CASE-BASED REASONING FOR COURSE TIMETABLING PROBLEMS

by Rong Qu, BSc

Thesis submitted to the University of Nottingham
for the degree of Doctor of Philosophy
August 2002

Table of Content

List of Figures

List of Tables

Abstract

The research in this thesis investigates Case-Based Reasoning (CBR), a Knowledge-Based Reasoning technique that proved to be capable of providing good solutions in educational course timetabling problems. Following the basic idea behind CBR, experiences in solving previous similar timetabling problems are employed to find the solutions for new problems.

A basic CBR system that is hierarchically organized with structured knowledge representations by attribute graphs is proposed in Chapter Four. The system is then further improved to solve a wider range of problems, which is described in Chapter Five. Evaluations on a large number of experiments indicate that this approach could provide a significant step forward in timetabling and scheduling research.

This basic system works well on relatively small problems. To deal with this drawback a multiple-retrieval approach that partitions large timetabling problems into small solvable sub-problems is presented in Chapter Six. Good results are obtained from a wide range of experiments.

In Chapter Seven, a new idea is introduced in CBR for solving timetabling problems by investigating the approach to select the most appropriate heuristic method rather than to employ it directly on the problem, in the attempt to raise the level of generality at which we can operate. All the evidence obtained from the first stage experiments indicates that there is a range of promising future directions.

Finally in Chapter Eight the results of the work are evaluated and some directions for future work are present.

Acknowledgements

I would like to thank my supervisors, Professor Edmund Burke, Dr. Bart MacCarthy and Dr. Sanja Petrovic for their valuable advice throughout the whole course of my research.

Also thanks to my husband, YiJun Xue, for his consistent support and my lovely daughter, Jessica Xue, for giving me immense motivation in my study. Thanks my parents, this thesis would not have been possible without their consistent support.

To all the ASAP group members for their effort in making this group a harmonious environment in which to work.

Chapter 1 Introduction

1.1 Background and Motivation

1.1.1 Background

In real-world problem solving, people usually use experience that was successful in solving previous, similar problems. Knowledge-Based techniques in Artificial Intelligence (AI) mimic the reasoning process people use by modelling the experiences, storing them in a knowledge base and reusing those experiences when solving new problems. In Expert Systems, the experiences are usually modelled as rules, which will be used to construct the solutions for new problems. In Case-Based Reasoning (CBR) (Kolodner, 1993), experiences are modelled into a different form as concrete problems with their solutions (*cases*). New problems are solved based on solutions of retrieved cases in previous similar situations from the knowledge base (*case base*).

The mechanism behind CBR is supported by the study of cognitive computer science and psychology that human reasoning based on experiences collected from previous problem solving in similar situations (Leake, 1996). In many real-world problem solving situations, people recall experiences in solving previous similar problems and reuse them with small modifications according to the different requirements from those of the previous ones. Reasoning is based on the assumption that ‘similar problems may have similar solutions’ and ‘the types of problems an agent encounters tend to recur’ (Leake, 1996). In problem domains where that assumption holds, previous solutions may be reused as good starting points in solving new similar problems. CBR’s problem solving by reusing the solutions of similar problems mimics the behaviour of experts and avoids reasoning from scratch.

1.1.2 Motivation

CBR is a knowledge-based paradigm that attempts to reuse previous knowledge for current, similar problems. This research investigates how CBR, a quite different methodology from many other AI and Operational Research (OR) techniques in timetabling, can be employed to solve the problem effectively. In schools and universities, altering “last year’s timetable” to create a solution for the problem in hand is an approach favoured by many timetabling officers. This is because the requirements in the new problem usually do not change significantly from previous instances. Thus parts of the previous timetable could be reused and a significant amount of effort and time could be saved.

1.1.3 Why CBR for Timetabling

One of the important contributions of Knowledge-Based techniques is their application in problems where rich experiences and knowledge are available. However, in practice, formalising experiences as a set of rules in complex domains such as timetabling may lead to the well-known bottleneck problem of knowledge acquisition. For example, in rule-based Expert Systems, collecting experiences and knowledge, modelling them in the form of rules and building up a complete rule base may take a considerable amount of work. Missing out just one rule may lead to a complete failure in reasoning. In some problem domains, it is impossible to ensure completeness. Also, sometimes the knowledge in some ill-structured domains (such as timetabling), with too many possible details, is difficult, or impossible, to be modelled explicitly as rules.

CBR is potentially a very good technique for addressing the bottleneck problems mentioned above (MacCarthy and Jou, 1996; Schmidt, 1998). Collecting cases, which can implicitly capture the knowledge, rather than modelling it as rules in timetabling may be relatively easy. When no exact match can be found from the case base, which is usually the case in complex problems, a similar case may be retrieved and the solution of this related problem might also be applicable to provide a good starting point for the new problem after a small amount of adaptation. By employing matching, selection and searching techniques, CBR is potentially good at solving complex timetabling problems, avoiding a large amount of computation.

Other problems that may exist in Knowledge-Based Systems rise from the evolved new experiences over time in real-world problem solving. In this case the whole rule-based system may need to be re-developed to build in the rules of these new experiences. In CBR, however, the system can be updated (to reflect new experiences) by retaining the newly solved problem (learning ability in CBR). Thus system performance can be improved by learning and embedding new knowledge automatically, saving a lot of human effort and solving new problems efficiently.

Timetabling problems are usually very large and have complicated constraints, which can be easily mapped as constraint satisfaction problems (CSP) (Carter and Laporte, 1995&1997). CBR may also be a valuable technique for such problems as it puts emphasis indirectly on constraint-directed search.

The observations presented above provide the motivations for our research on CBR for timetabling. To our knowledge, no other research has been reported applying CBR specifically to educational timetabling problems. However, recent work has been undertaken to investigate using CBR for nurse rostering problems, which is a special type of timetabling problem determining the shifts of staff in hospital over a fixed period of time (Scott and Simpson, 1998; Petrovic, Beddoe and Berghe, 2002).

1.2 Aims and Issues

There are many issues in CBR that have been investigated in research and practice. They can be mainly classified into the following five groups:

- Representation – How the problem should be represented to properly describe its situation?
- Indexing – How the indices should be selected so that cases can be organised in the case base and retrieved in certain situations?
- Retrieval – The most similar and reusable cases need to be retrieved efficiently and effectively from the case base?
- Adaptation – Retrieved cases need to be adapted using domain knowledge, according to the different requirements of the new problem;
- Retention – Newly solved cases need to be selected and stored into the case base. How should this be done so that the new knowledge in either success or failure can be learned?

Some surveys are available presenting the research on these issues (Marir and Watson, 1994; Mantaras and Plaza, 1997). Each of them may form a significant research topic in its own. To build an effective system in complex problem domains, these issues need to be addressed dependently and co-operatively. This thesis will investigate mainly the first four issues on CBR in timetabling problems. The most important issues addressed are:

- Representation – With the significant complexity in real world situations, timetabling problems are difficult to formalise. How to represent them to explore the deeper knowledge forms one of the most important objectives;
- Case base management – How should cases be indexed in the case base so that relevant cases can be retrieved effectively?
- Retrieval – How can the retrieval be carried out so that we can find reusable timetables efficiently from the case base?
- Adaptation – Retrieved cases are usually different from the new case. Research issues include how to adapt the retrieved timetables into the new situation concerning the domain knowledge.

Timetabling problems are a special type of scheduling problems, for which a wide range of techniques and approaches in both AI and OR have been studied (Burke and Ross, 1995; Barddadym 1995; Carter and Laporte, 1995&1997; Burke and Carter, 1997; Schaerf 1999; Burke and Erben, 2000; Burke, and Causmaecker, 2002). This thesis presents a major investigation of CBR for course timetabling, aiming at establishing a general framework of CBR for a range of scheduling problems.

1.3 Organisation of the Thesis

In this chapter we presented the background and motivation of applying CBR for timetabling problems. The remaining chapters of this thesis are organised in the following way:

Chapter Two reviews the research classified into different categories and applications in CBR. In particular, current research on CBR in scheduling is discussed.

Chapter Three introduces educational timetabling problems and presents different methods in research and practice for course timetabling.

Chapter Four presents a basic structure of our proposed CBR system employing a structured representation by attribute graphs. The mechanism is illustrated theoretically and a simple example is given to explain the retrieval, re-use and adaptation of structured cases.

Chapter Five investigates the issues of retrieval of a wider range of reusable cases. Evaluations on the system performance are given by experiments on a number of systematically constructed case bases. Showing the promising results on relatively small problems, the system will be used as the basis for developing a multiple-retrieval approach.

Chapter Six describes a multiple-retrieval approach based on the basic CBR system already developed, aiming at solving large course timetabling problems. Partitioning and composition techniques are used to decompose the problem into solvable sub-problems and to combine the sub-solutions into a final solution.

Chapter Seven presents a new approach using CBR as the selector of good problem-solving heuristics/strategies in solving previous similar problems. Problem-solving heuristics, not the concrete solutions, are reused to help constructing new timetables.

Chapter 8 presents concluding comments and some directions for future research work on CBR on timetabling problems.

Chapter 2 Case-Based Reasoning (CBR)

2.1 What is CBR?

Case-Based Reasoning (Kolodner, 1993) is a Knowledge-Based Reasoning technique that solves problems by retrieving the most similar previous problems (*cases*) from a store called the *case base* and by reusing the knowledge and experiences from these cases. If necessary, the retrieved solutions or problem solving strategies are adapted (using domain knowledge) so that they are applicable for the new problem. The solved new problems may be retained by updating the case base. Leake (1996) described CBR as:

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

In CBR, all the problems are represented as *cases*, which were defined by Kolodner and Leake (1996) as:

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

A case usually has two major parts: the problem itself with the context describing the environments it should be retrieved; and the solution of the problem or the lesson it will teach. Throughout this thesis, *source case* will be used to denote the cases in the case base; *target case* will be used to denote the new problem to be solved.

2.1.1 CBR Framework

CBR can be seen as a 4 REs' cyclic process: REtrieve, REuse, REvise and REtain (Aamodt and Plaza, 1994). This cycle is cited in Figure 2-1 below:

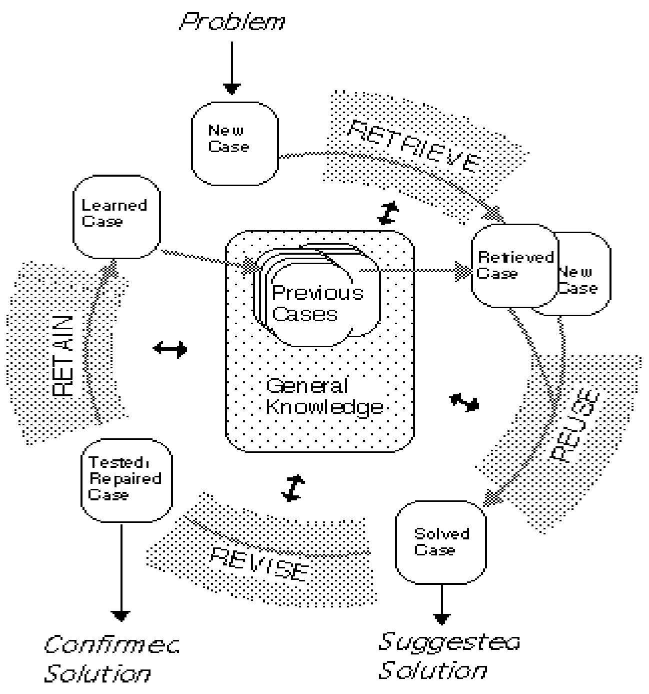


Figure 2-1 The CBR cycle (Aamodt and Plaza, 1994)

From Figure 2-1 we can see that when a target case is input into the CBR cycle, the following steps will be taken to solve it.

1. Retrieve – the most similar source cases efficiently;
2. Reuse – the retrieved solutions to solve the target case (new case);
3. Revise – the solution concerning the new requirements;
4. Retain – certain solved target cases into the case base.

2.1.2 Methodology and Issues in CBR

CBR methodology usually concerns the following issues so that different components can work co-operatively, contributing to efficient and effective system performance.

2.1.2.1 Case Representations

CBR's problem solving depends heavily upon the case representation that gives the important information for reasoning. Especially in complex problem domains, the cases need to be represented in a way that describes the sensible features of the problem that affect the solutions. Thus comparisons between cases can be carried out between source cases and target cases to find really reusable source cases.

2.1.2.2 Indexing

Indexing in CBR includes choosing proper indexing vocabulary that can distinguish between particular cases in certain situation within the case base. In different problem domains, indices could be abstract or specific details of the case, surface features or deep descriptions of the problem, etc. A good index should cover the correct and complete dimensions of the problem thus allowing an efficient matching between cases.

2.1.2.3 Case Base Maintenance and Management

It is usually easy to construct the initial case base with just a few cases as the starting point. Gradually users can update it interactively, or the system may retain some solved target cases automatically. Thus CBR inherently has a learning ability and the knowledge stored may evolve, increasing the system performance during the problem solving.

2.1.2.4 Adaptation

Adaptation, as one of the most difficult tasks (especially in a complex problem domain) in CBR, relies on both the retrieval of proper cases that need less adaptations and the utilisation of appropriate domain knowledge. Traditional methods include substitution method that replaces some part of the retrieved case, and transformation method that transfer some part of the retrieved case to fit the constraint to a new situation (see more details in

Kolodner, 1993). Recent research employing heuristic methods provides many promising prospects. This will be presented in the section under the heading “Hybridisation with CBR”.

2.1.2.5 Similarity Measure

Similarity assesses the right cases to be retrieved. In most of the CBR system, it considers the proper features and usually their importance for comparison between cases. Some researchers judge the similarity by concerning the adaptation that needs to be carried out (Smyth and Keane, 1998).

2.2 CBR in Research and Applications

CBR has been very successful in a wide range of problems over the last decade (Kolodner, 1993). A vast amount of work has been carried out concerning a wide range of issues and different techniques in CBR (Mantaras and Plaza, 1997). Concerning the interests of this thesis, in the following sections a review will be given mainly on case representation, decomposition techniques and hybridisation in CBR, from both the research and application points of view. Some open issues on CBR in scheduling are discussed.

2.2.1 Research Topics in CBR

2.2.1.1 Structurally Represented Cases

Case representation forms one of the most important issues in CBR, especially in problem domains concerning more complicated applications (e.g. scheduling, design and planning). Traditional case representation typically used a flat form of feature-value pairs (Kolodner, 1993). A value is given to describe different scales of these features in the problem. The nearest-neighbour method is extensively used to measure the similarity between two cases that gives every feature a weight and results in a weighted sum. Figure 2-2 presents an example of the similarity measure between New case and Case0 and Case1 employing the nearest-neighbour approach shown in the formula. New case, Case0 and Case1 are general timetabling problems represented in the form of feature-value pairs.

$$S(c_s, c_t) = [\sum_{i=0}^{n} sim(s_i - t_i) \times w_i] / \sum_{i=0}^{n} w_i$$

Feature	**Case0**	**Case1**	**Weight**	**New case**	**sim(s_i-t_i)**	
					Case0	**Case1**
Name	TTP0	TTP1	0	TTP		
No. of events	20	40	.5	30	0.33	0.33
No. of timeslots	7	20	.8	12	0.42	0.67
No. of students	600	1500	.2	1200	0.50	0.25
No. of rooms	5	12	.5	4	0.25	2.00
No. of clashes	10	23	.9	11	0.09	1.09
No. of consecutive events	3	10	.5	10	0.70	0.00
No. of non-consecutive events	2	12	.5	7	0.71	0.71
No. of before/after events	3	8	.3	4	0.25	1.00
No. of events with >5 clashes	8	23	.9	12	0.33	0.92
Similarity with new case	**0.37**	**0.83**				

Figure 2-2 An example of nearest neighbour method with feature-value pair representation

However, in complex domains where problem features are complicated and heavily interrelated, such as timetabling problems that are seen as constraint satisfaction problems, this flat representation is not adequate to represent the important situations where the problems occur, which in turn raises the problem of recognising the correspondence between the features in cases and qualities of the solutions. The traditional representation may lead to the retrieving of cases that are not strongly reusable and the adaptation may take as much effort as scheduling from scratch. A recent overview (Mantaras and Plaza, 1997) pointed out that the feature-value representation is the most severe limitations of existing CBR systems. This representation is not adequate for knowledge-rich applications that have higher-order relations between features. Smyth and Keane (1998) questioned the similarity assumption in CBR and introduced a concept called "adaptation-guided retrieval". It is unwarranted to assume that the most similar case is also the most appropriate from the re-use perspective. Similarity must be augmented by a deeper knowledge about how easy it is to modify a case to fit a target problem. Timetabling problems require much more complex case representations.

As the development of CBR has progressed, research has been conducted on more complex applications using more sophisticated methods that structurally represent the cases using graphs, semantic networks or trees, etc (Gebhardt, 1997), but no general theory or methodology has been identified.

Jantke (1993) defined the similarity as the mapping into a highly structured partially ordered space. The approach was studied further (Matuschek and Jantke, 1997) to formalise the structural similarity, with the aim of making it more flexible and expressive. Böner et al. (1996) proposed a CBR system to find the common structures between the target problem and a set of candidate cases that were transformed into a structural representation. Structural similarity is defined using maximum common sub-graphs that are employed as prototypes thus reducing much of the memory retrieval effort. Two systems, CHIRON and CAPER, were developed (Sanders, Kettler and Hendler, 1997) to show how the cases in graph-structured representation organised as semantic networks can support Case-Based Planning. The benefits and cost associated with graph-structured representation were discussed. Ricci and Sender (1998) used labelled trees associated with concepts to represent structured cases and the similarity measure takes into account both the structures and labels. A set of algorithms was explored to solve sub-tree-isomorphism problems. Praehofer and Kerschbaummayr (1999) proposed an approach that applied CBR in system design. Cases were structurally modelled and the retrieval was based on a graph matching algorithm (Messmer, 1995), which is also studied in this thesis. Similarity was assessed by computing the degree of fulfilment of requirements in the design.

The FABEL project (Gebhardt, 1995) and a survey (Gebhardt, 1997) provided more details of some existing systems that employed structured cases, which were classified into five groups: restricted geometric relationships; graphs; semantic nets; model-based similarities and hierarchically structured similarities.

2.2.1.2 Hybridisation with CBR

CBR is inherently suitable to be integrated with other AI/OR techniques especially in complex applications (Hunt and Miles, 1994). Some work combining CBR with other AI/OR techniques in some domains has shown the merits of this integration. For example rule-based techniques were used to solve problems where domain knowledge was understood reasonably well and CBR can be used when rule-based techniques failed (Leake, 1995; Surma and Vanhoof, 1998; and Golding and Rosenbloom, 1997). Constraint satisfaction techniques were also widely employed in integration with CBR (Sqalli, Purris and Freuder, 1999). This may also indicate the possible benefits from integrating AI/OR techniques with CBR in solving complex scheduling problems. One example in Case-Based Scheduling is CABINS (Miyashita and Sycara, 1994&1995) that utilised constraint satisfaction techniques to carry out the repair actions retrieved by CBR in solving dynamic job shop scheduling problems.

From the Cased-Based Reasoning Perspective

In CBR, the matching problems in retrieval could be solved by employing meta-heuristic methods such as Genetic Algorithms (GAs) (Shin and Han, 1999). Recently, adaptation in a wide range of CBR applications employed GAs other than traditional adaptation methods and showed promising results (Purvis and Athalye, 1997; Garza and Maher, 1999). Adaptation is one of the most difficult steps in CBR as it needs to integrate domain knowledge. Different AI techniques can be particularly suitable for using this knowledge to search for better solutions. Some work on constraint satisfaction techniques (Purvis and Pu, 1995) in adaptation has also been carried out to formalise this process on ill-structured domains.

From the Artificial Intelligence Search Methods Perspective

In Tabu Search, operator selection can be helped by employing CBR to improve the performance (Grolimund and Ganascia, 1997). GAs may benefit in solving optimisation problems from proper injection of cases into populations during the search (Louis and Li, 2000).

Initialisation plays an important role in Evolutionary Algorithms (Burke and Newall, 1998). With the assumption that similar problems may have similar solutions, retrieved good solutions of similar cases may be near to the good/optimal solution of the target case. Solutions of the source cases as initial starting points of different heuristic methods can help the search move toward the high quality/optimal solutions in the search space. This may indicate a high level of potential in investigating initialisation in AI search methods by CBR. Research on using CBR to seed GAs has shown different behaviour in solving optimisation problems (Oman and

Cunningham, 2001). Investigations on many potential issues such as how to select cases for seeding, etc need to be carried out.

2.2.1.3 Decomposition Techniques in CBR

In CBR, decomposition techniques have been mostly employed in design and planning domains where the cases were decomposed by sub-goals or abstractions, and the case bases were usually organised hierarchically. Marir and Watson (1995) proposed an approach that broke down the plans into small adaptable sub-problems by organising the refurbishment cases as a hierarchical structure composed of cases and sub-cases. Watson and Perera (1998) studied a case representation that decomposed problems of estimating construction costs into sub-problems, which were stored into a set of small case bases rather than a single large case base. This representation provided higher accuracy of retrieval than that of simple flat representations. Smyth, Cunningham and Keane (2001) presented an approach that decomposed cases by abstraction and solved the problem by reusing multiple cases at various levels of abstraction.

2.2.2 CBR Applications and Systems

CBR has been studied in many problem-solving applications. Successful areas include planning, design, explanation and diagnosis, legal advice, health and education (Mantaras and Plaza, 1997). A timetabling problem can be thought of as a special case of scheduling problems. The review in this section will be concentrate on CBR in scheduling and optimisation problems. Related problems such as planning and design problems will also be discussed.

2.2.2.1 CBR in Scheduling and Optimisation Problems

There are relatively few publications specifically on Case-Based Reasoning in scheduling problems. A brief survey of CBR in scheduling was given by MacCarthy and Jou (1996). Three Case-based scheduling systems in scheduling, SMARTplan, CBR-1 and CABINS, were reviewed in the survey and a general framework for applying CBR on a wide range of scheduling environments concerning the dynamic nature of the real-world problems was proposed. The authors claimed that CBR is a very good approach in expert scheduling systems and emphasised potential research areas in dynamic scheduling environments. A review of the current research in Case-based scheduling is given in the following two sub-sections: case-based reactive scheduling and other studies in case-based scheduling.

Case-Based Reactive Scheduling

One of the critical problems in scheduling is its dynamic nature (which is also referred to as ‘Reactive Scheduling’ in some research, see Smith, 1994; Szelke and Kerr, 1994), which describes situations where unexpected events (e.g. new user requirements, real-time changing environments) often occur. In some knowledge based techniques that solve problems from scratch, small changes in scheduling may lead to a re-construction and thus may not provide a high quality new schedule in time.

CBR is inherently a good technique to handle the uncertainty in real-time dynamic scheduling problems (MacCarthy and Ye, 1997; Schmidt, 1998). With the ability to "remember" the most appropriate repairing strategies in previous similar environments, CBR is capable of providing good/sub-optimal complete solutions in a bounded time, which is one of the key requirements in real-world scheduling problems to repair a schedule to satisfy the new requirements quickly. Current case-based scheduling approaches are focused on applications of reactive scheduling.

The CBR-1 project (Bezirgan, 1993) used CBR in a toy car job-shop scheduling problem to provide rules in dynamic environment as early as possible. However, the processing time of the system was not guaranteed because of the repeated retrieval and adaptation. The demand on memory may also lead to an efficiency problem.

Miyashita and Sycara (1994&1995) presented the CABINS system that selected heuristic repair actions in job shop scheduling problems, thus dynamically guiding the search procedure. Constraint satisfaction techniques were used incrementally to carry out these retrieved repair actions on a complete (but sub-optimal) seed schedule.

MacCarthy and Jou (1995) proposed a CBR system to solve scheduling problems involving sequence dependent set up times. They also reviewed different research issues and concluded that using CBR techniques might potentially improve problem solving in scheduling problems that are inherently dynamic, uncertain and complex.

Dorn (1995) proposed a CBR approach integrated with an iterative improvement method in the steel industry. The schedules retrieved by CBR were optimised by an iterative improvement method. Related work and the possible problems in implementing the approach were discussed.

Szelke and Markus (1997) developed a reactive scheduler, CBR/L, for complex dynamic manufacturing shop floor problems. Cases modelling the supervisory behaviours in industry were organised hierarchically to handle complex schedule repairs in fast changing environments and store the long-term valuable real-world knowledge.

Schmidt (1998) proposed a problem-solving CBR framework with the theory of scheduling to interactively make decisions in production planning problems. Well-known scheduling strategies/tactics associated with problems, which were represented by “transformation graphs”, were retrieved to solve target problems. The author claimed that the approach was applicable in reactive scheduling and pointed out work that needed to be done to model the scheduling problems mathematically.

Other Studies in Case-Based Scheduling

Research on case-based scheduling in a variety of scheduling and optimisation problems exist, employing a number of approaches representing cases in different ways and different techniques including Constraint Satisfaction techniques and Graph Heuristic methods.

Koton (1989) proposed the SMARTplan CBR system for a large-scale airlift management problem. The case base was organised into a two-tiered structure with the abstract features and the actual cases to reduce the retrieval cost significantly. Abstraction techniques were used in this system to deal with large problems with many features. However, the information of the later work in developing the system is not available from the literature and has not been reported eloquently.

Hennessy and Hinkle (1992) presented a CBR system, Clavier, for solving the autoclave management and loading problem. The system worked successfully in easily retrieving the autoclave loads that the experts would have chosen. The advantages of knowledge acquisition and representation and the difficulties of validating the CBR system for commercialisation in industrial scheduling were discussed.

Cunningham and Smyth (1997) illustrated two successful CBR approaches in scheduling using skeletons and portions of retrieved schedules. The approaches showed efficient performances in providing good quality solutions in less-complex scheduling problems. However the successful reusing of the retrieved cases would depend on proper adaptation methods and the retrieval time may increase linearly with the size of the case base.

Scott et al. (1998) proposed a CBR approach integrating Constraint Logic Programming in a nurse rostering problem. Cases of generalised high-level patterns of workforce allocation were used. However, the cases were relatively simple and more sophisticated details such as particular nurse preferences could be added into the cases to make the problem instances more realistic.

In this thesis we propose a CBR approach to solve educational course timetabling problem, which are modelled as attribute graphs. The cases in the case base with similar constraints are retrieved for reuse and a graph heuristic method is used for adaptation. When dealing with larger real-world problems, a multiple-retrieval process partitions the attribute graphs of the target case and a set of retrieved cases is reused by a combination technique.

Open Issues in Case-Based Scheduling

- Representation Issues in Case-Based Scheduling

  As one of the most important issues in CBR, representation needs to describe the complex scheduling problems concerning indexing and retrieval. From the work reviewed above on case-based scheduling, we observed that representations in all the existing systems or approaches fall into three types:

  1) Well-known repair strategies/tactics or optimisation heuristics in scheduling problems are either modelled as cases or associated with cases of actual problems. These strategies would be retrieved to guide the incremental repairs of a seed schedule. (See references cited in sub-section "Case-Based Reactive Scheduling" except Dorn, 1995). Application areas of all the reported work of this type are dynamic/reactive scheduling problems.

2) Whole scheduling problems are represented as cases including all the necessary details. Parts/components of a set of retrieved schedules are combined to compose the new schedule. (See Hennessy and Hinkle, 1992; Cunningham and Smyth, 1997; Burke et al. 2001b in the sub-section "Other Studies in Case-Based Scheduling"). The adaptation in these approaches usually needs to be carefully conducted to retain the highly optimised structures in components of the retrieved schedules. As the whole problems with all details are stored in the case base, this approach may suffer from the efficiency problem of retrieval on the case base that might be large.
3) Problems are abstracted in the form of high-level knowledge structures or generalised patterns. (See Koton, 1989; Dorn, 1995; Scott and Simpson, 1998). Appropriate abstract features or generalised schedule patterns, and the level of related and representative details of the cases need to be carefully dealt with to ensure an efficient retrieval that classifies the target cases to reusable cases.

Some of the research on case-based scheduling pointed out that it was impractical to represent the *whole* problem as a case to solve target problems (Miyashita and Sycara 1995; Dorn, 1995; Cunningham and Smyth 1997, Burke et al. 2001b). Real-world scheduling problems are usually very large and complex, so in practice it is rare that a whole scheduling problem can be seen as similar to another previous scheduling problem. Thus all the current research reported here either reuses repair strategies on a seed schedule, abstracted structures or patterns of previous schedules. In the situation where cases represent the whole problem, the sub-schedules of multiple cases corresponding to small matching parts of target case were reused to compose the new schedule, aiming at reusing the knowledge embedded in parts of the retrieved schedules to build a high quality schedule.

- General Methodology of Case-Based Scheduling

Due to the specific requirements and complicated characteristics in scheduling problems, so far no general mechanism for CBR in scheduling can be identified as being applicable in solving a wide range of scheduling problems. Some work has proposed general frameworks of CBR in scheduling (MacCarthy and Jou, 1995&1996). However, a deep study on standardisation of CBR in scheduling needs further investigation to develop an effective and flexible methodology that can be adopted to scheduling problems with specific requirements.

- Case Base Maintenance in Case-Based Scheduling

Due to their complexity, scheduling problems have been seen as ill structured and poor-understood. In case-based scheduling, good case base management is needed to obtain a high quality performance on efficient and effective retrieval. The issues include:

1) Selecting appropriate indices to properly organise the source cases so that retrieval can find reusable cases efficiently;
2) Selecting representative and necessary source cases so that the memory required does not lead to the efficiency problems;
3) Retaining carefully selected new learned cases for a case base without redundancy.

Although many researchers claimed that knowledge acquisition is easier in CBR, some problems do exist because of the complicated and sometimes interrelated constraints in scheduling problems. In this poorly understood area, it is difficult to detect the features that affect the retrieval and reuse of similar source cases. Current methodology on research in case-based scheduling is still far from being a mature research mechanism.

2.2.2.2 Case-Based Planning

Scheduling problems can be classified as a specific type of planning problem, which was one of the most important areas in CBR and has been heavily studied (Veloso, Munoz-Avila and Bergmann, 1996). It has been defined to be “constructing a course of actions to achieve a specified set of goals when starting from an initial situation” (Bergmann et al, 1998). Scheduling “deals with the allocation of scarce resources to tasks over time. It is a decision-making process with the goal of optimising one or more objectives" (Pinedo, 1995). In some research it is also claimed that in practice there is no distinct differences between them (Smith, Frank and Jonsson, 2000).

CBR is a suitable methodology for both planning and scheduling (MacCarthy and Ye, 1997). Case-Based Planning "is the idea of planning as remembering" (Hammond, 1990) that simulates the real-world planning problem solving of experts who modify previous plans according to the new requirements. One of the difficulties in Case-Based Planning is that there are too many features in the problem and thus representations become one of the most important issues (Arnold and Janke, 1994; Bergmann and Wilke, 1995; Marefat, 1997; Tah, Carr and Howes, 1999). Most of the work employs abstraction techniques (Bergmann and Wilke, 1995&1996) where different levels of information of source cases are represented and usually organised hierarchically in the case bases (Arnold and Janke, 1994; Prasad, 1995; Macedo et al., 1996). In other research, multiple cases are retrieved and sub-plans are combined for the target case (Tah, Carr and Howes, 1999). An in-deep study is required on the abstraction that ignores unnecessary details and also keeps enough concrete information so that the system can work effectively.

2.3 Knowledge Discovery

CBR works on previous knowledge/experience that are collected in the system. As mentioned above, representations that models the knowledge into cases is a key issue especially in complicated problems. In knowledge engineering, techniques in knowledge discovery and machine learning have been employed with success in a number of ill-structured domains (which many timetabling problems belong to). Knowledge discovery is the process of studying and investigating a collection of implicitly potential useful dataset to discover information such as rules, regularities, or structures in the problem domain. It was defined as a “non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data” (Fayyad, Piatetsky-Shapiro and Smyth, 1996). A key step in the knowledge discovery process is data mining that may employ a wide range of techniques in AI, machine learning, knowledge acquisition and statistics, etc. Knowledge discovery is usually carried out on databases and the application areas include medicine, finance, law and engineering, etc (Piatetsky-Shapiro, 1991). This thesis also investigates some issues in knowledge discovery to model specific heuristics within timetabling problems (see Burke, MacCarthy, Petrovic and Qu, 2002).

2.4 Chapter Summary

CBR has emerged as a mature research methodology and is extremely successful in a wide range of application domains over the last decade or so. Some CBR research in complex problem domains (e.g. planning, design) has shown promising results. However, its application in scheduling has just attracted the attention of research community and no general methodology has formed. In the case of timetabling, this thesis seeks to address this issue.

Course timetabling may be considered as a proper domain where CBR can be employed to make contributions in problem solving. Existing research in a variety of timetabling problems employing different techniques provides a foundation from both theoretical and application perspectives. It reveals the potential benefits of utilising CBR in course timetabling problems.

Chapter 3 Timetabling Problems

3.1 Timetabling Problems

Timetabling problems arise in many contexts including transportation timetabling (Wren and Rousseau, 1995), sports events timetabling (Schreuder, 1997), employee timetabling (Meisels and Lusternik, 1997) and university timetabling (Barddadym, 1995; Carter and Laporte, 1995&1997). These problems have been the subject of active research over the last 40 years (Wren, 1995; Burke et al, 1997; Schaef, 1999). However, 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 these problems (Burke and Erben, 2000; Smith, 2001; Burke and Petrovic, 2002; Burke and Causmaecker, 2002). Economics and resource utilisation are also important drivers for improved timetable generation.

3.1.1 What is the Timetabling Problem?

Timetabling problems are a specific type of scheduling problems that may be highly constrained and difficult to solve. It was defined by Wren (1995) as:

> *"the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives."*

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, while minimising the violations of a set of constraints. Different timetabling problems have different constraints. Constraints associated with each individual problem are usually classified into two particular types: *hard constraints* and *soft constraints*. Hard constraints should under no circumstances be violated. A common hard constraint is 'no person is assigned to two or more courses simultaneously'. Other constraints known as *soft constraints* are desirable but it is not essential to satisfy them. Indeed, it would usually be impossible to satisfy all of them in a given problem. Examples are when two events with common persons should or should not be consecutive, or when one event should be before another.

3.1.2 Course Timetabling Problems

This thesis addresses educational course timetabling problems. Course timetabling problem was defined by Carter and Laporte (1997) as:

> *"a multi-dimensional assignment problem in which students, teachers (or faculty members) are assigned to courses, course section or classes; events (individual meetings between students and teachers) are assigned to classrooms and times"*

In a course timetabling problem, a number of courses are assigned into classrooms and a limited number of timeslots within a week. Students and teachers are assigned to courses. Of course, course timetabling also comes along with a set of constraints that can also been classified as hard and soft constraints. Individual institutions usually have a variety of specific constraints and most of research in course timetabling investigated particular real world problems in their own institutions. The course timetabling is also referred as the class/lecture/school timetabling problem. In literature, research on course timetabling are grouped as class-teacher timetabling, student scheduling, teacher assignment and classroom assignment. This thesis deals with mainly the student scheduling concerning room capacities.

3.2 AI Techniques for Educational Timetabling Problems

3.2.1 Traditional Approaches in Educational Timetabling

Various methods have been investigated to solve educational timetabling problems (Carter and Laporte, 1995&1997). In the early days of educational timetabling research, graph theoretic methods represented the state of the art (Brelaz, 1979; Werra, 1985). Techniques such as graph colouring were widely used to solve the problems. For example, Burke, Elliman and Weare (1994) developed a heuristic based on graph colouring approach that split the exams into groups and schedule them together. The number of timeslots correspond the number of colours needed. Sequential assignment approach was also investigated in some recent work, where the events were ordered by heuristics to be scheduled one by one and backtracking was usually carried out to obtain a feasible solution (Carter and Laporte, 1996). Other research techniques that were also widely employed in the early days of timetabling research included integer linear programming (ILP), where constraints were modelled into formulas in which 0-1 variables represented the assignments (Tripathy, 1984; Carter, 1989). However this approach tended to be more impractical for real-world large timetabling problems.

3.2.2 Meta-Heuristic Methods in Educational Timetabling

More recently, meta-heuristic techniques have been very successful in a wide range of timetabling problems. A series of international conferences on the Practice and Theory of Automated Timetabling (PATAT) provides a forum for a wide variety of research work on timetabling and many relevant publications can be found in the proceedings of PATAT (Burke and Ross, 1995; Burke and Carter, 1997; Burke and Erben, 2001; Burke and Causmaecker, 2002). It is impossible to give an exhaustive review on all of the timetabling research. This thesis investigates course timetabling, and this section will present the work in meta-heuristics mainly for educational timetabling problems.

3.2.2.1 Tabu Search

In course timetabling, Tabu Search (TS) (Glover and Laguna, 1993) was mainly investigated on real-world course and general problems in different institutions with various specific requirements. The results reported were very well with variant of TS with properly selected parameters such as the tabu list, initial solutions and objective functions, etc (Hertz, 1991; Costa, 1994; Schaerf, 1996).

Nonobe and Ibaraki (1998) developed a Tabu-Based general problem solver for a range of constraint satisfaction problems including a high school timetabling problem. The results shown that this approach was competitive compare with other specially developed approaches for the respective problem domains. Alvarez-Valdes, Crespo and Tamarit (2002) developed a system with friendly user interface based on a Tabu Search with a set of heuristics. The package was tested and satisfactory results were obtained. Approaches that integrated TS with other techniques in timetabling were also investigated. For example, White and Zhang (1997) studied an approach that used the output of constraint logic technique as the starting solutions for TS on general timetabling problems. The results obtained were better than using either method alone.

Research on examination timetabling problems was also carried out (Cangalovic, et al., 1998; White and Xie, 2000; Gaspero and Schaerf, 2000), which studied different aspects (length of tabu lists, representations and initialisation methods of solutions) of utilising TS on timetabling.

3.2.2.2 Simulated Annealing

Simulated Annealing (SA) (Kirkpatrick, Gellat and Vecci, 1983; Reeves, 1996) was also a widely studied method on course/school timetabling problems. Abramson (1991) studied a SA that were implemented on a multiprocessor and presented further research issues arise from this approach. Some work concluded that the implementation of SA is highly dependent on various settings and parameters (e.g. solution space, cooling schedule, neighbourhood generation, cost function) on both examination (Bullnheimer, 1997; Thompson and Dowsland, 1998) and course/school timetabling problems (Elmohamed, Coddington and Fox, 1997; Melicio, Caldeira and Rosa, 1998; Abramson, Dang and Krisnamoorthy, 1999) thus careful selection on parameters and settings on this algorithm are needed.

3.2.2.3 Evolutionary Algorithms

Genetic Algorithms (GAs) () and Evolutionary Algorithms (EAs) () have been widely studied by researchers in timetabling, concerning different aspects of timetabling problems (Corne, Ross and Fang, 1994). In course timetabling, Abramson and Abela (1992) investigated a parallel GA that greatly reduced the execution time to solve the problem. Rich (1995) studied a GA with greedy algorithm that used domain knowledge for room and timeslot scheduling. Deris, et al. (1999) proposed an approach that embedded constraint-based techniques with GAs, where potential solutions for a real course timetabling problem were generated by GAs and then repaired and improved by using constraint-based techniques. By using the constraint-based reasoning, the search space for GAs can be significantly reduced and thus the convergence was much faster to produce nearly optimal solutions. Erben (2000) investigated a grouping GA in which the representation was based on the grouping character of the graph colouring problem. The author tested the GA on real-world examination timetabling problem and suggested that the fitness function should convey as much information about the quality of the solution as possible. Approaches that hybridise GAs with local search techniques during the evolution, which are known as Memetic Algorithms (Radcliffe and Surrey, 1994), have been investigated and the results obtained were promising on examination timetabling (Burke, Newall and Weare, 1995) and course timetabling (Rankin, 1995; Paechter, Rankin and Cumming, 1997).

Initialisation is also one of the important issues in GAs and EAs. Corne and Ross (1995) studied an approach using peckish initialisation and the results were better than both the greedy and random initialisation. Burke and Newall (1998) investigated different heuristic initialisation strategies and the results were very good. The authors suggested that good initial solutions are generated using heuristics with the condition of having a sufficient degree of diversity.

Of particular interests is that recently the encoding in GA/EA has attracted some research on timetabling problems. Paechter, Cumming and Luchian (1994) investigate an EA on general timetabling where chromosomes encode the suggestion lists for events to be scheduled to build the solutions. Ross, Hart and Corne (1997) carried out an extensive study on a GA with a direct encoding. Based on the observations that the direct encoding tended to lead the failure of solving parts of the problems, the authors suggested GA to be used for searching good heuristics rather than specific solutions in specific problems. Terashima-Marin, Ross and Valenzuela-Rendon (1999) also investigated an EA with un-direct representation in exam timetabling to evolve among the Constraint Satisfaction strategies, heuristics and conditions of changing from one strategy to another. The name "hyper-heuristic" is termed to name the heuristic that choose heuristics in later research using this method (see the sub-section "Hyper-heuristic methods").

3.2.2.4 Comparisons of Different Approaches

Comparisons concerning a range of issues in heuristic and meta-heuristic methods for timetabling have been also carried out. Ross and Corne (1995) compared GA, SA and stochastic hillclimbing with certain representation on a collection of real timetabling problems, concerning the solution quality and number of distinct useful solutions. The conclusions were that the stochastic algorithms perform generally well with respect of the solution quality. However different conclusions may be obtained if different representations and operators were employed. Dowsland (1997) investigated SA and TS on various timetabling problems and suggested that there is plenty of potential work to make it possible to develop general algorithms based on SA and TS, which work generally well on families of problems. Colorni, Dorigo and Maniezzo (1998) compared SA, TS, GA and GA with local search (known as Memetic Algorithm) on a high school timetabling problem. The authors claimed that TS obtained the best result and GA with local search was capable of giving a set of good quality solutions thus was much flexible to users who may have a variety of objectives.

Different algorithms within specific circumstances may perform differently on particular timetabling problems. In general, GA/EA is able of giving a number of useful distinct solutions thus in real-world problem solving may be more flexible on providing the users solutions that satisfy different aspects of requirements. There is much potential work on studying the real mechanism behind the reason why particular algorithms outperform others for a particular family of timetabling problems. The discoveries and knowledge/experiences on particular heuristics/meta-heuristics in specific circumstances on specific timetabling problem may lead to more effective knowledge-based techniques on solving a wide range of timetabling problems, which is the subject of this thesis.

3.2.3 Constraint Logic Techniques

Timetabling problems is a type of assignment problems with large amount of complex constraints thus usually can be easily modelled as Constraint Satisfaction Problems (CSP) (Brailsford, Potts and Smith, 1999). Constraint Logic programming (CLP) are suitable methods and have also been widely employed in course timetabling problems.

Most of research has been carried out to develop techniques that can be easily adapted into different problems. Two of the declarative languages: CHIP and ECL^{i}PS^{e} developed for modelling and solving CSP problems were widely used for course timetabling problems (Kambi and Gilber, 1996; Stamatopoulos, Viglas and Karaboyas, 1998; Goltz, 2000; Abdennadher and Marte, 2000). Other declarative languages developed for different specific timetabling problems included WPROLOG (Kang and White, 1992), COASTOOL (Yoshikawa, 1994), Oz (Henz and Wurtz, 1995) and EaCL (Tsang, Mills and Williams, 1999). Zervoudakis and Stamatopoulos (2000) also developed a constraint programming object-oriented C++ library that can model the possible common constraints within every problem, thus can be easily extended to instantiate different timetabling problems. Deris, Omatu and Ohta (2000) proposed an object-oriented approach in which course timetabling problems were formulated as constraint satisfaction model with forward checking and constraint propagation procedures. The author claimed that the approach would be potentially applicable in various environments by specifying different parameters.

Other work concerned different aspects in CLP for timetabling problems. Fahrion and Dollansky (1992) developed a Prolog rule system with simple heuristic priority scheme for a faculty (teacher) assignment problem. Boizumault, Delon and Peridy (1996) proposed an efficient CLP approach with finite domains for a real-world examination timetabling problem and presented potential future work. Banks, Beek and Meisles (1998) employed an approach in which constraints were iteratively added to the CSP representation before backtracking for high school course timetabling. Blanco and Khatib (1998) split a real course timetabling problem into two phases, each was modelled as a CSP and solved using optimisation techniques. Weekly lectures were grouped into timeslots and thus the domains of variables were greatly reduced. Also a variety of research on CLP for timetabling problems (Cheng et al, 1995; Gueret, et al. 1995; Lajos, 1995; David, 1997; Zervoudakis and Stamatopoulos, 2000) can be found in proceedings of PATAT conferences (Burke and Ross, 1995; Burke and Carter, 1997; Burke and Erben, 2000).

In timetabling research usually constraint-based techniques are used to model the problem into a CSP. The assignment of variables significantly affect the efficiency thus different special-purposed search heuristics were used to solve specific constraints. Most of CLP approaches produced feasible rather than optimal solutions, which were then improved by employing different techniques. For example, using CLP to produce the starting points for TS not only produce better quality results but also saved a lot amount of computation time (White and Zhang, 1997). Yoshikawa et al. (1994) proposed a constraint relaxation problem (the same as constraint satisfaction problem except constraints are associated with penalties) solver to produce good initial assignment, which was then improved using hill-climbing for a real course timetabling problem.

3.2.4 Knowledge-Based Techniques

The overall objective of using knowledge-based techniques for timetabling is to model the human knowledge for timetabling. Due to the complexity of constraints and implicit knowledge embedded in problem solving of timetabling, representations come to be one of the critical issues in using knowledge-based techniques for the problem.

A mixed approach was presented employing knowledge-based techniques and constraint networks on real-world employee timetabling (Meisels, Gudes and Solotoresky, 1995). The problems were explicitly represented on some constraints in the constraint-based processing and rules were incorporated into the scheduling process. The preliminary results shown that the explicit representation and the ordering heuristic are efficient for solving employ timetabling problems.

Gunadhi, Anand and Yong (1996) designed a timetable scheduler that used the knowledge modelled as rules, incorporated with heuristics, within course timetabling process to schedule data that was stored in separate bases. The results obtained were promising for real world timetabling problems and the authors claimed that the scheduler was flexible and general and thus was applicable to other university timetabling with the use of an object-oriented methodology.

Kong and Kwok (1999) proposed a conceptual model within a knowledge-based approach. The knowledge was modelled into heuristics that applied the rules to guide the scheduling process for course timetabling problems.

Foulds and Johnson (2000) developed a database decision support system for a real world course timetabling problem and emphasised that human judgement was crucial in timetabling processing. The system was designed to assist experienced timetabling officers in evolving a timetable from one year to the next by necessary modifications rather than automatically creating timetables from scratch.

All the existing knowledge-based techniques on timetabling use expert system, which models the knowledge of timetabling as rules, to generate course timetables. One possible problem with this is that usually the knowledge within the scheduling is implicit thus difficult to be modelled. This may be resolved by either the careful design of specific problems, or by employing techniques that can use the knowledge and avoid large amounts of work in modelling it. This thesis investigates an approach using case-based reasoning, which could be one of the solutions for this problem.

3.2.5 Hyper-Heuristic Methods

As mentioned before, hyper-heuristics are "heuristics that choose heuristics" (Cowling, Kendall and Soubeiga, 2000&2001, Cowling, Kendall and Han, 2002). The main difference between hyper-heuristics and the widely used meta-heuristics in timetabling is that hyper-heuristics is a method of using heuristics to select from a variety of different heuristics that may include meta-heuristics. So hyper-heuristics are potentially more general-purpose methods.

Some research in scheduling has investigated this approach although it may not have used the term "hyper-heuristics". Some approaches used Genetic Algorithms (GAs) to select from a set of heuristics encoded in the search space and quite good results were obtained. An approach was presented in (Fang, Ross and Corne, 1994) on open shop scheduling problems using GAs to search a space of abstractions of solutions to "evolve the heuristic choice". In a real-world scheduling problem for catching and transportation of large amount of chickens, GAs are used to construct a schedule builder that chooses the optimal combinations of heuristics (Hart, Ross and

Nelson, 1998). Another approach in (Terashima-Marin, Rossa and Valenzuela-Rendon, 1999) used a GA to select the heuristic to order the exam in a sequential approach for exam timetabling problems. A hybrid GA investigated on vehicle routing problems has also obtained promising results (Shaw, 1998; Berger, Sassi, and Salois, 1999).

Some research on hyper-heuristics has also been carried out on solving a variety of scheduling problems. Guided local search was used to select from a set of heuristics and also different parameters in these heuristics in the traveling salesman problems (Voudouris and Tsang, 1999). Cowling, Kendall and Soubeiga (2000&2001) used a hyper-heuristic approach to select from a set of lower level heuristics according to the characteristics of the current search space in a sales summit scheduling problem.

3.2.6 Decomposition in Timetabling

Real-world timetabling problems are usually very large and complex. To address this problem, decomposition and partition techniques have been studied with some success. The basic idea is to decompose the problem into a set of sub-problems that are small enough to be solved by using simple approaches. Then these (hopefully high quality) sub-solutions will be combined for the original problems. Robert and Hertz (1995) presented an algorithm decomposing the course timetabling problems into a series of easier assignment type sub-problems. An approach of decomposing the timetabling data to produce shorter flexible length timetables was studied by Weare (1995). Burke and Newall (1999) employed a multi-stage algorithm in an evolutionary approach to solve examination timetabling problems that were decomposed using graph colouring heuristics, and the sub-problems were solved by using the memetic approach presented in (Burke, Newall and Weare, 1995). Cangalovic, et al. (1998) used an approach that modelled a real exam timetabling problem into specially structured weighted graph and decomposed it into maximal cliques. Special purposed heuristic was used to generate a feasible good solution, which was then improved using TS. Carter (2000) presented an algorithm in course timetabling which decomposed the problem into relatively independent clusters that can be solved more easily using relatively simple approaches.

3.3 Chapter Summary

A large number of promising methodologies and algorithms have been investigated for university timetabling problems. Both problem specific and global techniques have been studied on a wide range of problems concerning variety of aspects.

Traditional techniques such as graph theoretical and integer programming can easily encode relatively simple timetabling problems and perform generally well. However, they are usually incapable of dealing with problems with large size and complex constraints. Global techniques in AI (e.g. GAs, TS, SA) have been reported to obtain generally good results on various problems. They are capable of performing well on a wide range of problems of different sizes but careful refinement concerning certain issues is usually needed for them to be fitted into different environments. Examples of these issues include initialisations and different parameters within different algorithms.

Research has shown that hybridised methods often perform better than individual approaches as they are benefited from the advantages of both techniques with careful design. For example, CLP can be easily applied on timetabling problems and solve problems quickly. It might be good initialisation techniques for GAs, TS or SA, whose starting points in the search space sometimes affect the quality of the evolved solutions significantly.

Timetabling as an example of a scheduling problem has become an application area with rich knowledge and experience. Comparisons have been carried out between different techniques and experiences on the problem solving have been accumulated. These provide the premises and foundation for utilising knowledge-based techniques like CBR in this area. All of the current knowledge-based systems on timetabling used the rule base to incorporate knowledge of problem solving. Due to the difficulties in modelling the knowledge that is implicit with complex constraints, most of these systems aim at assisting rather than reusing the deep knowledge within the timetabling process. This thesis investigates the benefits that CBR may offer on course timetabling problems.

Chapter 4 Structured Cases in CBR for Course Timetabling Problems

The work presented in this chapter was published in journal of Knowledge-Based Systems (Burke, MacCarthy, Petrovic, and Qu, 2000) as it was selected as one of the best six technical papers in the ES’99conference. The aim of this work is to present the possibilities and advantages of using attribute graphs to structurally model the course timetabling problems as cases in a CBR system. The attribute graphs are capable of describing the relations (constraints) between the events in a timetabling problem more concisely and explicitly, thus deeper knowledge such as the correspondence between structures of events and characteristics of the solutions can be expressed in cases. The retrieval aims at adaptability and reusability of the solutions of the retrieved cases, which are easy to be reused for the target case that has similar constraints.

4.1 Attribute Graphs for Course Timetabling Problems

In attribute graphs that model the course timetabling problems, nodes indicate courses and edges show the relation between any pair of courses. Nodes and edges have attributes that represent the problem more precisely. Each attribute corresponds to a label assigned to nodes and edges. Table 4-1 and Table 4-2 present the labels and attributes of nodes and edges that are used in our problems.

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 4-1 Node attributes of course timetabling problem

Label	**Attribute**	**Values(s)**	**Notes**
4	Before/after	1 or 0 (direction)	One before/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	Conflict with each other

Table 4-2 Edge attributes of course timetabling problem

A simple example is shown in Figure 4-1 to illustrate a course timetabling problem represented by an attribute graph. Nodes represent courses. Solid edges indicate hard constraints (labelled 7) which means that the adjacent courses cannot be held simultaneously. Dotted lines indicate soft constraints labelled 4, 5 or 6. The labels on the edges and inside the nodes correspond to the attributes shown in Table 4-1 and Table 4-2. For example, Maths, Physics and Chemistry are labelled with a 1 (to indicate that they are multiple courses) and with values 2, 3 and 2 that denote that they should be held 2, 3 and 2 times a week respectively. Other courses are labelled 0 (ordinary courses), which denote that they should be held just once a week. SpanishA should not be consecutive to Physics (because the edge between them is labelled by a 6) and Chemistry should be consecutive to SpanishB (labelled by a 5). The directed line between SpanishA and SpanishB has the label 4 (with value 1) which denotes that SpanishA should be held before SpanishB.

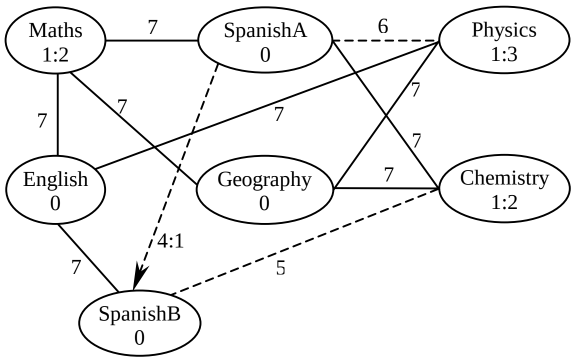


Figure 4-1 Attribute graph of a course timetabling problem

Using this approach, the course timetabling problems can be represented structurally. It enables us to describe the relations between events in the problem that is not possible by using feature-value pairs. Also the different cases of the problems can have different structures, unlike in traditional case representation using the list of feature-values pairs where all the cases have the same form of feature slots.

4.2 Implementation of the CBR System

4.2.1 The Graph Isomorphism Problem

Using attribute graphs to represent cases has many advantages. However, the matching problem between the structured cases is equivalent to that of the graph isomorphism or sub-graph isomorphism problem that is known to be NP-Complete (Garey and Johnson, 1979). A graph, G, is isomorphic to graph G' if there exists a one to one correspondence between nodes and edges of the two graphs. A graph G is sub-graph isomorphic to graph G' if G is isomorphic to a sub-graph of G'. Some methods have been attempted to solve this problem in CBR by detecting cliques of the graph (Borner, 1993). The system being proposed here is based on Messmer's algorithm (Messmer, 1995) where graphs are organised in a decision tree.

The attribute graph is represented by its adjacency matrix $M = m_{i,j}$, where $m_{i,j} \in L_e$ indicates the attribute of the edge between node i and node j and $m_{i,i} \in L_n$ indicates the attribute of node i. L_e and L_n are the sets of labels defined in Table 4-2 and Table 4-1. There are n! different adjacency matrices for an n-node attribute graph when the nodes are in different permutations. The idea of Messmer's algorithm is to pre-store all the adjacency matrices of some known graphs with their permutation matrices $P = p_{i,j}$ to the corresponding nodes in a decision tree. If graph G is isomorphic to graph G’, then if $p_{i,j} = 1$, node i in graph G corresponds to node j in graph G’. If a target graph can be classified to a node in the decision tree at level k, then the permutation matrix(matrices) stored in this node indicate the matching between the k nodes of the target graph and that of previously stored graph(s). If the time spent on building up the decision tree is ignored, this algorithm guarantees that all the graph isomorphism(s) or sub-graph isomorphism(s) stored in the tree can be found in polynomial time (quadratic to the number of nodes of the target graph).

For example, in Figure 4-2, attribute graph G represents a 3-course timetabling problem. Maths is labelled 1 with value 2 (multiple course, held twice a week). Physics and Spanish are labelled 0 (ordinary course, held once a week). Physics should be held before Maths. Spanish should not be scheduled simultaneously with Physics as Maths. There are 6 adjacency matrices M0~M5 representing graph G, X denotes that there is no edge between two nodes and the labels in the matrices are described in Table 4-1 and Table 4-2.

$$
\begin{array}{c|c|c|c|}
 & a & b & c \\ \hline
a & 0 & 7 & 7 \\ \hline
b & 7 & 0 & 4 \\ \hline
c & 7 & x & 1 \\ \hline
\end{array}
\quad
\begin{array}{c|c|c|c|}
 & a & c & b \\ \hline
a & 0 & 7 & 7 \\ \hline
c & 7 & 1 & X \\ \hline
b & 7 & 4 & 0 \\ \hline
\end{array}
\quad
\begin{array}{c|c|c|c|}
 & b & a & c \\ \hline
b & 0 & 7 & 4 \\ \hline
a & 7 & 0 & 7 \\ \hline
c & x & 7 & 1 \\ \hline
\end{array}
\quad
\begin{array}{c|c|c|c|}
 & b & c & a \\ \hline
b & 0 & 4 & 7 \\ \hline
c & x & 1 & 7 \\ \hline
a & 7 & 7 & 0 \\ \hline
\end{array}
\quad
\begin{array}{c|c|c|c|}
 & c & a & b \\ \hline
c & 1 & 7 & x \\ \hline
a & 7 & 0 & 7 \\ \hline
b & 4 & 7 & 0 \\ \hline
\end{array}
\quad
\begin{array}{c|c|c|c|}
 & c & b & a \\ \hline
c & 1 & x & 7 \\ \hline
b & 4 & 0 & 7 \\ \hline
a & 7 & 7 & 0 \\ \hline
\end{array}
$$

M0 M1 M2 M3 M4 M5

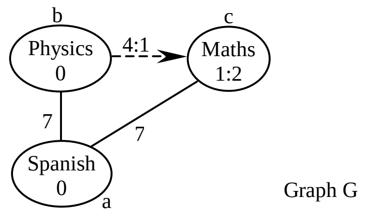


Figure 4-2 Matrices of attribute graph G of a course timetabling problem

These matrices are used to build the decision tree (see Figure 4-3). If a matrix M can be seen as consisting of an array of so-called row-column elements $a_i = (m_{1i}, m_{2i}, \dots m_{ii}, m_{i(i-1)}, \dots, m_{i1})$, then a 3 X 3 matrix consists of 3 elements: $a_1 = a_{11}$, $a_2 = a_{21}a_{22}a_{12}$ and $a_3 = a_{31}a_{32}a_{33}a_{23}a_{13}$. The first element of each of the matrices M0~M5 can be 1 or 0, and therefore there are two branches from the root node with label 0 and 1 on the first level. The second level under branch 1 can be 707 and 40x in M4 and M5, thus two branches below branch 1 are built. Then the following levels of the decision tree can be built by the same process, each branch on level *i* leads to a successor node that is associated with a specific value for the *i*th element of M0~M5. Each permutation matrix is stored in the corresponding node in the decision. Then all the other known attribute graphs can be added into the tree in the same way.

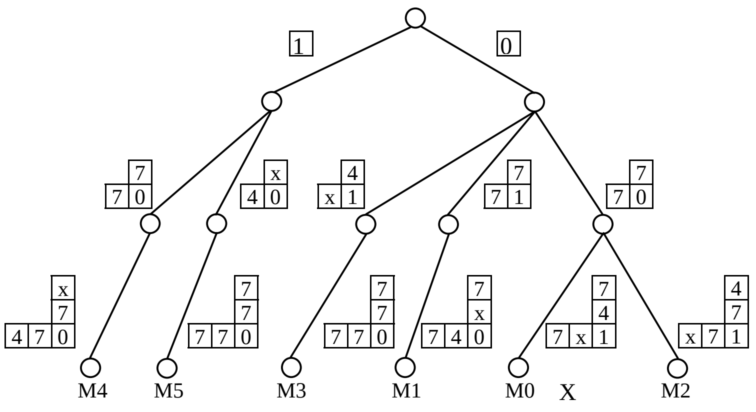


Figure 4-3 A decision tree storing matrices of attribute graph G

Let us suppose that we are presented with a target problem represented by matrix M for attribute graph G' (see Figure 4-4). The matrix M is inserted into the tree and can be classified to node X according to the values of each branch. The permutation stored to node X gives the isomorphism that tells us that Maths(c), Physics(b) and Spanish(a) in attribute graph G correspond to English(b), Chemistry(a) and Maths(c) in attribute graph G' respectively.

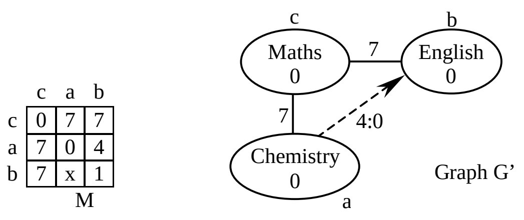


Figure 4-4 Matrices of attribute graph G’ for a target course timetabling problem

4.2.2 Retrieving Structurally Similar Cases

Some course timetabling problems are generated randomly and their attribute graphs are used to build up a decision tree in the proposed system. The solutions to these problems are obtained by using a heuristic graph colouring method described in (Burke, Newall and Weare, 1998).

Penalties are associated (see Appendix A) with pairs of labels described in Table 4-1 and Table 4-2 and are used in the retrieval process. A threshold is also set to judge whether two labels are similar or not. When the system tries to match each pair of events in the target problem with source cases, the events can be seen as similar if the penalty between their labels is below the threshold. They are identified as similar and returned to be matched with each other. The penalties are set so that the constraints of the target problem are never released. For example, soft constraints in source cases cannot be mapped to hard constraints in target cases so the solutions of the retrieved source cases will guaranteed to be feasible for the target problem.

If an event in the target problem has the same label and the same value as the source case, then they match with no penalty. Two events that are labelled the same are further analysed to see if they have the same values. Penalties are given for the differences between the values and are taken into account in the similarity measure.

Every label is also given a weight using domain knowledge for the similarity measure. The similarity measure is thus given by formula (1):

$$S = 1 - \sum_{i,j=0}^{n} p_{ij} \times w_i / P \qquad (1)$$

where symbols are defined as following:

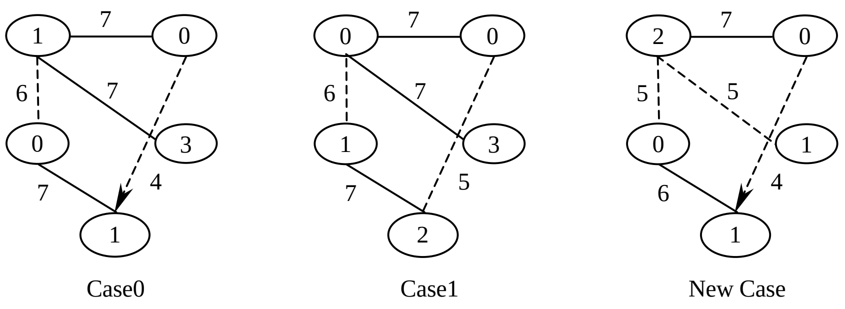
n: the total number of the labels

$p_{i,j}$: the penalty between label i of node or edge in the target problem and label j of node or edge of source cases

w_i: the weight of label i in the target problem

P: the sum of the penalty for every pair of labels times the weight of every label.

Figure 4-5 presents an example of how the similarities are calculated between two pairs of cases, New Case and Case0, and New Case and Case1. By employing the similarity measure shown in formula (1) with penalties and weights presented in Appendix A, similarities are calculated as 0.86 and 0.78 for New Case between Case0 and Case1, indicating New Case is more similar with Case0 than with Case1.



Case0 labels	Case1 labels	New case					
		labels	w_i	Case0		Case1	
				p_{ij}	$p_{ij}*w_i$	p_{ij}	$p_{ij}*w_i$
7	7	7	0.9	0	0	0	0
6	6	4	0.6	0.7	0.42	0.7	0.42
7	7	5	0.6	0.8	0.48	0.8	0.48
4	5	4	0.4	0	0	0.7	0.28
0	1	0	0.5	0	0	0.4	0.2
3	3	1	0.6	0.5	0.3	0.5	0.3
7	7	6	0.4	0.7	0.28	0.7	0.28
1	2	1	0.6	0	0	0.7	0.42
				P = 10.6	$\sum$ = 1.48		$\sum$ = 2.38
				S = 1- $\sum p_{ij}*w_i$ / P	S = 0.86		S = 0.78

Figure 4-5 An example of similarity measure between cases

Using the penalties assigned to each pair of labels in the course timetabling problems, the retrieval is targeted at matching between every pair of events, not just a single judgement between the whole cases. The system can retrieve the case(s) suitable for adaptation for the target problem from the case base.

When a target problem is entered in the system, it is classified to a node in the decision tree and the system retrieves all the cases stored in and below that node as candidates. As the tree stores cases hierarchically, all the cases that have more events and/or more relations are stored below those having less events and/or relations. It is observed that solutions of more constrained cases can be adapted easily for less constrained problems. Thus all the cases in and below the node are retrieved.

Using the penalties for every pair of the labels of nodes and edges, the system calculates the similarity between the target problem and the candidate cases in and below the node. The most similar case(s) are selected for adaptation.

4.2.3 Reuse and Adaptation of the Solutions

After the system finds the most similar case(s), the solutions or part of the solutions of the retrieved case(s) can be reused. The system substitutes the events in the solution(s) of the retrieved case(s) with the matching events in the target problem according to the isomorphism(s) found. After the substitution, a partial solution for the target problem can be obtained although there may be some violations of constraints. If there is no violation of hard constraint in the retrieved solutions, there is also no violation of hard constraint in the solutions after substitution.

The graph heuristic method which tries to minimise the violations of constraints is used in the adaptation process. Events that violate the constraints are collected from the partial solution, and all the unscheduled events are ordered first by their degrees (number of conflicts of an event with other events) decreasingly and then are assigned one by one to the first available timeslot. If some events cannot be assigned to a timeslot without violation of constraints, they will be kept until all the other events have been scheduled. Then they are scheduled to the timeslots that lead to the fewest number of violations of constraints.

4.3 A Simple Illustrative Example

Let us suppose that the problem shown in Figure 4-1 is the target problem. All the cases and their isomorphism are retrieved from the node that the target problem is classified to in the case base. Not only the case(s) that are graph isomorphic to the target problem can be adapted, but also the case(s) which the target problem is sub-graph isomorphic can be adapted, although they may not be “good” solutions for the target problem. Two cases whose similarities pass a given threshold (a score set) are considered to be the most similar to the target problem and are retrieved from the case base. The structures of these two cases are shown in Figure 4-6. It is possible to find more than one isomorphism between two graphs. Two isomorphism were found for each of the retrieved cases in this example.

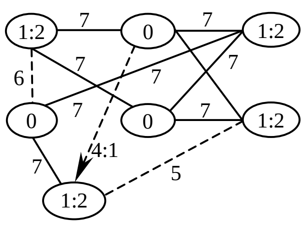
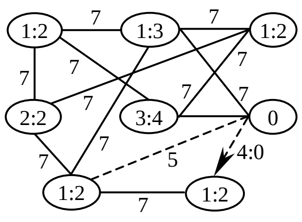



Figure 4-6 Two retrieved cases from case base

After substituting the events of the retrieved cases shown in Figure 4-4 by matching events indicated by the isomorphism, four solutions can be obtained for the target problem (see Table 4-3).

	Timeslot1	**Timeslot2**	**Timeslot3**	**Timeslot4**	**Timeslot5**
Solution1	Physics, Maths, Chemistry	English, Geography	SpanishA	SpanishB, Physics, Maths	Chemistry, Maths
Solution2	Maths, Physics, Chemistry	English, Geography	SpanishA	SpanishB, Maths, Physics	SpanishB, Physics
Solution3	Physics, Maths, Chemistry	English, Geography	SpanishA	SpanishB, Physics, Maths	Chemistry, Maths
Solution4	Maths, Physics, Chemistry	English, Geography	SpanishA	SpanishB, Maths, Physics	Chemistry, Physics

Table 4-3 Solutions after substitution by using isomorphism

It can be seen that there are 3 violations of soft constraints in solution 1: SpanishA is consecutive to Physics, Physics is held only 2 times and Maths is scheduled one more time. Using the graph heuristic method takes 2 adaptation steps: It deletes Maths from timeslot1 and adds another Physics to timeslot 5. It can also be seen that there are 1, 3 and 1 violation(s) of soft constraints in solution 2, 3 and 4 respectively. Using the graph heuristic method takes 1 and 2 adaptation step(s) respectively for solution 2 and 3. There is no adaptation for solution 4. After adaptation, there is only one violation of a soft constraint in each solution.

The simple example has demonstrated that only a few adaptations are needed to get solutions for the target problem on the basis of the solutions of the retrieved similar cases. Cases can explore deeper knowledge in course timetabling problems by the structural representation. Retrieval that targets the adaptability of every pair of events between the target problem and the retrieved case(s) finds the most adaptable cases for the target problem, thus a corresponding relation between the events and adaptation requirements is built up. Employing the adaptation requirements in the definition of the similarity between every event pair gives a more elaborate description for the similarity measure. Thus the knowledge and experiences previously stored in the retrieved cases' solutions can be exploited for re-use for target similar problems. It is noted that the CBR can re-use the sub-solutions of previously solved problems within the case-base, a manner similar to that of experts in timetabling.

4.4 Chapter Summary

In this chapter, a CBR approach is proposed in which attribute graphs are used to represent cases for course timetabling problems. To our knowledge, the CBR approach proposed in this chapter is new in solving the timetabling problems. *Retrieval targets every pair of nodes and edges between the cases so that the retrieved case(s) are the most adaptable for the target problem***.** The retrieved cases' solutions store optimised or sub-optimised schedules for the previously solved problems. These schedules can be exploited and re-used for the new similar cases, after only limited adaptations for solutions that are then applicable for the target problem. The graph data structure gives a detailed description of the timetabling problem. The relations between any events can be described clearly, and therefore the application of *this method to timetabling problems is likely to find the similar cases that are adaptable for the target problem*. In the next chapter, this method is improved and systematically analysed to solve a wider range of course timetabling problems.

Chapter 5 Improved CBR Attribute Graph Approach

In the CBR system presented in the last chapter, it is assumed that some pre-compiled cases exist so that the target problems can find isomorphic or sub-graph isomorphic source cases. The overriding motivation is that previous timetables with similar constraints will provide a sensible starting point for solving a target problem. However, attribute graphs of source cases that have common or partially similar (sub-)structures could also be reusable for the target case. The work, which is published in the Fourth International Conference on Case-Based Reasoning (ICCBR'01) and presented in this chapter (Burke, MacCarthy, Petrovic, and Qu, 2001a) improves the previous CBR system to deal with a wider range of problems than those dealt with in the last chapter.

5.1 Improved Retrieval of Structurally Similar Cases

5.1.1 Partially Similar Cases with Differences

In the improved approach, not only the source cases that are graph or sub-graph isomorphic to the target case are retrieved, but also (partial) matches with some differences are examined. Source cases do not need to contain all the corresponding similar edges to be reused for the target cases. For example in Figure 5-1, graph B is neither graph nor sub-graph isomorphic to graph A shown in Figure 5-2. However, graph B can be graph isomorphic to graph A if some vertices and edges are inserted. When dealing with difficult real world timetabling problems our approach has to be more flexible than just considering cases in the case base that are graph isomorphic to the target case. Note that graph B is partially similar to graph A. In graph A, not all of its vertices and edges can match those of graph C in Figure 5-1 (*Physics*, *ComputerA* and *ComputerB* cannot find a matching course in graph C). Also, not all of the vertices and edges in graph C can find a match with those in graph A (the course labelled with 1:2 with adjacent edges illustrated by light lines does not have a matching course with matching edges in graph A). These two cases have common parts that are partially similar with each other in either vertices or edges.

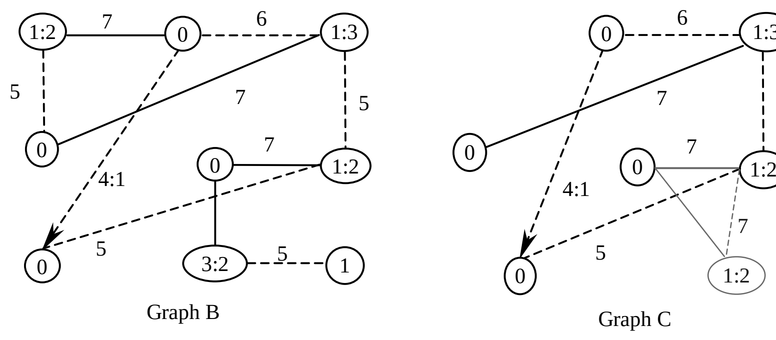


Figure 5-1 Cases partially similar with some differences with case in Figure 5-2

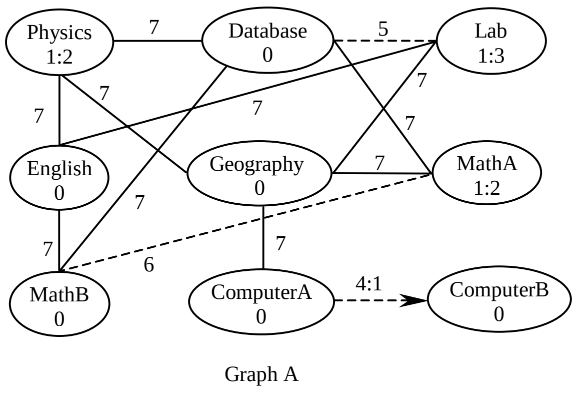


Figure 5-2 A course timetabling problem represented by an attribute graph

In the approach developed here, target cases like graphs B and C can all be seen as partially similar (but clearly have some significant differences) to graph A. The timetable associated with graph A could be reusable for the cases of graph B and C. This approach retrieves a large number of useful cases thus allowing an investigation of a much wider range of timetabling problems.

5.1.2 Similarity Measure

The similarity measure takes into account the costs assigned to the substitutions, deletions and insertions of vertices and edges labelled with particular attributes from or into the target case. Deleting vertices and edges with different attributes from the cases in the case base are assigned lower costs than those of inserting vertices and edges into them. Also inserting and deleting the edges of hard constraints is assigned a higher cost than for the soft constraints. Costs are assigned so that the operations of deletion, insertion and substitution on the attribute graphs simulate the adaptation steps (explained in the later subsection) on the timetables retrieved. Deleting, inserting and substituting the less important vertices and edges have less of an effect on adapting the timetables. Thus such cases have lower costs assigned because of the need for less adaptation. The similarity measure between target case C_2 and source case C_1 is presented in formula (2).

$$S(C1,C2) = 1 - \frac{\sum_{i,j=0}^{n} p_{i,j} + \sum_{a=0}^{m} a_a + \sum_{d=0}^{k} d_d}{P + A + D} \qquad (2)$$

The notations in formula (2) represent the following:

n: number of matched vertices

m, k: numbers of the vertices or edges needed to be inserted into and deleted from C_2 respectively

$p_{i,j}$: cost assigned for substituting vertex or edge i in C_2 with vertex or edge j of C_1

a_a, d_d: costs assigned for inserting and deleting a vertex or edge labelled with attribute into and from C_2

P: the sum of the costs of substitution of every possible pair of vertices or edges in C_2 to those of C_1

A, D: the sum of the costs of inserting and deleting all of the vertices or edges into and from C_2 respectively

We can see that the closer the value $S(C_1, C_2)$ is to 1, the more similar C_1 and C_2 are.

5.1.3 Branch and Bound in Retrieval

The retrieval needs to search through the decision tree to find all the cases in the case base that are similar to the target case. The size of the decision tree storing all the possible permutations of the previous cases may be large, resulting in extensive searching. Thus the retrieval process may be difficult and time consuming. Branch and bound (Williams, 1999) is employed to reduce the size of the search tree in the retrieval phase. When the permutation of the courses of the target case is input into the case base, the retrieval starts from the root node and first searches down along the branches as far as possible in the tree that stores the most similar (sub-)structures. All of the possible candidate branches under one node that have a similar sub-structure and attributes with the target case are sorted by their summed costs. The branches storing the (sub-)structures whose costs exceed the given threshold are considered not to be similar to the (sub-)structures of the target cases and are all discarded. Thus the size of the search tree for retrieval can be greatly reduced because the retrieval does not need to search all the branches in the decision tree.

Backtracking is used when the retrieval cannot find a complete match. The retrieval backtracks to the parent node and the branch that has the lowest cost among the remaining branches will be chosen. This process continues until a complete match is found. All the complete and partial matches identified during the retrieval will be collected for potential adaptation.

Usually in timetabling problems, the more conflicts a course has with the other courses, the more difficult it is to schedule it. All the courses of the target case are sorted by their difficulties (here the degrees of the vertices in the attribute graph) in decreasing order and input into the decision tree for retrieval. Thus the retrieval process can first try to find a match for the more important courses.

5.1.4 Reuse and Adaptation of the Solutions

Adaptation of the timetables of all the retrieved cases is performed according to the (partial) matches found. The adaptation steps for each retrieved case are:

1. According to the match found, matched courses are substituted and all the un-matched courses in the retrieved case are deleted.

2. All the courses that violate the constraints in the newly formed timetable are removed and inserted into an unscheduled list sorted by their difficulties in decreasing order. The courses in the target case that are not yet scheduled are also inserted into the sorted unscheduled list.
3. All the courses in the unscheduled list are rescheduled by the graph heuristic method described below.

Different constructive methods can be used to generate the timetables based on the partial solutions. The CBR approach presented here employs a simple graph heuristic method in the adaptation that is the same as that employed in (Burke, Elliman and Weare, 1994) to construct a timetable based on the retrieved cases. It is briefly described below.

1. From the first one that is the most important, the courses in the unscheduled list are scheduled to the first timeslot with no violations (penalty-free);
2. The 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;
3. In the case of a tie, randomly assign the course to the first timeslot available.

The best timetable with the lowest penalty is selected as the solution of the target case.

5.1.5 Penalty Function

Every timetable generated for the target case is evaluated by the following formula (3):

$$\text{Penalty} = H \times 100 + S \times 5 \qquad (3)$$

H is the number of violations of hard constraints (the clashes between courses). It is assigned a cost of 100 to ensure that an infeasible timetable has a high cost. S is the total number of the violations of the soft constraints. They are assigned lower costs (at 5) because it is desirable to avoid them but not essential when a penalty-free timetable cannot be found.

To test our system and carry out the comparisons, the cost of violations of soft constraints is set as 5. In the experiments we found this value is not critical but should be limited within 20 when the cost of violations of hard constraints is set as 100. In different real-world timetabling problems, soft constraints could have different weights.

5.2 Experiments with Different Case Bases

To test the computational performance of the system on different case bases, different groups of random cases with different features have been defined systematically and stored in the case base. The determination of a number of cases needed to build a case base is not an easy task. In order to have different case bases, cases with a range of properties that real-world problems may have are generated. Thus an investigation of the system on a range of possible case bases can be carried out. Also different target cases are randomly generated so that the general performance of the system can be tested on a set of different target cases that the system may meet.

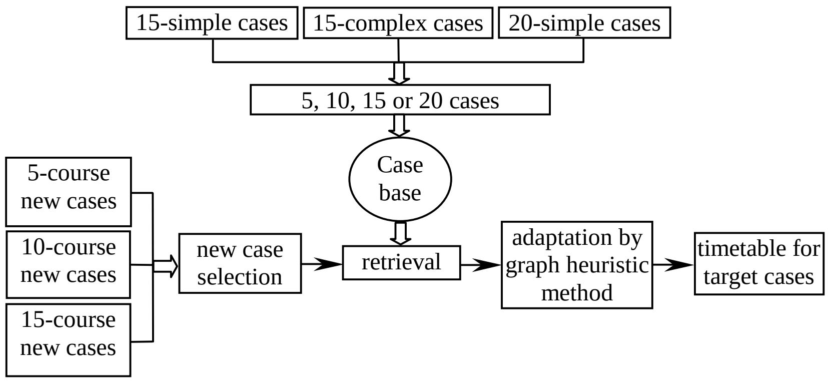


Figure 5-3 Schematic diagram of the CBR system used for evaluation

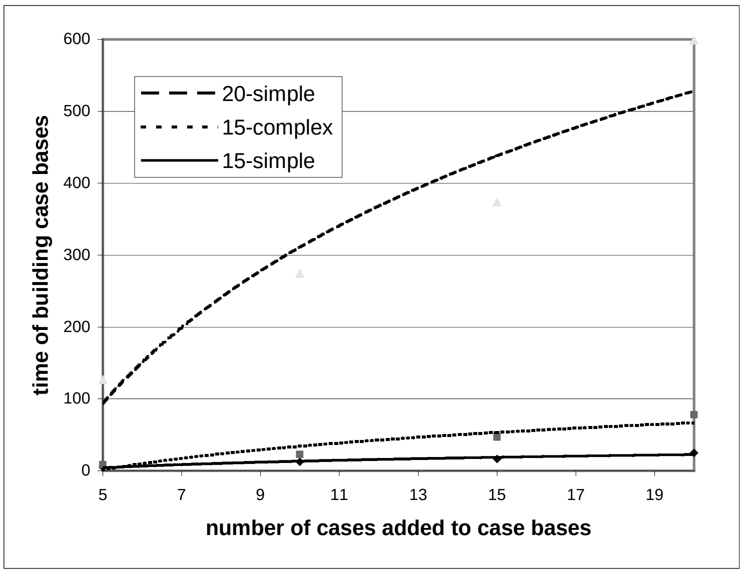
A schematic diagram of the system is given in Figure 5-3. Case bases with three different types of random cases were produced to solve a group of small target cases. These are 15-course simple, 15-course complex and 20-course simple cases. The complex cases have vertices whose degrees are at the lowest 1 and at the highest 4. The degrees of vertices in simple cases are at the lowest 1 and at the highest 3. The complex cases have more constraints than those simple cases and are usually more difficult to solve. The attributes are randomly selected from Table 4-1 and Table 4-2. The timetables of these cases are generated by using the graph heuristic method and stored in the case base. Small target cases with 5, 10 and 15 courses, also randomly generated, are tested to give an easy evaluation on the CBR approach developed. The system is developed in C++ and the experiments are run on Pentium 450Mhz PC with 128MB of RAM under the Windows environment.

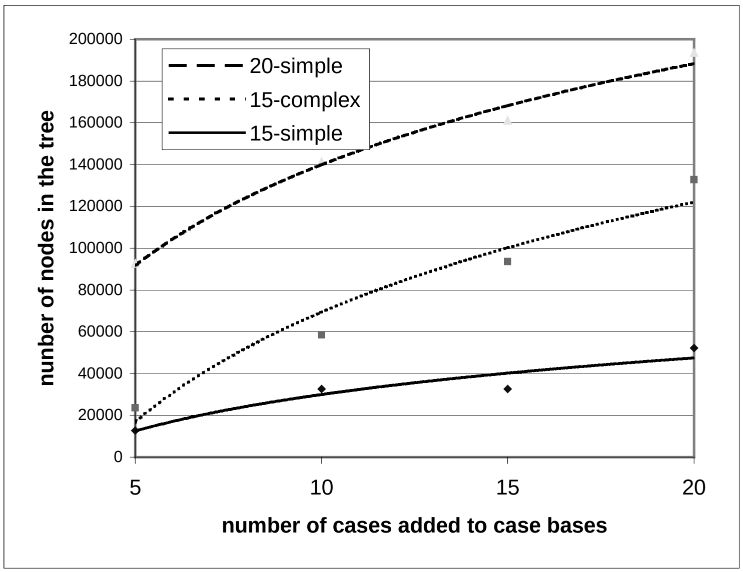
5.2.1 Algorithm Complexity Evaluation

5.2.1.1 Time and Memory Needed to Build the Decision Tree

In every case base 5, 10, 15 or 20 of the three types of cases are stored. Figure 5-4 gives the time spent and space needed to build these 12 different case bases. In the notation x/y in the table, x gives the time in seconds and y is the number of nodes in the decision tree.

We can see that from the table that because the number of permutations grows rapidly (but not exponentially) with the number of vertices in the graph, adding 20-course cases into the case base takes much more time and space than for both simple and complex 15-course cases. We can also observe from the charts shown that the time and number of nodes grows rapidly but not explosively with the number of cases in the case base. This is because many of the (partial) permutations of the cases may be stored under the node that is built for previous cases if they have the same (sub-)structures.



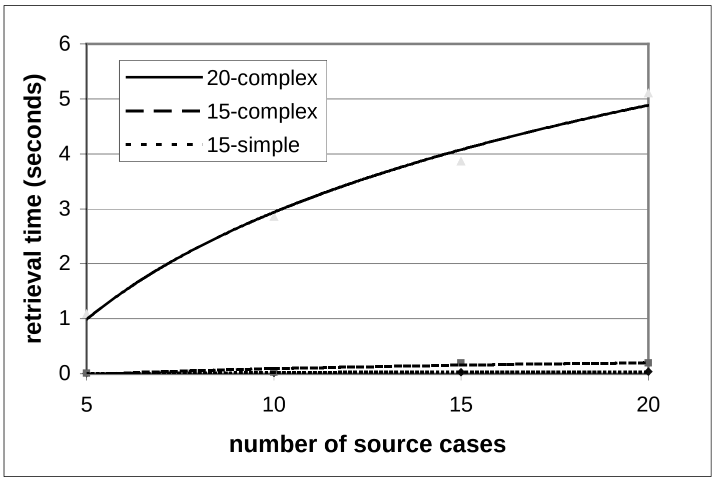


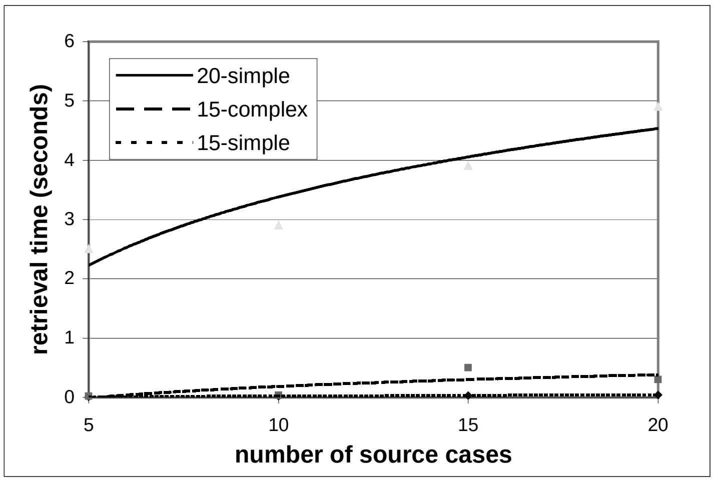
	5	10	15	20
15-simple	5.04/12689	12.58/32647	16.57/32647	24.83/52153
15-complex	8.48/23569	22.76/58475	46.88/93523	77.69/132750
20-simple	125.85/92449	273.84/141163	373.09/160887	598.36/193473

Figure 5-4 Time of building case bases of 15-simple, 15-complex and 20-simple cases

5.2.1.2 Time Spent in Retrieval

Figure 5-5 gives the retrieval time for different target cases of 5-course, 10-course and 15-course, respectively. We can see that the retrieval time changes in the same way as that for building the same case bases. With the number of source cases increase, the retrieval time grows rapidly but not exponentially as many of the permutations of source cases added into the decision tree are stored under the same nodes previously built for the similar sub-graphs.





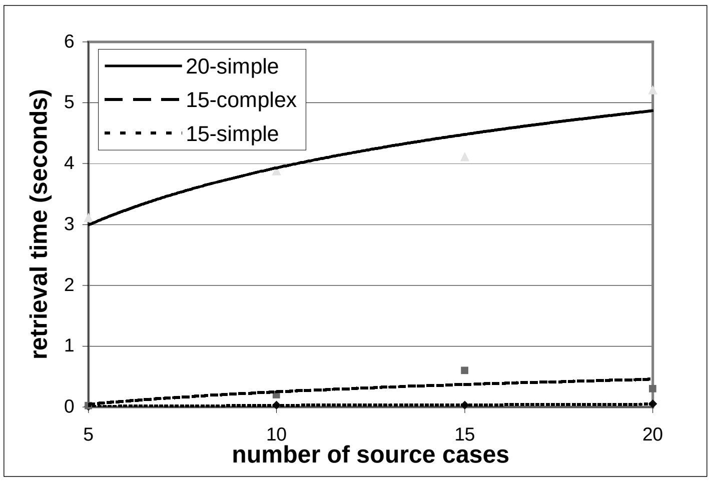


Figure 5-5 Retrieval time in different case bases for target cases (upper: 5-course; middle: 10-course; lower: 15-course target cases)

5.2.2 Performance Evaluation

5.2.2.1 The Number of Target Cases That Find Matches

With too few matched vertices, the retrieved cases cannot provide enough information for adaptation. Only matches that have enough courses (here more than half) in the retrieved cases are seen as helpful and retrieved for adaptation. From all the retrieved cases, a set of the most similar cases is selected as a set of candidates for the adaptation.

To test how many target cases can retrieve cases from the case base with different complexity, two groups of experiments were conducted on the case bases storing simple or complex 15-course cases. The results are given in Tables 5 and 6 respectively. The values before and after '/' give the percentages of target cases that could retrieve partial and complete matches from the case base respectively. The values in parentheses give the overall percentage, as either partial or complete matches found.

No. of 15-simple cases in case base	**5-course target case**	**10-course target cases**	**15-course target case**	**Average percentages**
5	100/100 (100)	100/0 (100)	30/0 (30)	76.67
10	100/100 (100)	100/0 (100)	70/0 (70)	90
15	100/100 (100)	100/0 (100)	70/0 (70)	90
20	100/100 (100)	100/45 (100)	70/0 (70)	90

Table 5-1 Percentages of target cases that find case(s) from the 15-course simple case base

No. of 15-complex cases in case base	**5-course target case**	**10-course target cases**	**15-course target case**	**Average percentages**
5	100/100(100)	100/0(100)	35/5(35)	78.3
10	100/100(100)	100/0(100)	70/5(70)	90
15	100/100(100)	100/70(100)	85/75(85)	98.33
20	100/100(100)	100/70(100)	85/80(85)	98.33

Table 5-2 Percentages of target cases that find cases from the 15-course complex case base

It can be seen from Table 5-1 that all of the 5-course and 10-course target cases can find (partial) match(es) from a case base with simple 15-course cases. No complete match can be found for target cases with 10 or more courses when the case base consists of less than 20 cases. Table 5-2 shows that storing complex cases in the case base enables more target cases to find matches. Higher percentages of larger target cases (10-course and 15-course target cases) retrieve cases (complete or partial matches) from the case base.

We can also see that when 10, 15 or 20 simple cases are stored in the case base, the same number of target cases (90 percent) can retrieve matches. Also, the same number of target cases (98.3 percent) can find matched cases in the case bases with 15 or 20 complex cases. This is because the attribute graphs of a certain number of cases in the case base provide a certain number of different (sub-)structures in the decision tree. Additional cases do not provide new (sub-)structures in the decision tree. Attribute graphs of complex cases can provide more (sub-)structures, thus more target cases can retrieve cases from the case base with more than 10 or 15-course complex cases.

The effect of storing larger cases with 20 courses in the case base is tested in a further experiment and the results are given in Table 5-3. The overall percentages of successful retrievals are higher than those with smaller simple cases but lower than those with smaller complex cases.

No. of 20-simple cases in case base	**5-course target case**	**10-course target cases**	**15-course target case**	**Average percentages**
5	100/100(100)	100/0(100)	85/0(85)	95
10	100/100(100)	100/0(100)	85/0(85)	95
15	100/100(100)	100/0(100)	85/0(85)	95
20	100/100(100)	100/45(100)	85/0(85)	95

Table 5-3 The percentages of target cases that find cases from the 20-course case base

Figure 5-6 gives a chart of average percentages of target cases that can retrieve case(s) from the case base with different numbers of three types of cases. We can observe that storing more than 15 complex 15-course cases provides a higher percentage of success in retrieval than storing both simple 15-course and simple 20-course cases. By storing a sufficient number of complex cases, sufficient (sub-)structures can be stored in the decision tree for reuse. It is actually the number of (sub-)structures, not the number and size of the cases, that affects the percentage of successful retrievals. Thus it is not necessary to store more cases.

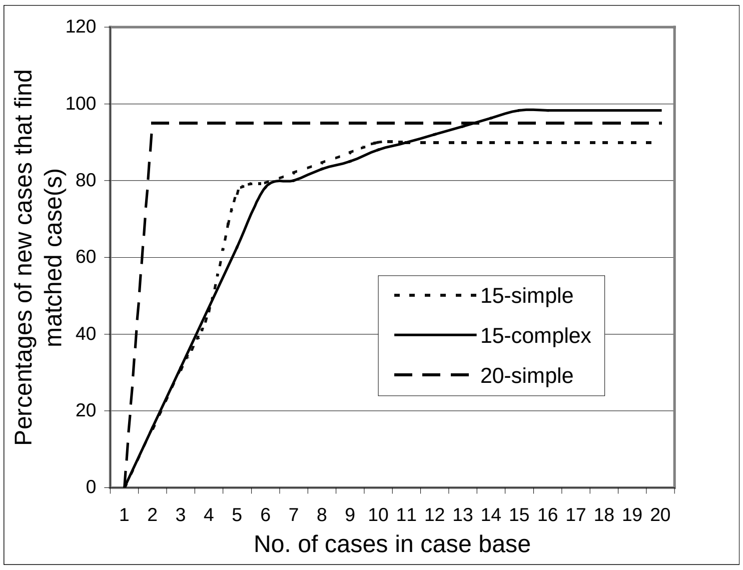


Figure 5-6 Percentage of target cases that retrieve case(s) from different case bases

5.2.2.2 Adaptation of Retrieved Cases

20 different cases with 5, 10 or 15 courses are tested on the case bases with 5, 10 15 or 20 of the three types of cases respectively. So altogether 720 (=20×3×4×3) experiments were carried out. The graph heuristic method described in Section 3 is used in the adaptation to adapt all the retrieved cases and the timetable that has the lowest penalty is used as the solution for the target cases. For comparison, the same graph heuristic method is also used to generate a timetable from scratch for each target case that can retrieve cases from the case base. All the timetables generated by these methods are evaluated by using the penalty function given in (2). The number of schedule steps needed during adaptation is also taken into account in the comparison. The average penalties and schedule steps for these two methods are presented in Table 5-4, Table 5-5 and Table 5-6.

The y in ‘x/y’ gives the number of schedule steps needed to obtain a timetable that has a penalty x. Values in parentheses give the penalty and schedule steps of the timetables generated by adapting complete matches for the target cases.

No. of cases	**5-course target case**		**10-course target cases**		**15-course target case**	
	CBR	**GH**	**CBR**	**GH**	**CBR**	**GH**
5	6/7(6/8)	11/15	22.8/35.8	30.5/45.6	39.2/68	39.2/76
10	6/6(5/6)	11/15	16.5/30.2	30.5/45.6	33.2/59	36.1/59
15	6/6(5/6)	11/15	16.5/30.3	30.5/45.6	33/59.8	36.1/69
20	6/5(5/6)	11/15	17/28(23/40)	30.5/45.6	30/54.3	34/66.1

Table 5-4 Penalties and schedule steps by graph heuristic (GH) and CBR approach with different 15-course simple case bases

No. of cases	**5-course target case**		**10-course target cases**		**15-course target case**	
	CBR	**GH**	**CBR**	**GH**	**CBR**	**GH**
5	7/7(6/5)	11/15	19.3/30.5	30.5/45.6	30/49	15/50
10	6/6(6/5)	11/15	18.5/31.2	30.5/45.6	30/49	15/50
15	6/6(5/5)	11/15	17/31(28/39)	30.5/45.6	30/60(39/65)	39.7/69
20	6/6(5/5)	11/15	16/27(28/39)	30.5/45.6	27/61(39/68)	39.7/69

Table 5-5 Penalties and schedule steps by graph heuristic (GH) and CBR approach with different 15-course complex case bases

No. of cases	**5-course target case**		**10-course target cases**		**15-course target case**	
	CBR	**GH**	**CBR**	**GH**	**CBR**	**GH**
5	6/6.7(5/6)	11/15	16.5/28.7	30.5/45.6	37.9/55	40/66.4
10	6/6(5/5.5)	11/15	15.8/28.3	30.5/45.6	36.8/55.7	39.4/67
15	6/6.5(5/5.3)	11/15	16.4/27.3	30.5/45.6	61.7/79.3	53.4/81
20	6/6(5.3/5.4)	11/15	18/29(10/4)	30.5/45.6	62.2/76.5	46/72.4

Table 5-6 Penalties and schedule steps by graph heuristic (GH) and CBR approach with different 20-course case bases

From the results shown in Table 5-4, Table 5-5 and Table 5-6 we can see that in all of the experiments solving 5-course and 10-course target cases, the timetables constructed by the graph heuristic method based on the partial solutions from the proposed CBR approach need much fewer scheduling steps and have less penalties than those constructed from scratch using the graph heuristic (GH) approach. The knowledge and experiences stored in the previously solved problems that are structurally similar to the target problems are re-used and not too much effort needs to be taken to get high quality results.

In solving the larger 15-course target cases by the case base with 5 or 10 15-course complex cases, the CBR approach finds timetables with higher penalties than those from the graph heuristic approach and takes almost the same number of schedule steps in adaptation. This is because only storing a small number of (less than 10) complex cases cannot provide enough good cases (sub-structures) and the complexity of the retrieved cases makes the adaptation difficult. Storing more complex cases provides much better results. Also, larger retrieved cases may cause more adaptation because more courses in the timetables of these cases may need more adaptation. This is why in Table 5-6 some of the retrieved larger cases provide high penalty timetables for the target cases.

It can also be seen that not all of the timetables adapted from the complete matching cases are better than those from the partial matching cases (although most of them are much better than those generated by the graph heuristic approach). This might be because the larger good structures of the complete matches in the timetables are more likely to be destroyed in the adaptations for the target cases.

5.3 Chapter Summary

The improved CBR approach presented in this chapter shows that the retrieved cases that have similar (sub-)structures can provide high quality partial solutions for the target cases. This is because by retrieving structurally similar cases from the case base, solutions generated on similar constraints may be easily reused for the target case without significant adaptations. Timetables constructed by using the graph heuristic method on the basis of these partial solutions take less scheduling effort to get lower penalty solutions than those constructed by only using the same graph heuristic method from scratch.

The CBR system also shows that storing a certain number of cases in the case base can provide the same number of (sub-)structures as those obtained by storing more cases. Also storing a certain number of complex cases works better than storing larger or more simple cases for providing the sub-structures for re-use. It is the number of (sub-)structures, not the number of cases in the case base that contributes to the successful retrieval of partial solutions for adaptation. It is important to build a case base with just a certain number of cases because the size of the decision tree grows rapidly when the size and the number of the cases in the case base increases. The work presented in the next chapter is to tackle this issue.

Chapter 6 Multiple-Retrieval CBR for Course Timetabling

In previous chapters we have shown that a basic structured CBR approach 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. Also a large timetabling problem with complicated constraints and attributes will rarely match a case of the same size in the case base. With the limited help from a single retrieved case of small, the larger new case may not obtain good solutions based on the small matched part.

Based on the CBR system presented in the last chapters, this chapter will 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. The work (Burke, MacCarthy, Petrovic, and Qu, 2001b) presented in this chapter has been resubmitted to the

Journal of Operational Research Society. It draws upon the structured CBR approach presented in previous chapters.

6.1 Multiple-Retrieval Approach on a Decision Tree

The partition is made by carrying out the retrieval process recursively. 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 recursive retrievals partition the problem into smaller solvable sub-problems based on the retrieval process employed in the previous CBR system. A schematic diagram illustrating the process is presented in Figure 6-1.

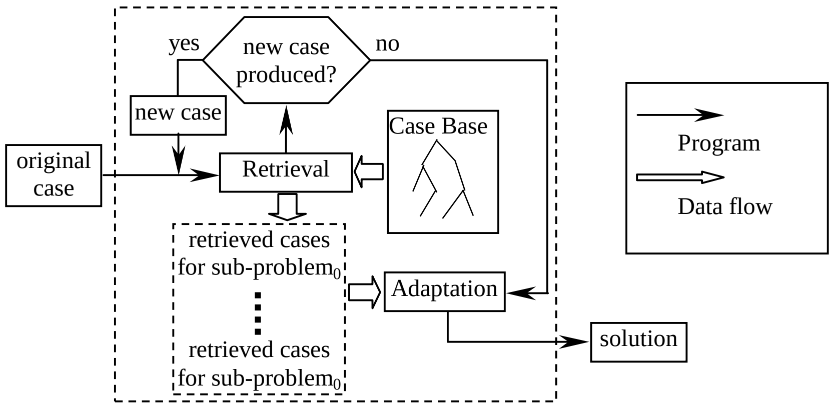


Figure 6-1 Schematic Diagram of the Multiple-Retrieval CBR System

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. The matched part in the attribute graph of the new case in the last cycle is combined into one vertex, which we call a *super vertex*. Edges that are originally adjacent to the matched vertices are combined and adjacent to the super vertex. The attributes of the newly combined edges are decided by the following:

- If one of the original edges is labelled 7 (conflict), the new attribute will be set as *conflict*.
- In other cases, the new attribute will be set as one of the original ones.

By never releasing the constraints (attributes) using above rules, we can guarantee that the combined final solutions (combining process shown in the next section) will always be feasible with hard constraints.

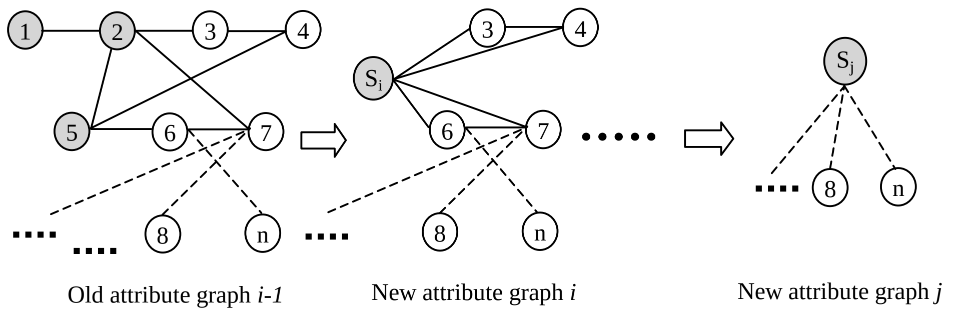


Figure 6-2 New attribute graph generated after each retrieval

Figure 6-2 illustrates in some of the cases 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.

This multiple-retrieval approach is carried out on the same decision tree and partitions the problem upon the case base rather than by employing fixed rules. It generates sub-problems automatically depending on the cases in the case base. Usually more than one possible match can be found for each sub-problem partitioned. The most similar cases are used to generate a number of candidate timetables. The one with the lowest penalty (calculated by formula (3)) is selected as the best solution for the new timetabling problem.

The new multiple-retrieval approach requires some changes on the similarity measure that was used in the single retrieval process. In the new similarity measure, the individual similarity between each sub-problem and the retrieved cases for it is calculated in the same way as when using single retrieval, 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 on 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. This new similarity measure is shown in formula (4):

$$S = 1 - \frac{\sum_{1}^{r}(\sum_{b=0}^{n} p_b + \sum_{i=0}^{m} a_i + \sum_{j=0}^{k} d_j)}{r(P + A + D)} \qquad (4)$$

The notation used in formula (4) 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 the new case;

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

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, respectively;

P is the sum of the substitution cost of every possible pair of vertices or edges;

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.

6.2 Adaptation on Multiple Cases Retrieved

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.

6.2.1 Combining Sub-Solutions

Starting from the sub-solution of the last sub-problem, we combine all these sub-solutions into a final solution for the original new case by substituting the corresponding super vertices with their sub-solutions repeatedly. The combined solution is guaranteed to be feasible as we never release the constraints and all the sub-problems are feasible.

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

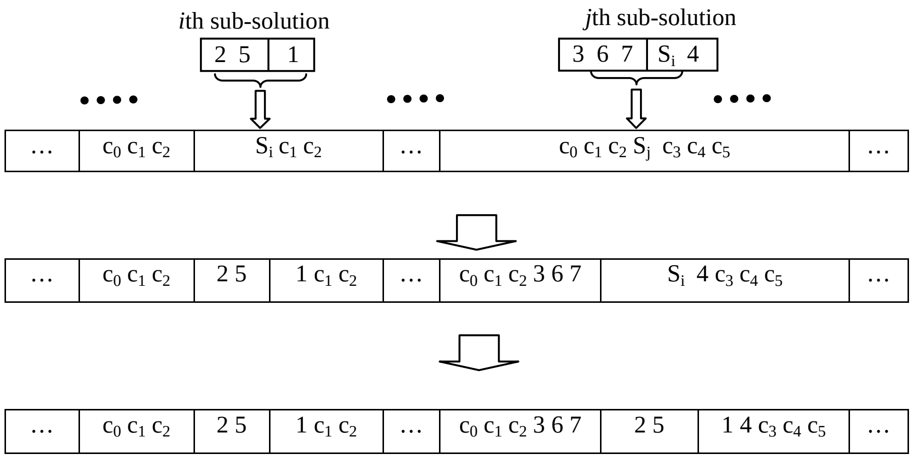

*i*th sub-solution
2 5 | 1
*j*th sub-solution
3 6 7 | S_i 4
… | c_0 c_1 c_2 | S_i c_1 c_2 | … | c_0 c_1 c_2 S_j c_3 c_4 c_5 | …
… | c_0 c_1 c_2 | 2 5 | 1 c_1 c_2 | … | c_0 c_1 c_2 3 6 7 | S_i 4 c_3 c_4 c_5 | …
… | c_0 c_1 c_2 | 2 5 | 1 c_1 c_2 | … | c_0 c_1 c_2 3 6 7 | 2 5 | 1 4 c_3 c_4 c_5 | …

Figure 6-3 Combining the solutions of the sub-problems

The combined partial solution is adapted by the following steps to generate the final solution. The adaptation process uses a basic timetabling method to allocate rooms and improve the CBR generated solution with soft constraints.

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 explain below.

6.2.2 Graph Heuristic Method with Tournament Selection

The graph heuristic with tournament selection (GHT) presented by Burke, Newall and Weare (1998) is used to schedule the courses in the unscheduled list one by one to the first timeslot and room with no violations (penalty-free). Tournament selection is used to select the first course every time from a 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 hard constraint or no room is available.

6.2.3 Penalty Function

The penalty function given in formula (3) is used to evaluate every timetable generated in the experiments carried out in the next session. The violations of unscheduled courses are assigned a high cost of 100. Violations of soft constraints, indicated by S, are assigned a relatively low cost of 5.

6.3 Experiments and Results

In this section we carry 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 and it is very important for the analysis of the CBR approach that we understand exactly the structure of the sets that we are working with.

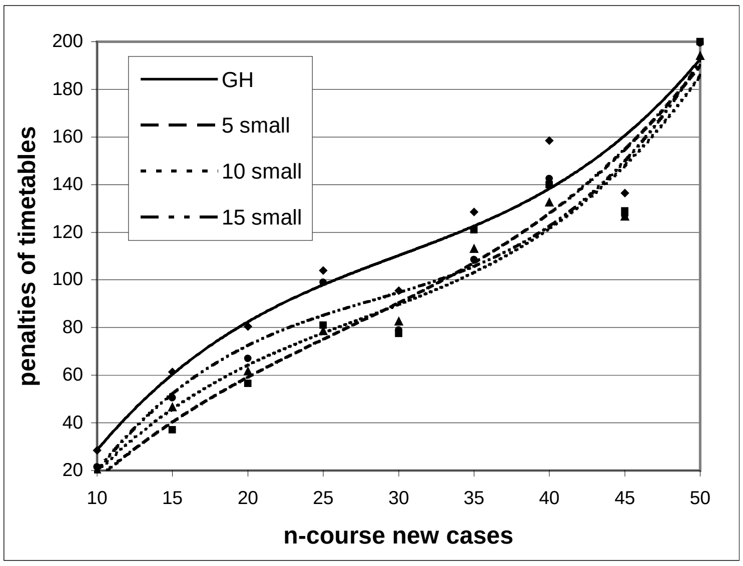
A large number of experiments have been carried out to solve timetabling problems of different size on case bases with different types and sizes of cases. We use two types of cases in the case bases: simple and complex (of small or large size). In complex cases, every course has at most 4 and at least 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 GHT (Burke, Newall and Weare, 1998).

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. Also, we investigated the employment of the multiple-retrieval CBR as the initialisation approach for Tabu Search in order to determine whether CBR might provide solutions which are a good starting point for meta-heuristic methods.

6.3.1 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 multiple-retrieval approach with adaptation employing the GHT. These solutions are compared with those generated from scratch by the same GHT. Figure 6-4 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 best average result for each new case type is highlighted in the table.

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



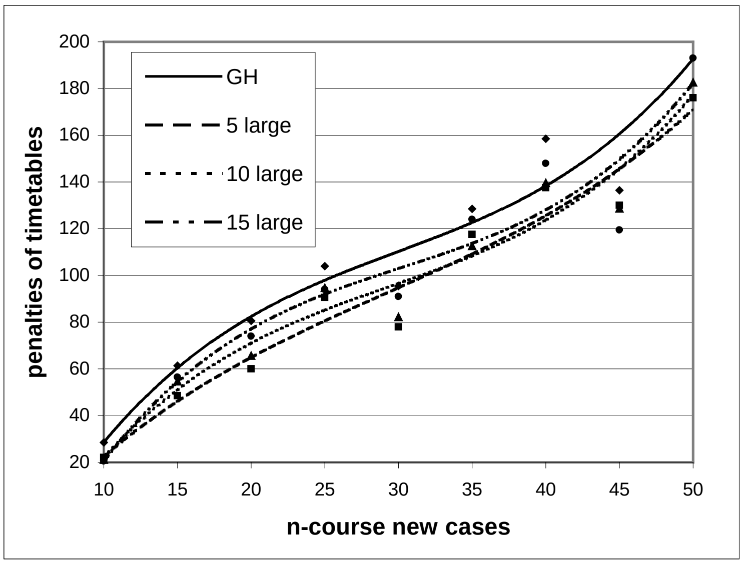


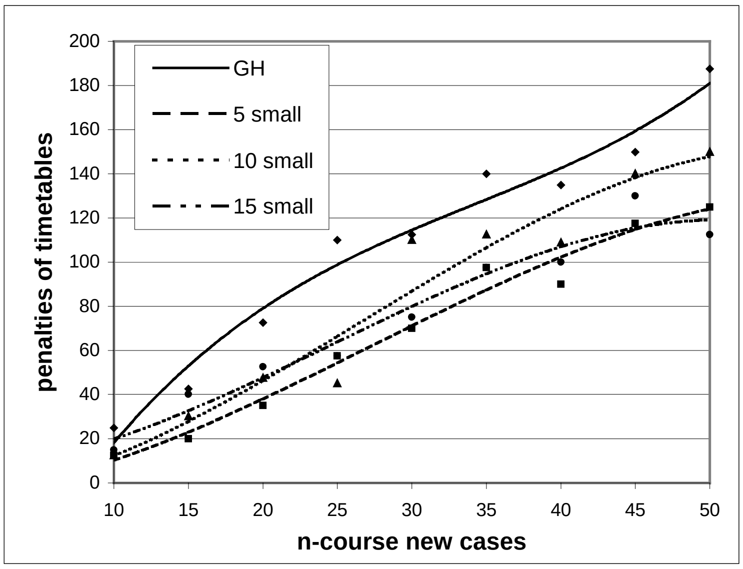
Figure 6-4 Penalties of timetables by using graph heuristic (GH) and CBR with case bases of simple cases (upper: small cases; lower: large cases)

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. 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, although we have not tested this statistically.

6.3.2 Case Bases with Complex Cases

Another group of experiments have been conducted on the nine sets of new cases to investigate the use of case bases with complex cases. Figure 6-5 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.

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



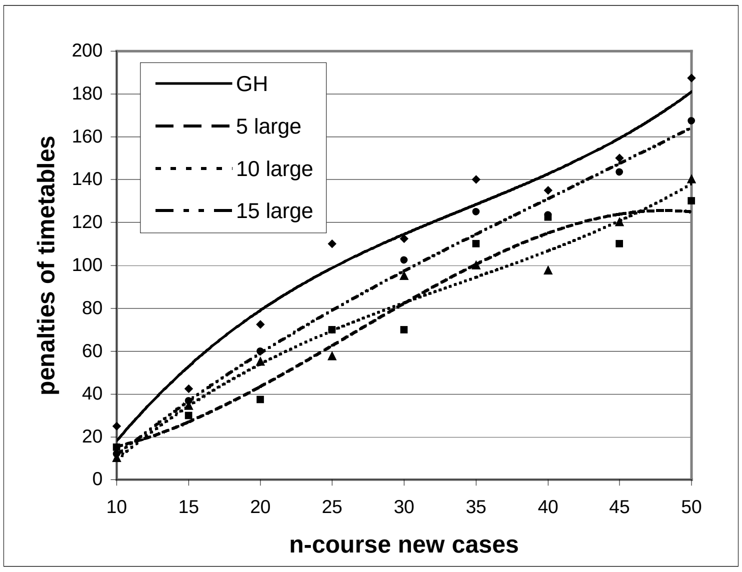
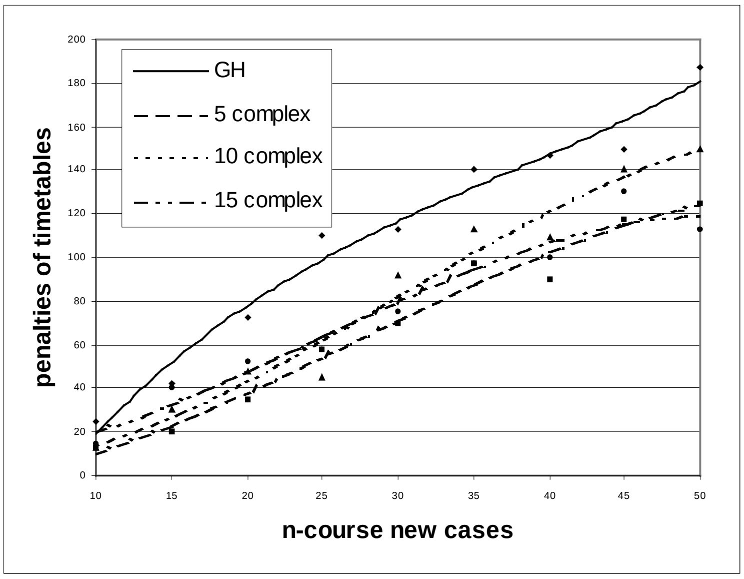


Figure 6-5 Penalties of timetables by using graph heuristic (GH) and CBR with case bases of complex cases (upper: small cases; lower: large case)

6.3.3 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 performs better than reusing larger sub-solutions on solving the same problems.

The results of experiments on case bases of small simple and complex cases are illustrated in Figure 6-6. 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 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.



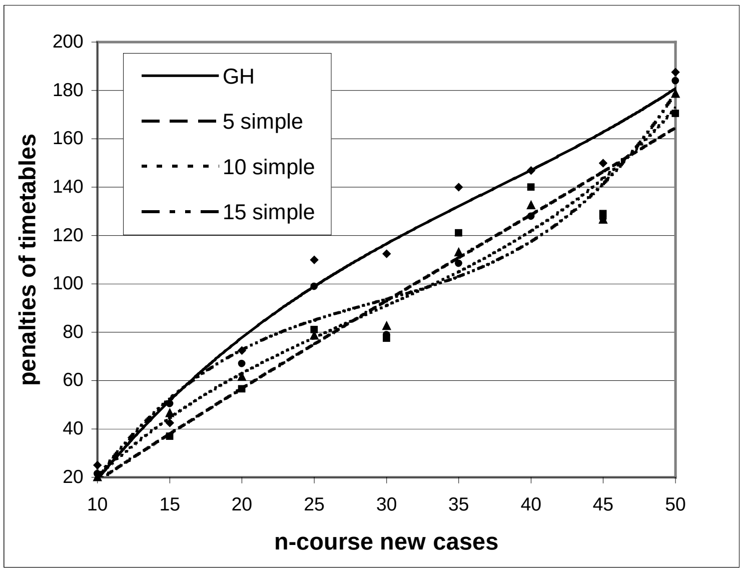
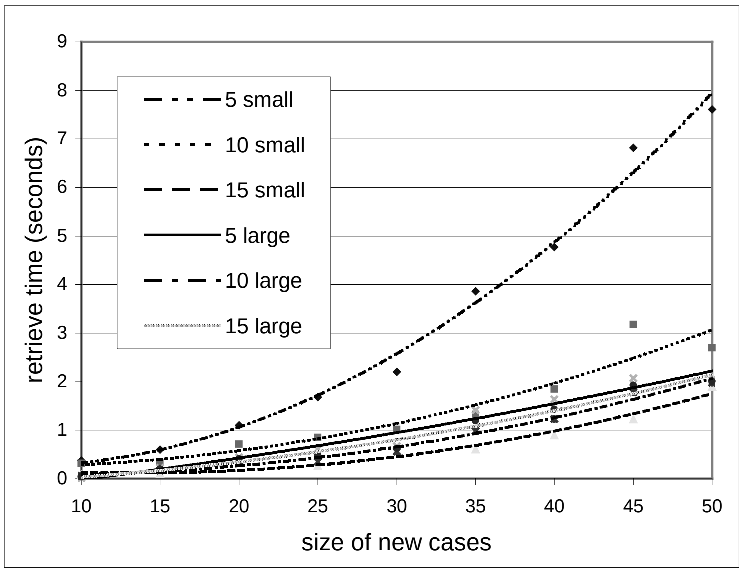


Figure 6-6 Penalties of timetables by using graph heuristic (GH) and CBR with case bases of small cases (upper: complex cases; lower: simple cases)

6.3.4 Comparison on Retrieval Time in Different Case Bases

The retrieval time of the multiple-retrieval CBR approach varies on different case bases for different new cases. The overall retrieval times for new problems on the case bases with simple and complex cases are presented in Figure 6-7, 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 for the case base are needed for the new case. With the limited number of scheduling structures that 5 simple cases can provide, a longer 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.

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.



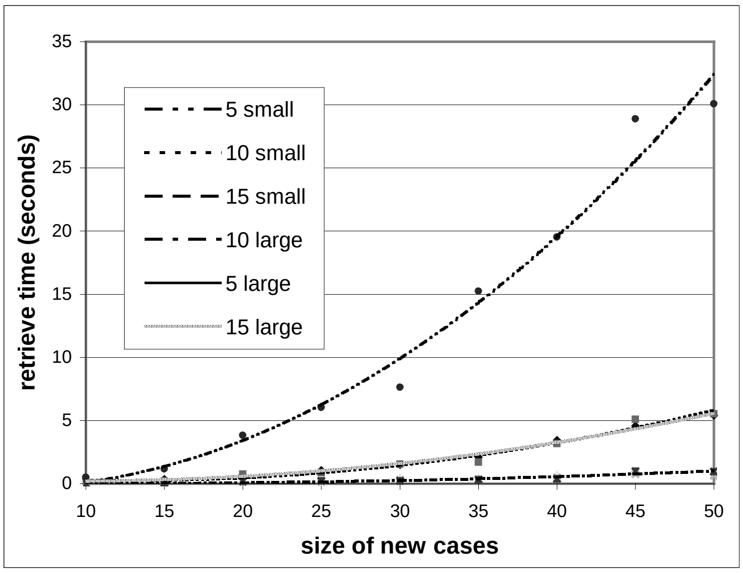
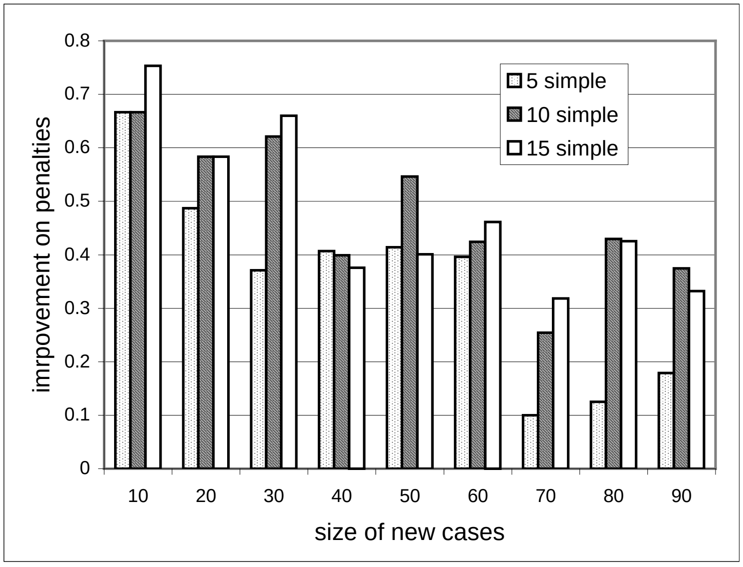


Figure 6-7 Retrieve time on case bases of simple cases (upper: simple cases; lower: complex cases)

6.3.5 Multiple-Retrieval CBR as the Initialisation Method for Tabu Search

The results of our experiments led to a natural question: would the suggested CBR approach provide good starting point for local search meta-heuristics such as Tabu Search. The motivation here is that the CBR approach might be able to generate good solutions which Tabu Search could “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 cases as an initialisation method for Tabu Search. We compare this with the results of Tabu Search with initialisation from GHT alone. The table in Figure 6-8 presents the penalties of timetables generated by Tabu Search with the multiple-retrieval CBR and with GHT alone as the initialisation methods. We can observe that Tabu Search with multiple-retrieval CBR as initialisation outperforms that of Tabu Search with GHT as initialisation. The significant improvement on the penalties of the timetables generated from the multiple-retrieval CBR over GHT as initialisation is drawn in the charts in Figure 6-8. The multiple-retrieval CBR does indeed provide a good starting point for the Tabu Search algorithm for these problems. By reusing good schedule structures in timetables of previous similar problems, the multiple-retrieval CBR approach may also decrease the possibility of becoming stuck in a local optimum.

n-course new case	Tabu Search + GHT	5 simple	10 simple	15 simple	5 complex	10 complex	15 complex
10	150	50	50	37	**36**	51	100
20	156	80	**65**	**65**	83	90	136
30	256	161	97	**87**	100	115	187
40	253	**150**	152	158	161	191	207
50	302	177	**137**	181	180	291	192
60	245	148	141	**132**	148	199	235
70	220	198	164	**150**	181	186	245
80	235	205.6	134	135	**118**	174	181
90	235	193	**147**	157	166	187	196



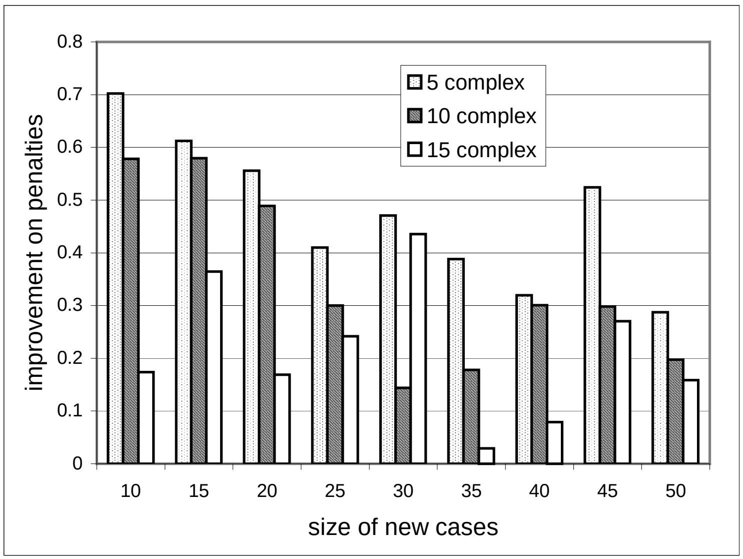


Figure 6-8 GHT and multiple-retrieval CBR with small cases as the initialisation method for Tabu Search

6.4 Chapter Summary

Real-world timetabling problems are usually very large and complex with a number of complicated constraints. The multiple-retrieval CBR approach provides promising results quickly on solving timetabling problems of different sizes. Large timetabling problems are tackled by a partitioning process that is carried out recursively on the same case base to automatically decompose the problems into small solvable sub-problems. The solutions of the partitioned sub-problem can be obtained by adapting high quality timetables of the retrieved problems that have common similar constraints. High quality scheduling structures in the sub-solutions found by multiple retrievals are retained after the combination in the adaptation phase and 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, 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 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.

Chapter 7 Knowledge Discovery in Hyper-Heuristic using CBR on Course Timetabling

The work presented in the previous chapters investigated the contributions that CBR can make to solve course timetabling problems by reusing previous good quality timetables. In real-world problem solving, people also reason by reusing the heuristics or procedures that were successful in solving previous similar problems. In timetabling (and also other scheduling problems), sometimes a small change in the constraints may lead to a quite different solution, thus research issues in representation and similarity need to be carefully conducted in CBR to detect the differences in solutions that result from the differences in problems. Modelling and reusing the knowledge of methods people use rather than the actual timetables in solving similar problems would also be useful.

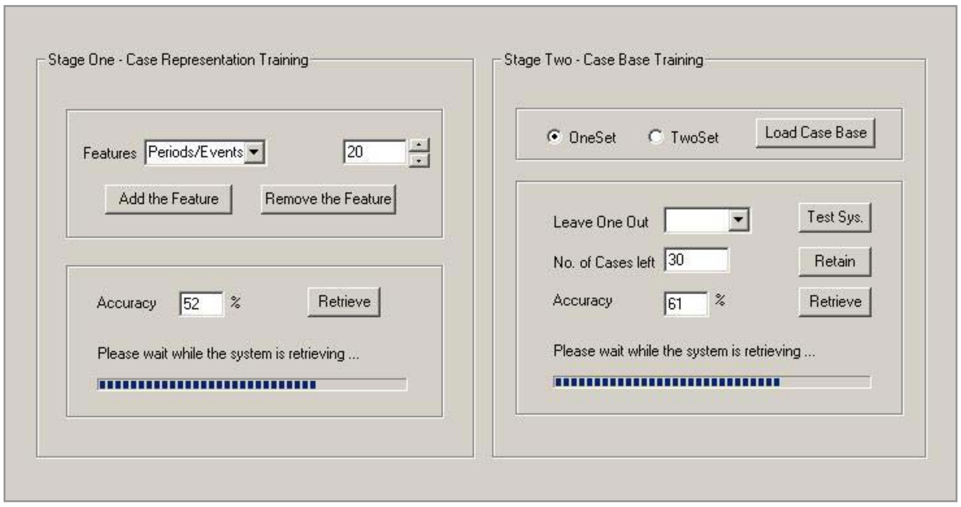
This chapter presents a new hyper-heuristic method using CBR for solving course timetabling problems (Burke, MacCarthy, Petrovic, and Qu, 2002). One of the overriding motivations of hyper-heuristic methods is the attempt to develop techniques that can operate with greater generality than is currently possible. The basic idea behind this is that we maintain a case base of the information about the most successful heuristics for a range of previous timetabling problems to predict the best heuristic for the new problem in hand using the previous knowledge. Knowledge discovery techniques are used to carry out the training on the CBR system to improve the system performance on the prediction.

7.1 CBR as a Heuristic Selector

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

7.1.1 Knowledge Discovery on Heuristic Selection

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



Stage One - Case Representation Training

Features: Periods/Events 20

Accuracy: 52 %

Please wait while the system is retrieving ...

Stage Two - Case Base Training

[✓ OneSet] TwoSet

Leave One Out:

No. of Cases left: 30

Accuracy: 61 %

Please wait while the system is retrieving ...

Figure 7-1 Screenshot of the 2-stage knowledge discovery process for course timetabling

Knowledge discovery techniques are employed to extract the knowledge of meaningful relationships within the case-based heuristic selector via iterative training processes on cases of course timetabling problems. There are two iterative training stages used in the process. Figure 7-1 presents the screenshot of this 2-stage process. The first stage tries to discover the representation of cases with a proper set of features and weights. The second stage trains the case base so that it contains the proper collection of source cases. Both of the processes are carried out iteratively. The overall objective is to obtain the highest accuracy on retrievals for predictions of heuristics for target cases.

7.1.2 Getting Started

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

The retrieval is a similarity-driven process that searches through the case base to find the most similar source cases. The similarity measure employs a nearest-neighbor method that calculates a weighted sum of the similarities between each pair of individual features between cases. Formula (6) presents the similarity measure between the source case C_s and the target case C_t in the system:

$$S(C_s, C_t) = \frac{1}{\sqrt{\sum_{i=0}^{j} w_i * (fs_i - ft_i)^2 + 1}} \tag{6}$$

the notation is described as follows:

j: the number of features in the case representation

w_i: the weight of the *i*th feature reflecting the relevance on the prediction

fs_i, ft_i: the values of the *i*th feature in source case C_s and target case C_t respectively

The possible values of the features describing timetabling problems are all integers (see Appendix D). So the higher the value of $S(C_s, C_t)$, the more similar the two cases are.

The performance of the system is tested on different sets of target cases. The training on the system is targeted at a reasonably high accuracy on all of the retrievals for the target cases quickly. Within each retrieval, the best two heuristics of the retrieved case are compared with the best heuristic of the target case. If the best heuristic of the target case maps onto any of the best two heuristics of the retrieved case, the retrieval is concluded as successful. Actually, in the training processes, we found that sometimes penalties of the timetables produced by different heuristics are close or equal to each other.

7.1.3 Training on the Case Representation

An initial case base is built up containing a set of different source cases with artificially selected specific constraints and requirements from Appendix B. An initial list of features is firstly randomly selected to represent cases. Each of the features is initially assigned with the same normalized weights. There are 11 features (details of the which are given in Appendix D) in the initial case representation.

Our knowledge discovery on the case representation to train the features and their weights in the system adopts the iterative methodology (Cunningham and Bonzano, 1999). In every iteration, we:

a) Analyse the retrieval failures.

b) Propose new features to address retrieval failures.

c) Select a discriminating set of features for the new case representation.

d) Evaluate the competence of this representation.

In the CBR system, the training for case representation is a recursive failure-driven process carried out to refine the initial features and their weights. A schematic diagram of the knowledge discovery on case representation is given in Figure 7-2. The knowledge discovery process in the system includes the following steps:

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

Removing irrelevant features. After certain rounds of iterative adjustments, the weights of some of the features may be small enough to be removed from the feature list. This means that these features are either irrelevant or less important, thus are not needed in the case representation. Retaining the irrelevant features may confuse the retrieval process, as the similarities between cases maybe too close to each other, thus reduce the number of the successful retrievals and decrease the system performance (John, G.H. Kohavi, R. and Pfleger, K., 1994).

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

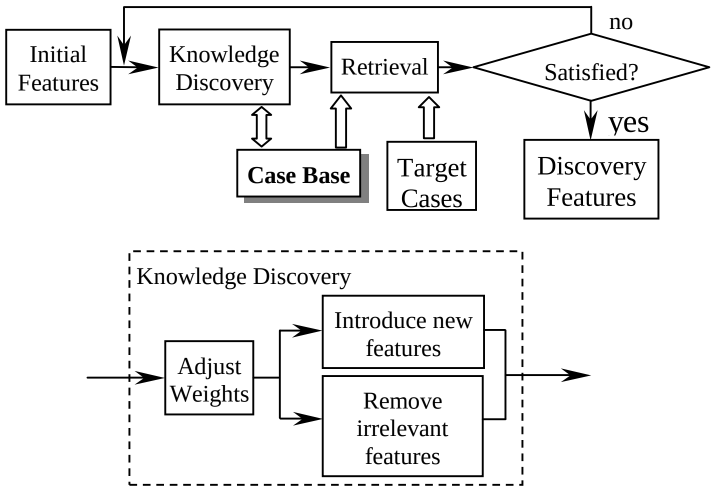


Figure 7-2 Schematic Diagram of Knowledge Discovery on Featrues and Their Weights

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

7.1.4 Training on the Case Base

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

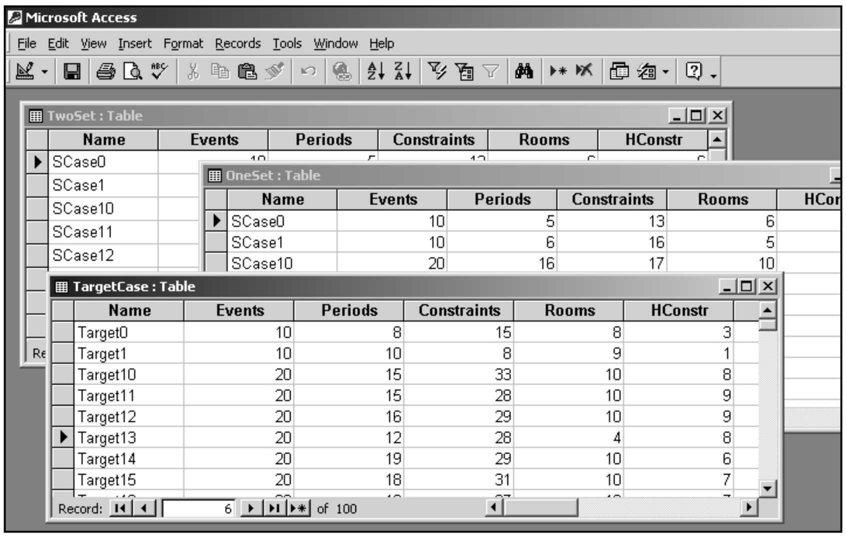
Firstly we build up two initial case bases with source cases of 9 different sizes with 10, 15, … to 50 courses in them:

“OneSet” – For each size, 5 source cases are produced, each has one of the 5 heuristics in Appendix B as its best heuristic. We name this case base “OneSet” as it contains one set of the 5 heuristics for cases with different sizes (thus in OneSet there are 9 * 5 = 45 source cases).

“TwoSet” – The case base consists of two sets of the 5 heuristics for each source case with 9 different sizes (in total 9 * 5 * 2 = 90 source cases).

A database is built up containing these two case bases and the target case set. Figure 7-3 presents a screenshot of the database.

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



TwoSet : Table

Name	Events	Periods	Constraints	Rooms	HConstr
SCase0					
SCase1					
SCase10					
SCase11					
SCase12					

OneSet : Table

Name	Events	Periods	Constraints	Rooms	HCon
SCase0	10	5	13	6	
SCase1	10	6	16	5	
SCase10	20	16	17	10	

TargetCase : Table

Name	Events	Periods	Constraints	Rooms	HConstr
Target0	10	8	15	8	3
Target1	10	10	8	9	1
Target10	20	15	33	10	8
Target11	20	15	28	10	9
Target12	20	16	29	10	9
Target13	20	12	28	4	8
Target14	20	19	29	10	6
Target15	20	18	31	10	7

Record: 6 of 100

Figure 7-3 Screenshot of the Case Bases and Target Cases for Course Timetabling Problems

The training process on these two initial case bases is carried out recursively using the "Leave-One-Out" strategy: Each time when a source case is removed from the case base we test to see if the number of successful retrievals on the case bases for all of the target cases are increased. If removing a source case decreases the number of successful retrievals, it will be restored back to the case base as it may contribute to successful retrievals for certain types of cases. Otherwise if the number of successful retrievals increases or does not change, it will be removed from the case base as a redundant case, or because it may not be a representative case for a specific type of cases. The process stops when the highest number of retrievals is obtained on all the target cases. The process is presented at the right part of the screenshot in Figure 7-1.

After the second stage of training, finally there are 14 and 15 source cases left in the original two case bases, respectively. To test the system performance, an experiment is carried out on both the initial and trained case bases for another set of target cases that are, of course, not the same as those of the training set. The accuracies of the system performance on these case bases are shown in Table 7-1.

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

Table 7-1 Accuracies of system performance on initial and trained case bases

We can see that the second training process removes quite a lot of source cases that are redundant or that are harmful for the performance of the CBR system. With a smaller number of relevant source cases retained in the case bases, the system performance is improved to provide higher accuracies of predictions on heuristics.

7.2 Chapter Summary

This chapter presents the first step of our work on a hyper-heuristic method using CBR for heuristic selection on course timetabling problems. Knowledge discovery techniques employ relatively simple methods and just a few training processes are carried out to obtain the good results. This approach is applicable of using the CBR as the heuristic selector for guiding the problem solving using previous experience. It may provide potential benefits in course timetabling when a good solution is needed within a limited time. Further work may to be carried out to fulfil the possible advantages of employing knowledge discovery techniques in the course timetabling domain.

Chapter 8 Conclusions and Future Work

This thesis investigates the CBR for solving course timetabling problems. The overall objective is to study how CBR can help to solve this type of scheduling problem by reusing previous knowledge collected and stored in a case base. Mainly in two ways, by reusing good quality solutions, and by reusing good heuristics, CBR will help solving the course timetabling problem. We will conclude these in the following sections, which are followed by future work on using CBR for timetabling problems.

8.1 Summary of the Structured CBR

8.1.1 Structured Representation in CBR

Structured representation has attracted more attention along with the wider and more complicated application areas being conducted in CBR research. The approach investigated in this thesis showed some potential benefits that can be obtained through the structured representation in course timetabling by attribute graphs.

The similarity measure considers the actions need to be taken in the adaptation thus is adaptation targeted. Different costs are associated with necessary actions in adaptation and recorded in similarity measure according to their difficulties. Adaptations are associated with the matching information on constraints of the retrieved cases and the new case, which is provided by (sub-)graph isomorphism. Based on the retrieval and similarity measure used in the structured CBR, good scheduling structures within previous problem solutions with similar constraints can be retrieved and reused as the components of the starting point, contributing high quality schedules in solving similar new course timetabling problems quickly.

The work presented in this thesis in course timetabling is trying to provide a CBR mechanism that can be easily adopted 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 type of constraint satisfaction problems.

8.1.2 Multiple-Retrieval Approach

In Case-based Scheduling, complex problems usually need to borrow several previous schedules, each of which contributes different parts of the problem. The multiple-retrieval CBR approach conducted here partitions the large timetabling problems on a small case base pre-built. One of the differences between many of the approaches investigated and the approach here is that the sub-problems are not partitioned by the rules but according to the source cases in the case base. This provides more flexibility when dealing with the complicated timetabling problems. Also once a sub-problem is partitioned, it is guaranteed that a sub-solution will always be obtained based on the corresponding source cases retrieved.

8.2 Summary on CBR as a Heuristic Selector

This thesis also presents the first step of the work on using CBR for heuristic selection on course timetabling problems. The results are good and indicate more potential on employing CBR within the hyper-heuristic approach in the course timetabling domain. Knowledge on what specific heuristics are good for solving which types of problems can be discovered and stored in the system. This on a higher level improves the generality of the problem solving, providing generally good heuristics quickly and avoiding the comparisons between different heuristics.

8.3 Future Work

8.3.1 Improving the Current Structured CBR Approach

8.3.1.1 Similarity Measure

When consider the similarity between the target problem and a set of cases, one of the problems in the current multiple-retrieval similarity measure is that sometimes it cannot precisely define the similarity due to some unexpected possibilities on characteristics in multiple cases. For example, during the multiple-retrieval process, the previously matched parts of the target case are combined into one super vertex and the adjacent edges between the matched and un-matched parts are combined with certain new attributes accordingly. This may in some way affect the accuracy of the similarity measure and thus the existing similarity measure needs to be refined in a more precise way.

8.3.1.2 Issues on Real World Course Timetabling Problems

A large number of experiments have been carried out on the current system concerning issues of time and space complexities. In real-world educational timetabling, it is well known that different institutions have their own specific requirements for course timetabling. Much of the research work carried out in the literature is aimed at problems in the authors' own institution. It is known that so far there is no exclusive set of real-world benchmark course timetabling problems available upon which to test our multiple-retrieval CBR system and compare it with other research results. We are currently putting together these benchmark course timetabling problems. They will be available at http://www.asap.cs.nott.ac.uk/themes/tt and the authors welcome further contribution from other timetabling researchers.

Much of the current work on course timetabling employs meta-heuristics methods and constraint logic programming, in which the problems are represented in quite different ways. Reformatting the real data will also form part of our future work.

8.3.2 Hybridisation within the Structured CBR Approach

Recent research in timetabling has reported many promising results by employing a variety of heuristic and meta-heuristic techniques, which are much flexible in solving a wide range of complex problems and thus is also adoptable to be hybridised to the CBR approach studied here.

This thesis presents some results in which the CBR approach works as the initialisation method for tabu search. Initialisation as one of the important factors on searching in meta-heuristic methods has formed an important research subject. Good initial solutions usually provide good starting points and save a significant amount of computing time. From the heuristic perspective, the multiple-retrieval CBR approach here fit well to provide good initial solutions embed good scheduling blocks. The future work will study more potential contributions of the hybridisation in solving more general complex timetabling problems.

The retrieval phase that finds the (sub-)graph isomorphism can be seen as a searching process, which the meta-heuristic methods are good at and may potentially be beneficial. Some state-search approaches such as Meta-heuristic methods (Williams, Wilson and Hancock, 1999) and

Memetic Algorithms (Cross, Myers and Hancock, 2000) are recently studied for graph matching in research. Investigation may be carried out on using meta-heuristic method to search for matching parts between the attribute graphs. The case base then can be organised into a flat structure to store a list of attribute graphs and the size of it will grow linearly thus reducing the storage complexity. However, the required searching time of the meta-heuristic for (sub-) graph isomorphism might be increased and it is not guaranteed the meta-heuristic will find all of the good matched (sub-)structures.

8.3.3 CBR as the Heuristic Selector

The approach that uses CBR as the heuristics selector presented in the thesis employed relatively simple techniques in knowledge discovery. There are many more complex and elaborate techniques that can be investigated and integrated into the CBR system to improve its performance. For example, for the case representation we currently use a simple technique that is manually carried out to choose the features and adjust their weights. This can be seen as a feature selection task, which is the problem of selecting a set of features to be used as the data format in the system to achieve high accuracy of prediction. Feature selection is an important issue in machine learning (Hall and Smith, 1996) for which a variety of traditional techniques exist (e.g. wrappers Kohavi and John, 1997; relief Kononenko, 1994 models). Some recent work employing AI methods such as hill climbing (Caruana and Freitag, 1994) and evolutionary algorithms (Freitas, 2002) to optimise the feature selection also provide a wider range of possible research. For complex timetabling problems, these more efficient algorithms can be employed to carry out the searching on features more effectively when dealing with larger data sets. Our future work will study and compare these different techniques to optimise the case representation and improve the system performance on wider range of larger timetabling problems. New features are being studied and introduced into the system. For example, some refined features such as the number of rooms with a range of capacities, the number of courses with more than a certain number of constraints, etc can be introduced to give a more specific description of problems. Other issues of knowledge discovery in the CBR system may include how to deal with the incomplete data in case bases and how to involve the domain knowledge in the system. User interaction in knowledge discovery is also important on tasks like judgment and decision-making, for which human usually do better than a machine.

The current system uses 5 simple heuristic to implement the analysis and testing on the case-based heuristic selection. Future work will study more heuristics in the system. Also the testing cases are artificially produced to give a systematic analysis on as many possible types of problem as possible. After the initial study of using CBR as a heuristic selector we have increased our understanding of the area. Real-world benchmark timetabling data (such as that presented in Carter and Laporte, 1996) will be collected and stored in the case base for solving real-world problems. Adaptation may also need to be conducted to utilize domain knowledge on some of the heuristics retrieved for the new problem.

The knowledge discovery techniques we studied in choosing heuristics may also be employed to discover knowledge in the search space that may guide the search towards a more promising region in problem solving using a variety of AI methods. In the Case-Based heuristic selection presented in this thesis, knowledge of what specific heuristic work well on what types of problems is modelled as cases. It is also possible that the knowledge of heuristics worked well during the problem solving within a particular periods of problem solving can be modelled and memorized in a case base and suggests heuristics during the problem solving process by employing heuristics which worked well on previous similar situations. Our future work will also investigate more complicated hyper-heuristic methods using CBR for general timetabling problems.

References

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

ABDENNADHER, S. and MARTE, M. (2000) University course timetabling using constraint handling rules. *Journal of Applied Artificial Intelligence*, **14**(4), 311 – 326.

ABRAMSON, D. (1991) Constructing school timetables using simulated annealing: sequential and parallel algorithms. *Management Science*, **37**, 98 – 113.

ABRAMSON, D. and ABELA, J. (1992) A parallel genetic algorithm for solving school timetabling problem. *The fifteenth Australian Computer Science Conference*, 1 – 11.

ABRAMSON, D.A. DANG, H. and KRISNAMOORTHY, M. (1999) Simulated annealing cooling schedules for the school timetabling problem, *Asia-Pacific Journal of Operational Research*, **16,** 1 – 22.

ALVAREZ-VALDES, R. CRESPO, E. and TAMARIT, J.M. (2002) Design and implementation of a course scheduling system using tabu search. *EJOR*, **137**, 512 – 523.

ARNOLD, O. and JANKE, K. (1994) Therapy plans as hierarchically structured graphs. *The Fifth International Workshop on Graph Grammars and Their Applications to Computer Science*. Virginia, USA.

BANKS, D. BEEL, P. and MEISLES (1998) A heuristic incremental modelling approach to course timetabling. *Proceedings of the Canadian Conference on Artificial Intelligence*, 16 – 29.

BARDDADYM, V. (1995) Computer-aided school and university timetabling: the new wave. In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 22 – 45, Lecture Notes in Computer Science 1153, Springer-Verlag: Berlin.

BERGER, J. SASSI, M. and SALOIS, S. (1999) A hybrid genetic algorithm for the vehicle routing problem with windows and itinerary constraints. *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, Morgan Kaufmann, 44 – 51.

BERGMANN, R. and WILKE, W. (1995) Building and refining abstract planning cases by change of representation language. *Journal of Artificial Intelligence Research*, **3**, 53 – 118.

BERGMANN, R. M-AVILA, H. VELOSO, M. and MELIS E. (1998) Case-based reasoning applied to planning. In: LENZ, M. BARTSCH, B. BURARD, H-D.

and WESS, S. (eds.) *Case-Based Reasoning Technology From Foundation to Applications*, pp. 169 – 200, Lecture Notes in Artificial Intelligence 1400, Springer-Verlag.

BERGMANN, R. and WILKE, W. (1996) On the role of abstraction in CBR. *European Workshop on Case-Based Reasoning*. Lecture Notes on Computer Science 1168, 28 – 43.

BEZIRGAN, A. (1993) A case-based approach to scheduling constraints. In: DORN, J. and FROESCHL, K.A. (eds.) 48 – 60. *Scheduling of Production Processes*. Ellis Horwood Limited.

BLANCO, J.J. and KHATIB, L. (1998) Course scheduling as a constraint satisfaction problem. *Proceedings of PACT98*.

BOIZUMAULT, P. Delon, Y. and PERIDY, L. (1996) Constraint logic programming for examination timetabling. *The Journal of Logic Programming*, **26**, 217 – 233.

BÖRNER, K. PIPPIG, E. TAMMER, E.C. COULON, C.H. and (1996) Structural similarity and adaptation. In: SMITH, I. and FALTINGS, B. (eds.) *Advances in Case-based Reasoning*, 58 – 75. Springer-Verlag, Switzerland.

BRAILSFORD, S.C. POTTS, C.N. and SMITH, B.M. (1999) Constraint satisfaction problems: algorithms and applications. *EJOR*, **119**, 557 – 581.

BRELAZ, D. (1979) New methods to colour the vertices of a graph. *Communication of the Association for Computing Machinery*, **22**, 251 – 256.

BULLNHEIMER, B. (1997) An examination scheduling model to maximize students' study time. In: BURKE, E.K. and CARTER W. (eds.) *The Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference*, 78 – 91. Lecture Notes in Computer Science 1408. Springer-Verlag: Berlin.

BURKE, E.K. ELLIMAN, D.G. and WEARE, R.F. (1994) A university timetabling system based on graph colouring and constraint manipulation. *Journal of Research on Computing in Education*, **27**, 1 – 18.

BURKE, E.K. NEWELL J.P. and WEARE, R.F. (1995) A memetic algorithm for university exam timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*. 241 – 250. Lecture Notes in Computer Science 1153. Springer-Verlag: Berlin.

BURKE, E.K. and ROSS, P. (eds.) (1995) *The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

BURKE, E.K. and CARTER, M. (eds.) (1997) *The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference*. Lecture Notes In Computer Science 1408. Springer-Verlag, Berlin.

BURKE, E.K. JACKSON, K.S. KINGSTON, J.H. and WEARE, R.F. (1997) Automated timetabling: the sate of the art. *The Computer Journal*, **40**(9), 565 – 571.

BURKE, E.K. and NEWALL, J. (1998) Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation,* **6**(1), 81 – 103.

BURKE, E.K. NEWALL, J. and WEARE, R. (1998) A simple heuristically guided search for the timetable problem. *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems (EIS'98)*, 574 – 579.

BURKE, E.K. and NEWALL, J.P. (1999) A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, **3**, 63 – 74.

BURKE, E.K. and ERBEN, W. (eds.) (2000) *The Practice and Theory of Automated Timetabling: Selected Papers from the Third International Conference*. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin.

BURKE, E.K. MACCARTHY, B. PETROVIC, S. and QU, R. (2000) Structured cases in CBR – re-using and adapting cases for timetabling problems. *Journal of Knowledge-based System*, **13**(2-3), 159 – 165. (Also published in ES’99 as one of the best technique papers).

BURKE, E.K. MACCARTHY, B.L. PETROVIC, S. and QU, R. (2001a) Case-based reasoning in course timetabling: an attribute graph approach. In: AHA, D.W. and WATSON, I. (eds.) *Case-Based Reasoning Research and Development. Proceedings of the Fourth International Conference on Case-Based Reasoning* (ICCBR2001). 90 – 104. Lecture Notes in Artificial Intelligence 2080. Vancouver, Canada.

BURKE, E.K. MACCARTHY, B.L. PETROVIC, S. and QU, R. (2001b). A multiple-retrieval case-based reasoning system for course timetabling problems. Technical Report NOTTCS-TR-2001-7. School of Computer Science and Information Technology, University of Nottingham.

BURKE, E.K. MACCARTHY, B.L. PETROVIC, S. and QU, R. (2002). Knowledge discovery in a hyper-heuristic for course timetabling using case-based reasoning. In: BURKE, E.K. and CAUSMAECKER, P. (eds.) *Proceedings of the Fourth International Conference on the Practice and Theory of Automated Timetabling*.

BURKE, E.K. and CAUSMAECKER, P. (eds.) (2002) *Proceedings of the Fourth International Conference on the Practice and Theory of Automated Timetabling.*

BURKE, E.K. and PETROVIC, S. (2002) Recent research directions in automated timetabling. *European Journal of Operational Research* **140**(2), 266 – 280.

CANGALOVIC, M. KOVACEVIC-VUJCIC, V. IVANOVIC, L. and DRAZIC, M. (1998) Modelling and solving a real-life assignment problem at universities. *EJOR*, **110**, 223 – 233.

CARTER, M.W. (1989) A Langrangian relaxation approach to the classroom assignment problem. *INFOR*, **2**, 230 – 246.

CARTER, M.W. and LAPORTE, G. (1995) Recent developments in practical examination timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the First*

International Conference. 3 – 21. Lecture Notes in Computer Science 1153. Springer-Verlag: Berlin.

CARTER, M.W. and LAPORTE, G. (1996) Examination timetabling: algorithmic Strategies and applications, *Journal of the Operational Research Society*, **74**, 373 – 383.

CARTER, M.W. and LAPORTE, G. (1997). Recent developments in practical course timetabling. In: BURKE, E.K. and CARTER W. (eds.) *The Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference*, 3 – 19. Lecture Notes in Computer Science 1408. Springer-Verlag: Berlin.

CARTER M.W. (2000). A comprehensive course timetabling and student scheduling system at the university of waterloo. In: BURKE, E.K. and ERBEN, W. (eds.) *The Practice and Theory of Automated Timetabling: Selected papers from the third International Conference*, 64 – 82. Lecture Notes in Computer Science 2079. Springer-Verlag: Berlin.

CARUANA, R. and FREITAG, D. (1994) Greedy attribute selection. In: *Proceedings of the International Conference on Machine Learning*, 28 – 36.

CHENG, C., KANG, L., LEUNG, N. and WHITE, G.M. (1995) Investigations of a constraint logic programming approach to university timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*. 112 – 129. Lecture Notes in Computer Science 1153. Springer-Verlag: Berlin.

COLORNI, A. DORIGO, M. and MANIEZZO, V. (1998) Metaheuristics for school timetabling. *Computational Optimisation and Applications*, **9**(3), 277 – 298.

CORNE, D. ROSS, P. and FANG, H.L. (1994) Evolutionary timetabling: practice, prospects and work in progress. *Proceedings of UK Planning and Scheduling SIG Workshop,*

CORNE, D. and ROSS, P. (1995) Peckish initialisation strategies for evolutionary timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected papers from the First International Conference*. 227 – 240. Lecture Notes in Computer Science 1153. Springer-Verlag: Berlin.

COSTA, D. (1994) A tabu search for computing an operational timetable. *EJOR,* **76,** 98 – 110.

COWLING, P. KENDALL, G. and SOUBEIGA, E. (2000) A hyperheuristic approach to scheduling a sales summit. In: BURKE, E.K. and ERBEN, W. (eds.) *Proceedings of the Third International Conference on the Practice and Theory of Automated Timetabling*. 176 – 190.

COWLING, P. KENDALL, G. and SOUBEIGA, E. (2001) Hyperheuristic: a tool for rapid prototyping in scheduling and optimisation. *Second European Conference on Evolutionary Computing for Combinatorial Optimisation (EvoCop 2002),* 1 – 10.

COWLING, P. KENDALL, G. and HAN, L. (2002) An investigation of a hyperheuristic genetic algorithm to a training scheduling problem. *Proceedings* *of 2002 World Congress on Computational Intelligence (CEC 2002)*, 1185 – 1190.

CROSS, A.D.J. MYERS, R. and HANCOCK, E.R. (2000) Convergence of a hill-climbing genetic algorithm for graph matching. *Pattern Recognition*, **33**, 1863 –1880.

CUNNINGHAM, P. and SMYTH, B. (1997) Case-based reasoning in scheduling: reusing solution components. *The International Journal of Production Research*. **35**, 2947 – 2961.

CUNNINGHAM, P. and BONZANO, A. (1999) Knowledge engineering issues in developing a case-based reasoning application. *Knowledge-Based Systems*, **12**, 371 – 379.

DAVID, P. (1997) In: BURKE, E.K. and CARTER, M.W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference*, 169 – 186. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

DERIS, S. OMATU, S. and OHTA, H. (2000) Timetable planning using the constraint-based reasoning. *Computers & Operations Research*, **27**, 819 – 840.

DERIS, S. OMATU, S. OHTA, H. and SAAD, P. (1999) Incorporating constraint propagation in genetic algorithm for university timetable planning. *Engineering Applications of Artificial Intelligence*, **12**, 241 – 253.

DORN, J. (1995) Case-based reactive scheduling. Technical Report CD-TR 95/75.

DOWSLAND, K.A. (1997) Off-the-peg or made to measure? Timetabling and scheduling with SA and TS, In: BURKE E.K. and CARTER M.W. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference*, 37 – 52. Lecture Notes in Computer Science 1408. Springer-Verlag: Berlin.

ELMOHAMED, M.A.S. CODDINGTON, P. and FOX, G. (1997) A comparison of annealing techniques for academic course timetabling. In: BURKE E.K. and CARTER M.W. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference*. 92 – 112. Lecture Notes in Computer Science 1408, Springer-Verlag: Berlin.

ERBEN, W. (2000) A grouping genetic algorithm for graph colouring and exam timetabling. In: BURKE, E.K. and ERBEN, W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Third International Conference,* 132 – 156. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin.

FAHRION, R. and DOLLANSKY, G. (1992) Construction of university faculty timetables using logic programming. *Discrete Applied Mathematics*. **35**(3), 221 – 236.

FANG, H.L., ROSS P. and CORNE, D. (1994) A promising hybrid GA/heuristic approach for open-shop scheduling problems. *The Eleventh European Conference on Artificial Intelligence (ECAI'94)*. John Wiley & Sons, Ltd.

FAYYAD, U., PIATETSKY-SHAPIRO, G., SMYTH, P. (1996) From data mining to knowledge discovery in databases. In: FAYYAD, U., PIATETSKY-SHAPIRO, G., SMYTH, P. and UTHURUSAMY, R. (eds.):

Advances in Knowledge Discovery and Data Mining. AAAI Press, Melo Park, CA, 1 – 34.

FOULDS, L.R. and JOHNSON, D.G. (2000) SlotManager: a microcomputer-based decision support system for university timetabling. *Decision Support Systems*, **27**, 307 – 381.

FREITAS, A. (2002) A survey of evolutionary algorithms for data mining and knowledge discovery. To appear in: GHOSH, A. and TSUTSUI, S. (eds.): *Advances in Evolutionary Computation*, Springer.

FUCHS, B. MILLE, A and CHIRON, B. (1996) Using explanations to guide adaptation. *Proceedings of Workshop on Adaptation in Case-Based Reasoning, ECAI96*.

GAREY, M.R. and JOHNSON, D.S. (1979) *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Company. New York.

GARZA, A.G.S. and MAHER, M.L. (1999) An evolutionary approach to case adaptation. *Proceedings of the Third International Conference on Case-Based Reasoning*.

GASPERO, L.D. and SCHAERF, A. (2000) Tabu search techniques for examination timetabling. In: BURKE, E.K. and ERBEN, W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Third International Conference*, 104 – 117. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin.

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

GEBHARDT, F. (1997) Survey on structure-based case retrieval. *The Knowledge Engineering Review*, **12**(1), 41 – 58.

GLOVER, F. and LAGUNA, M. (1993) Tabu search. In: REEVES, C.R. (eds.) *Modern Heuristic Techniques for Combinational Problems*. Scientific Publications, Oxford.

GOLDING, A.R. and ROSEBLOOM, P.S. (1997) Improving rule-based systems through case-based reasoning. 22 – 27. *Proceedings of the Ninth National Conference on Artificial Intelligence*.

GOLTZ, H.J. (2000) On methods of constraint-based timetabling. *Proceedings of PACLP'2000*, 167 – 177.

GROLIMUND, S. and GANASCIA, J. (1997) Driving tabu search with case-based reasoning. *EJOR*. **103**, 326 – 338.

GUERET, G. JUSSIEN, N. BOIZUMAULT, P. and PRINS, C. (1995) Building university timetables using constraint logic programming. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 130 – 145. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

GUNADHI, H. ANAND, V.J. and YONG, Y.W. (1996) Automated timetabling using an object-oriented scheduler. *Expert Systems with Applications*, **10**(2), 243 – 256.

HALL, M.A. and SMITH, L. (1996) Practical feature subset selection machine learning. *Proceedings of the Australian Computer Science Conference*.

HAMMOND, K.J. (1990) Case-based planning: A framework for planning from experience. *Cognitive Science*, **14**(3), 385 – 443.

HART, E. ROSS, P. and NELSON, J. (1998) Solving a real-world problem using an evolving heuristically driven schedule. *Evolutionary Computation*, **6**, 61 – 80.

HENNESSY, D. and HINKLE, D. (1992) Applying case-based reasoning to autoclave loading. *IEEE Expert*, **7**, 21 – 26.

HENZ, M. and WURTZ, J. (1995) Using oz for college timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 162 – 180. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

HERTZ, A. (1991) Tabu search for large scale timetabling problems. *EJOR*, **54**, 39 – 47.

HUNT, J. and MILES, R. (1994) Hybrid case-based reasoning. *The Knowledge Engineering Review*, **9**(4), 383 – 397.

JANTKE, K.P. (1993) Nonstandard concepts of similarity in case-based reasoning. *Proceedings of the Seventeenth Annual Conference of the GfKI*, Springer-Verlag, Kaiderslautern.

JOHN, G.H. KOHAVI, R. and PFLEGER, K. (1994) Irrelevant features and the subset selection problem. *Proceedings of the Eleventh International Conference on Machine Learning*, 121 – 129.

KAMBI, M. and GILBER, D. (1996) Timetabling in constraint logic programming. *Proceedings of the INAP-96: Symposium and Exhibition on Industrial Applications of Prolog*, 79 – 88.

KANG, L. and WHITE, G.M. (1992) A logic approach to the resolution of constraints in timetabling. *EJOR*, **61**, 306 – 317.

KIRKPATRICK, S. GELLAT, JCD and VECCI, MP (1983) Optimisation by Simulated Annealing. *Science*, **220**, 671 – 680.

KOHAVI, R. and JOHN, G.H. (1997) Wrappers for feature subset selection. *AI*, **97**(1-2), 273 – 324.

KOLODNER, J.L. (1993) *Case-Based Reasoning*. Morgan Kaufmann.

KOLODNER, J.L. and LEAKE, D (1996) A tutorial introduction to case-based reasoning. In: LEAKE, D. (ed.) *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, 31 – 65. AAAI Press/The MIT Press.

KONG, S.C. and KWOK, L.F. (1999) A conceptual model of knowledge-based timetabling system. *Knowledge-Based Systems*, **12**, 81 – 93.

KONONENKO, I. (1994) Estimating attributes: analysis and extensions of relief. *Proceedings of the Seventh European Conference on Machine Learning*, 171 – 182.

KOTON, P. (1989) SMARTlan: A case-based resource allocation and scheduling system. *Proceedings of Workshop on Case-Based Reasoning* (DARPA), 285 – 289.

KUSIAK, A. and CHEN, M. (1998) Expert systems in planning and scheduling manufacturing systems. *EJOR*, **3**, 113 – 130.

LAJOS, G. (1995) Complete university modular timetabling using constraint logic programming. In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 146 – 161. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

LEAKE, D. (1995) Combining rules and cases to learn case adaptation. *Proceedings of the Seventh Annual Conference of Cognitive Science Society*.

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

LOUIS, S.J. and LI, G. (2000) Case injected genetic algorithms for travelling salesman problems. *Information Sciences*, **122**, 201 – 225.

MACCARTHY, B.L. and JOU, P. (1995) A case-based expert system for scheduling problems with sequence dependent set up times. In: ADEY, R.A. and RZEVSKI, G. (eds.) *Applications of Artificial Intelligence in Engineering X*. 89 – 96. Computational Machines Publications, Southampton.

MACCARTHY, B.L. 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, 1996) 211 – 218.

MACCARTHY, B.L. and YE, R. (1997) Intelligent planning and scheduling: the state of the art. *The Third International Symposium on Logistics*, 275 – 280. SGE publications Padova.

MACCARTHY, B.L. and WILSON, J.R. (eds.) (2001) *Human Performance in Planning and Scheduling: Fieldwork Studies, Methodologies and Research Issues*. Taylor and Francis, London.

MACEDO, L., PEREIRA, F.C., GRILO, C. and CARDOSO, A. (1996) Plans as structured networks of hierarchically and temporally related case pieces. *The Third European Workshop on Case-Based Reasoning*. 234 – 248, Lecture Notes on Artificial Intelligence 1168.

MANTARAS, R.L. and PLAZA, E. (1997) Case-based reasoning: an overview. *AI Communications*, **10**, 21 – 29.

MAREFAT, M. (1997) Case-based process planning using an object oriented model representation. *Robotics and Computer Integrated Manufacturing*, **13**(3), 229 – 251.

MARIR, F. and WATSON, I. (1994) Case-based reasoning: a categorised bibliography. *The Knowledge Engineering Review*, **9**(4), 355 – 381.

MARIR, F. and WATSON, I.D. (1995) Representation and indexing building refurbishment cases for multiple retrieval of adaptable pieces of cases. In: MATUSCHEK, D. and JANTKE, K.P. (1997) Axiomatic characterizations of structural similarity for case-based reasoning. *FLAIRS-97*, 432 – 436.

MCKENNA, E and SMYTH, B. (1998) A competence model for case-based reasoning. *Proceedings of the Ninth Irish Conference on Artificial Intelligence and Cognitive Science.*

MEISELS, A. and GUDES, E. and SOLOTOREVSKY, G. (1995) Employee Timetabling, Constraint Networks and Knowledge-Based Rules: A Mixed Approach. In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the First International Conference,* 93 – 105. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

MEISELS, A. and LUSTERNIK, N. (1997) Experiments on networks of employee timetabling problems. In: BURKE, E.K. and CARTER, M.W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference,* 130 – 141. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

MESSMER, B.T. (1995) Efficient graph matching algorithms for preprocessed model graph. PhD thesis, University of Bern, Switzerland.

MELICIO, F. CALDEIRA, J.P. and ROSA, A.C. (1998) Timetabling implementation aspects by Simulated Annealing. In: GU J. (ed.) *IEEE - Systems Science and Systems Engineering*, 553 – 557.

MIYASHITA, K. and SYCARA, K. (1994) Adaptive case-based control of scheduling revision. In: ZWEBEN, M. and FOX, M.S. (eds.) *Intelligent Scheduling*, 291 – 308.

MIYASHITA, K. and SYCARA, K. (1995) CABINS: A framework of knowledge acquisition and iterative revision for schedule improvement and reactive repair. *AI*, **76**, 377 – 426.

Nonobe K. and Ibaraki T. (1998) A tabu search approach to the constraint satisfaction problem as a general problem solver, *European Journal of Operational Research*, **106**(2-3), 599 – 623.

NORONHA, S.J. and SARMA, V.V.S. (1991) Knowledge-based approaches for scheduling system in a job shop. *IEEE Transactions on Knowledge and Data Engineering*, **3**(2), 160 – 171.

OMAN, S. and CUNNINGHAM, P. (2001) Using case retrieval to seed genetic algorithms. *International Journal of Computational Intelligence and Applications*, **1**(1), 71 – 82.

PAECHTER, B. CUMMING, A. and LUCHIAN, H. (1995) The use of local search suggestion lists for improving the solution of timetable problems with evolutionary algorithms. In: GOOS, G. HARTMANIS, J. and LEEUWEN, J. (eds.) *Evolutionary Computation, AISB Workshop*, 86 – 93. Lecture Notes in Computer Science 993. Springer-Verlag, Sheffield.

PAECHTER, B. RANKIN, R.C. and CUMMING, A. (1997) Improving a lecture timetabling system for university-wide use. In: BURKE, E.K. and CARTER, M.W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference,* 156 – 168. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

PAUL, J. (1993) Building expert systems for knowledge-poor domains. *Proceedings of Expert Systems 93*, 223 - 233. Cambridge.

PETROVIC, S. BEDDOE G.R. and BERGHE G.V. (2002) Storing and adapting repair experiences in personnel rostering. In: BURKE, E.K. and CAUSMAECKER, P. (eds.) *Proceedings of the Fourth International Conference on the Practice and Theory of Automated Timetabling*. 185 – 186.

PRAEHOFER, H. and KERSCHBAUMMAYR, J. (1999) Case-based reasoning to support reusability in a requirement engineering and system design tool. *Engineering Applications of Artificial Intelligence*, **12**, 717 – 731.

PIATETSKY-SHAPIRO, G. (1991) *Knowledge Discovery in Databases*. AAAI Press.

PINEDO, M. (1995) *Scheduling Theory, Algorithms and Systems*. Prentice Hall.

PLAZA, E. (2000) The ABC of adaptation: towards a software architecture for adaptation-centred case-based reasoning system. *Foundations of Intelligent Systems, Twelfth International Symposium, ISMIS.*

PRASAD, B. (1995) Planning with case-based structures. In: AHA, D. JONES, E. LEAKE, D. and RAM A. (eds.) *Proceedings of the American Association for Artificial Intelligence (AAAI) Fall Symposium on Adaptation of Knowledge For Reuse*, MIT, Cambridge, USA.

PURVIS, L. and ATHALYE, S. (1997) Towards improving case adaptability with a genetic algorithm. *Proceedings of the Second International Conference on Case-Based Reasoning*. 403 – 412.

PURVIS, L. and PU, P. (1995) Adaptation using constraint satisfaction techniques. In: VELOSO, M.M. and AAMODT, A. (eds.) *Case-Based Reasoning Research and Development, Proceedings of the First International Conference on Case-Based Reasoning, ICCBR-95,* 289 – 300, Lecture Notes in Computer Science 1010. Springer.

QUINLAN, J.R. (1986) Induction of decision trees. *Machine Learning*, **1**, 81 – 106.

RADCLIFFE, N.J. and SURREY, P.D. (1994) Formal memetic algorithms. In FORGARTY, T.C. (ed.): *Lecture Notes in Computer Science 865*, 1 – 16.

RANDHAWA, S.U. and MACDOWELL, E.D. (1990) An investigation of the applicability of expert system to job shop scheduling. *International Journal on Man-Machine Studies*, **32**, 203 – 213.

RANKIN, R.C. (1995) Automatic timetabling in practice. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 266 – 282. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

REEVES, C.R. (1996) Modern heuristic techniques. *Modern Heuristic Search Methods*. John Willey & Sons Ltd.

RICCI, F. and SENTER, L. (1998) Structured cases, trees and efficient retrieval. *Proceedings of the Fourth European Workshop on Case-based Reasoning*. Springer-Verlag, Dublin.

RICH, D.C. (1995) A smart genetic algorithm for university timetabling. In: BURKE, E.K. and ROSS, P. (eds.) *The Practice and Theory of Automated*

Timetabling: Selected Papers from the First International Conference, 181 – 197. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

ROBERT, V. and HERTZ, A. (1995) How to decompose constrained course scheduling problems into easier assignment type subproblems. In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated Timetabling: Selected Papers from the First International Conference*, 364 – 373. Lecture Notes in Computer Science 1153. Springer-Verlag, Berlin.

ROSS, P. and CORNE, D. (1995) Comparing genetic algorithms, simulated annealing, and stochastic hillclimbing on timetabling problems. In: GOOS, G. HARTMANIS, J. and LEEUWEN, J. (eds.) *Evolutionary Computation, AISB Workshop*, 94 – 102. Lecture Notes in Computer Science 993. Springer-Verlag, Sheffield.

ROSS, P. HART, E. and CORNE, D. (1997) Some observations about GA-based exam timetabling. In: BURKE, E.K. and CARTER, M.W. (eds.) *Practice and Theory of Automated Timetabling: Selected Papers from the Second International Conference,* 115 – 129. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

SANDERS, K.E. KETTLER, B.P. and HENDLER, J.A. (1997) The case for graph-structured representations. *Proceedings of the Second International Conference on Case-Based Reasoning*. Springer-Verlag, Berlin.

SCHAERF, A. (1996) Tabu search for large high-school timetabling problems. *IEEE Transactions on Systems, Man, and Cybernetics*, **29**(4), 368 – 377.

SCHAERF, A. (1999) A survey of automated timetabling. *Artificial Intelligence Review*, **13**, 87 – 127.

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

SCHREUDER, JAN A.M. (1997) Historical developments, present situation and future perspectives on sports timetabling. *Proceedings of the Second International Conference on the Practice and Theory of Automated Timetabling*. 353 – 357. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

SCOTT S, SIMPSON, R. (1998) Case-based incorporating scheduling constraint dimensions – experiences in nurse rostering. In: SMYTH, B. and CUNNINGHAM, P. (eds.) *Advances in Case-Based Reasoning. The Fourth European Workshop on Case-Based Reasoning*. Lecture Notes on Artificial Intelligence 1488. 392 – 401.

SHAW, P. (1998) Using constraint programming and local search methods to solve vehicle routing problems. *Proceedings of CP-98*. 417 – 431.

SHIN, K.-S. and HAN, I. (1999) Case-based reasoning supported by genetic algorithm for corporate bonding rating. *Expert Systems with Applications*. **16**, 85 – 95.

SMITH, D.E., FRANK, J. and JONSSON, A.K. (2000) Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review*. **15**(1), 61 – 94.

SMITH, S. (2001) Is scheduling a solved problem? Special Session *“The Next 10 Years of Scheduling Research”* on *Genetic and Evolutionary Computation Conference 2001*. 116 – 120.

SMITH, S.F. (1994) OPIS: An architecture and methodology for reactive scheduling. In ZWEBEN, M. and FOX, M.S. (eds.) *Intelligent Scheduling*. Morgan Kaufmann Publishers, Inc.

SMYTH, B. and KEANE, M.L. (1998) Adaptation-guided retrieval: questioning the similarity assumption in reasoning. *Artificial Intelligence*, **102**, 249 – 293.

SMYTH, B. CUNNINGHAM, P. and KEANE, M. (2001) Hierarchical case-based reasoning. *IEEE Transactions on Knowledge and Data Engineering*, **13**, 793 – 812.

SQALLI, M.H. PURVIS, L. and FREUDER, E.C. (1999) Survey of applications integrating constraint satisfaction and case-based reasoning. *Proceedings of the First International Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming*.

STAMATOPOULOS, P. VIGLAS, E. and KARABOYAS, S. (1998) Nearly optimum timetable construction through CLP and intelligent search. *International Journal on Artificial Intelligence Tools*, **7**(4), 415 – 442.

SURMA, J. and VANHOOF, K. (1998) Integrating rules and cases for the classification task. *Proceedings of the AAAI’98 Spring Symposium on Multimodel Reasoning*, 130 – 136.

SZELKE, E. and KERR, R.M. (1994) Knowledge based reactive scheduling. *International Journal of Production Planning and Control*. **5**, 124 – 145.

SZELKE, E. and MARKUS, G. (1997) A learning reactive scheduler using CBR/L. *Computer Industry*, **33**, 31 – 46.

TAH, J.H.M., CARR, V. and HOWES, R. (1999) information Modelling for Case-Based Construction Planning of Highway Bridge Projects. *Advances in Engineering Software*, **30**, 495 – 509.

TERASHIMA-MARIN, H. ROSS, P. and VALENZUELA-RENDON, M. (1999) Evolution of constraint satisfaction strategies in examination timetabling. *Proceedings of the Genetic and Evolutionary Computation Conference 1999 (GECCO-99)*, Morgan Kaufmann, 635 – 642.

THOMPSON, J.M. and DOWSLAND, K.A. (1998) A robust simulated annealing based examination timetabling system. *Computers and Operations Research,* **25,** 637 – 648.

TRIPATHY, A. (1984) School timetabling **-** A case in large binary integer linear programming. *Management Science*, **30**, 1473 – 1489.

TSANG, E. MILLS, P. and WILLIAMS, R. (1999) A computer aided constraint programming system. *The First International Conference on the Practical Application of Constraint Technologies and Logic Programming*, 81 – 93.

VELOSO, M. MUNIOZ, H. and BERGMANN, R. (1996) Case-based planning: selected methods and systems. *AI Communications* **9**(3), 128 – 137.

VOUDOURIS, C. and TSANG, E.P.K. (1999) Guided local search and its application to the travelling salesman problem. *EJOR*, **113**(2), 469 – 499.

WATSON, I. and PERERA, S. (1998) A hierarchical case representation using context guided retrieval. *Knowledge-Based Systems,* **11**, 285 – 292.

WEARE, R.F. (1995) Automated examination timetabling. PhD dissertation, University of Nottingham, Department of Computer Science.

WERRA, D. (1985) Graphs, hypergraphs and timetabling. *Methods of Operations Research* (Germany F.R.), **49**, 201 – 213.

WHITE, G.M. and ZHANG, J. (1997) Generating complete university timetables by combining tabu search with constraint logic. In: BURKE, E.K. and CARTER, M.W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Second International Conference,* 187 – 200. Lecture Notes in Computer Science 1408. Springer-Verlag, Berlin.

WHITE, G.M. and XIE, B.S. (2000) Examination timetables and tabu search with long-term memory. In: BURKE, E.K. and ERBEN, W. (eds.) *Practice and Theory of Automated Timetabling: Selected papers from the Third International Conference,* 85 – 103. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin.

WILLIAMS, H.P. (1999) *Model Building in Mathematical Programming*. Wiley: Chichester.

WILLIAMS, M.L. WILSON, R.C. and HANCOCK, E.R. (1999) Deterministic search for relational graph matching. *Patter Recognition* **32**(7), 1255 – 1271.

WREN, V. (1995) Scheduling, timetabling and rostering – a special relationship? In: BURKE, E.K. and ROSS, P. (eds.) *Practice and Theory of Automated*

Timetabling: Selected Papers from the First International Conference 46 – 75. Lecture Notes in Computer Science 1153. Springer-Verlag: Berlin.

WREN, A. and ROUSSEAU, J.-M. (1995) Bus driver scheduling - an overview. In DADUNA, J.R. BRANCO, I. And PAIX'AO, J.M.P. (eds.) *Computer-Aided Transit Scheduling*, 173 – 187. Springer-Verlag.

YE, P. and HUGHES, J.G. (1994) A method for solving the job shop scheduling problem in the CLP paradigm. *Proceedings of Expert Systems 94*. Cambridge.

YOSHIKAWA, M. KANEKO, K. NOMURA, Y. and WATANABE, M. (1994) A constraint-based approach to high-school timetabling problems: a case study. *Proceedings of the twelfth National Conference on Artificial Intelligence (AAAI-94)*, 1111 – 1116.

ZERVOUDAKIS, K. and STAMATOPOULOS, P. (2000) A generic object-oriented constraint-based model for university course timetabling. In: BURKE, E.K. and ERBEN, W. (eds.) (2000) *The Practice and Theory of Automated Timetabling: Selected Papers from the Third International Conference*, 28 – 47. Lecture Notes in Computer Science 2079. Springer-Verlag, Berlin.

ZWEBEN, M. and FOX, M.S. (eds.) (1994) *Intelligent Scheduling*. Morgan Kaufmann Publishers, Inc.

Appendices

Appendix A: Penalties Between Mapped Attributes

The values in Table A 1 the give the penalties of mapping the nodes or the edges of labels on rows in source attribute graph with those in the target attribute graphs. Threshold is set as 1 to define if the nodes or edges are similar (values below 1) or not similar (values above 1).

Labels	**Weights**	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
0	0.5	0	0.7	0.9	0.8	/	/	/	/
1	0.6	0.4	0	1.2	1.2	/	/	/	/
2	0.7	0.1	0.7	0	0.8	/	/	/	/
3	0.4	0.1	0.5	0.8	0	/	/	/	/
4	0.4	/	/	/	/	0	0.7	0.5	1.2
5	0.6	/	/	/	/	0.7	0	1.2	1.2
6	0.4	/	/	/	/	0.7	1.2	0	1.2
7	0.9	/	/	/	/	0.7	0.8	0.7	0

Table A 1 Penalties of Mapping Attributes of Nodes and Edges

Appendix B: Course Timetabling Problems Specification

Hard constraints:

- A course is in conflict with another thus they can not be scheduled into the same timeslot
- A course should be carried out n times a week
- Each course has a specific room requirement with type and capacity
- Certain number of periods is given for each problem

Soft constraints:

- One course should be scheduled before or after another
- Inclusive/exclusive - a course should/should not be scheduled into a fixed timeslot
- Consecutive - a course should/should not be scheduled into a timeslot consecutive to that of another

Appendix C: Heuristics Used in the System

- LD – Largest degree first

  All the courses not yet scheduled are inserted into an "unscheduled list" in descending order according to the number of constraints the course has with the other courses. This heuristic tries to schedule the most difficult courses first.

- LDT – Largest degree first with tournament selection

It is similar with the LD except every time the most difficult course is selected from a subset of the "unscheduled list" by a percentage given. Here 30 percent is used to get a subset from the list. This heuristic tries to schedule the most difficult courses first but also give some randomness.

- HC – Hill climbing

An initial timetable is constructed randomly then is improved by hill climbing.

- CD – Colour degree

Courses in the "unscheduled list" are ordered by the number of constraints it has with those courses that are already scheduled in the timetable. Usually these courses left are more difficult to be scheduled than those with less number of constraints with already scheduled ones.

- SD – Saturation degree

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

Appendix D: Initial Features and Their Weights for Cases

f_0: number of hard constraints / number of events

f_1: number of soft constraints / number of events

f_2: number of constraints / number of events

f_3: number of periods / number of events

f_4: number of rooms / number of events

f_5: number of not consecutive courses / number of constraints

f_6: number of consecutive courses / number of constraints

f_7: number of hard constraints / number of constraints

f_8: number of soft constraints / number of constraints

f_9: number of hard constraints / number of periods

f_{10}: number of soft constraints / number of periods

normalized weight w_i = factor$_i$ * 1 / sum of weights of all the features

initial factor$_i$ = {1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1}

Appendix E: Trained Features and Their Weights

f_0: number of exclusive courses / number of events

f_1: number of inclusive courses / number of events

f_2: number of constraints / number of events

f_3: number of rooms / number of events

f_4: number of hard constraints / number of periods

f_5: number of not consecutive courses / number of constraints

normalized weight w_i = factor$_i$ * 1 / sum of weights of all the features

factor$_i$ = {45, 10, 10, 15, 30, 6}