CASE-BASED INITIALISATION OF METAHEURISTICS FOR EXAMINATION TIMETABLING

Sanja Petrovic, Yong Yang
School of Computer Science and Information Technology,
The University of Nottingham, Nottingham NG8 1BB, UK

Moshe Dror
University of Arizona, Tucson, AZ 85721, USA

Abstract Examination timetabling problems are traditionally solved by choosing a solution procedure from a plethora of heuristic algorithms based either on a direct construction principle or on some incremental improvement procedure. A number of hybrid approaches have also been examined in which a sequential heuristic and a metaheuristic are employed successively. As a rule, best results for a problem instance are obtained by implementing heuristics with domain-specific knowledge. However, solutions of this kind are not easily adoptable across different problem classes. In order to lessen the need for a problem-specific knowledge we developed a novel solution approach to examination timetabling by incorporating the case-based reasoning methodology. A solution to a given problem is constructed by implementing case-based reasoning to select a sequential heuristic, which produces a good initial solution for the Great Deluge metaheuristic. A series of computational experiments on benchmark problems were conducted which subsequently demonstrate that this approach gives comparable or better results than solutions generated not only by a single Great Deluge algorithm, but also the state-of-the-art approaches.

Keywords: case-based reasoning, heuristic selection, graph matching and retrieval.

1. INTRODUCTION

Examination timetabling problem is a difficult combinatorial optimisation problem. The task is to assign a set of examinations into a limited number of time periods and classrooms subject to constraints (Carter *et al.*, 1996). The constraints are usually divided into two categories: hard and soft constraints. Hard constraints are those that must not be violated. One such constraint is

that no student can attend two examinations at the same time. An examination timetable is considered to be feasible only when it meets all hard constraints. On the other hand, soft constraints are not essential to a timetable, but highly desirable. For example, students might wish to have a two or three day interval between two consecutive examinations. The quality of a timetable is measured in terms of its satisfaction of soft constraints. A good review of a variety of constraints that are usually imposed on examination timetabling is given in Burke *et al.* (1996).

A timetabling problem can be modelled by a graph where each vertex represents an examination while an edge represents a conflict between two examinations (e.g. two examinations have some students in common and therefore cannot be scheduled into the same period). Thus, a timetabling problem is analogous to a graph colouring problem when neglecting soft constraints and resource requirements, with colour-coding for time slots (Welsh *et al.*, 1967). The vertices of the graph have to be coloured in such a way so that no two adjacent vertices share the same colour. Note that finding a minimal number of colours to colour a graph is one of the classical NP-Complete problems (Garey and Johnson, 1977).

1.1 Examination Timetabling

Over the last 40 years, various approaches to examination timetabling have been developed. A number of review papers discuss approaches and research issues in examination timetabling (Burke and Petrovic, 2002; Carter, 1986; Carter and Laporte, 1996). Approaches based on applications of graph colouring heuristics for solving timetabling problems were widely employed in the early days of timetabling research (Carter, 1986; Foxley and Lockyer, 1968). The idea behind these heuristics is to schedule examinations sequentially commencing with the examinations estimated to be the most difficult to schedule and ending with the easiest ones. By the beginning of the 1990s, sequential heuristics had been superseded by various metaheuristics such as Tabu search and Simulated Annealing (SA), which take into consideration soft constraints and therefore produce more satisfactory solutions (Carter and Laporte, 1996).

Recently, there has been a growing interest in employing various sequential heuristics to generate initial solutions for metaheuristics. Saleh Elmohamed *et al.* (1998) used sequential heuristics which consider size of examinations to find a feasible solution and handled soft constraints by simulated annealing. Burke and Newell (1999) used sequential heuristics to decompose a large problem into several sub-problems, which were then solved by memetic algorithm. Di Gaspero and Schaerf (2001) designed sequential heuristics, which are hybridised with Tabu Search. The hybrid approaches (Burke and Newell 1999; Casey and Thompson, 2003; Merlot *et al.*, 2003; White *et al.*, 2004)

produced the best results on a number of benchmark problems, and represent the state of the art in timetabling. Sequential heuristics serve an important role in the successful subsequent implementation of metaheuristics because they cannot only shorten the search time but may also greatly enhance their performance (Burke *et al.*, 1998; Burke and Newell, 2002).

However, a successful development of such a metaheuristic is a difficult task since it usually involves incorporation of problem domain-specific knowledge. For example in a simulated annealing timetabling algorithm (Merlot *et al.*, 2003), a sophisticated neighbourhood structure (such as the Kempe chains), and an appropriate cooling schedule, which involves choosing a cooling formula and setting values for parameters such as initial temperature and cooling factor, have to be defined. Similarly, a Tabu Search timetabling algorithm (White *et al.*, 2004) requires an appropriate setting of parameters such as the length of the tabu list, the stopping criteria, and a candidate list strategy to restrict the neighbourhood size. Generally, the current approaches suffer from limitation in their applicability when faced with changes in problem description.

It is well known in the timetabling community that a solution procedure which generates good results at one university might perform poorly for timetabling problems in another university (Carter and Laporte, 1996). Naturally, the following question arises: which sequential heuristic should be used with a given metaheuristic for solving a timetabling problem at hand? In practice, a preferred solution for a given problem is usually obtained after appropriately selecting and "tailoring" both sequential heuristics and metaheuristics based on domain-specific knowledge of the problem.

In light of the above limitations, Burke *et al.* (2003a) applied a local search method, the Great Deluge algorithm (GDA), to solve timetabling problems. The "beauty" of the GDA is that it is much easier to develop a GDA algorithm compared to other metaheuristics, because it only requires one input parameter and therefore requires the least effort to "tailor" it for a given problem. It is worth noting that the authors showed that GDA is effective even by using a very simply defined neighbourhood structure. Burke and Newell (2002, 2003) extended this research further by applying an adaptive initialisation heuristic before running GDA. This adaptive heuristic firstly solves the problem a number of times in order to learn how to adjust the heuristic's parameters. Both methods produced best-published results on a range of benchmark problems.

It is desirable to develop a general timetabling system which works equally well for a variety of problem descriptions from different universities. Hyper-heuristic solution methodology, which is "an emerging methodology in search and optimisation" (Burke *et al.*, 2003a) aims at addressing these needs. Broadly speaking, the term of hyper-heuristics is defined as "the process of using (meta-)heuristics to choose (meta-)heuristics to solve the problem in hand"

(Burke *et al.*, 2003b). Terashima-Marín *et al.* (1999) presented a hyper-heuristic Evolutionary Approach for solving examination timetabling problems. The choices of different sequential heuristics, parameter value settings and the conditions for swapping sequential heuristics during the search process are encoded as chromosomes and evolved by a genetic algorithm. The timetable is built by the best chromosome founded by the genetic algorithm. Petrovic and Qu (2002) proposed a novel case-based hyper-heuristic to intelligently select sequential heuristics. A timetable is constructed by applying iteratively a number of sequential heuristics. The selection of a heuristic to improve the current partial solution is based on the performance of each heuristic in a similar situation. Their system requires a training process using the knowledge discovery techniques.

1.2 Case-Based Reasoning in Scheduling

Case-Based Reasoning (CBR) is an artificial intelligence methodology in which a new problem is solved by reusing knowledge and experience gained in solving previous problems (Leake, 1996; Kolodner, 1993). A case contains a description of the problem, and its solution. Cases are stored in a case base. The CBR process is divided into four phases (Aamodt and Plaza, 1994): retrieval of the case most similar to the new problem, reuse and revision of its solution, and inclusion of the new case in the case base.

Only a few applications of CBR to scheduling have been reported. The work on CBR so far can be classified into two categories. Approaches in the first category reuse the past problem solving methods or operators within a method for solving a new problem. Miyashita and Sycara (1995) built a CBR system CABINS, which improves sub-optimal solutions for job scheduling problems by applying iteratively a number of moves, chosen by CBR. A case in CABINS consists of a move operator and the context in which it proved to be useful. Schirmer (2000) applied CBR to choose the most suitable heuristic for solving different project scheduling problems. Petrovic *et al.* (2003a) developed a CBR system for nurse rostering problems, which stores scheduling repair knowledge of experts as cases and uses CBR to drive the constraint satisfaction procedure.

The second category of CBR approaches to scheduling reuse the whole solutions to a problem. Coello and Santos (1999) solved the real-time job scheduling problem by reusing solutions to similar problems. Similarly, Burke *et al.* (2001) established a CBR scheduler in which a new course timetabling problem is solved by revising the solution of a previously stored similar timetabling problem.

In this paper, (an early version of which appeared in Petrovic *et al.*, 2003b), we aim to develop a new approach which enhances the performance of GDA

on examination timetabling problems by intelligently applying an appropriate sequential heuristic for its initialisation. Section 2 briefly introduces a GDA and different sequential heuristics. The CBR approach developed for examination timetabling is described in Section 3. Section 4 presents experimental results and related discussion. Conclusions and future research work are given in Section 5.

2. GREAT DELUGE ALGORITHM AND SEQUENTIAL HEURISTICS

2.1 Great Deluge Algorithm

The GDA is a local search method introduced by Dueck (1993) that has been successfully applied to examination timetabling problems (Burke *et al.*, 2003b). It represents a modification of the SA approach (Kirkpatrick *et al.*, 1983). Apart from accepting a move that improves the solution quality, GDA also accepts a move that results in a decrease of the solution quality as long as the decrease of the solution quality is smaller than a given upper boundary value, referred to as "water-level". In this work, the water-level is initially set to be the objective function value of the initial solution multiplied by a predefined factor. The neighbouring solutions of the current solution are obtained by moving an examination to a different time slot. After each move, the water-level is iteratively decreased by a fixed rate, which is equal to the initial value of the water-level divided by the time that is allocated to the search (expressed as the total number of moves). Not surprisingly, the GDA produces better solutions with the prolongation of the search time of the algorithm. This does not hold for a number of other local search algorithms where the user does not control the search time.

2.2 Sequential Heuristics

A variety of sequential heuristics can be used to construct initial solutions for GDA. They sort examinations based on the estimated difficulty of their scheduling. A number of sequential heuristics are briefly described as follows.

1 Largest degree (LD). Examinations with the largest number of conflicts are scheduled first.

2 Largest enrolment (LE). A modification of LD: it schedules examinations with the largest student enrolment first.

3 Largest colour degree (CD). A dynamic version of LD: it prioritises examinations by the largest number of conflicts with other examinations, which have already been scheduled.

4 Largest weighted degree (LWD). LWD is a combination of LD and LE. The highest priority is given to the examination with the largest sum of the weighted conflicts, where each conflict is weighted by the number of students who are enrolled in both examinations.

5 Least saturation degree (SD). Examinations with the least number of available periods for placement should be scheduled first (Brelaz, 1979).

These sequential heuristics can be enriched in a variety of ways. The most common ones are listed below:

1 Maximum clique detection (MCD). The maximum clique is the largest completely connected subgraph of a graph. The cardinality of the maximum clique determines the lower bound on the number of time periods needed for the timetable (Carter, 1986). Finding the maximum clique is an NP-Complete problem (Garey and Johnson, 1977). Vertices of the maximum clique are regarded as the most difficult examinations to schedule and therefore should be scheduled first (Carter *et al.*, 1996). In this research, a tabu search heuristic approach proposed by Gendreau *et al.* (1993) was implemented to find the vertices in the maximum clique of a given graph.

2 Adding random elements (ARE). The examination to be scheduled next is selected from a subset of randomly chosen examinations (Burke *et al.*, 1998). The size of the subset is given as the percentage of the full set of examinations.

3 Backtracking (BT). Some examinations cannot be assigned to any time period without violating hard constraints. In order to schedule these examinations, some previously scheduled examinations that are in conflict with the examinations at hand are rescheduled. Several rules are used to prevent cycles (Laporte and Desroches, 1984).

Sequential heuristics investigated in this research are hybridized with MCD, and/or BT, and/or ARE where 30%, 60% or 90% of examinations not yet scheduled are chosen randomly to form the subset of examinations to choose from. Selecting a suitable heuristic to generate an initial solution for the GDA is of high importance, because it can significantly affect the quality of the final solution.

3. CBR SYSTEM FOR EXAMINATION TIMETABLING

It is not an easy task to select an appropriate sequential heuristic to construct a good initial solution for GDA. It would be computationally very expensive to

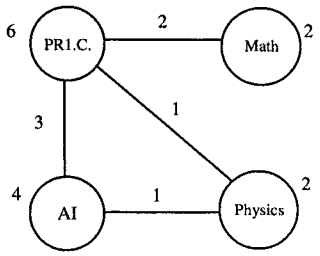
try every combination of sequential heuristics and GDA. Thus, we developed a CBR system, which selects a sequential initialisation heuristic for GDA in order to produce a high quality solution for a given problem. The rationale behind this study is that given an effective hybridisation of a certain sequential heuristic and GDA for a specific timetabling problem, it is likely that it will also be effective for a "similar" problem.

In our CBR system, a case memorises an examination timetabling problem and an effective sequential heuristic, which has generated an appropriate initial solution for GDA. For solving a new input timetabling problem, the sequential heuristic of the most similar case is proposed. The main research issue is how to define the "similarity" measure between two timetabling problems.

3.1 Case Representation

In this section, we explain how the important features of examination timetabling problems are incorporated into the case representation. An examination timetabling problem is represented by an undirected weighted graph $G = (V, E, \alpha, \beta)$, where V is the set of vertices that represent examinations, $E \subseteq V \times V$ is the finite set of edges that represent conflicts between examinations, $\alpha : V \mapsto N^+$ assigns positive integer weights to vertices that correspond to the number of students enrolled in the examination, and $\beta : E \mapsto N^+$ is an assignment of weights to edges which correspond to the number of students enrolled in two examinations that are in conflict. $|V|$ is used to denote the cardinality of the set V For illustration purpose, a simple example is given in Figure 1. In this figure the weight of Math is 2 because two students are enrolled in this course. The edge connecting AI and Physics is assigned weight 1 because there is one student who is enrolled in both examinations. Important features of the timetabling problem, such as number of examinations, number of enrolments, and number of constraints, are incorporated into the weighted graph case representation. Moreover, the weighted graph case representation is capable of describing highly inter-connected constraints that are imposed between examinations and on examinations themselves.

A solution to an examination timetabling problem is denoted by a vector $S = (s_1, s_2, \ldots, s_{|V|})$, where $s_n, n = 1, \ldots, |V|$, represents the time period assigned to the examination n. A feasible (conflict free) solution is a solution in which for any two vertices $a \in V$ and $b \in V$, then s_a, must be different from s_b if $(a, b) \in E$. The cost function often used in timetabling community for solution evaluation soft constraints was proposed by Carter *et al.* (1994). The common cost function enables comparison of quality of solutions produced by different approaches. The cost function gives a cost w_s to a solution whenever a student has to sit two examinations s periods apart. Costs that are used are $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$. The cost function sums all the



Student ID.	Enrolled Examination
2000101	AI; PR1 C.
2000102	AI; PR1 C.
2000103	AI; PR1 C.
2000104	AI; Physics.
2000105	Math; PR1 C.
2000106	Math; PR1 C.
2000107	Physics; PR1 C.

Figure 1. An examination timetabling problem represented by a graph.

costs of each student and divides the obtained sum by the number of students. The value obtained is the average cost for each student.

A case C can be formally represented by an ordered pair (G, H), where G is the graph representation of an examination timetabling problem, while H is the sequential heuristic that produced an initial solution appropriate for GDA.

3.2 Similarity Measure

An adequate definition of similarity measure is of great importance for a CBR system because it enables the retrieval of the case that is most closely related to the new problem. Since weighted graphs are used to represent timetabling problems, the retrieval of the most similar case from the case base requires solving a graph isomorphism problem, which is known to be NP-Complete (Garey and Johnson, 1977).

The following notation will be used. We denote a new timetabling problem to be solved (a query case) by C_q and a source case in the case base by C_s, while their weighted graphs are denoted by $G_q = (V_q, E_q, \alpha_q, \beta_q)$ and $G_s = (V_s, E_s, \alpha_s, \beta_s)$, respectively. In order to compute the similarity degree between C_q and C_s, a vertex-to-vertex correspondence has to be established that associates vertices in V_q with those in V_s. The correspondence is represented by the function $f : V_q \rightarrow V_s$.

Latin and Greek letters are used to denote vertices and edges in G_q and G_s, respectively. For instance, $f(a) = \chi$ denotes that vertex $a \in V_q$ is mapped to the vertex $\chi \in V_s$ by the correspondence f. In this research, the computation of the similarity degree between pairs of vertices, edges and graphs proposed by Wang and Ishii (1997) is modified to include the concept of weights employed in our problem.

The similarity degree between two vertices in G_q and G_s determined by the correspondence f is denoted by $DS_f(a, \chi)$:

$$DS_f(a,\chi) = \begin{cases} \text{Min}(\alpha_q(a), \alpha_s(\chi)), & \text{if } f(a) = \chi \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

Similarly, $DS_f(x, \varphi)$ represents the similarity degree between two edges determined by the correspondence f, where $x = (a, b) \in E_q$ and $\gamma = (\chi, \delta) \in E_s$:

$$DS_f(x,\gamma) = \begin{cases} \text{Min}(\beta_q(x), \beta_s(\gamma)), & \text{if } f(a) = \chi \text{ and } f(b) = \delta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

We use the label ϕ to denote an extraneous vertex or edge in a graph, which is not mapped by the correspondence f. We set $DS_f(a, \phi)$, $DS_f(\phi, \chi)$, $DS_f((a, b), \phi)$ and $DS_f(\phi, (\chi, \delta))$ to be equal to 0. Finally, the similarity degree $DS_f(G_q, G_s)$ between G_q and G_s determined by the correspondence f is calculated in the following way:

$$DS_f = (G_q, G_s) = \frac{F_v + F_e}{M_v + M_e} \tag{3}$$

where

$$F_v = \sum_{a \in Vq} \sum_{\chi \in V_s} DS_f(a, \chi) \tag{4}$$

$$F_e = \sum_{x \in Eq} \sum_{\gamma \in E_s} DS_f(x, \gamma) \tag{5}$$

$$M_v = \text{Max}\left(\sum_{a \in Vq} \alpha_q(a), \sum_{\chi \in V_s} \alpha_s(\chi) \right) \tag{6}$$

$$M_e = \text{Max}\left(\sum_{x \in Eq} \beta_q(x), \sum_{\gamma \in E_s} \beta_s(\gamma) \right) \tag{7}$$

Note that the value of $DS_f(G_q, G_s) \in [0, 1]$ is subject to correspondence f. The task is to find the correspondence f that yields as high a value of $DS_f(G_q, G_s)$ as possible.

3.3 Case Retrieval

The goal of the case retrieval is to find a case in the case base whose graph is the most structurally similar to that of the new problem. The retrieval of the

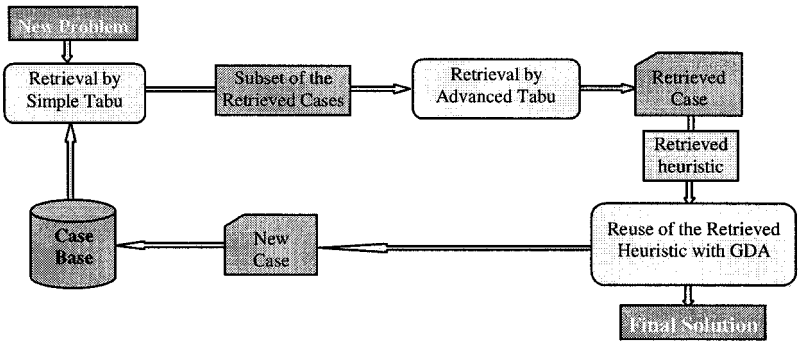


Figure 2. Architecture of the CBR system for heuristic initialisation of meta-heuristics.

graph-structured cases is a difficult process. Firstly, it is difficult to develop a valid indexing scheme to manage the organisation of graph-structured cases in the case base. Secondly, there is an expensive computational cost for calculating the similarity degree between two cases that involves graph matching.

For case retrieval we employ a two-stage Tabu Search described in more detail in Petrovic *et al.* (2002). The search procedure is guided by the short-term and long-term adaptive memories (Glover and Laguna, 1997). The short-term memory is used to prevent the search process from cycling by forbidding moves, which have been made recently. The long-term memory holds the history of all moves and is used to guide the search process to vicinities of elite solutions or regions that have not yet been explored. In order to reduce the computational cost required in the retrieval process, it is divided into two phases. Firstly, the simple Tabu Search with its short-term memory is used to quickly select a subset of cases from the case base considered to be similar enough to the new problem. Then the advanced Tabu Search enriched with long-term memory is used for the final more precise retrieval of the case.

3.4 Architecture of the CBR System

The architecture of our CBR system is depicted in Figure 2. The retrieval process is performed by the simple and advanced Tabu Search algorithms. The sequential heuristic, which has been shown to be the most appropriate for generating the initial solution for GDA for solving the retrieved case, is then proposed for the initialisation of GDA to be applied to the new problem. Once the problem is solved, the new problem together with the retrieved sequential heuristic will be stored as a new case in the case base.

4. EXPERIMENTS

The purpose of the designed experiments is twofold: evaluation of effectiveness and efficiency of the case retrieval and evaluation of system performance on a range of real world examination timetabling problems. Experiments were run on a PC with an Athlon 1400 Mhz CPU and 256 MB RAM.

4.1 Description of Seeding Cases

A number of real-world examination problems that are often used as benchmark problems within the timetabling community are used for the construction of cases, which will form a case base. The characteristics of these timetabling problems are given in Table 1. The conflict matrix is used to represent conflicts between pairs of examinations. Rows and columns of the matrix represent examinations, while each element of the matrix shows the number of students common for a pair of examinations. The density of the conflict matrix is calculated as the ratio of the number of exams in conflict to the total number of exams.

Seven benchmark problems (ear-f-83, hec-s-92, kfu-s-93, lse-f-91, sta-f-83, tre-s-92, and yor-f-83) and their specially designed variations were used to seed the case base. Variation problems were created by adding or deleting a given a number of students and examinations from the benchmark problem. Each variation problem is denoted by x/y, where x gives the percentage of both examinations and students of a benchmark problem which were added, and y gives the same percentage of those which were deleted. We defined five categories of variation problems: $5/15$, $5/10$, $5/5$, $10/5$, $15/5$. For example, the expression $5/15$ denotes a variation problem that was created by randomly adding and deleting 5% and 15% of the numbers of examinations and students of an associated corresponding benchmark problem, respectively. Two case bases of different sizes were created. The large case base contains ten variations of each seeding benchmark problem (two for each category). Thus it contains 77 cases (seven benchmark problems and 70 variations). The small case base contains five variations of each benchmark problem, which gives in total 42 cases (seven benchmark problems and 35 variations).

For each seeding problem, all possible sequential heuristics were used to produce initial solutions for GDA. The sequential heuristic, which led to the best final solution, was stored in the case. In this experiment, GDA was run for 20,000,000 iterations while water-level was set to be 1.3. The number of iterations was set empirically and the "water-level" was taken from (Burke *et al.*, 2003b).

The modifications of benchmark problems were used as seeding cases, because we noticed that heuristics proven to be the best for GDA initialisation for variation problems could be different from that of the original benchmark

Table 1. The benchmark problems used for seeding the case base.

Data	Institution	Periods	Number of exams	Number of students	Number of enrolments	Density of conflict matrix
Car-f-92	Carleton University, Ottawa	32	543	18,419	55,522	0.14
Car-s-91	Carleton University, Ottawa	35	682	16,925	56,877	0.13
Ear-f-83	Earl Haig Collegiate Institute, Toronto	24	190	1,125	8,109	0.29
Hec-s-92	Ecole des Hautes Etudes Commercials, Montreal	18	81	2,823	10,632	0.20
Kfu-s-93	King Fahd University Dharan	20	461	5,349	25,113	0.06
Lse-f-91	London School of Economics	18	381	2,726	10,918	0.06
Rye-s-93	Ryeson University, Toronto	23	486	11,483	45,051	0.07
Sta-f-83	St Andrew's Junior High School, Toronto	13	139	611	5,751	0.14
Tre-s-92	Trent University, Peterborough, Ontario	23	261	4,360	14,901	0.18
Ute-s-92	Faculty of Engineering, University of Toronto	10	184	2,750	11,793	0.08
Uta-s-92	Faculty of Arts and Science, University of Toronto	35	622	21,276	58,979	0.13
Yor-f-83	York Mills Collegiate Institute, Toronto	21	181	941	6,034	0.27

problem. In addition, the system performance will be improved by adding timetabling problems with different graph structures.

The aim of the first experiment is to show how often each heuristic appears to be the best for GDA in the timetabling problems of the large case base (77 cases). Note that the best initial solution (with respect to the cost function) does not necessarily lead to the best final solution produced by the GDA.

Figure 3 shows how many times each of the sequential heuristics appears in 77 seeding cases. A triplet (a, b, c) is assigned to each heuristic SD, LD, CD, LE and LWD, which denotes whether the heuristic was enriched with maximum clique detection ($a = 1$, otherwise $a = 0$), backtracking ($b = 1$, otherwise $b = 0$), and/or adding random element ($c = 30\%$, or 60%, or 90%). We can notice that although some heuristics are used more than the others, there is no single heuristic that outperforms all the rest.

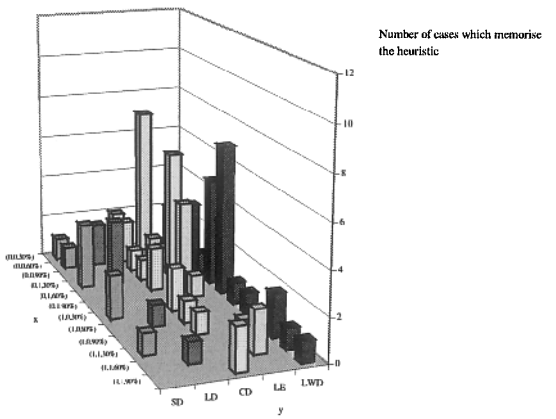


Figure 3. Use of heuristics in GDA initialisation.

4.2 Evaluation of the Case Retrieval Process

Since the retrieval process uses the simple and advanced Tabu Search sequentially, our second set of experiments investigates how precisely and with what computational cost for each of the approaches the similarity degree between two timetabling problems can be calculated. Due to the NP-Complete nature of the graph isomorphism problem, it is not possible to evaluate the performance of the two Tabu Search algorithms on two randomly chosen graphs. In this study we used the following method that is based on the experiments of Luo and Hancock (2001).

In our experiments, for each graph G of the seeding case, a copy of it is denoted by G_c. The similarity degree between G and G_c is 1 when each vertex in G is mapped to its corresponding vertex in G_c. This vertex-to-vertex mapping is considered as the ideal one. Then for each pair of G and G_c, both simple and advanced Tabu Search algorithms were run where the initial solution was a random vertex-to-vertex mapping. We evaluate the performance of our Tabu Search algorithms by examining whether they could find this ideal mapping between G and G_c, or how closely they approach it. We use DS_{tabu} to denote the similarity degree obtained after running a Tabu search algorithm. The closer the value of DS_{tabu} to 1, the better the vertex-to-vertex mapping that has been found.

Figure 4 shows the performance and search time of the two Tabu Search algorithms, respectively. A test running is terminated either when a solution has not improved for 2,000 moves or when an ideal mapping has been found. Circles and squares show the average value of DS_{tabu} obtained for the bench-

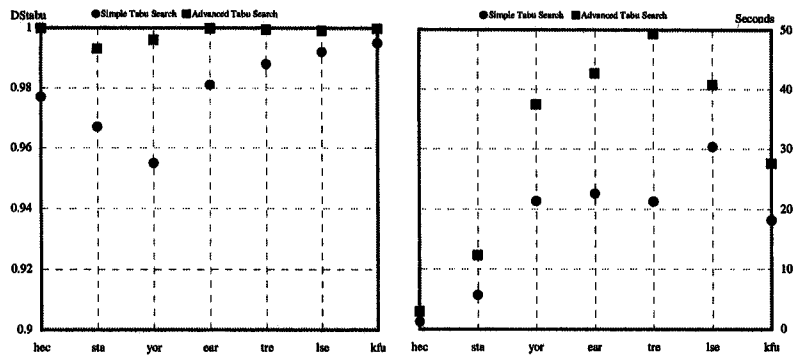


Figure 4. Performance of the simple and advanced Tabu Search.

mark problem (denoted on the x-axis) and its 10 variation problems by simple and advanced Tabu Search algorithm, respectively. Benchmark problems are sorted by their number of examinations in ascending order.

In Figure 4, it can be seen that the simple Tabu Search can obtain DS_{tabu} in the range from 0.95 to 0.995. The advanced Tabu Search could find the "ideal" mapping of vertices in five out of seven benchmark problems. For the remaining two problems Sta-f-83 and Yor-f-83, the advanced Tabu Search did not provide the "ideal" mapping, but the average values of DS_{tabu} are still high. On the other hand, the time required for advanced Tabu Search is not significantly higher than for simple Tabu Search. This justifies the implementation of the advanced Tabu Search within the retrieval process.

The experimental results show that the simple Tabu Search is capable of calculating the similarity degree approximately within a relatively short time and is used to filter the cases from the case base and to pass a predefined number of cases with the highest similarity degree with the new problem to the advanced tabu search. The advanced Tabu Search then spends more time finding mapping for graph isomorphism on the smaller set of cases.

4.3 Performance on Benchmark Problems

The purpose of the third set of experiments is to test performance of our CBR system on benchmark problems. In addition we want to investigate whether the size of the case base has an impact on the performance of the system. For each of the five benchmark problems (Car-f-92, Car-s-91, Rye-s-93, Uta-s-92, and Ute-s-92), the problem was solved twice by using the small and the large case base. For each of the other seven seeding bench-

Table 2. Results obtained using case bases of different sizes.

Data	Small case base				Large case base			
	Time (s)		Cost		Time (s)		Cost	
	Retrieval	Run GDA	Best	Avg.	Retrieval	Run GDA	Best	Avg.
Car-f-92	2274	1231	3.97	4.08	3772	1231	3.97	4.08
Car-s-91	3301	1321	4.54	4.65	5788	1321	4.54	4.65
Ear-f-83	1164	1250	34.49	36.06	2158	811	34.49	36.06
Hec-s-92	377	1540	10.92	11.29	699	1540	10.92	11.29
Kfu-s-93	1982	735	14.82	15.11	3701	735	14.82	15.11
Lse-f-91	1769	587	11.48	**11.60**	3077	586	10.60	**10.83**
Rye-s-93	1775	872	9.0	**9.66**	3097	930	9.0	**9.39**
Sta-f-83	373	695	160.29	**160.83**	673	680	159.89	**160.49**
Tre-s-92	1878	715	7.96	8.09	2936	786	7.96	8.09
Ute-s-92	430	508	25.74	**26.22**	786	637	25.64	**26.09**
Uta-s-92	2534	1271	3.26	3.29	4768	1271	3.26	3.29
Yor-f-83	1096	1300	36.82	**37.26**	2006	1320	36.69	**37.08**

mark problems, it was also used as a new problem and solved twice by the two case bases (the problem itself and its variation problems were removed from the case base in the retrieval process). Therefore, for each seeding benchmark problem, the small case base includes the other six benchmark problems and 30 associated variation problems; the large case base includes the other six benchmark problems and 60 associated variation problems. We ran each experiment five times to obtain the average results. The results are summarised in Table 2.

We can see that, as expected, the retrieval of more similar cases can lead to better solutions. For five problem instances Lse-f-91, Rye-s-93, Sta-f-83, Ue-s-92 and Yor-f-83 (highlighted by bold characters), the large case base yielded better solutions, while for the remaining instances both case bases gave the same solutions. The price to be paid is of course longer time spent on the case retrieval, which is proportional to the number of cases.

Table 3 provides the comparison of the average results obtained by three other GDA initialisation approaches previously tried in the timetabling literature on these benchmark problems, namely GDA where SD is used to provide the initial solution (Burke *et al.*, 2003b), GDA where initial solutions drive the adaptation of the parameters of the algorithm throughout the search (Burke and Newell, 2003; Burke and Newell, 2002), and GDA where the combination of SD, MCD and BT is used to construct an initial solution (Carter *et al.*, 1996). Again GDA was run for 200×10^6 iterations, because that was the number of iterations used in the methods that we compare our approach with. For illustration purpose, we also provide the time spent on the search (in seconds).

Table 3. Comparison of results for benchmark problems obtained by different initialisation of GDA.

Data	SD			Adaptive			SD & MCD & BT			CBR			
	GDA Time	Best Cost	Avg. Cost	GDA Time	Best Cost	Avg. Cost	GDA Time	Best Cost	Avg. Cost	Retrieval Time	GDA Time	Best Cost	Avg. Cost
Car-f-92	1120	4.03	4.07	416	–	4.10	1220	3.97	**4.04**	3772	1231	3.97	4.08
Car-s-91	1400	4.57	**4.62**	681	–	4.65	1441	4.62	4.66	5788	1321	4.54	4.65
Ear-f-83	806	34.85	**36.04**	377	–	37.05	767	33.82	36.26	2158	811	34.49	36.06
Hec-s-92	1471	11.27	12.43	516	–	11.54	1411	11.08	11.48	699	1540	10.92	**11.29**
Kfu-s-93	843	14.33	14.64	449	–	**13.90**	996	14.35	14.62	3701	735	14.82	15.11
Lse-f-91	646	11.61	11.65	341	–	**10.82**	675	11.57	11.94	3077	586	10.60	10.83
Rye-s-93	845	9.19	9.66	–	–	–	881	9.32	9.50	3097	930	9.0	9.39
Sta-f-83	675	165.12	169.7	418	–	168.73	674	166.07	166.31	673	680	159.89	**160.49**
Tre-s-92	907	8.13	8.29	304	–	8.35	751	8.19	8.27	2936	786	7.96	**8.09**
Ute-s-92	716	25.88	26.05	324	–	**25.83**	653	25.53	26.02	786	637	25.64	26.09
Uta-s-92	1070	3.25	3.30	517	–	**3.20**	1101	3.24	3.31	4768	1271	3.26	3.29
Yor-f-83	1381	36.17	**36.59**	695	–	37.28	1261	36.31	37.27	2006	1320	36.69	37.08
Rank			2.58			2.55			2.5				**2.08**

Although each algorithm was allocated the same number of iterations, the time is different due to computers of different characteristics. The table shows the average time spent on the search (each problem instance was solved five times).

In order to examine the performance of each initialisation method further, we also show the rank of the average cost that a method obtained on problem instances. This evaluation method was introduced by White *et al.* (2004). For example, the rank on the problem instance Car-f-92 is computed as: SD, 2; Adaptive, 4; MCD&BT&SD, 1; CBR, 3. The bottom row of Table 3 shows the average of the ranks for the 12 problems (excluding the rank of Rye-s-93 for the Adaptive initialisation method) of four different approaches.

We can see that our CBR system outperforms other initialisation methods for GDA. The CBR initialisation obtained the best rank (2.08) among all the methods. More significantly, we obtained the best average results for four benchmark problems (highlighted by bold characters). For the remaining seven problems, the other three methods only slightly outperformed our approach except for the problem instance Kfu-s-93. It is evident that our CBR system spent additional time on the case retrieval. However, the quality of the obtained results justifies the time spent on the selection of an appropriate heuristic, which determines a good starting point for the GDA.

Finally, we compare our results with those produced by the state-of- the-art timetabling metaheuristics: SA (Merlot *et al.*, 2003), Tabu search (White *et al.*, 2004), and GRASP (Casey and Thompson, 2003) by Table 4.

Table 4. Results for benchmark problems obtained by different timetabling approaches.

Data	SA			Tabu			GRASP			CBR			
	Time	Best Cost	Avg. Cost	Time	Best Cost	Avg. Cost	Time	Best Cost	Avg. Cost	Retrieval Time	GDA Time	Best Cost	Avg. Cost
Car-f-92	233	4.3	4.4	–	4.63	4.69	–	4.4	4.7	3772	1231	**3.97**	**4.08**
Car-s-91	296	5.1	5.2	–	5.73	5.82	–	5.4	5.6	5788	1321	**4.54**	**4.65**
Ear-f-83	26	35.1	35.4	–	45.8	45.6	–	34.8	**35.0**	2158	811	**34.49**	36.06
Hec-s-92	5.4	**10.6**	**10.7**	–	12.9	13.4	–	10.8	10.9	699	1540	10.92	11.29
Kfu-s-93	40	**13.5**	**14.0**	–	17.1	17.8	–	14.1	14.3	3701	735	14.82	15.11
Lse-f-91	35	**10.5**	11.0	–	14.7	14.8	–	14.7	15.0	3077	586	10.6	**10.83**
Rye-s-93	70	**8.4**	**8.7**	–	11.6	11.7	–	–	–	3097	930	9.0	9.39
Sta-f-83	5	157.3	157.4	–	158	158	–	**134.9**	**135.1**	673	680	159.89	160.49
Tre-s-92	39	8.4	8.6	–	8.94	9.16	–	8.7	8.8	2936	786	**7.96**	**8.09**
Ute-s-92	9	**25.1**	**25.2**	–	29.0	29.1	–	25.4	25.5	786	637	25.64	26.09
Uta-s-92	233	3.5	3.6	–	4.44	4.49	–	–	–	4768	1271	3.26	3.29
Yor-f-83	30	37.4	37.9	–	42.3	42.5	–	37.5	38.1	2006	1320	**36.69**	**37.08**

Table 5. Average of the ranks for benchmark problems obtained by different approaches.

Approaches	SA	Tabu	GRASP	SD GDA	Adaptive GDA	SD & MCD & BT GDA	CBR GDA
Average Rank	**3.00**	6.17	4.0	3.58	3.36	3.67	3.08

We obtained six best average costs and six best costs (highlighted) out of 12 benchmark problems.

Table 5 shows the average of the ranks for the 12 problem instances. Due to the incomplete results in Burke and Newell (2002) and Casey and Thompson (2003), we exclude the rank of Rye-s-93 for the Adaptive GDA method, and the rank of Rye-s-93 and Uta-f-92 for the GRASP method.

We can see that SA and the GDA initialised by CBR obtained the best rank (3.00) and the second best rank (3.08) among the seven different approaches investigated. However, opposite to SA our approach does not require parameter "tuning" for a particular timetabling problem and design of appropriate neighbourhood structure. In addition, the experience gained in solving one timetabling problem is not wasted but can be used in solving new similar timetabling problems.

5. CONCLUSIONS

In this paper we have presented a case-based reasoning system, which selects an appropriate sequential heuristic for generating an initial solution for

Great Deluge algorithm (GDA). We have shown that with an appropriate definition of "similarity" measure, such initialisation of GDA provides high-quality solutions for a range of real-world problems. One of the insights of this study is that our CBR system significantly contributes to the attempt of building a general metaheuristic framework for timetabling. Usually in metaheuristics, a random initialisation is employed or a thorough investigation of heuristics needs to be performed, which is useful only for a given problem instance. In this paper, we demonstrated that knowledge gained in initialisation of one timetabling problem can be used for solving new similar timetabling problems.

The developed CBR system examined in this paper contains cases with complex structures represented by weighted graphs. We have shown that the two-phase Tabu Search approach is capable of retrieving graph-structured cases where graphs are of large size and the case base contains hundreds of cases. We believe that the graph-structured case representation, the similarity measure, and the proposed case retrieval are applicable to other domains such as job shop scheduling, planning and other CBR applications.

The results obtained so far provide us with a good foundation for the development of a more general CBR system for solving timetabling problems. Our future research direction will include improvements aimed to shorten the required time for the case retrieval. We will also investigate hierarchical case representation that would enable the case retrieval process to examine only a subset of the case base. Finally, we will investigate the hybridisation of sequential heuristics and other local search methods such as tabu search and simulated annealing.

Acknowledgments

The authors wish to thank Dr Jim Newall for offering the source code of the timetabling library, and the anonymous reviewers for their valuable remarks on this work.

References

Aamodt, A. and Plaza, P. (1994) Case-based reasoning: foundational issues, methodological variations and system approaches. *The European Journal on Artificial Intelligence*, **7**:39–59.

Brelaz, D. (1979) New methods to color the vertices of a graph. *Communication of ACM*, **22**:251–256.

Burke, E. K. and Newell, J. P. (1999) A multi-stage evolutionary algorithm for the timetable problem. *IEEE Transactions on Evolutionary Computation*, **31**:63–74.

Burke, E. K. and Newall, J. P. (2002) Enhancing Timetable Solutions with Local Search Methods. In *The Practice and Theory of Automated Timetabling IV*, Lecture Notes in Computer Science, Vol. 2740, Springer, Berlin, pp. 195–206.

Burke, E. K. and Newell, J. P. (2003) Solving examination timetabling problems through adaptation of heuristic orderings. *Annals of Operations Research*, accepted for publication.

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

Burke, E. K., Elliman, D. G., Ford, P. H. and Weare, R. F. (1996) Examination timetabling in british universities—A survey. In *The Practice and Theory of Automated Timetabling I*, Lecture Notes in Computer Science, Vol. 1153, Springer, Berlin, pp. 76–92.

Burke, E. K., Newall, J. P. and Weare, R. F. (1998) Initialisation strategies and diversity in evolutionary timetabling. *Evolutionary Computation Journal*, **6**:81–103.

Burke, E. K., Newell, J. P. and Weare, R. F. (1998) A simple heuristically guided search for the timetable problem. In *Proceedings of the International ICSC Symposium on Engineering of Intelligent Systems*, University of La Laguna, pp. 574–579.

Burke, E. K., MacCarthy, B., Petrovic, S. and Qu, R. (2001) Case-based reasoning in course timetabling: an attribute graph approach. In *Proceedings of 4th International Conference on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, Vol. 2080, Springer, Berlin, pp. 90–104.

Burke, E. K., Hart, E., Kendall, G., Newall, J., Ross, P. and Schulenburg, S. (2003a) Hyper-heuristics: an emerging direction in modern search technology. In *Handbook of Meta-Heuristics*, Chapter 16, pp. 457–474, Kluwer, Dordrecht.

Burke, E. K., Bykov, Y., Newall, J. P. and Petrovic, S. (2003b) A time-predefined local search approach to exam timetabling problems. *IIE Transactions on Operations Engineering*, **36**:509–528.

Carter, M. W. (1986) A survey of practical applications on examination timetabling. *Operations Research*, **34**:193–202.

Carter, M. W. and Laporte, G. (1996) Recent developments in practical examination timetabling. In *The Practice and Theory of Automated Timetabling I*, Lecture Notes in Computer Science, Vol. 1153, Springer, Berlin, pp. 3–21.

Carter, M. W., Laporte, G. and Chinneck, J. W. (1994) A general examination scheduling system. *Interfaces*, **24**:109–120.

Carter, M. W., Laporte, G. and Lee, S. Y. (1996) Examination timetabling: algorithmic strategies and applications. *Journal of the Operational Research Society*, **47**:373–383.

Casey, S. and Thompson, J. (2003) GRASPing the examination scheduling problem. In *The Practice and Theory of Automated Timetabling IV*, Lecture Notes in Computer Science, Vol. 2740, Springer, Berlin, pp. 232–246.

Coello, J. M. A. and Santos, R. C. (1999) Integrating CBR and heuristic search for learning and reusing solutions in real-time task scheduling. In *Proceedings of 3rd International Conference on Case-Based Reasoning*, Lecture Notes in Artificial Intelligence, Vol. 1650, Springer, Berlin, pp. 89–103.

Di Gaspero, L. and Schaerf, A. (2001) Tabu search techniques for examination timetabling. In *Proceedings of Practice and Theory of Automated Timetabling III*, Lecture Notes in Computer Science, Vol. 2079, Springer, Berlin, pp. 104–117.

Dueck, G. (1993) New optimization heuristics. *Journal of Computational Physics*, **104**:86–92.

Foxley, E. and Lockyer, K. (1968) The construction of examination timetable by computer. *The Computer Journal*, **11**:264–268.

Garey, M. R. and Johnson, D. S. (1977) Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco.

Gendreau, M., Soriano, P. and Salvail, L. (1993) Solving the maximum clique problem using a tabu search approach. *Annals of Operations Research*, **41**:385–403.

Glover, F. and Laguna, M. (1997) Tabu Search. Kluwer, Dordrecht.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983) Optimisation by simulated annealing. *Science*, **220**:671–680.

Laporte, G. and Desroches, S. (1984) Examination timetabling by computer. *Computers and Operations Research*, **11**:351–360.

Leake, D. B. (1996) CBR in context: the present and future. In *Case-Based Reasoning: Experiences, Lessons, and Future Directions*, D. Leake (Ed.), AAAI Press/MIT Press, Menlo Park, CA.

Luo, B. and Hancock, E. R. (2001) Structural graph matching using the em algorithm and singular value decomposition. *IEEE Transactions Analysis and Machine Intelligence*, **23**:1120–1136.

Kolodner, J. (1993) Case-Based Reasoning. Morgan Kaufmann, San Mateo, CA.

Merlot, L. T. G., Boland, N., Hughs, B. and Stucky, P. J. (2003) A hybrid algorithm for the examination timetabling problem. In *The Practice and Theory of Automated Timetabling IV*, Lecture Notes in Computer Science, Vol. 2740, Springer, Berlin, pp. 207–231.

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

Petrovic, S. and Qu, R. (2002) Case-based reasoning as a heuristic selector in a hyper-heuristic for course timetabling problems. In *Proceedings of Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies*, Frontiers in Artificial Intelligence and Applications, Vol. 82, IOS Press, Amsterdam, pp. 336–340.

Petrovic, S., Kendall, G. and Yang, Y. (2002) A tabu search approach for graph-structured case retrieval. In *Proceedings of the STarting Artificial Intelligence Researchers Symposium*, IOS Press, Amsterdam, pp. 55–64.

Petrovic, S., Beddoe, G. R. and Berghe, G. V. (2003a) Storing and adapting repair experiences in employee rostering. In *Practice and Theory of Automated Timetabling IV*, Lecture Notes in Computer Science, Vol. 2740, Springer, Berlin, pp. 149–166.

Petrovic, S., Yang, Y. and Dror, M. (2003b) Case-based initialisation of metaheuristics for examination timetabling. In *Proceedings of the 1st Multidisciplinary International Conference on Scheduling: Theory and Applications*, pp. 137–155.

Saleh Elmohamed, M. A., Coddington, P. and Fox, G. (1998) A comparison of annealing techniques for academic course scheduling. In *The Practice and Theory of Automated Timetabling II*, Lecture Notes in Computer Science, Vol. 1408, Springer, Berlin, pp. 92–112.

Schirmer, A. 2000 Case-based reasoning and improved adaptive search for project scheduling. *Naval Research Logistics*, **47**:201–222.

Terashima-Marín, H., Ross, P. and Valenzuela-Rendón, M. (1999) Evolution of constraint satisfaction strategies in examination timetabling. In *Proceedings of the Genetic and Evolutionary Conference*, pp. 635–642.

Wang, Y. and Ishii, N. (1997) A method of similarity metrics for structured representations. *Expert Systems with Applications*, **12**:89–100.

Welsh, D. J. A. and Powell, M. B. (1967) An upper bound on the chromatic number of a graph and its application to timetabling problems. *The Computer Journal*, **10**:85–86.

White, G. M. Xie, B. S. and Zonjic, S. (2004) Using tabu search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, **153**:80–91.