20**	30	40	30	34.4	36.9
20-course new case	80.5	**35**	47.5	52.5	37.5	55	60
25-course new case	104	57.5	**45**	57.5	70	57.5	70
30-course new case	95.5	**70**	110	75	**70**	95	102.5
35-course new case	128.5	**97.5**	112.5	**97.5**	110	100	125
40-course new case	158.5	**90**	108.8	100	122.5	97.5	123.5
45-course new case	136.5	117.5	140	130	**110**	120	143.5
50-course new case	200.5	125	150	**112.5**	130	140	167.5

Figure 7 Penalties of Timetables by using GHT alone and CBR with (left: small, right: large) Complex Cases



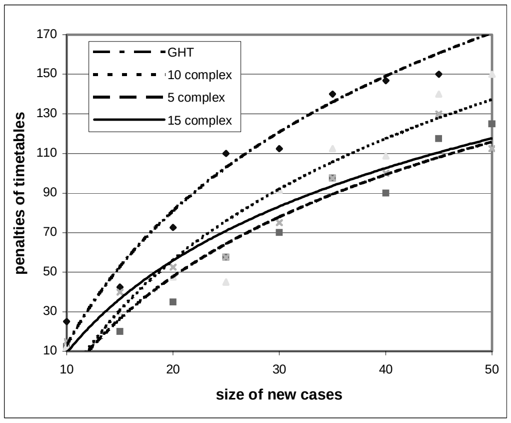
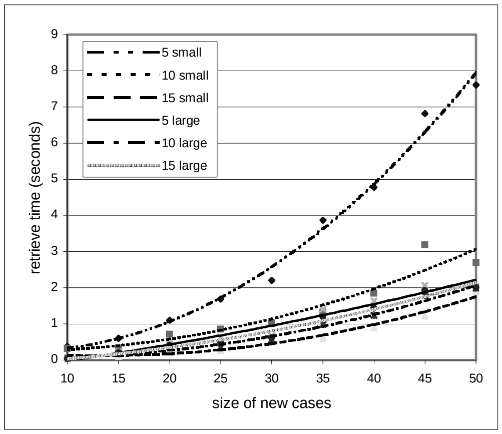


Figure 8 Penalties of Timetables by using GHT alone and CBR with (left: simple, right: complex) Small cases



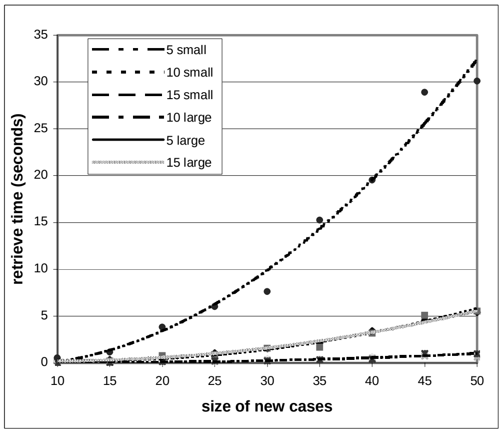


Figure 9 Retrieval Time on Case Bases (Left: simple cases; Right: complex cases)

	10-course	20-course	30-course	40-course	50-course	60-course	70-course	80-course	90-course
GHT + TS	135	172	305	266	295	294	307	266	245
CBR with 5 cases + TS	49	99	175	169	186	159	199	196	197
CBR with 10 cases + TS	46	80	130	166	117	130	149	153	171
CBR with 15 cases + TS	31	55	101	178	120	133	149	134	151
GHT + SA	22	45	55	65	77	96	121	103	134
CBR with 5 cases + SA	16	33	49	63	76	98	121	107	134
CBR with 10 cases + SA	**14**	**25**	**40**	**58**	**64**	**81**	**104**	**98**	**128**
CBR with 15 cases + SA	**14**	**26**	**42**	**59**	**68**	**87**	**97**	100	**124**
GHT+HC	152(3)	56(2)	66(2)	87(1)	85	106	127	111	156
CBR with 5 cases + HC	144	63	74	134	85	103	113	101	144
CBR with 10 cases + HC	148(3)	42(1)	59(2)	79(1)	74	92	112	**99**	140
CBR with 15 cases + HC	150(3)	43(1)	57(2)	78(1)	78	93	105	103	139

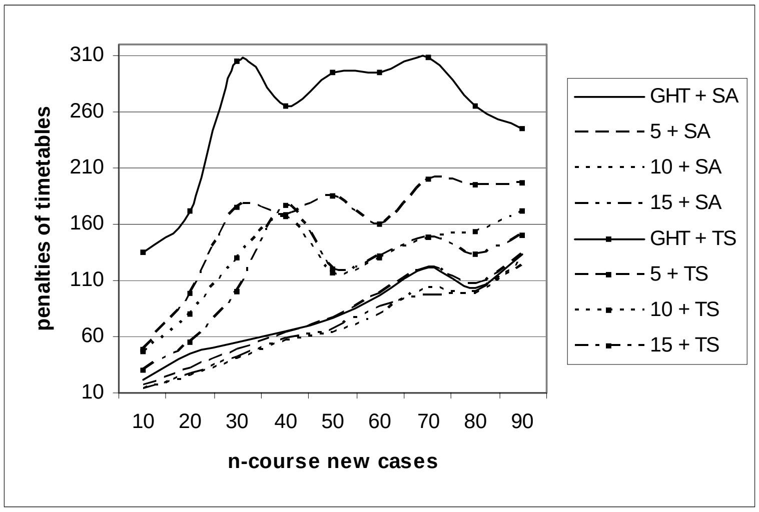


Figure 10 GHT and Multiple-Retrieval CBR with Small Complex Cases as the Initialization Methods for Local Search Methods

Label	Attribute	Value(s)	Notes
0	Ordinary course	N/A	Takes place once a week
1	Multiple course	N (No. of times)	Takes place N times a week
2	Pre-fixed course	S (Slot No.)	Assigned to timeslot S
3	Exclusive course	S (Slot No.)	Not assigned to timeslot S

Table 1 Vertex Attributes of Course Timetabling Problems

Label	Attribute	Value(s)	Notes
4	Before/after	1 or 0 (direction)	Before or after another course
5	Consecutive	N/A	Be consecutive with each other
6	Non-consecutive	N/A	Not consecutive with each other
7	Conflict	N/A	Not assigned simultaneously

Table 2 Edge Attributes of Course Timetabling Problems

Table 1 Vertex Attributes of Course Timetabling Problems

Table 2 Edge Attributes of Course Timetabling Problems