ELSEVIER

European Journal of Operational Research 140 (2002) 266–280

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/dsw

Recent research directions in automated timetabling

Edmund Kieran Burke, Sanja Petrovic

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

Abstract

The aim of this paper is to give a brief introduction to some recent approaches to timetabling problems that have been developed or are under development in the Automated Scheduling, Optimisation and Planning Research Group (ASAP) at the University of Nottingham. We have concentrated upon university timetabling but we believe that some of the methodologies that are described can be used for different timetabling problems such as employee timetabling, timetabling of sports fixtures, etc. The paper suggests a number of approaches and comprises three parts. Firstly, recent heuristic and evolutionary timetabling algorithms are discussed. In particular, two evolutionary algorithm developments are described: a method for decomposing large real-world timetabling problems and a method for heuristic initialisation of the population. Secondly, an approach that considers timetabling problems as multicriteria decision problems is presented. Thirdly, we discuss a case-based reasoning approach that employs previous experience to solve new timetabling problems. Finally, we outline some new research ideas and directions in the field of timetabling. The overall aim of these research directions is to explore approaches that can operate at a higher level of generality than is currently possible.

Keywords: Combinatorial optimisation; Timetabling/Scheduling; Meta-heuristic approaches; Multiple criteria analysis; Case-based reasoning; Hyper-heuristics

1. Introduction

Educational timetabling is a major administrative activity for a wide variety of institutions. A timetabling problem can be defined to be the problem of assigning a number of events into a limited number of time periods. Wren (1996) defines timetabling as follows:

> "*Timetabling* is 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."

In this paper, we will concentrate on university timetabling problems. Such problems can be divided into two main categories: course timetabling and exam timetabling. These problems are subject to many constraints that are usually divided into two categories: "hard" and "soft" (e.g., Burke et al., 1997).

Hard constraints are rigidly enforced. Examples of such constraints are:

- No resource (students or staff) can be demanded to be in more than one place at any one time.

E-mail addresses: ekb@cs.nott.ac.uk (E.K. Burke), sxp@cs.nott.ac.uk (S. Petrovic).

0377-2217/02/$ - see front matter
PII: S0377-2217(02)00069-3

- For each time period there should be sufficient resources (e.g., rooms, invigilators, etc.) available for all the events that have been scheduled for that time period.

Soft constraints are those that are desirable but not absolutely essential. In real-world situations it is, of course, usually impossible to satisfy all soft constraints. Examples of soft constraints (in both exam and course timetabling) are:

- *Time assignment*. A course/exam may need to be scheduled in a particular time period.
- *Time constraints between events*. One course/exam may need to be scheduled before/after the other.
- *Spreading events out in time*. Students should not have exams in consecutive periods or two exams on the same day.
- *Coherence*. Professors may prefer to have all their lectures in a number of days and to have a number of lecture-free days. These constraints conflict with the constraints on spreading events out in time.
- *Resource assignment*. Professors may prefer to teach in a particular room or it may be the case that a particular exam must be scheduled in a certain room.

There are, of course, significant differences between the two broad categories of university timetabling problem: course and exam timetabling. For example, a number of exams may be scheduled in one room or an exam may be split across several rooms, while in course timetabling one course (usually) has to be scheduled into exactly one room. Exam timetabling often aims to decrease the number of students who have exams in adjacent periods while in course timetabling it is usually desirable for students to have two or more courses in a row.

A more detailed range of exam timetabling constraints that are in use in British universities can be seen in Burke et al. (1996a). This paper is an analysis of the examination requirements of over 50 British universities. The soft constraints and their importance differs very much from university to university and some of these differences are discussed in the same paper. More details of examination timetabling constraints and approaches arising in practice can be seen in Carter and Laporte (1996), while the same authors discuss course timetabling in Carter and Laporte (1998).

The large number of events (courses/exams) to be scheduled and a wide variety of constraints imposed on timetabling makes the set of all possible solutions (i.e., the search space of the problem) very large indeed. The construction of a timetable can be an extremely difficult task and its manual solution can require much effort. Timetabling problems have attracted the attention of the scientific community from a number of disciplines (including OR and Artificial Intelligence) for about 40 years and over the last decade or so there has been an increased interest in the field.

A wide variety of approaches to timetabling problems have been described in the literature and tested on real data. They can be roughly divided into four types (Carter, 1986; Carter and Laporte, 1996, 1998): (1) sequential methods, (2) cluster methods, (3) constraint-based methods, and (4) meta-heuristic methods.

(1) *Sequential methods*. These methods order events using domain heuristics and then assign the events sequentially into valid time periods so that no events in the period are in conflict with each other (Carter, 1986). In sequential methods, timetabling problems are usually represented as graphs where events (courses/exams) are represented as vertices, while conflicts between the events are represented by edges (de Werra, 1985). For example, if some students have to attend two events there is an edge between the nodes which represents this conflict. The construction of a conflict-free timetable can therefore be modelled as a graph colouring problem. Each time period in the timetable corresponds to a colour in the graph colouring problem and the vertices of a graph are coloured in such a way so that no two adjacent vertices are coloured by the same colour.

A variety of graph colouring heuristics for constructing conflict-free timetables is available in the literature (Brelaz, 1979; Carter and Laporte, 1996). These heuristics order the events based on an estimation of how difficult it is to schedule them. The heuristics that are often used are:

- *Largest degree first*. Events that have a large number of conflicts with other events (i.e., a large degree) are scheduled early. The rationale

is that the events with a large number of conflicts are more difficult to schedule and so should be tackled first.
- *Largest weighted degree*. This is a modification of the *Largest degree first* which weights each conflict by the number of students involved in the conflict.
- *Saturation degree*. In each step of the timetable construction an event which has the smallest number of valid periods available for scheduling in the timetable constructed so far is selected.
- *Colour degree*. This heuristic priorities those events that have the largest number of conflicts with the events that have already been scheduled.

Once the events are ordered for scheduling, a number of approaches can be employed to select a valid period for each event. For example, the earliest valid period can be selected, or the "best" valid period in the context of a defined objective function, etc.

(2) *Cluster methods*. In these methods the set of events is split into groups which satisfy hard constraints and then the groups are assigned to time periods to fulfill the soft constraints. An early paper to describe this approach was written by White and Chan (1979). Different optimisation techniques have been employed to solve the problem of assigning the groups of events into time periods (Fisher and Shier, 1983; Balakrishnan et al., 1992). The main drawback of these approaches is that the clusters of events are formed and fixed at the beginning of the algorithm and that may result in a poor quality timetable.

(3) *Constraint-based approaches*. In these methods a timetabling problem is modelled as a set of variables (i.e., events) to which values (i.e., resources such as rooms and time periods) have to be assigned to satisfy a number of constraints (Brailsford et al., 1999; White, 2000). Usually a number of rules is defined for assigning resources to events. When no rule is applicable to the current partial solution a backtracking is performed until a solution is found that satisfies all constraints.

(4) *Meta-heuristic methods*. Over the last two decades a variety of meta-heuristic approaches such as simulated annealing, tabu search, genetic algorithms and hybrid approaches have been investigated for timetabling. Some very good results have been reported in Burke and Ross (1996), Burke and Carter (1998), Burke and Erben (2001) and the timetabling section of Fogarty (1995). Meta-heuristic methods begin with one or more initial solutions and employ search strategies that try to avoid local optima. All of these search algorithms can produce high quality solutions but often have a considerable computational cost.

We have just presented a very brief outline of timetabling research over the years. Interested readers can see a number of surveys of existing timetabling methods and applications (de Werra, 1985; Carter, 1986; Carter and Laporte, 1996, 1998; Bardadym, 1996; Burke et al., 1997; Schaerf, 1999). The aim of this paper is not to provide another survey of timetabling research. Instead we aim to present an overview of some of the latest approaches and research directions that have been, or are being undertaken, in the Automated Scheduling, Optimisation and Planning Research Group (ASAP) at the University of Nottingham. A number of approaches are described which explore an interdisciplinary strategy along promising avenues that have not yet been fully studied in the timetabling field. Heuristic and meta-heuristic methods developed for exam timetabling are described in Section 2. Section 3 presents an approach to exam timetabling that considers exam timetabling problems as multicriteria problems. The application of case-based reasoning to course timetabling problems is presented in Section 4. Section 5 looks at some new research directions in timetabling research with an overall aim of building timetabling systems that handle a higher level of generality than is currently available.

2. Heuristic and meta-heuristic methods for timetabling

This Section gives a brief tour of our research into timetabling over the last five years which resulted in innovations in heuristic and meta-heuristic methods for timetabling. This research was focused on exam timetabling problems.

2.1. Hybrid heuristic methods

In order to direct a search of the solution space to more desirable areas, a simple efficient search method based on heuristics was presented in 1998 (Burke et al., 1998a). The events are ordered using graph colouring heuristics. The ordered events are scheduled in valid periods with respect to the evaluation function. The evaluation function $cost(t)$ takes into consideration a number of soft constraints in a timetable t:

$$cost(t) = 5000 * unscheduled(t) + 3 * sameDay(t) + overNight(t),$$

where $unscheduled(t)$ holds the number of exams not scheduled in a valid period, $sameDay(t)$ and $overNight(t)$ present the number of conflicts where students have exams in adjacent periods on the same day and in overnight adjacent periods, respectively.

Two types of selection, which include random elements, have been proposed to replace the sequential scheduling of the ordered events:

(1) *Tournament selection* which generates a subset of events randomly and then selects the event from the subset that is ranked the first with respect to the employed heuristics.
(2) *Bias selection* where an event is randomly selected from the list of the first n events.

Experiments have shown the advantages of such a hybrid method over the "pure" heuristic sequential methods (Burke et al., 1998a). It can be observed that the most notable improvements of the pure heuristic sequencing strategies have been obtained using the *Largest degree first* heuristic within the described heuristic search with the random elements.

2.2. Genetic and memetic algorithms for timetabling

The possibilities and advantages of the application of meta-heuristic methods to timetabling problems have been studied extensively within the timetabling research community. Within the ASAP research group, special interest has been given to genetic and memetic approaches to exam timetabling problems.

The memetic algorithm attempts to improve the performance of a genetic algorithm by incorporating local neighbourhood search (Moscato and Norman, 1992). The main idea of the memetic algorithm is to explore the neighbourhood of the solutions obtained by a genetic algorithm and to navigate the search toward the local optima (for each solution) before passing back to the genetic algorithm and continuing the process. Paechter et al. (1995, 1996) have used this approach for course timetabling.

However, it is the case that, although meta-heuristics have (relatively) recently provided successful approaches, domain specific heuristics have been utilised in timetabling problems for many years. It is a well established fact that methods incorporating domain specific heuristics can give acceptable solutions very quickly, but they often lack the optimisation capabilities of the more intensive search methods provided by meta-heuristics. This observation generated the motivation to use a range of graph colouring heuristics within the memetic algorithm to improve and accelerate the search process. The main characteristics of this approach for exam timetabling are outlined below (Burke et al., 1995, 1996b, 1998b; Burke and Newall, 1999).

2.2.1. Representation of the solution

A population of a number of chromosomes which encode solutions to the problem is generated. Each member of the population consists of a number of time periods that are allocated for the examination. Each time period contains a list of the exams that are scheduled in that period and rooms assigned to these exams.

2.2.2. Evaluation function

A linear weighted cost function has been defined to assess the quality of the timetables. It penalises incomplete timetables and occurrences where students have to attend two exams in consecutive periods which are on the same day or on adjacent days.

2.2.3. Evolutionary operators

A number of different operators have been developed and used within the memetic algorithms

which manipulate the chromosomes of the population to form chromosomes of the offspring. They are briefly described as follows.

(1) The *light mutation operator* makes minor changes to the timetable by rescheduling randomly chosen exams in other valid periods. The purpose of this mutation operator is to direct the search away from local optima.
(2) The *heavy mutation operator* reschedules the exams from "bad" time periods in timetables. In order to determine the "bad" time periods, a probability of disruption is calculated for each time period. This probability takes into consideration the penalties of the exams that are scheduled in the period. All exams from disrupted periods are then rescheduled in random order in the valid periods that cause the lowest increment of the cost function. The application of heuristics to the heavy mutation operator results in substantially better final timetables and also increases the speed at which acceptable solutions can be found.
(3) The *heuristic heavy mutation operator* reschedules exams from disrupted periods in heuristic order. The *Largest degree first* graph colouring heuristic is used to order exams for rescheduling.
(4) The specialist *recombination operator* produces a new solution by combining two different timetables (parents). The exams that are scheduled in the same time periods in both parents are scheduled in the same period in the child. The exams that are scheduled differently in the parents form the list of unscheduled events. Different heuristics have been investigated that order the list of unscheduled exams which are then rescheduled in the valid periods with respect to the evaluation function. However, the recombination operator (enhanced with heuristics) has not significantly improved the performance of the algorithm.

2.2.4. Hill-climbing

After each mutation or recombination operator, a hill-climbing operator is applied to direct the search towards the local optima. The hill-climbing operator takes exams sequentially from periods that are randomly ordered, and reschedules them in valid periods that cause the least penalties. Experiments have shown that the designed operators show better results when they are combined with hill-climbing.

2.2.5. Selection phase

The selection process orders the members of the population by the values of their evaluation functions and decides (according to a certain probability) whether to leave the solution for the next generation or not.

2.2.6. Decomposition

Real-world university timetabling problems usually have a very high number of events that have to be scheduled. Algorithms for timetabling might be more efficient if they consider subsets of events, rather then the whole set of events. Carter proposed an algorithm for timetabling decomposition (Carter, 1983). The algorithm schedules subsets of events sequentially, adding the events from the current subset into an already created schedule. Each subset of events is small enough to be solved by conventional approaches such as linear integer programming.

Following the lines of decomposition proposed by Carter, a memetic algorithm has been modified to decompose large exam timetabling problems into smaller ones (Burke and Newall, 1999). The drawback of such a decomposition could be that it may be very difficult (or indeed impossible) to schedule events belonging to later subsets having fixed the schedule for earlier subsets. To alleviate this potential problem subsets are chosen using a heuristic ordering of events and the algorithm handles two subsets at a time, but fixes only the events from the first subset. It has been shown that this "look ahead" process combined with a good choice of heuristic and a proper size for the subsets can significantly reduce the computational cost and can also improve the quality of the solutions.

The experiments have been performed on available real data from a number of universities around the world (obtainable from ftp://ftp.cs.nott.ac.uk.ttp/Data and ftp://ie.utoronto.ca/mwc/testprob). Some of the characteristics of the

Table 1
Characteristics of examination data used in experiments

University	No. of periods	No. of exams	No. of students	No. of students' enroll.	No. of seats per period
Carleton University, Ottawa – CARF	36	543	18 419	55 552	2000
King Fahd University, Dharan – KFU	21	461	5349	25 118	1955
Nottingham University, Nottingham – NOTT	23	800	7896	34 265	1550
Purdue University, Indiana – PUR	30	2419	30 032	120 690	5000

examination data from four universities are given in Table 1.

It is beyond the scope of this paper to present and discuss all the interesting results obtained. However, we summarise in Table 2 some of the most recent results reported in (Di Gaspero and Schaerf, 2001). This paper compares three different approaches: A heuristic method with backtracking based on *Saturation degree* graph colouring heuristics (Carter et al., 1996), tabu search (Di Gaspero and Schaerf, 2001), and a memetic algorithm enhanced with decomposition (Burke and Newall, 1999). The problem considers capacity constraints (the total number of seats available in the period is taken into account but not the number of rooms because one room can accommodate more than one exam). The cost function penalises occurrences where students have two exams in adjacent periods, but penalises overnight adjacent periods less than the other adjacent periods. The values obtained for the cost function in three different approaches are shown in Table 2. The comparisons show that the heuristic method and tabu search perform similarly, but a decomposition-based memetic algorithm outperforms them.

An overall conclusion is that there are considerable benefits to be gained from decomposing large problems into smaller ones and from hybridising three approaches: meta-heuristic, heuristic and local search, to produce advanced search methods that are greater than the sum of their individual parts.

The algorithms have also been evaluated using randomly generated data that represented timetabling problems of different difficulties. The algorithms did not perform as well on these problems due (we believe) to the even spread of difficult exams (with respect to scheduling) which decreased the usefulness of the heuristics (Newall, 1999). We may conclude that heuristics exploiting knowledge about the domain may direct the search towards promising areas of the solution space. They can miss the area with the better solutions, but that is the price that has to be paid in order to find acceptable results at a reasonable computational cost.

2.2.7. Initialisation

The effects of heuristic initialisation in evolutionary algorithms for timetabling problems have been reported in (Burke et al., 1998b). Employment of graph colouring heuristics in the initialisation phase results in initial solutions that are of higher quality than randomly generated solutions. However, in order to preserve the diversity of the initial population (that is desirable in evolutionary algorithms), two types of ran-

Table 2
Values of the cost function for different heuristic and meta-heuristic methods (Di Gaspero and Schaerf, 2001; Burke and Newall, 1999)

University	Heuristic method	Tabu search	Memetic algorithm based on decomposition
CARF	2915	3048	1765
KFU	2700	1733	1608
NOTT	918	751	524
PUR	97 521	123 935	65 461

dom selection of an event from the list of ordered events have been employed: tournament and bias selection.

In order to study the effects of heuristic initialisation in the evolutionary algorithms three different measures of diversity have been defined:

(a) *Events are in the same time period*. This measure counts the number of occurrences where an event is scheduled in the same time period in both timetables. The drawback of this measure is that it considers absolute positions of the events in timetables. For example, if one timetable has events which are scheduled one period later than in another timetable then this diversity measure would say that they are very different, although they could be considered to be similar to each other.
(b) *Pairs of events are in the same period*. This measure takes into consideration the temporal relationships between events. It counts pairs of events that are scheduled in the same period in both timetables.
(c) *Coverage of the search space*. This measure counts how many times each pair (event, time period) appears in the members of the whole population.

Method (b) was favoured because it allowed us to represent diversity as a single value average and did not have the drawback of method (a).

Experiments have shown that the employment of heuristic initialisation strategies can provide significant benefits in terms of the quality and diversity of the members of the population. A variety of experiments have been performed which involved timetabling problems of different difficulties (i.e., low and high conflicting timetabling problems), different graph colouring heuristics and different sizes of subsets in the tournament and bias selection. Dynamic heuristics, which change the order of events for scheduling in each step of the timetable construction (e.g., *Saturation degree* and *Colour degree* heuristics), provide better initial timetables. The timetables obtained were shorter and have higher degrees of diversity in comparison with static heuristic sequencing strategies. The use of heuristic initialisation can significantly improve the performance of evolutionary timetabling algorithms.

3. A multicriteria approach to timetabling

Many timetabling algorithms are relatively simple and address only a subset of the constraints. In this way a reasonable computational cost is achieved at the expense of "comprehensiveness". In order to overcome this drawback we have considered timetabling problems as multicriteria decision problems (Burke et al., 2001a). If we assume that solutions which satisfy hard constraints exist then the quality of the solutions can be assessed on the basis of how well they satisfy the soft constraints. One hard constraint has been introduced which is common to all universities, that events which require the same resources must not be scheduled in the same time period. Consequently, a number of criteria have been defined with respect to the other constraints. Each criterion expresses the measure of violations of the corresponding constraint. A constraint which is normally considered to be hard (namely that the total resources at any time must not exceed the resources available) has been taken as one of the criteria.

Initial research has been focused on exam timetabling problems. Different groups of criteria (with 9 criteria in total) have been introduced whose values describe the quality of the timetables from different aspects:

(a) *Room capacities*. Criterion C_1 expresses the number of students who cannot be seated in the assigned rooms in the timetable.
(b) *Proximity of exams*. Criteria C_2, C_3, C_4 and C_5 give the number of conflicts where students have two exams in adjacent periods, on one day, in adjacent days, and in overnight adjacent periods, respectively.
(c) *Time and order of exams*. Criterion C_6 represents the number of students' enrollments affected when an exam is not scheduled in the time period of appropriate duration. Criterion C_7 represents the number of students' enrollments when an exam is not scheduled in the desired time period. Criteria C_8 and C_9 represent the number of students' enrollments when exams do not satisfy temporal relationships before/after and immediately before/after, respectively.

The criteria are of a different nature. They are incommensurable due to their different units of measure with different scales. In addition, criteria are partially or totally conflicting. For example, two exams which should be scheduled immediately before/after each other may have common students, thus causing two criteria C_2 and C_9 to be conflicting. Criteria weights, which represent relative importance of criteria, are subjectively defined by the timetable officer. They reflect different regulations and requirements of universities and their attitudes toward certain timetabling constraints.

Each timetable is represented as a point in the criteria space whose dimension is equal to the number of criteria. A point in the criteria space, which optimises all criteria simultaneously, is called an ideal point. It is generally the case that the solution that corresponds to the ideal point does not exist. The difficulties caused by different units of measure and different scales of criteria are overcome by the introduction of a preference space in which criteria values are comparable (Petrovic and Petrovic, 1995). A linear mapping of the criteria space into the preference space is defined. The ideal point I in the criteria space is mapped onto the point W in the preference space whose co-ordinates are equal to the weights of criteria.

A new algorithm has been developed for heuristic search of the preference space. It is based on the principal idea of compromise programming – a multicriteria decision making method that attempts to determine so-called compromise solutions with respect to all criteria simultaneously that are closest to the ideal point (Zeleny, 1973, 1974). A family of L_p metrics which has been used to measure the distance between the solutions and the ideal point is defined by the following formula:

$$L_p(S,W) = \left\{ \sum_{k=1}^{K} [s_k - w_k]^p \right\}^{1/p}, \quad (1)$$

where K is a number of criteria, p is a parameter to control the metrics and $L_p(S,W)$ is the distance between a solution S with co-ordinates s_k in the preference space and the ideal point W with the co-ordinates w_k. Three values of p are of particular interest: $p = 1$, $p = 2$, and $p = \infty$.

The algorithm starts the search of the preference space from a set of high quality timetables in terms of each criterion separately. The procedure then iteratively searches the neighbourhood of each of these timetables to improve other criteria values while approaching the ideal point. Two operators have been used to explore the neighbourhood of the timetables. They are based on the concepts of hill-climbing and the heavy mutation operators employed in the memetic algorithm described in Section 2. These operators have been modified to take into consideration the distances of the timetables from the ideal point.

The multicriteria approach has been tested on the examination data of the University of Nottingham whose description is shown in Table 3.

Experiments have been carried out with two parameters: w_k and p. We present the results of different experiments in Table 4 when all the criteria are of the same importance and when the criterion C_2 (which expresses the number of conflicts where students have two exams in adjacent periods) is more important than the other criteria. The table presents the distances of the obtained timetables from the ideal point in the preference space, and the criteria values of the timetables which are also expressed percentages of the worst possible values of criteria. The higher importance of the criterion C_2 led to solutions with a lower number of violations of the corresponding constraint. Smaller values of p imply that a bad value of one criterion can be offset by a good value of another. Higher values of p do not favour compensation of criteria

Table 3
Characteristics of the constraints of the NOTT examination data (23 periods)

Characteristics of constraints on the NOTT examination data	Value
Number of time periods per day	3
Duration of time periods (hours)	3,2,2
Number of exams which require particular time periods	9
Number of times that exams should be scheduled before/after another	2
Number of times that exams should be scheduled immediately before/after another	1

Table 4
Criteria values of the timetables obtained for different weights of criteria W and different p in L_p metrics

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9
$W = (1,1,1,1,1,1,1,1,1), L_2 = 0.16$								
1038	1111	3518	4804	405	4	0	0	0
(4.3%)	(5.0%)	(8.5%)	(12.1%)	(2.7%)	(0.0%)	(0.0%)	(0.0%)	(0.0%)
$W = (1,2,1,1,1,1,1,1,1), L_2 = 0.18$								
982	713	3511	5216	486	39	0	0	0
(4.0%)	(3.2%)	(8.5%)	(13.2%)	(3.2%)	(2.7%)	(0.0%)	(0.0%)	(0.0%)
$W = (1,10,1,1,1,1,1,1,1), L_2 = 0.23$								
1275	212	3525	6584	855	100	0	0	0
(5.3%)	(1.0%)	(8.5%)	(16.7%)	(5.7%)	(6.9%)	(0.0%)	(0.0%)	(0.0%)
$W = (1,1,1,1,1,1,1,1,1), L_1 = 0.30$								
0	879	3623	6381	264	0	0	0	0
(0.0%)	(4.0%)	(8.8%)	(16.1%)	(1.7%)	(0.0%)	(0.0%)	(0.0%)	(0.0%)
$W = (1,1,1,1,1,1,1,1,1), L_\infty = 0.16, L_\infty(S,W) = \max_k[s_k, w_k]$								
2848	2608	4886	4658	807	170	40	0	0
(11.8%)	(11.8%)	(11.8)	(11.8%)	(5.4%)	(11.8%)	(10.0%)	(0.0%)	(0.0%)

values in the timetable construction and therefore all the criteria values are reasonably good. However, in some cases they are not as good as in solutions for $p = 1$ and $p = 2$.

The initial results of this multicriteria approach to timetabling have confirmed that multicriteria decision making methods can lead to a significant insight into timetabling problems and a flexibility in handling the constraints that is not provided by existing methods. Our future research work will be devoted to the investigation of the possibilities of a multicriteria approach to meta-heuristic methods for timetabling such as genetic algorithms, tabu search, simulated annealing and hybrids.

4. The application of case-based reasoning to timetabling

Very often, a guideline for constructing a timetable in practice is to start from ''last year's'' timetable and make as few changes as possible. Of course, this is appropriate only if there is little change in the problem (such as student demand patterns) from one year to the next. This observation provided some of the motivation for investigating a case-based reasoning approach (CBR) to timetabling problems. CBR has been successfully applied to various scheduling problems including the planning and scheduling of large-scale airlift operations (Koton, 1989), dynamic job-shop scheduling (Bezirgan, 1993), repair problems in job-shop scheduling (Miyashita and Sycara, 1995) and production planning and control problems (Schmidt, 1998). These applications indicate that CBR is potentially a valuable tool in solving timetabling problems.

Following the basic idea behind case-based reasoning (Kolodner, 1993), a number of previously solved timetabling problems are stored in a case base. These are used for constructing solutions for new timetabling problems. The following issues have been considered (Burke et al., 2000):

1) How to represent complex timetabling problems.
2) How to organise timetabling problems into a case base.
3) How to define a similarity measure which enables the retrieval of the timetable from a case base which is the most useful for solving a new problem.
4) How to adapt the solution of the retrieved case to meet the requirements of the new timetabling problem.

These issues are discussed here using course timetabling problems as an example.

4.1. Representation of the cases

A list of (feature, value) pairs is often used in case-based reasoning to represent cases (Leake, 1996). However, such data structures are inadequate for representing complex timetabling problems with various relations between courses.

Attribute graphs have been used to represent timetables structurally. Vertices of the graph represent courses while edges represent relations between the courses. Attributes have been assigned to both nodes and edges to represent constraints on courses and on pairs of courses, respectively. For example, attributes assigned to vertices may denote that the course should be held once per week, or n times per week, that the course should be scheduled in a particular time period, etc. Attributes of the edges may denote that two courses are in conflict with each other (i.e., they have common students), that one course should be held before/after the other, that two courses should be consecutive to each other, etc.

4.2. Organisation of the case base

The attribute graphs are represented by adjacency matrices whose diagonal elements contain the attributes of the vertices, while the other elements denote the attributes of the edges between the corresponding vertices. In order to handle different orders of courses of a new problem and the timetabling problems stored in the case base, an adjacency matrix has been introduced for each permutation of the courses. Timetabling problems are stored hierarchically in a decision tree. Nodes and branches of the decision tree store the attributes of the attribute graphs which represent the timetabling problems. In this way the decision tree stores the information about the structures of the timetabling problems contained in the case base.

4.3. Retrieval process and the similarity measure

The goal of the retrieval process is to find a case from the case base which is structurally the most similar to the new timetabling problem. The matching process between two timetabling problems represented by attribute graphs is based on graph isomorphism. An algorithm for graph isomorphism presented in (Messmer, 1995) has been enriched with a similarity measure in order to retrieve the attribute graph which is structurally the most similar to the new problem and thus can be reused to solve the new problem. Instead of using an exact match between the attributes of the new timetabling problem and the attributes stored in the decision tree, a partial match has been introduced which is based on the penalties defined for the pairs of attributes. It has been observed that not only the cases which are graph isomorphic to the new problem can be successfully reused, but also the cases that have (sub)structures which are partially similar to the new problem (Burke et al., 2001b).

In the retrieval process a new timetabling problem is classified to a node in the decision tree. The cases stored with the selected node and the nodes below the selected ones are retrieved. The similarity measure between the new timetabling problem T_2 and the timetabling problem from the case base T_1 is defined in the following way:

$$S(T_1, T_2) = 1 - \frac{\sum_{i=0}^{n} p_i + \sum_{k=0}^{m} a_k + \sum_{l=0}^{k} d_l}{P + A + D}, \quad (2)$$

where

- n is the number of matched attributes between the cases T_1 and T_2;
- p_i is the cost assigned for substituting a vertex or edge of the retrieved case T_1, with a vertex or edge of T_2;
- m and k are the total numbers of the vertices or edges that have to be inserted into and deleted from the retrieved case T_1, respectively, to obtain a match with T_2;
- a_k, d_l are the penalties assigned for inserting and deleting a vertex or edge k and l into and from the retrieved case T_1, respectively, to obtain a match with T_2;
- P is the sum of the penalties for substitution of each possible pair of vertices or edges in the retrieved case T_1 to match T_2;
- A and D are the sums of the costs of inserting and deleting all of the vertices or edges into and from the retrieved case T_1, respectively, to match T_2.

4.4. Adaptation

According to the isomorphism found, the courses in the solution of the retrieved cases are replaced with matching courses from the new problem. Very often the solution of the retrieved case violates some of the constraints of the new timetabling problem. Therefore, it has to be adapted to fulfill the constraints of the new problem. However, as the retrieval process compares the structures of the timetabling cases and the similarity measure takes into consideration how difficult it is to adapt the solution of the retrieved case, the adaptation process usually requires a small number of steps. The courses that violate the constraints are ordered using a graph heuristic method described in Burke et al. (1994) and then rescheduled.

A large number of systematically designed experiments have been performed on a different number of cases (5, 10, 15, 20) with differing complexity (15-course simple, 15-course complex, and 20-course simple cases) in the case base (Burke et al., 2001b). We denote as complex cases those cases which have vertices whose degree is between 1 and 4, while simple cases have vertices whose degree is between 1 and 3. The complex cases are usually more difficult to solve. The generated cases had a range of properties that real-world problems may have and were designed with the aim of investigating how the possible cases in the case base affect the retrieval process. Table 5 shows average percentages of successful retrievals which have found either a partial or a complete match between the new timetabling problem and a case from the case base. Experiments have shown that storing complex cases in the case base enables more successful retrievals than storing simple cases because more new timetabling problems can find matching cases. Also, it is not the number of cases in the case base that is important for the successful retrieval but the number of (sub)structures. For example, the average percentages of successful retrievals are the same independently whether the case base contains 15 or 20 cases of the same complexity. Storing larger (20-courses) simple cases in the case base shows better performance than storing smaller (15-courses) simple cases. However, it seems that storing a certain number of complex cases (here 15-course complex cases) produces a better performance than storing larger cases or simpler cases.

The performance of the developed CBR system has been compared with a graph heuristic method which has been used to construct a timetable from scratch (Burke et al., 2001c). The cost function takes the number of violations of the soft constraints and the number of courses that cannot be scheduled without violation of the hard constraints into account. It has been shown that in the majority of experiments, the CBR approach has produced better solutions with respect to the cost function than those constructed by the graph heuristic method. However, the graph heuristic method outperformed the CBR approach when an insufficient number of complex cases or large number of large cases which are difficult to adapt have been stored in the case base. This is, however, in line with the previous experiments on the size of the case base and the complexity of its cases.

The drawback of the described CBR approach is that it requires all of the permutations of the courses of the timetabling problems to be stored. Therefore it is appropriate to store only small size timetabling problems. However, if the timetabling problem has a larger size than the cases in the case base, the retrieved process is not successful. In order to overcome this, an ongoing research is

Table 5
Average percentages of new timetabling problems that find a match in the case base

Number of cases in the case base	15-course simple case base	15-course complex case base	20-course simple case base
5	76.67%	78.3%	95%
10	90%	90%	95%
15	90%	98.33%	95%
20	90%	98.33%	95%

devoted to a multiple retrieval process which retrieves in each cycle a case from the case base that matches a part of the new problem (Burke et al., 2001c). The vertices of the retrieved case that match the vertices of the new timetabling problem are replaced with a new vertex. Such a constructed graph has been used as an input case in the next cycle of the retrieval process.

5. Directions for future research work

Upon examination of the variety of existing approaches to timetabling, the following question naturally arises: "Where do we stand now with regard to the automatic solution of timetabling problems and in what directions should we aim future research efforts?" Whatever timetabling problem is considered (university exam/course timetabling, employee timetabling, school timetabling etc.) that problem usually varies significantly from institution to institution in terms of specific requirements and constraints. Many current successful university timetabling systems are often applied only in the institutions where they were designed. Indeed, Carter and Laporte (1996) say (of examination timetabling), "There have been hundreds of research papers on the subject and probably thousands of computer programs (mostly by amateurs at each of these schools) to 'solve' their own particular variation on the theme". There is also a very wide variety of approaches (heuristics, meta-heuristics, hybrids, constraint-based methods etc.) that have been investigated and reported in the literature. Many of these methods have been very successful and have helped to increase our understanding of how to solve these difficult real-world problems. However, they are usually approaches that are specifically designed for the particular version of the timetabling problem that is being addressed. There has been little timetabling research work that has been carried out on how to develop systems that can intelligently choose the right method in the right situation. There is currently a school of thought within the timetabling community that believes that, rather than developing a specific algorithm for particular problems, the next significant step should be taken to investigate methodologies for automatically and intelligently choosing an appropriate algorithm for the problem in hand. For example, Ross et al. (1998) say (in terms of genetic algorithms for timetabling), "..... However, all this naturally suggests a possibly worthwhile direction for timetabling research involving Genetic Algorithms (GAs). We suggest that a GA might be better employed in searching for a good algorithm rather than searching for a specific solution to a specific problem".

It is, of course, very expensive to build special purpose timetabling systems for one institution or narrow class of problems. One of the overall motivations for this line of work is that more general systems built by employing such methods would be much less expensive and would still address the issues that are of concern to practitioners. Such people do not usually care about provable optimality or about algorithms that can shave a few percent off the current best approaches on benchmark problems. They usually care about "good enough – soon enough" solutions (Ross, 2001) and a major challenge is to develop approaches that can generate such solutions within a generalised framework that can handle a wide range of problems. It would be difficult for a "general" algorithm to beat a special purpose algorithm on its own problem but it would be a major achievement if the "general" algorithm could compete with the special purpose algorithms for a wide range of problems.

The question of how to develop approaches that can generate algorithms for a range of timetabling problems raises a very important issue for the timetabling (and indeed the wider scheduling community) to address. One of our main motivations for future timetabling research is to investigate ways of building systems that can operate at a higher level of generality than current timetabling methods can support. We will very briefly outline two major directions in our future research work:

1. *A case-based reasoning approach to heuristic selection*. Earlier on in this paper we described a case-based reasoning approach to directly solve course scheduling problems. This was, essentially, motivated by the observation that timetabling

officers tend to re-use timetables where possible. Timetabling researchers also tend to re-use methods that they have found to be effective on problems that they consider to have certain similarities. This provided us with the motivation for one of our major research initiatives: To explore case-based reasoning approaches for picking out a suitable heuristic for a particular timetabling problem. The basic idea is to generate a case base which stores a variety of cases. Each case would include a timetabling problem (together with an algorithm for solving it). When presented with a new problem, the approach would consult the case base and (based on previous experience) it would pick out suitable heuristics for the new problem. Our aim is to investigate the circumstances under which specific approaches are appropriate for particular instances of the problem. We believe that the meta-heuristic, heuristic and hybrid methods used for solving previous timetabling problems can be re-used in solving new timetabling problems which are "similar" (in a certain sense). Of course, the question of when two timetables are "similar" in terms of when particular methods work well on both of them is a major research issue in itself. Indeed, we see it as one of the main challenges that needs to be addressed in establishing a case-based approach to heuristic selection. However, it is certainly not the only research challenge. The establishment of a case-based reasoning methodology will also involve: Defining a set of training cases, organisation of the case base, evaluation, learning from failure (as well as success) and controlling the growth of the case base.

2. *Hyper-heuristic methods*. A hyper-heuristic denotes a heuristic that selects heuristics for a wide variety of problems, including timetabling. Note that this differs from the widely used term *meta-heuristic* in that the term *meta-heuristic* usually refers to a heuristic which manages one other heuristic for a particular problem. A hyper-heuristic can be thought of as a *heuristic to choose heuristics*. Of course, meta-heuristics can be used as hyper-heuristics as Ross et al. (1998) suggest when they discuss a genetic algorithm to find suitable timetabling heuristics. It may also be the case that a hyper-heuristic chooses from a range of heuristics, meta-heuristics and hybrids. The main idea is to try and design an algorithm that will choose the right algorithm to carry out a certain task in a certain situation. There are a number of research issues that need to be tackled. In particular, we need to investigate and analyse what methods may or may not be successful as hyper-heuristics and we need to investigate the use of "knowledge poor" algorithms on a range of problems. This work, of course, complements our research on a case-based approach to heuristic selection and is being carried out in collaboration with Peter Ross and his research team.

Acknowledgements

We would like to thank Prof. J. Krarup and the Programme Committee of EURO 2000 for inviting us to present this work in a semi-plenary talk. We would also like to thank Prof. Krarup, Dr. J.P. Newall and the anonymous referees for their helpful comments in the preparation of this paper. Some of the research described in this paper was funded (or is being funded) by the UK Engineering and Physical Sciences Research Council (EPSRC).

References

Balakrishnan, N., Lucena, A., Wong, R.T., 1992. Scheduling examinations to reduce second-order conflicts. Computers and Operations Research 19, 353–361.

Bardadym, V.A., 1996. Computer-aided school and university timetabling: The new wave. In: Burke and Ross (1996) pp. 22–45.

Bezirgan, A., 1993. An application of case-based expert system technology to dynamic job-shop scheduling. In: Bramer, M.A., Milne, R.W. (Eds.), Research and Development in Expert Systems IX, Proceedings of Expert Systems 92, The Twelfth Annual Technical Conference of the British Computer Society, Specialist Group on Expert Systems, 15–17 December 1992, London, UK, pp. 225–235.

Brailsford, S.C., Potts, C.N., Smith, B.M., 1999. Constraint satisfaction problems: Algorithms and applications. European Journal of Operational Research 119, 557–581.

Brelaz, D., 1979. New methods to color the vertices of a graph. Communications of the ACM 22 (4), 251–256.

Burke, E., Carter, M. (Eds.), 1998. The Practice and Theory of Automated Timetabling II: Selected Papers from the 2nd International Conference on the Practice and Theory of Automated Timetabling, University of Toronto, August

20–22, 1997. Springer Lecture Notes in Computer Science Series, vol. 1408.

Burke, E., Ross, P. (Eds.), 1996. The Practice and Theory of Automated Timetabling: Selected Papers from the 1st International Conference on the Practice and Theory of Automated Timetabling, Napier University, August/September 1995. Springer Lecture Notes in Computer Science Series, vol. 1153.

Burke, E.K., Newall, J.P., 1999. A multi-stage evolutionary algorithm for the timetable problem. IEEE Transactions on Evolutionary Computation 3 (1), 63–74.

Burke, E., Erben, W. (Eds.), 2001. The Practice and Theory of Automated Timetabling III: Selected Papers from the 3rd International Conference on the Practice and Theory of Automated Timetabling, University of Applied Sciences, Konstanz, August 16–18, 2000. Springer Lecture Notes in Computer Science Series vol. 2079.

Burke, E.K., Elliman, D.G., 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., Elliman, D.G., Weare, R.F., 1995. A hybrid genetic algorithm for highly constrained timetabling problems. In: Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, USA, 15–19 July 1995. Morgan Kaufmann, Los Altos, CA, pp. 605–610.

Burke, E.K., Elliman, D.G., Ford, P., Weare, R.F., 1996a. Examination timetabling in British Universities – A survey. In: Burke and Ross (1996) pp. 76–92.

Burke, E.K., Newall, J.P., Weare, R.F., 1996b. A memetic algorithm for University exam timetabling. In: Burke and Ross (1996) pp. 241–250.

Burke, E., Kingston, J., Jackson, K., Weare, R., 1997. Automated university timetabling: The state of the art. The Computer Journal 40 (9), 565–571.

Burke, E.K., Newall, J.P., Weare, R.F., 1998a. A simple heuristically guided search for the timetable problem. In: International ICSC Symposium on Engineering of Intelligent Systems EIS'98. ICSC Academic Press, New York, pp. 574–579.

Burke, E.K., Newall, J.P., Weare, R.F., 1998b. Initialisation strategies and diversity in evolutionary timetabling. Evolutionary Computation 6 (1), 81–103 (special issue on Scheduling).

Burke, E., MacCarthy, B., Petrovic, S., Qu, R., 2000. Structured cases in CBR – Re-using and adapting cases for timetabling problems. Knowledge-Based Systems 13 (2-3), 159–165.

Burke, E., Bykov, Y., Petrovic, S., 2001a. A multicriteria approach to timetabling problems. In: Burke and Erben (2001), pp. 118–131.

Burke, E., MacCarthy, B., Petrovic, S., Qu, R., 2001b. Case-based reasoning in course timetabling: An attribute graph approach. In: Aha, D.W., Watson, I., Yang, Q. (Eds.), Case-Based Reasoning Research and Development, Proceedings of the 4th International Conference on Case-Based Reasoning, ICCBR-2001, Vancouver, Canada, 30 July–2 August 2001. Springer-Verlag Lecture Notes in Artificial Intelligence vol. 2080, pp. 90–104.

Burke, E., MacCarthy, B., Petrovic, S., Qu, R., 2001c. Multi-retrieval in a structured case based reasoning approach for course timetabling problems. School of Computer Science and IT Technical Report, University of Nottingham, 2001.

Carter, M.W., 1983. A decomposition algorithm for practical timetabling problems. Technical Paper 83-06, Department of Industrial Engineering, University of Toronto.

Carter, M.W., 1986. A survey of practical applications of examination timetabling algorithms. Operations Research 34, 193–202.

Carter, M.W., Laporte, G., 1996. Recent developments in practical examination timetabling. In: Burke and Ross (1996) pp. 3–21.

Carter, M.W., Laporte, G., 1998. Recent developments in practical course timetabling. In: Burke and Carter (1998) pp. 3–19.

Carter, M.W., Laporte, G., Lee, S.Y., 1996. Examination timetabling: Algorithmic strategies and applications. Journal of the Operational Research Society 74, 373–383.

de Werra, D., 1985. An introduction to timetabling. European Journal of Operational Research 19, 151–162.

Di Gaspero, L., Schaerf, A., 2001. Tabu search techniques for examination timetabling. In: Burke and Erben (2001), pp. 104–117.

Fisher, J.G., Shier, D.R., 1983. A heuristic procedure for large-scale examination scheduling problems. Technical Report 417, Department of Mathematical Sciences, Clemson University.

Fogarty, T.C. (Ed.), 1995. Evolutionary Computing, Timetabling Section (3 papers). Springer LNCS, vol. 993.

Kolodner, J., 1993. Case-Based Reasoning. Morgan–Kaufmann, Los Altos, CA.

Koton, P., 1989. SMARTPlan: A case-based resource allocation and scheduling system. In: Hammond, K. (Ed.), Proceedings: Workshop on Case-Based Reasoning (DARPA), Pensacola Beach, Florida, San Mateo, CA. Morgan–Kaufmann, Los Altos, CA, pp. 285–289.

Leake, D. (Ed.), 1996. Case-Based Reasoning, Experiences and Future Directions. AAAI Press.

Messmer, B.T., 1995. Efficient graph matching algorithms for preprocessed model graph. Ph.D. Thesis, University of Bern, Switzerland.

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

Moscato, P., Norman, M., 1992. A "Memetic" approach for the travelling salesman problem – Implementation of a computational ecology for combinatorial optimisation on message passing systems. In: Proceedings of the International Conference on Parallel Computing and Transputer Applications. IOS Press, Amsterdam, pp. 177–186.

Newall, J.P., 1999. Hybrid methods for automated timetabling. Ph.D. Thesis, Department of Computer Science, University of Nottingham, UK.

Paechter, B., Cumming, A., Luchian, H., 1995. The use of local search suggestion lists for improving the solution of timetabling problems with evolutionary algorithms. In: Burke and Ross (1996) pp. 86–102.

Paechter, B., Cumming, A., Norman, M.G., Luchian, H., 1996. Extensions to a memetic timetabling system. In: Burke and Ross (1996) pp. 251–266.

Petrovic, S., Petrovic, R., 1995. Eco-Ecodispatch: DSS for multicriteria loading of thermal power generators. Journal of Decision Systems 4 (4), 279–295.

Ross, P., 2001. Tutorial on evolutionary scheduling and routing. In: 2001 Genetic and Evolutionary Computation Conference, San Fransisco, 2001, Tutorial Program, pp. 193–210.

Ross, P., Hart, E. Corne, D., 1998. Some observations about GA-based exam timetabling. In: Burke and Carter (1998) pp. 115–129.

Schaerf, A., 1999. A survey of automated timetabling. Artificial Intelligence Review 13 (2), 87–127.

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

White, G.M., 2000. Constrained satisfaction, not so constrained satisfaction and the timetabling problem. In: A Plenary Talk in the Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling, University of Applied Sciences, Konstanz, August 16–18, 2000, pp. 32–47.

White, G.M., Chan, P.W., 1979. Towards the construction of optimal examination timetables. INFOR 17, 219–229.

Wren, A., 1996. Scheduling, timetabling and rostering – A special relationship? In: Burke and Ross (1996) pp. 46–75.

Zeleny, M., 1973. Compromise programming. In: Cochrane, J.L., Zeleny, M. (Eds.), Multiple Criteria Decision Making. University of South Carolina Press, Columbia, pp. 262–301.

Zeleny, M., 1974. A concept of compromise solutions and the method of displaced ideal. Computers and Operations Research 1 (4), 479–496.