A time-predefined local search approach to exam timetabling problems

EDMUND BURKE[1], YURI BYKOV[1], JAMES NEWALL[2] and SANJA PETROVIC[1]

[1] *Automated Scheduling, Optimisation and Planning Group, School of Computer Science & IT, University of Nottingham, Wollaton Road, Nottingham NG8 1BB, UK*

E-mail: {ekb,yxb,sxp}@cs.nott.ac.uk

[2] *EventMAP Limited, 21 Stranmillis Road, Belfast BT9 5AF, N. Ireland*

E-mail: Jim.Newall@eventmaponline.com

Abstract

In recent years the computational power of computers has increased dramatically. This in turn has allowed search algorithms to execute more iterations in a given amount of real time. Does this necessarily always lead to an improvement in the quality of final solutions? This paper is devoted to the investigation of that question. We present two variants of local search algorithms where the search time can be set as an input parameter. These two approaches are: a time-predefined variant of simulated annealing and an adaptation of the “Great Deluge” method. We present a comprehensive series of experiments, which show that these approaches significantly outperform the previous best results (in terms of solution quality) on the most popular benchmark exam timetabling problems. Of course, there is a price to pay for such better results: increased execution time. We discuss the impact of this trade-off between quality and execution time. In particular we discuss issues involving the proper estimation of the algorithm’s execution time and the assessment of its importance.

1. Introduction

1.1 *The Exam Timetabling Problem*

The University Examination Timetabling Problem has attracted significant research interest over the years. Its various instances usually appear as large-scale, highly constrained and difficult to solve NP-hard problems. These problems are often varied in their structure, which can contribute to making them a difficult class of problems, offering some serious research challenges. From a practical point of view they are also very important problems. The quality of solutions to these problems often has a significant impact on the institutions concerned.

At its most basic, the exam timetabling problem is concerned with distributing a collection of university exams among a limited number of timeslots (periods). This is, of course, subject to a set of regulations and limitations (often termed constraints), which vary widely from institution to institution. There are certain constraints which must be satisfied under any circumstances such as the requirement that no student can sit two exams simultaneously, or that exam rooms have a certain physical capacity which must not be exceeded. Such constraints are known as "hard". Solutions, which satisfy all the hard constraints, are often called "feasible" solutions.

In addition to the hard constraints there are usually various constraints that are considered to be desirable but not essential. These are often called "soft". Of course, there is significant difference across institutions as to which constraints they consider important and which they do not. The situation in British universities is discussed in more detail in (Burke et al. 1996) which analyses the responses of over 50 British universities to a questionnaire on exam timetabling. Examples of commonly occurring soft constraints can reflect the situation where students prefer to spread exams evenly throughout the examination session or at least have some time interval between exams. On the other hand, the institution often wants to schedule large exams

earlier (in order to leave more time for marking). Specific preferences may also be expressed by particular members of staff concerning, for example, invigilation duties. Of course, in any real world situation it would be extremely rare if it were possible to satisfy all the soft constraints. Therefore, a useful measure of the quality of a timetable can be taken to be the number of violations of these constraints. Minimising these violations is often one of the over-riding objectives for development of software systems to solve the problem. Traditionally the violated soft constraints are aggregated into an objective function (cost function or "fitness" or “penalty”), which serves as an index of the solution quality. Thus, the goal of the examination timetabling process can be taken to be that of producing the feasible timetable of the highest possible quality (minimum value of the particular cost function under consideration).

Managing a high variety of different constraints is quite a difficult task. Every additional constraint can increase the total complexity of the problem and can make the solution more resource-consuming. Therefore, in the real-world, there is often a high level of user-intervention and relaxation of constraints. Often, it becomes clear during the timetabling process that the particular problem instance in hand is over-constrained.

What we are aiming for in this paper is a mechanism that allows a user to define a certain period of time in which the algorithm should run in order to try to find a high quality solution. This mechanism should ensure that the algorithm searches in an intelligent manner for the specified amount of time. We do not want the algorithm to converge on a solution prematurely. We want the algorithm to use all of its specified time in trying to improve the solution. The overall motivation behind the techniques described in this paper is that we want to be able to employ as much (or as little) computing resource as the user may desire to find the level of solution quality that the user is happy to pay for (in terms of computational time).

The paper is organised as follows. In the next section we present an overview of algorithmic approaches to exam timetabling. Section 2 discusses the importance of time

characteristics during search and presents two variants of time-predefined algorithms. In Section 3 we investigate the properties of the proposed techniques and compare their performance with the current state-of-the-art in exam timetabling. In section 4, we discuss the question of how to effectively evaluate and compare results produced by algorithms with and without time-predefinition.

1.2 *Algorithmic Approaches to Examination Timetabling*

One of the major approaches in exam timetabling over the years has been the investigation of constraint based approaches. Examples of such approaches are provided by Boizmault, Delon and Peridy (1996), David (1998) and Reis and Olivera (2000). While such approaches play an important role in the examination timetabling literature, they have little direct impact on the research work that is presented in this paper.

Some early approaches to solving the problem which do, in a certain sense, underpin the research presented in this paper were developed in the 1960's. These approaches tended to concentrate on the hard constraint which says that, "exams in the same period should not have common students". The generation of such clash-free timetables is analogous to solving the classical graph colouring problem where the vertices correspond to examinations, edges between two vertices indicate the presence of common students who should attend both exams and every colour conforms to a particular time slot. Correspondingly, the methods for solving such timetabling problems are based upon methods for solving graph colouring problems. Indeed, in recent times such approaches have been hybridised with modern meta-heuristics to produce high quality solutions (eg. (Burke Newall and Weare 1998), (Burke and Newall 1999), (Di Gaspero and Shaerf 2001)). In essence, these basic graph colouring based methods fill a blank timetable with exams, taken in a certain order. The particular methods of ordering vary according to the different heuristics employed. More details are given in the following surveys (Carter 1986), (Carter and Laporte 1996). Of course, the ordering motivation is to place firstly

the most “difficult” exams depending upon your measure of “difficulty”. As a measure of this “difficulty”, several authors (Cole (1964), Peck and Williams (1966)) have used the degree of each vertex (the number of conflicting exams). Another example is the sum of degrees of adjacent vertices (Williams 1974). The inverse ordering (recursive removing of exams with the smallest degree) was proposed by Matula, Marble and Isaacson (1972). Possibly the most successful idea in this respect involves the calculation of the exam's degree with the number of available slots and its recalculation after each placement (saturation degree) (Brelaz 1979). Further modifications of sequencing heuristics use backtracking (rescheduling of certain exams in conflict situations) and can consider some soft constraints. A comprehensive investigation of their effectiveness with a collection of real-world examination timetabling problems is presented in (Carter, Laporte and Chinneck 1994), (Carter, Laporte and Lee 1996).

As briefly alluded to above, in addition to a straightforward application to simplified graph colouring analogous timetabling problems, such heuristics can be incorporated into more recent "metaheuristics" techniques. These approaches can represent the gradual improvement of a current solution (or solutions) starting from initial one(s) until some stopping condition is satisfied. Some of these metaheuristics are based upon a simple approach called "hill-climbing". This algorithm iteratively inspects the neighbourhood (the set of candidate solutions, produced from the current one by swapping of exams or by certain other modification) and replaces the current solution by the candidate with better fitness. It is a very fast algorithm but due to relatively poor performance it is no longer used (on its own) as a serious approach for solving real world timetabling problems except as a comparative measure. However, the approach, along with some level of hybridisation still has a role to play in modern research as is evidenced by Burke, Bykov and Petrovic (2001) who hybridised hill climbing with a mutation operator while investigating a multiobjective approach to examination timetabling.

A more fruitful idea is to allow the occasional acceptance of solutions, which are worse than the current one. Of course, this prolongs the search time but can lead to better final results. This basic mechanism is implemented in “simulated annealing”, which is one of the most well studied metaheuristics and was presented by Kirkpatrick, Gellat and Vecci (1983). Here, the candidate solutions with worse objective function values than the current one are accepted with probability $P=e^{-\delta/T}$ where δ is the difference of the values of the cost function between the current and the candidate solutions and T is a parameter called the “temperature”, which usually gradually reduces during the search. The reduction scheme that is employed is known as the “cooling schedule”.

Over the last few years simulated annealing has been investigated for examination timetabling with some level of success. In 1996 Thompson and Dowsland considered an adaptive cooling technique, where the temperature is automatically reduced or increased depending upon the success of the move (Thompson and Dowsland 1996a). The same authors proposed launching the algorithm several times starting from a random seed, temporary allowing unfeasible solutions under a certain threshold and increasing the size of the neighbourhood using Kempe chains (Thompson and Dowsland 1996b). The last idea was expanded into the use of S-chains (ordered lists of S colours) (Thompson and Dowsland 1998). In 1998 Bullnheimer investigated simulated annealing while dividing the problem into several subproblems (Bullnheimer 1998).

Another metaheuristic, which can be considered to be based on hill climbing, is known as "tabu search". The basic idea was proposed by Glover (1986). The overall defining feature of this approach is the keeping of a list of previous moves or solutions (a "tabu list") in order to avoid cycling. It has been successfully applied to timetabling by Hertz (1991) and Boufflet and Negre (1996) among others (see (Shaerf 1999)). In (White and Xie 2001) the authors demonstrated a frequency-based long-term memory mechanism, which restricted the

movement of over-active exams and vice-versa forced the movement of exams with low activity. "Tabu relaxation" (emptying the tabu list after a number of idle moves) was also suggested as a way to move the searched region into one where better solutions could perhaps be found. The benefits of the use of a variable length tabu list in examination timetabling were investigated in (Di Gaspero and Shaerf 2001) where the authors presented a comparison of their approach with other techniques by (Carter, Laporte and Lee 1996) and (Burke and Newall 1999).

Probably the most attention in examination timetabling over the last decade has been in exploring evolutionary solving methods. Corne, Fang and Mellish suggested the use of square pressure of fitness, elitism (keeping the best solutions into later generations) and fixed point uniform crossover (Corne, Fang and Mellish 1993), delta-evaluation for fast calculation of fitness (Corne, Ross and Fang 1994) and "peckish" initialisation (i.e. partially greedy algorithms being used in the timetabling process) (Corne and Ross 1996). Burke et al. considered special crossover and mutation operators (Burke et al. 1995a), (Burke et al. 1995b), based on graph colouring heuristics. Ergul (1996) improved the mutation operator with a certain mechanism for ranking the solutions and he penalised conflicts in order to better satisfy the preassignment constraints. An advanced representation of the problem's structure was presented by Erben (2001) who suggested employing a grouping genetic algorithm for examination timetabling.

The performance of genetic algorithms in examination timetabling (without special recombinative operators) was compared with hill-climbing and simulated annealing by Ross and Corne (1995). Both of these techniques produced better results than the genetic algorithm. One of the ways of improving the performance of this method is by hybridising with other techniques. The memetic algorithm (which can be considered to be a hybrid of a genetic algorithm and a local search operator) for examination timetabling problem was presented by

Burke, Newall and Weare (1996) who also proposed an advanced initialisation strategy, which involved the inclusion of a random aspect into graph colouring heuristics. This particular memetic algorithm hybridised a "mutation only" genetic algorithm with hill climbing. The effect of heuristic seeding was presented in a detailed investigation in (Burke, Newall and Weare 1998) which explored the use of different diversity measures. Another evolutionary approach to examination timetabling is to investigate evolutionary algorithms for selecting the right heuristic (Terashima-Marin, Ross and Valenzuela-Rendon 1999). Indeed, this is a major research direction that Burke and Petrovic discuss in (Burke and Petrovic 2002).

In 1999, Burke and Newall (Burke and Newall 1999) presented an approach to the examination timetabling problem, which decomposed the larger problem into a series of smaller subproblems. The subproblems were ordered by "how difficult" each exam in the subproblem was (to schedule). This difficulty measure was provided by graph colouring heuristics. In addition a "look ahead" technique was used where the current subproblem was not fixed in place until the next one had been dealt with. The motivation here was to try and avoid situations where scheduling decisions that were taken in an earlier subproblem would lead to infeasibilities in later subproblems. A memetic algorithm was employed on the subproblems. This approach produced the best published results on certain benchmark problems at the time. The techniques described in this paper improve these results further.

This paper is important because it introduces new examination timetabling algorithms which consider computational expense as a major input parameter. In particular, it introduces an adapted version of the Great Deluge Algorithm for exam timetabling. This algorithm requires just two input parameters: computational time (that the user is willing to spend) and an approximation of the objective function value that would be desirable. This employment of just two input parameters represents a significant achievement in itself, particularly as the parameters are measures that real world users can readily understand (time and desired solution

quality) rather than abstract concepts (such as number of generations in a genetic algorithm, or cooling rate for simulated annealing etc). However, this paper also makes a significant contribution to the timetabling literature in that it demonstrates that the performance of a local search algorithm can be significantly improved by incorporating a controlled management of the processing time into the approach. This is evidenced by our results which show that the algorithm produces the best published results on a range of well known exam timetabling benchmark problems. The approach concentrates upon the exam timetabling problem but there is significant scope to investigate this approach for other optimisation problems.

2. Time Characteristics of the Search Process

2.1 *The Importance of Time*

The "trade-off" between the quality of solution and the search time in examination timetabling has been discussed in several papers. In (Thompson and Dowsland 1996a) the authors demonstrated that a longer search produced better results. The "tabu relaxation" method presented in (White and Xie 2001) is also a certain way of prolonging the search process in an attempt to improve the quality of the overall solution.

This proposition seems to be logical: the longer search allows the exploration of a greater part of the search space and, thus, the probability of reaching a good solution is increased. The main challenge is to ensure that the approach does not converge too quickly and that all the allocated time is used to intelligently explore the search space.

The prolonging of the search process is also motivated by the progress of hardware facilities. Let us suppose that the real processing time (T_p) is calculated by the formula: $T_p = T_{mov} * N_{mov}$ (where T_{mov} is the time required for one move (iteration) and N_{mov} is the number of moves). The first factor (T_{mov}) depends on the size of a particular problem as well as the particular environment in which the algorithm is launched. Factors that could affect time

include computer hardware, the operating system, the compiler and programming style. A detailed study of these factors exceeds the scope of this paper. However, the increase in the power of computer hardware always leads to a reduction in T_{mov} . This is due not only to the processor speed, but also to an increase in the amount of RAM (avoids relatively slow dynamic reallocations of memory), to widening the set of Assembler operators for new processors, etc. Thus, while using more powerful computing facilities, the increasing of N_{mov} can be compensated by reducing T_{mov} and therefore the prolongation of the search time can be less tangible or even virtual. Thus managing the search time promotes the optimal utilisation of computational resources.

In different real situations when computational time and the quality of solution are dependent upon each other a user can attempt to find some preferable balance between their values. In some cases the user needs an “average quality” result very quickly, but in other cases the user may want to spend more time to improve the solution. A certain estimation of the importance of computing time can be carried out in the context of attendant processes. For example, let us say that an examination timetable has to be compiled twice a year and preparation of input data and utilising the results often requires several days. In such an environment it is obvious that a computing period of 3 seconds or 3 minutes will make insignificant difference to the time taken by the timetabling process as a whole. The difference becomes important when the computing time reaches 3 hours or 3 days say, as it begins to significantly increase the time taken by the whole process. A computing time of, say, 3 weeks would mostly be regarded as unacceptable. If computing time becomes a significant part of the time taken for the whole process of developing a timetable it should obviously be taken into account by the user when planning the complete administrative process. Thus it is safe to assume that for the purpose of improving the result a reasonable prolongation of computing time can be acceptable and indeed desirable to a user who has catered for a significant amount

of time in the overall administrative plan. Of course, if the algorithm requires several hours, it is fairly painless for the user to launch it overnight or over a weekend in order to obtain the result at the beginning of the next working day.

2.2 *Time-Predefined Simulated Annealing*

In order to run a simulated annealing algorithm for a given number of steps the user should precisely determine the required parameters. An approximate indication of such parameters is not sufficient. Even a small deviation of parameter values can cause a dramatic deterioration in time spent. Also, experimental adjustment of parameter values by manual tests is not practical given the (often) high computational expense of a single run. Although different parameters have some influence on search time, they may not be suitable for its direct regulation or calculation. What is required is an additional time-predefinition mechanism, which guarantees reaching convergence in the given time.

To make simulated annealing able to produce a result in a given number of moves we slightly modify the algorithm. These modifications relate to an acceptance condition only and do not affect the initialisation and neighbourhood aspects, which are discussed in Sections 3.3 and 3.4. We use the basic geometric cooling algorithm (Reeves 1996), which stops when reaching a certain temperature (which we call T_f). The fact that T_f can be obtained from the initial temperature T_0 by multiplying it by α (where α is a value yet to be determined) during a desired number of steps N_{mov} can be expressed by the following formula.

$$T_f = T_0 \cdot \alpha^{N_{mov}} \tag{1}$$

From this equation the necessary value of α can be expressed as:

$$\alpha = \exp\left(\frac{\ln T_f - \ln T_0}{N_{mov}} \right) \tag{2}$$

To enable slow cooling it is desirable that α is close to one and using $\lim_{x\to 0} e^x = 1$ the formula (2) can be approximated to the more simple expression:

$$\alpha = 1 - \frac{\ln(T_0) - \ln(T_f)}{N_{mov}} \tag{3}$$

Using either rule (2) or (3) we can now define a value for the parameter α based on the predefined time that we want the simulated annealing to run for. But this algorithm still requires the determination of both temperatures. Their values are problem-specific and conventionally they are estimated while using general empiric rules. For T_0 Kirkpatrick (1984) suggested that its value should provide some reasonable probability for acceptance of the average-sized uphill move δ_{av}^{+} at the beginning of the search. The value of the initial temperature can be calculated by formula (4) which is derived from the main condition of acceptance of the Simulated Annealing algorithm:

$$T_0 = \frac{-\delta_{av}^{+}}{\ln(P_0)} \tag{4}$$

In this formula, P_0 denotes the chosen initial probability of acceptance. We have found that this rule is also suitable for our time-predefined version, even though the definition of the best value of P_0 requires several runs of the algorithm.

The situation with the final temperature is less certain. The common suggestion is to choose the value of T_f to small enough to guarantee the convergence of the algorithm. However, this does not normally take into account a restraint on the processing time. In the time-predefined variant of Simulated Annealing the value of T_f may need to be significantly higher, but we cannot give accurate recommendations for its value. This uncertainty complicates the procedure and is, of course, the drawback of this algorithm (see Section 3.5).

2.3 *The Great Deluge Algorithm*

In 1994, Dueck (Dueck 1994) introduced a local search procedure called the “Great Deluge Algorithm”. It was introduced as an alternative to Simulated Annealing. This algorithm like Simulated Annealing may accept worse candidate solutions (than the current one) during its run. The worse solution is accepted if its fitness is less than or equal to some given upper limit B (in the paper by Dueck it was called a “level”). Its value does not depend upon the current solution: at the beginning B is equal to the initial cost function and at every iteration it is lowered by the fixed decay rate ΔB whose value is the only input parameter for this technique.

During the search, a particular value for B makes the corresponding part of the search space infeasible and forces the current solution to "escape" into the remaining feasible region. In other words, the neighbourhood (see Section 3.4) appears to be cut down from one side. Thus the decreasing of B could be thought of as a control process, which drives the search towards a desirable solution. If the controlling process is relatively slow, the current “level” does not exceed the current solution - it only prohibits the longest backward moves. The current solution has the chance to produce several successful moves (in both directions) inside the remaining neighbourhood and improve its value before the “level” comes too close. While approaching the end of the search, the number of possible forward moves in the neighbourhood (and correspondingly the chance of improving the current solution) decreases. Here a large part of the neighbourhood is cut down and the percentage of successful moves is thus reduced. This situation progresses until the “level” eventually passes the current solution.

To manage this situation, we integrated the acceptance of all better moves (the hill-climbing rule) into the basic Great Deluge algorithm. This extension helps the current solution to jump into the unrestricted region (below the “level”). Also, at the end of the search this algorithm has a chance to make a certain improvement of the current solution independently from the current value of B until convergence (when a further improvement becomes

impossible). Thus the stopping criterion of this algorithm can be the same as for Hill-Climbing: no improvement during a given number of steps. The pseudocode of the final variant of our Great Deluge Exam Timetabing algorithm is given in Figure 1.

```
Set the initial solution s
Calculate initial cost function f(s)
Initial level B=f(s)
Specify input parameter ΔB = ?
While further improvement is impossible
   Define neighbourhood N(s)
   Randomly select the candidate solution  s* ∈ N(s)
   Calculate f(s*)
   If  f(s*) ≤ f(s)
   Then accept  s*
   Else if  f(s*) ≤ B
        Then accept  s*
   Lower the level B = B –ΔB
```

Fig. 1. The extended Great Deluge algorithm

In this algorithm the decay rate ΔB actually defines the speed of the "level" reduction. For the desired number of moves N_{mov} its value can be calculated by formula (5).

$$\Delta B = \frac{B_0 - f(s')}{N_{mov}} \tag{5}$$

where B_0 is the starting value of the "level" and $f(s')$ is the cost function of the final result.

The first parameter B_0 is defined exactly (initial cost function), but the final cost function can only be proposed (as the goal value which the user intends to reach by prolonging the computing time). If this goal is completely unknown, a user can apply some quick technique (eg. hill-climbing) for the same problem. Its average-quality result will give an idea of the range of possible solutions. Thus, in contrast to time-predefined simulated annealing, the Great Deluge algorithm allows only an approximate predefinition of the search time. However, practice shows that the possible deviation between the expected solution and the real one is insignificant (relative to the complete search interval). Therefore, the inaccuracy in the predefinition of the operational time does not usually exceed a few percent.

3. Experiments and Results

Both of the described techniques were implemented with Microsoft Visual C++ 6.0 and experiments were undertaken using a PC with an Athlon 750 MHz processor and Windows 98. The overall aims of our experiments were:

- To investigate the properties of the time-predefined techniques by generating the "cost progress" diagrams (see Section 3.5) for the search processes. These diagrams track the evolution of the cost function during the search.

- To explore the manner in which the prolongation of the search can increase the quality of solution and if true, to define the manner of such a feature by plotting the time-cost diagrams. This can be achieved by several runs of the software with different predefined search times (number of steps).

- To evaluate the quality of the results produced by the time-predefined search in an acceptable time by comparison of its range with the outcomes of other techniques applied to the same datasets and published in the literature.

3.1 *Experimental Data*

The examination timetabling research community has established a publically available set of examination timetabling data and supplementary instructions (for example: the way of calculating cost functions, etc.). Our experiments were carried out with datasets, taken from the following open sources:

a) The disposition of exams and students at Nottingham University in 1994, which is available from: ftp://ftp.cs.nott.ac.uk/ttp/Data/Nott94-1/. This dataset includes 800 exams and 7 896 students, which compose 33 997 "student-exam" pairs (enrolments).

b) Michael Carter's collection of examination timetabling data, which can be downloaded from archive at: ftp://ftp.mie.utoronto.ca/pub/carter/testprob, comprising of 13 sets of

examination data, which took place at different Universities during 1983-1993. Their parameters are presented in Table 1.

Table 1. The parameters of Carter's collection of examination datasets

Data set	Institution	Exams	Students	Enrolments
CAR-F-92	Carleton University, Ottava	543	18 419	55 552
CAR-S-91	Carleton University, Ottava	682	16 925	56 877
EAR-F-83	Earl Haig Collegiate Institute, Toronto	189	1 125	8 108
HEC-S-92	Ecole des Hautes Etudes Commercials, Montreal	80	2 823	10 632
KFU-S-93	King Fahd University, Dharan	461	5 349	25 118
LSE-F-91	London School of Economics	381	2 726	10 919
PUR-S-93	Purdue University, Indiana	2419	30 032	120 690
RYE-S-93	Ryeson University, Toronto	481	11 483	45 052
STA-F-83	St Andrew's Junior High School, Toronto	138	611	5 751
TRE-S-92	Trent University, Peterborough, Ontario	261	4 360	14 901
UTA-S-92	Faculty of Arts and Sciences, University of Toronto	638	21 267	58 981
UTE-S-92	Faculty of Engineering, University of Toronto	184	2 750	11 796
YOR-F-83	York Mills Collegiate Institute, Toronto	180	941	6 029

3.2 Definition of the Problems

The input data is given as:

- N is the number of exams;
- M is the number of students;
- P is the given number of timeslots (which are defined below);

- The conflict matrix $C=(c_{ij})_{NXN}$ where each element (denoted by c_{ij} where $i,j \in \{1,\ldots,N\}$) is the number of students that have to take both exams i and j. This is a symmetrical matrix of size N, where diagonal elements C_{ii} equal the number of students who have taken exam i.

The solution of the problem can be represented as a vector $T=(t_k)_N$, where t_k specifies the assigned timeslot for exam k ($k \in \{1,\ldots,N\}$). Each timeslot can be thought of as a non-negative integer ($1 \le t_k \le P$).

The requirement for a clash-free timetable (hard constraint) is expressed by formula (6).

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot clash(t_i, t_j) = 0 \qquad \text{where} \qquad clash(t_i, t_j) = \begin{cases} 1 & if\ t_i = t_j \\ 0 & otherwise \end{cases} \tag{6}$$

In our basic experiments the cost function F_C (formula (7)) summarises the proximity between exams. If a student has two consecutive exams then we assign a penalty value equal to 16. Two exams with one empty period between them will be assigned a penalty value of 8. Two empty periods correspond to a penalty of 4 and so on. In order to have a relative measure, this sum is divided by the total number of students. Thus we are required to minimise expression (7).

$$F_C = \frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij} \cdot prox(t_i, t_j)}{M} \qquad \text{where} \qquad prox(t_i, t_j) = \begin{cases} 2^{5-|t_i - t_j|} & if\ 1 \le |t_i - t_j| \le 5 \\ 0 & otherwise \end{cases} \tag{7}$$

This can be considered to be one of a number of additional requirements (that are characteristic of real timetabling problems). For more advanced experiments (that are closer to the real-world situation) additional input data is specified.

A "room capacities" constraint assumes the specification of the number of seats S that are available for each timeslot. Thus the requirement that the total number of students in any period should not exceed S is expressed by formula (8).

$$\sum_{p=1}^{P} (S_p - S) \cdot exc(S_p) = 0 \qquad \text{where} \qquad exc(S_p) = \begin{cases} 1 & if\ S_p > S \\ 0 & otherwise \end{cases} \tag{8}$$

and where S_p is the number of students taking exams in period p (which is of course a particular timeslot). This can be calculated by the next formula:

$$S_p = \sum_{i=1}^{N} C_{ii} \cdot per(t_i, p) \qquad \text{where} \qquad per(t_i, p) = \begin{cases} 1 & if\ t_i = p \\ 0 & otherwise \end{cases} \tag{9}$$

The vector $D=(d_l)_P$ with elements d_l (where $l \in \{1,\ldots,P\}$) specifies the number (for every timeslot p) which represents the day in an examination session. In our experiments we consider the examination session, to start from Monday and we have 3 exams every day, except Saturdays (only one timeslot) and Sundays (no exams). Thus, the first three timeslots correspond to day "1" (Monday of the 1st week), the 4th,5th and 6th timeslots correspond to day "2" (Tuesday of the 1st week), etc. Day "6" only one timeslot (Saturday of the 1st week), after which the second week starts. This list continues until all given timeslots are represented. Thus the distribution of days can be expressed in the following way:

$$\boldsymbol{D_P} = (1,1,1,2,2,2,3,3,3,4,4,4,5,5,5,6,8,8,8,9,9,9,10,10,10,11,11,11,12,12,12,13,15\ldots) \tag{10}$$

Note, that Sundays (for example: day "7") have a number even though it is not actually used. This is done in order to aid the calculation of overnight conflicts (formula (12)).

For the described type of problem specification the cost function is calculated as a weighted sum of the number of students who have two exams in adjacent periods and overnight. Here the following formulas are useful:

- The number of students having two exams in adjacent periods on the same day F_1 can be found by formula (11).

$$F_1 = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \cdot adjs(t_i, t_j) \qquad \text{where} \qquad adjs(t_i, t_j) = \begin{cases} 1 & if\ \left(\left|t_i - t_j\right| = 1\right) \wedge \left(d_{t_i} = d_{t_j}\right) \\ 0 & otherwise \end{cases} \tag{11}$$

- The number of students having two exams overnight F_2 (adjacent periods at adjacent days except the pair: "Saturday – first slot on Monday") is expressed in formula (12).

$$F_2 = \sum_{i=1}^{N-1} \sum_{j=i+1}^{N} c_{ij} \cdot ovnt(t_i, t_j) \quad \text{where} \quad ovnt(t_i, t_j) = \begin{cases} 1 & if \ \left(\left|t_i - t_j\right| = 1\right) \wedge \left(\left|d_{t_i} - d_{t_j}\right| = 1\right) \\ 0 & otherwise \end{cases} \tag{12}$$

Thus the cost function F_n can be calculated as the weighted sum of those soft constraints (with weights 3 and 1 to represent their relative importance). It can be seen in formula (13) and should be minimized.

$$F_n = 3 \cdot F_1 + F_2 \tag{13}$$

3.3 *Initialisation*

In this paper we do not attempt to investigate the initialisation phase of the presented techniques. It could be the topic of a separate study. However, we expect that the initialisation strategy could have a crucial influence on the performance of the algorithms (as it can for other search methods), especially when the search space is disconnected, which is typical for Examination Timetabling problems. Therefore we made the initial solution as good as possible in as little time as possible and we made it independent of the applied heuristic. In our experiments, for every problem 20 solutions were generated and the one with the minimum cost function was chosen as the initial one.

Those solutions were produced by Brelaz's saturation degree graph colouring sequencing algorithm (Brelaz 1979), which chooses the vertex (exam) with the least number of available colours (timeslots) and assigns the timeslot for it. In order to have different solutions with different runs, the assigned timeslot was chosen randomly among the available ones. This stochastic feature allows us to capture different areas of the search space. The given algorithm produces feasible solutions in a few seconds, so we consider the initialisation time negligible and do not include it in the estimated search time.

3.4 *Neighbourhood Structure*

In this paper we also do not investigate the possible effect of the utilisation of different neighbourhood structures. In the interests of ensuring that the experiments are consistent, we

use the same (most common) variant of the neighbourhood everywhere. Here all candidate solutions can be produced from the current one by a simple replacement of one exam into a different timeslot. An advantage of such a move is a very fast evaluation procedure, i.e. at each iteration, our algorithms calculate only the difference in the cost function caused by this move. This allows us to increase the number of moves produced in a given amount of processing time and hence (in terms of our research) to improve the performance of algorithm. Otherwise, the utilisation of a more advanced neighbourhood (e.g. Kempe chains) can extend the search space and produce better results in the same number of moves. However, these moves can be more computationally expensive, which could mean that the algorithm actually could do "less" in the amount of time given. A detailed investigation into the effect of utilising different neighbourhoods for time-predefined approaches represents a future research direction.

3.5 *Investigating the Properties of the Algorithms*

In the first phase of our experiments the influence of algorithmic parameters was investigated for both of the described algorithms. The algorithms were launched on a series of benchmark datasets while employing the cost function presented by formula (7). In the experiments N_{mov} was determined as 3,000,000. On the graphs presented in Figures 2-3 the *y* axis represents the current cost and the *x* axis represents the current number of moves together with the processing time in seconds. When the figure is labelled with "TPSA" it is describing the time-predefined Simulated Annealing approach and when it is labelled "GD" it describes the Great Deluge algorithm.

The progress diagrams for two sample runs (with different cooling schedules) of the time-predefined Simulated Annealing on the CAR-F-92 problem are shown in Figure 2. To define the values of initial and final temperatures we have carried out the following preliminary manipulations.

The initial temperature was the same for both schedules and was calculated by formula (4). The average uphill move δ_{av}^{+} was defined while recording 1000 uphill moves at the beginning of the search and calculating their average value. Our further experiments showed that for the given problems a highly suitable value of P_0 is around 0.9. In this way we calculated the initial temperatures for all our problems. In particular, for the CAR-F-92 problem, T_0 was set up to be 0.06.

In order to calculate the final temperature, we launched the Simulated Annealing algorithm until no improvement was indicated during 10,000 moves (0.3% of the number of moves of the whole procedure). The algorithm converged at $T_f \approx 1.2*10^{-5}$. Based on this value we calculated the cooling rate of our first sample run by formula (3). The evolution of the cost function is shown as curve *A* in Figure 2.

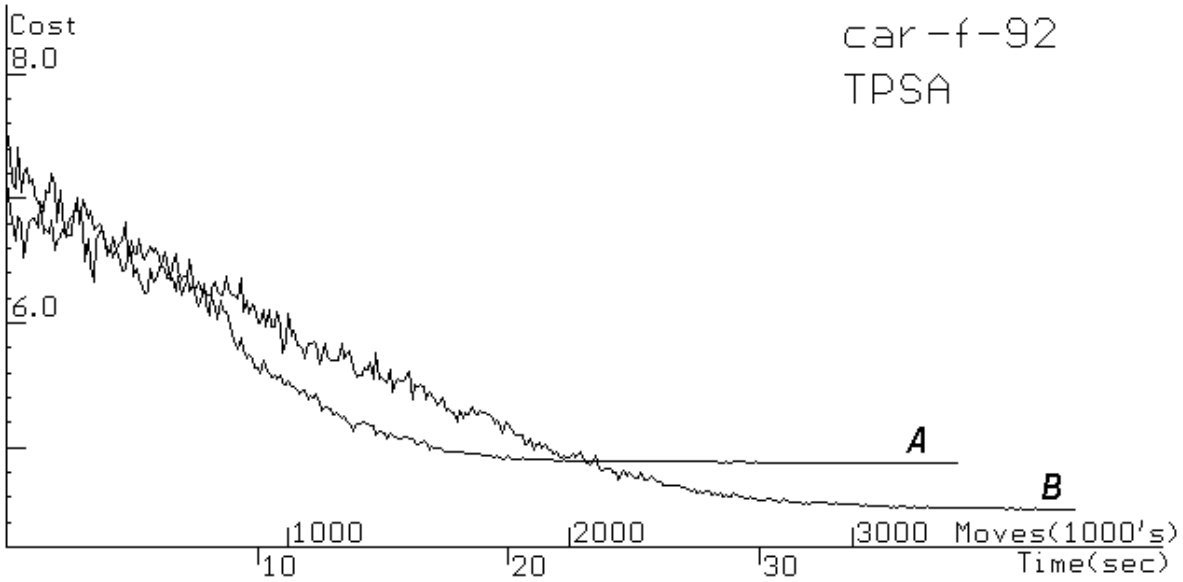

Fig. 2. Cost progress diagrams for time-predefined Simulated Annealing on CAR-F-92

As we proposed, the process *A* converges in approximately 3,000,000 moves. However, this diagram has a long "tail" – it spends almost half of its processing time very close to the point of convergence. To investigate this behaviour further and to look at its effect on the time-quality value, we carried out a second sample run of the algorithm (the curve *B* in Figure 2). The initial temperature and the number of moves were the same as for the process *A*, but we

increased the final temperature 35 times: $T_f = 4*10^{-4}$. This value does not provide the convergence in the given number of moves: the current solution continues to improve after passing it. Nevertheless, at the point of 3,000,000 moves the process *B* has reached a substantially better solution than the process *A*.

However, our attempts to further increase the final temperature led to a worsening of the quality of solutions at this point. It is clear that the optimal value of T_f is problem-dependent and quite difficult to determine. For the experiments presented in this paper we determined it manually by making a high number of *short-time* launches. Of course, the problem of choosing the right parameters usually presents a difficulty when employing Simulated Annealing.

In terms of setting the initial parameters, the uncertainty of the time-predefined Simulated Annealing approach is also characterised by the fact that the process *B* continues to improve after the allocated time is expired. In circumstances when a time limit is not too strict, the user can decide not to terminate the algorithm in order to achieve a better quality solution. However, the extra time for the improvement can be quite long which can mean that even if the solution is improved, certain users may consider it too high a price to pay.

Next we investigated the behaviour of the Great Deluge algorithm with the same given number of moves. Before the first launch of the algorithm on each problem we applied a Hill-Climbing algorithm to get a base for a provisional solution. This was used in formula (5) for the definition of the decay rate. This process lasted only a few seconds and so the time is considered to be negligible. For the CAR-F-92 problem, the Hill-Climbing produced a solution with penalty value of 5.5. Of course, we expected the Great Deluge algorithm to achieve higher quality solutions. Therefore for assigning $f(s')$ in formula (5) we made correspondent fractional decrements of this value. The progress of two sample runs is shown in Figure 3. In the first run (the curve *A*) the final "level" was determined to be 4.4 (5.5*0.8) and in the second one (the curve *B*) it was taken to be 4.95 (5.5*0.9).

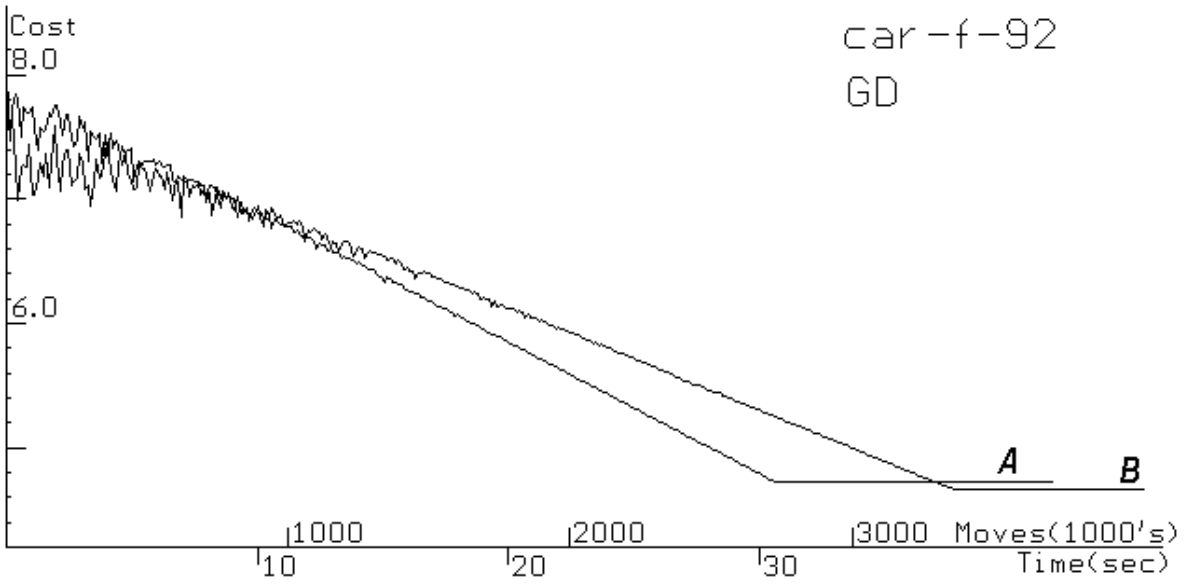

Fig. 3. Cost progress diagrams for the Great Deluge algorithm

These diagrams display how strictly the search follows the linear movement of the "level". The fluctuations are seen only in the first half, but later all solutions become quite close to the "level" and make very small oscillations. The moment of convergence is quite recognisable (the process *A* converged at 2,700,000 moves and process *B* at 3,400,000 moves). We should definitely terminate the algorithm at the point of convergence. The presented way of assessing the final "level" causes an inaccuracy in the time predefinition of around ±13%. However, this is the only uncertainty presented by setting initial parameters for the Great Deluge algorithm.

The same experiments were also conducted on other datasets from Carter's collection. The progress diagrams are not presented here but can be found on the following web page: http://www.cs.nott.ac.uk/~yxb/tpls. They show quite similar properties to the presented ones. The over-riding feature is that the behaviour of time-predefined Simulated Annealing is much more uncertain and parameter depended than for the Great Deluge Algorithm. It requires more preliminary work for the definition of problem-specific parameters and does not guarantee the best time-quality value.

3.6 *Analysis of the Relationship between Time and Cost.*

A second set of experiments was performed on datasets from Carter's collection, and again the cost function presented in (7) was employed. For all datasets, both techniques were launched a number of times with different values of N_{mov}. Simulated Annealing operated with the same temperatures as in the previous experiments. In Great Deluge algorithm B_f of each launch was assigned to be equal to the final solution of the previous launch (for the first one the Hill-Climbing method was applied). The final results of those launches were collected into 26 tables (13 datasets x 2 methods), where each table comprises approximately 50 results. All these tables are available on the following web page: http://www.cs.nott.ac.uk/~yxb/tpls. The most typical samples of them (in the form of diagrams) are presented and discussed below. In all following diagrams the *y* axis represents the final cost and the *x* axis represents the number of moves and computing time in seconds taken by the search session. We discuss the results in the context of the number of enrolments for each problem (given in the right-hand column in Table 1), which can be considered as a certain measure of the problem's size.

Firstly, we consider the three "largest" problems: PUR-S-93, UTA-S-92, CAR-S-91. The resulting diagrams are shown in Figures 4-6. For every dataset the given number of timeslots, enrolments and average search speed is indicated.

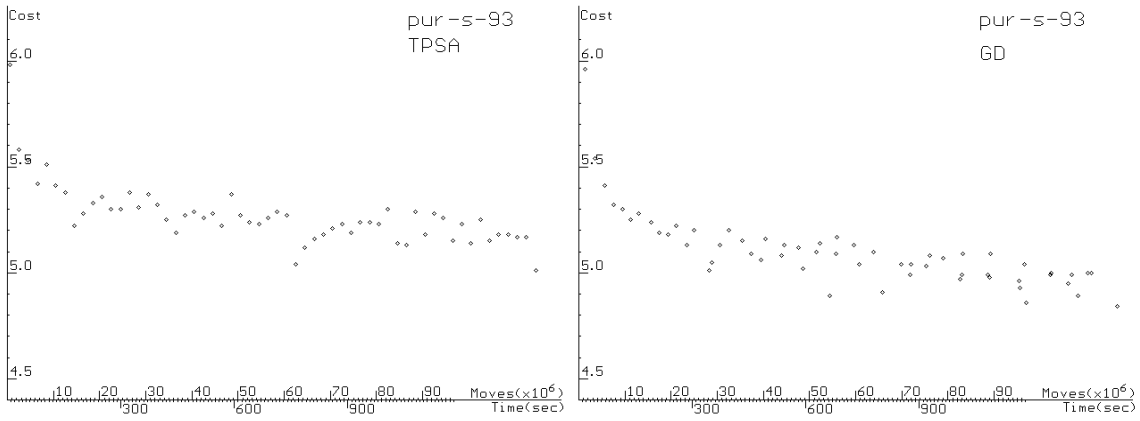

Fig. 4. Time-cost diagrams for PUR-S-93 problem (120690 enrolments, 43 timeslots, search speed 82000 moves/sec)

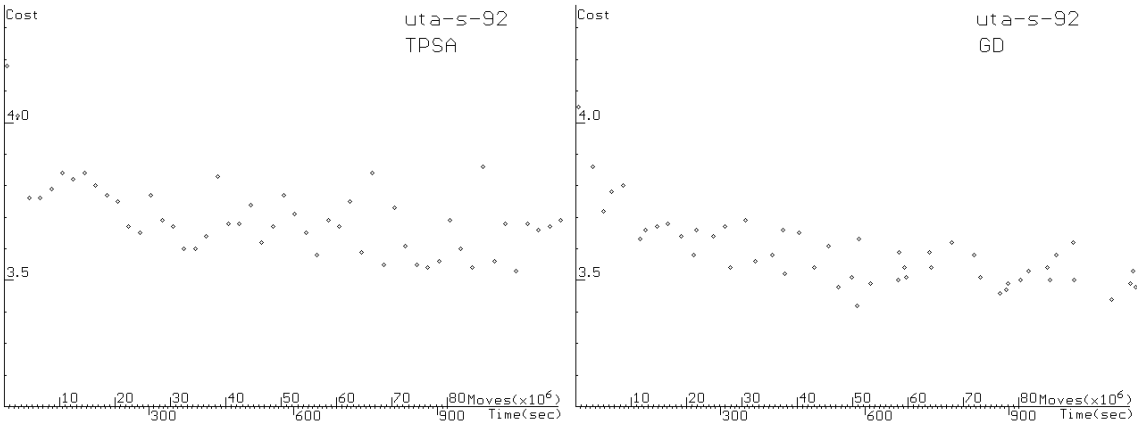

Fig. 5. Time-cost diagrams for UTA-S-92 problem (58981 enrolments, 35 timeslots, search speed 87000 moves/sec)

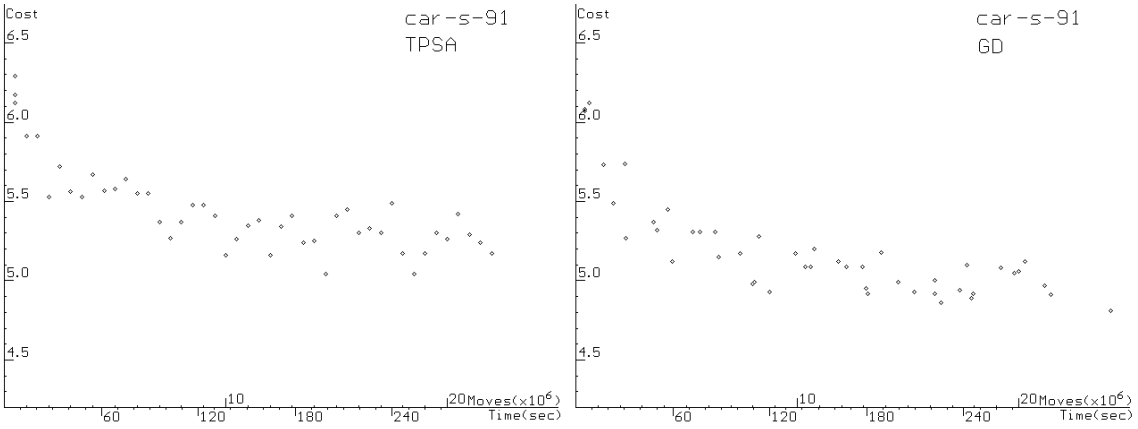

Fig. 6. Time-cost diagrams for CAR-S-91 problem (56877 enrolments, 35 timeslots, search speed 73000 moves/sec)

The distribution of points in these diagrams demonstrates the trade-off between the search time and overall solution quality. Even though the results are relatively scattered, there is a clear general tendency to improve the cost function as time increases. Moreover this tendency, for both techniques appears (in almost the same way) for all presented problems. Indeed all three diagrams presented here are surprisingly similar (although, the scale of both axes is individual for each problem).

The analysis of the diagrams shows that the slope of the curves is relatively steep on the left hand side of the diagrams (i.e. a small increase in time leads to a high improvement in

quality). As the search time gets longer the improvement of solutions becomes slower. Thus, the time-cost diagram of a time-predefined algorithm can be approximated as a monotonically lowered function, which asymptotically approaches some limit. This limit is obviously better than the local optimum (produced by Hill-Climbing). Possibly it is the minimum for the explored area (which can be separated in the case of a disconnected search space).

The particular shape of the time-cost curve depends on the different problem's characteristics. To investigate the influence of the problem's size we next consider the diagrams for the "smallest" problems from Carter's collection: EAR-F-83, YOR-F-83, STA-F-83 (Figures 7-9).

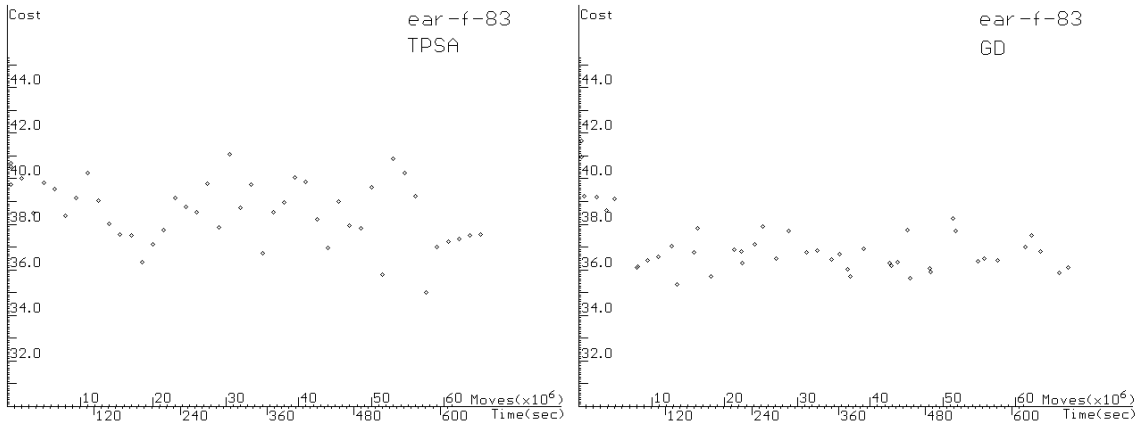

Fig. 7. Time-cost diagrams for EAR-F-83 problem (8108 enrolments, 24 timeslots, search speed 99000 moves/sec)

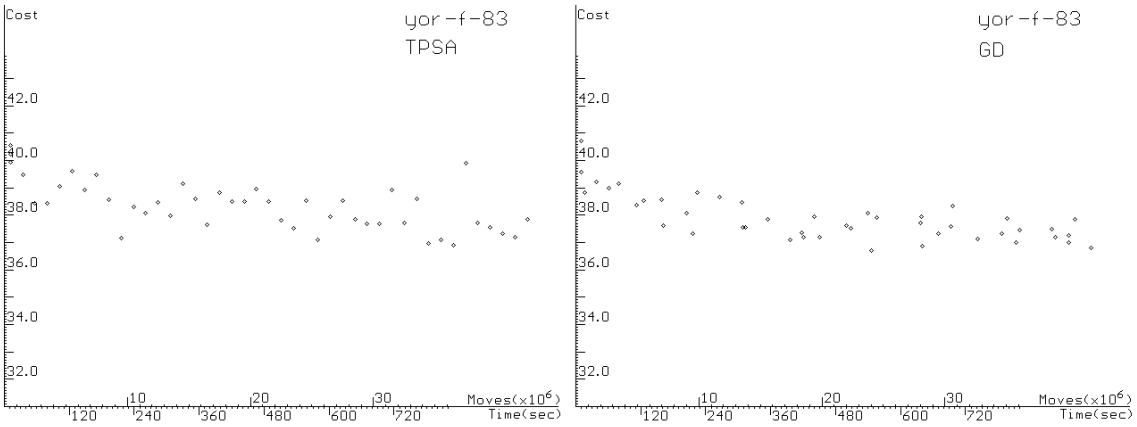

Fig. 8. Time-cost diagrams for YOR-F-83 problem (6029 enrolments, 21 timeslots, search speed 44000 moves/sec)

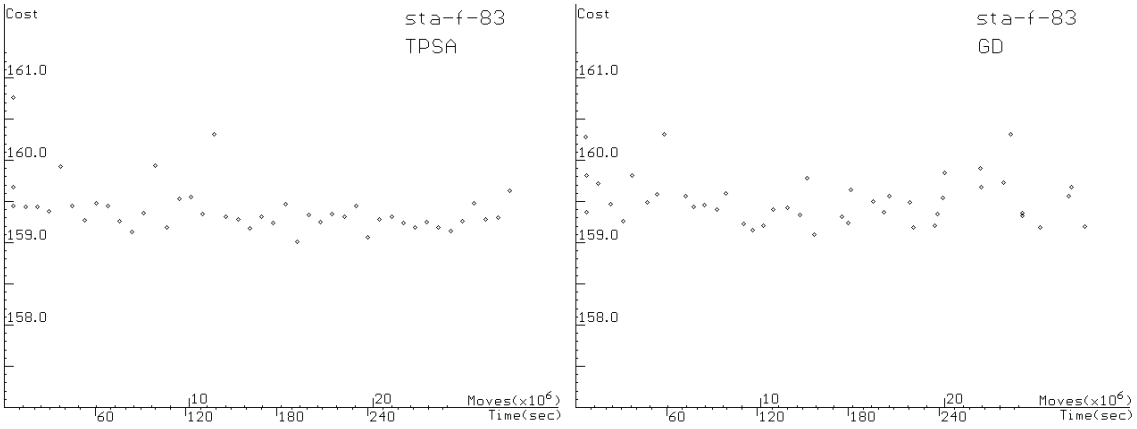

Fig. 9. Time-cost diagrams for STA-F-83 problem (5751 enrolments, 13 timeslots, search speed 82000 moves/second)

These diagrams are clearly different from the ones obtained for the “largest” problems. They show a sharp rise in the final quality with relatively short computational times but further prolongation of the search has very low influence on the result (i.e. the rest of the diagram remains almost flat).

Thus the trade-off between the search time and the quality of results holds mostly for the “large” exam timetabling problems while it plays less of a role for the smaller size problems. This is in accordance with what we would expect: the larger problems need more work! With the “middle-sized” problems the presented techniques usually behave in an intermediate way

(which is also in accordance with what we would expect). An example of such a dataset (UTE-S-92) is presented on Figure 10.

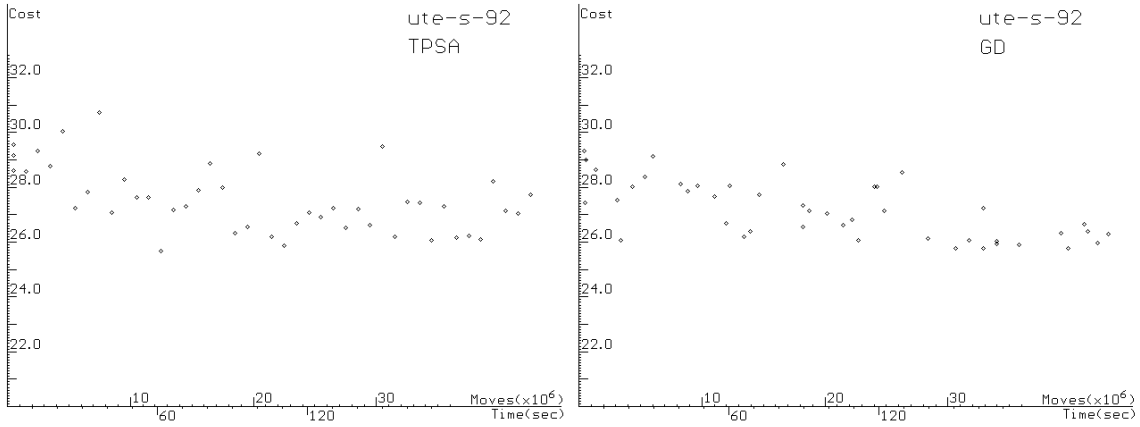

Fig. 10. Time-cost diagrams for UTE-S-92 problem (11796 enrolments, 10 timeslots, search speed 203000 moves/sec)

3.7 *A Comparison of Time-Predefined Simulated Annealing and the Great Deluge Algorithm with the Current State-of-the-Art*

To fully evaluate our techniques, we compare them with recently published results on the same benchmark problems. We consider the results produced by Carter, Laporte and Lee (1996) by employing several sequencing heuristics with backtracking. We also compare against some very recent results that were produced by Di Gaspero and Shaerf (2001). They used tabu search with a variable tabu list. For every dataset Carter et al. published four results, obtained with different heuristics. Di Gaspero and Shaerf presented their best and average values. For comparison purposes we present the best and worst of Carter's et al. results, the best and average results of Di Gaspero and Shaerf and the best, worst and average results for both of our algorithms. Note that Carter's et al. results are obtained by using different algorithms whereas Di Gaspero and Shaerf's results (and our results) employ the same approach.

The average values of our results can be estimated while selecting a proper range of samples. The solutions of "short-time" searches (placed in the left hand sides of our diagrams

in the previous section) are too poor and are shown mainly for illustration purposes. Therefore, as the search time (at the end of each diagram) is quite acceptable (i.e. we consider its right hand side as within the recommended range). Thus our average values are calculated as an average cost of the five rightmost points on our diagrams.

All of the results are summarised in Table 2. Dashes show that the corresponding data was not published. We also include, in the table, a computational time (in seconds) for each best cost. However, this is done only to give a notion about a range of useful time periods because as the published results were produced with different hardware and algorithmic solutions, the search times of the different techniques cannot be sensibly compared.

Table 2. Published and our results for proximity cost

Data set	Time slots	Carter et al.				Di Gaspero and Schaerf				TPSA				GD			
		Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)
		best	worst	average		best	worst	average		best	worst	average		best	worst	average	
CAR-F-92	32	6.2	7.6	-	47.0	5.2	-	5.6	860.6	4.4	5.1	4.5	295	**4.2**	5.1	4.3	274
CAR-S-91	35	7.1	7.9	-	20.7	6.2	-	6.5	30.2	5.0	6.3	5.3	256	**4.8**	6.1	5.0	328
EAR-F-83	24	36.4	46.5	-	24.7	45.7	-	46.7	4.6	**35.0**	41.1	37.3	589	35.4	41.7	36.7	134
HEC-S-92	18	10.8	15.9	-	7.4	12.4	-	12.6	3.7	**10.6**	13.2	11.4	1730	10.8	12.4	11.5	278
KFU-S-93	20	14.0	20.8	-	120.2	18.0	-	19.5	12.3	14.2	16.2	14.9	586	**13.7**	15.6	14.4	729
LSE-F-91	18	10.5	13.1	-	48.0	15.5	-	15.9	20.3	11.0	14.0	12.0	541	**10.4**	12.6	11.0	1030
PUR-S-93	43	**3.9**	5.0	-	21729		-	-	-	5.0	6.0	5.1	1395	4.8	6.0	4.9	1412
RYE-S-93	23	**7.3**	10.0	-	507.2		-	-	-	9.2	10.6	9.6	959	8.9	10.3	9.3	752
STA-F-83	13	161.5	165.7	-	5.7	160.8	-	166.8	3.9	**159.0**	160.8	159.4	190	159.1	160.3	159.4	157
TRE-S-92	23	9.6	11.0	-	107.4	10.0	-	10.5	16.2	8.6	9.8	8.9	326	**8.3**	9.6	8.4	392
UTA-S-92	35	3.5	4.5	-	664.3	4.2		4.5	50.7	3.5	4.2	3.6	906	**3.4**	4.1	3.5	585
UTE-S-92	10	25.8	38.3	-	9.1	29.0	-	31.3	42.4	**25.7**	30.7	27.2	65	**25.7**	29.3	26.2	236
YOR-F-83	21	41.7	49.9	-	271.4	41.0	-	42.1	25.2	36.9	40.6	37.5	820	**36.7**	40.7	37.2	546

The figures in the table demonstrate the power of the two time-predefined approaches. Their performance is better (by all quality indices) than Di Gaspero and Shaerf's tabu search on all the benchmark problems. For eleven of the thirteen problems the two time-predefined algorithms achieved a better cost function value than any of the currently published ones. For five of those problems the best published results are worse than even our average outcome (in a sixth problem they are equal). It is interesting to note that for two of the problems (the largest one and a medium sized one) Carter et al. have an approach, which produces better quality solutions than both of our techniques although, as mentioned above, Carter's et al. results do not represent one single approach (they employ different heuristics).

When comparing the outcomes of our two techniques with each other, it is evident that the Great Deluge Algorithm performs slightly better than the time-predefined Simulated Annealing. In addition, for most of the problems the time-cost diagrams for the Great Deluge Algorithm are less scattered. In Table 2 it can be seen that the Great Deluge Algorithm outperforms the time-predefined Simulated Annealing in 9 cases by the best values and in 11 cases by the average ones. In contrast, time-predefined Simulated Annealing is correspondingly better in terms of best values in 3 cases and in terms of average values in just one case. Moreover, it is only with small-size problems. Note that the two approaches have an equal best value for UTE-S-92 and an equal average value for STA-F-83. Perhaps the time-predefined Simulated Annealing approach does not perform as well because of imperfect values of the initial and final temperatures. However, such a situation is characteristic of Simulated Annealing approach and the Great Deluge approach is free of this uncertainty. This alone leads us towards the conclusion that the Great Deluge algorithm is superior to the time-predefined Simulated Annealing approach. When this particular advantage is taken together with its superior performance (overall) on the benchmark problems then the

evidence of its superiority over time-predefined Simulated Annealing becomes overwhelming (at least for these benchmark examination timetabling problems).

3.8 *Experiments with More Advanced Problems*

Finally, we evaluated the performance of our approach with more complex problems (in terms of more constraints). Three datasets were chosen where, in addition to the clash-free requirement, the seats constraint (formula (8)) is considered as a hard constraint. The distribution of timeslots among days (formula (10)) is also taken into account. Hence the cost function is calculated by formula (13). The characteristics of these problems are presented in Table 3.

Table 3. Additional characteristics of problems

Data	Seats	Periods	Enrolments
KFU-S-93	1955	21	25,118
NOTT-94	1550	23	33,997
CAR-F-92	2000	36	55,552

In these experiments the method of tuning the algorithmic parameters was the same as that employed in the previous experiments. Also, the same type of time-cost diagram was employed for the interpretation of the produced results. These diagrams are presented in Figures 11-13.

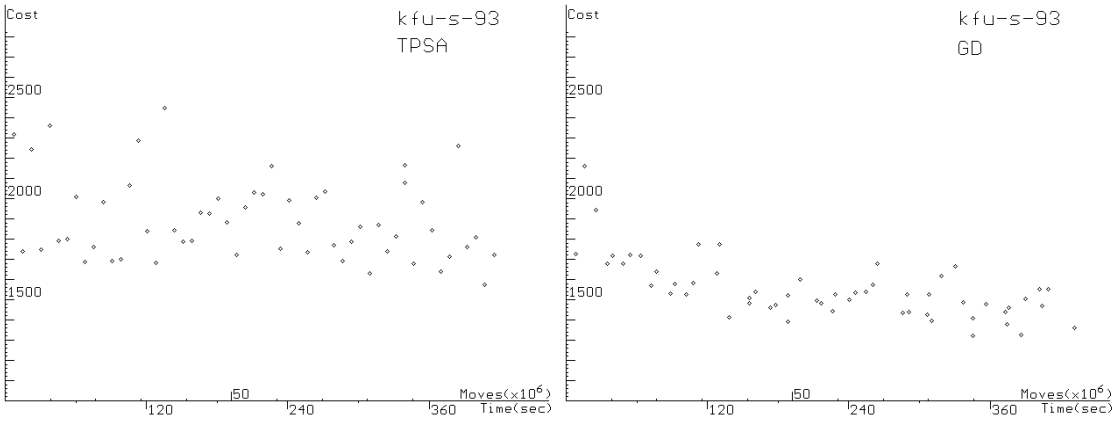

Fig. 11. Time-cost diagrams for Kfu-s-93 problem (speed 260000 moves/second)

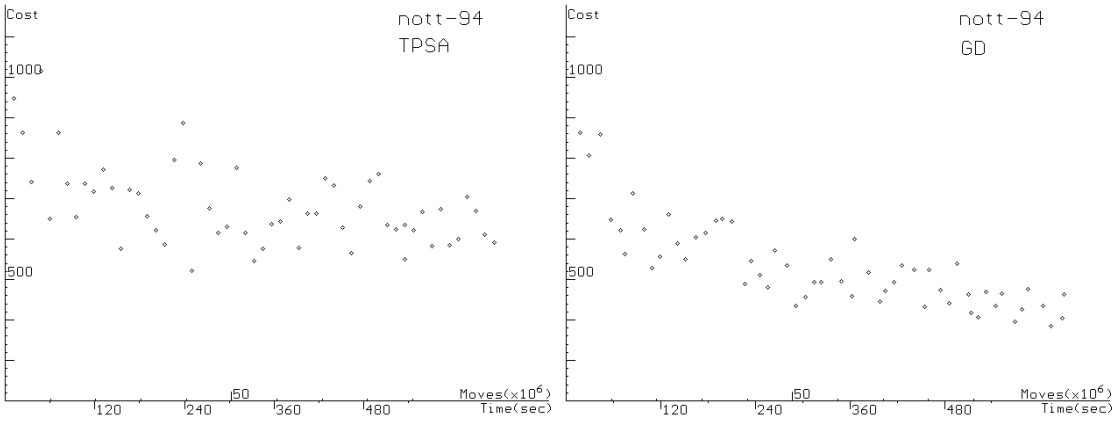

Fig. 12. Time-cost diagrams for Nott-94 problem (speed 174000 moves/second)

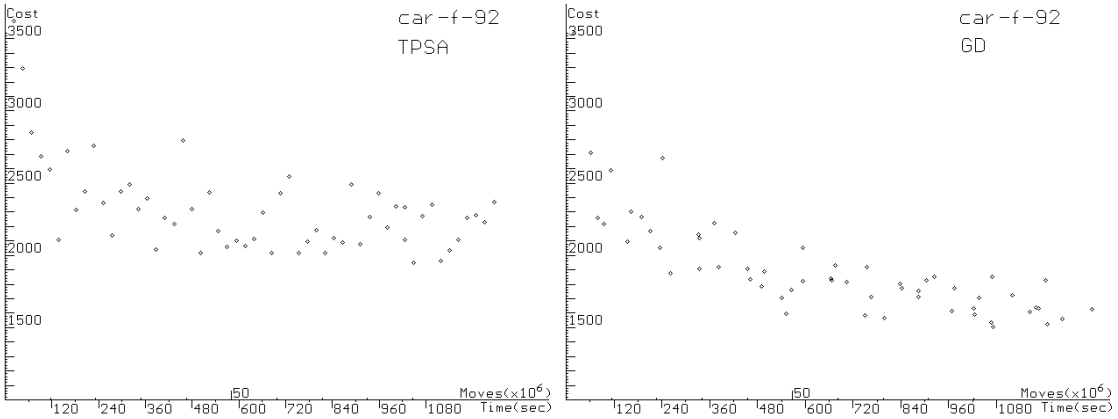

Fig. 13. Time-cost diagrams for Car-f-92 problem (speed 87000 moves/second)

The results obtained by both of our algorithms are compared with the published results obtained by the multi-stage memetic algorithm of Burke and Newall (1999). There is a series of comparisons in (Burke and Newall 1999) which establish that the multistage memetic algorithm had the best published results on these problems up until the development of the algorithms presented in this paper. In addition, to give a more complete evaluation we also present the tabu search results obtained by Di Gaspero and Shaerf (2001) on these 3 problems. In (Burke and Newall 1999) and (Di Gaspero and Shaerf 2001) there is a fourth benchmark problem (PUR-S-93) that is evaluated and discussed. However, the number of timeslots was taken to be 30, which we

strongly believe is insufficient for producing a feasible timetable. Both of the approaches in those papers publish results which are infeasible i.e. they generate high values of the cost function because a very high additional value is included when a hard constraint is broken. The problem was intentionally excluded from this comparison because our two approaches consider only feasible solutions. If we cannot find a solution which satisfies the hard constraints then they cannot cope. However the definition of hard constraints is, of course, that they must be satisfied in all cases. The final figures are presented in Table 4.

Table 4. Published and our results for weighted sum of adjacent and overnight conflicts

Data set	Time slots	Multi-stage memetic algorithm				Di Gaspero and Schaerf				TPSA				GD			
		Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)	Cost			Time for best (sec)
		best	worst	average		best	worst	average		best	Worst	average		best	worst	average	
KFU-S-93	21	1388	2662	1626	105	1733	-	1845	n/a	1573	2448	1825	407	**1321**	2161	1470	345
NOTT-94	23	490	1022	552	467	751	-	810	n/a	522	1015	635	249	**384**	862	433	612
CAR-F-92	36	1665	4806	1765	186	3048	-	3377	n/a	1951	3622	2250	1220	**1506**	3536	1610	1268

This comparison once more justifies the power of the two time-predefined approaches. Indeed, it is clear here that the Great Deluge algorithm is far superior to time-predefined Simulated Annealing. In fact it is far superior (in terms of the quality of its results) to all previously published approaches on these problems. Moreover, the presented time-cost diagrams show the potential for further improvement of results (i.e. there is a clear slope in the right hand side of the diagrams). We did not continue with experiments, which went beyond the right hand side of the diagrams for all 3 problems because of the extreme computational expense. However, the Great Deluge algorithm was launched on the NOTT-94 problem for extremely long periods to give us some indication of what it could produce with enough computational time (Table 5).

Table 5. The best results, obtained for Nott-94 problem by Great Deluge search

Time (hours)	Cost
2.5	256
67	225

As we expected from examining the diagrams, an extremely long search can produce extremely good results. Note that although the solution with a cost of 225 took over 67 hours, we ran it over a weekend (when the computer would otherwise have been idle) and we produced a result which is more than twice as good as the previous best published solution. It is clear that while this ability may not be appreciated by all users, there is a definite potential for incorporating this approach into real world examination timetabling systems (and indeed other scheduling systems).

4. A Brief Note about the Comparison of Time-Predefined Algorithms with other Approaches

The comparison of our results with previously published ones, given in Tables 2 and 4 is presented in a conventional way (as it is usually conducted with non time-predefined

algorithms). It only expresses a rough idea about the range of the presented values of the cost function. However, a formal comparison (which takes into account both cost and time) of the proposed approach with non time-predefined techniques requires a more detailed study. In this section we sketch the main aspects of this investigation.

When embedding time predefinition into search algorithms it is possible to express their performance as a function: one can get any solution (within some range) as long as one pays the necessary cost in time. From this point of view the processing time can be thought of as an additional objective, which should be minimised together with the cost function. Thus, the problem becomes bi-objective where the time-cost goal corresponds to the so-called "Pareto-front" in multiobjective optimisation. The comparison of the quality of multiobjective solutions produced by different techniques is a growing area of study. The investigation of several metrics suitable for this comparison can be found in (Knowles and Corne 2002).

The comparison of the performance of a time-predefined and a conventional technique (which can be thought as having a single time-cost point) is quite uncertain and can be carried only within some limits. We illustrate the issues in Figure 14. The dotted line represents the time-cost trade-off curve of a time-predefined algorithm $\mathfrak{A}$. A solution, produced by a traditional algorithm $\mathfrak{B}$ (point *B*) has worse time and cost than $\mathfrak{A}$. Thus a complete outperformance of the algorithm $\mathfrak{A}$ over $\mathfrak{B}$ is evident. However, while evaluating the point *C* (the solution, produced by another non time-predefined algorithm $\mathfrak{C}$) we cannot make any general decision about the preference of either algorithm. Algorithm $\mathfrak{C}$ partially outperforms $\mathfrak{A}$. The preference is clear between points *x* and *y* but unclear outside these points.

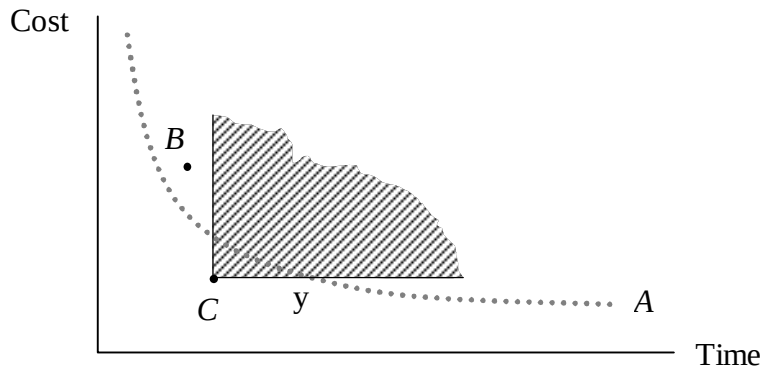

Fig. 14. The example of uncertainty in comparison of the time-cost diagram with the single solution.

Of course, more uncertainty is caused by the presence of a number of parameters whose values are tuned by performing many experiments. This time should be taken into account when the total time of the solution process is calculated. We should point out that Great Deluge algorithm is almost free of such parameters and spends almost all of its time only in the search procedure.

5. Conclusions and Future Work

This paper investigates a number of aspects in exploring the time-cost trade-off for improving the quality of exam timetables. In many higher educational institutions, it is quite acceptable to have examination timetable processing times of up to several hours. However, this length of time is justified only if an algorithm uses the specified amount of time in an intelligent manner in order to produce a solution of a suitably high quality in this period. This can be achieved while embedding time-predefinition mechanisms into local search methods. However such mechanisms function differently for different techniques.

Two time-predefined algorithms are discussed and their properties are investigated. We point out that the time-cost trade-off is mostly revealed in large-scale problems. The time-predefined algorithms between them produced the best published results on all but two

benchmark problems. The Great Deluge algorithm is preferred because it produces most of the best results on these problems (and all of the best results on a further three, more complicated, benchmark problems) in addition to having much less uncertainty in setting up algorithmic parameters. A detailed study of possible user strategies while operating with time-predefined algorithms is the subject of future work. It is possible that a multiobjective approach may yield benefits while considering the quality of solution and computational time as two user objectives. Additionally, some methods suitable for the formal comparison of the time-cost indexes of different algorithms could be defined and investigated.

To expand the practical benefits of the presented approaches, it would be worth more accurately investigating their properties, such as the dependence of the time-cost diagram on a problem's size (and on other characteristics of a problem). Another direction would be to investigate the influence of an initial solution on the overall result while exploring different initialisation methods. Also in our work we did not touch on the question about neighbourhood variation, which probably influences the result as well as questions about disconnected search spaces, relaxation of a problem, kempe or s-chains, etc. The described Great Deluge algorithm employs a linear reduction of the "level" as the simplest variant. It seems reasonable to investigate other possible reductions. Also the Great Deluge algorithm is open to different extensions and hybridisations. Our future work will include investigation of the possibilities of the algorithms to operate with population-based metaheuristics and also for solving multiobjective problems. Also, for certain, the family of time-predefined techniques is not limited to the two proposed methods. We believe that the predefinition of time can indeed be embedded into other techniques.

6. Acknowledgments

We would like to thank the anonymous referees for their helpful comments. The research described in this paper was supported by EPSRC grant GR/N36837/01.

References

Boizumault, P., Delon, Y. and Peridy, L. (1996) Constraint Logic Programming for Examination Timetabling. *Journal of Logic Programming*, **26**(2), 217-233.

Boufflet, J.P., and Negre, S. (1996) Three Methods Used to Solve an Examination Timetabling Problem. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 327-344.

Brelaz, D. (1979) New Methods to Color the Vertices of a Graph. *Communication of the ACM*, **22**(4), 251-256.

Bullnheimer, B. (1998) An Examination Scheduling Model to Maximize Students' Study Time. *Lecture Notes in Computer Science vol. 1408. The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*, Burke, E.K., Carter, M. (eds). Springer-Verlag, Berlin, Heidelberg, New York, 78-91.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1995a) A Hybrid Genetic Algorithm for Highly Constrained Timetabling Problems. *Genetic Algorithms: Proceedings of the 6th International Conference*. *San Francisco*. Eshelman, L.J. (editor). Morgan Kaufmann, 605-610.

Burke, E.K., Elliman, D.G. and Weare, R.F. (1995b) Specialised Recombinative Operators for Timetabling Problems. *Lecture Notes in Computer Science vol. 993. AISB Workshop on Evolutionary Computing*. Fogarty, T.C. (editor). Springer-Verlag, Berlin, Heidelberg, New York, 75-85.

Burke, E.K., Elliman, D.G., Ford, P.H. and Weare, R.F. (1996) Examination Timetabling in British Universities: a Survey. *Lecture Notes in Computer Science vol. 1153. The Practice*

and Theory of Automated Timetabling: Selected Papers (ICPTAT '95), Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 76-90.

Burke, E.K., Newall, J.P. and Weare, R.F. (1996) A Memetic Algorithm for University Exam Timetabling. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 241-250.

Burke, E.K., Newall, J.P. and Weare, R.F. (1998) Initialization Strategies and Diversity in Evolutionary Timetabling. *Evolutionary Computation* **6**(1), 81-103.

Burke, E.K. and Newall, J.P. (1999) A Multi-Stage Evolutionary Algorithm for the Timetabling Problem. *The IEEE Transactions of Evolutionary Computation* **3**(1), 63-74.

Burke, E.K., Bykov, Y. and Petrovic, S. (2001) A Multicriteria Approach to Examination Timetabling. *Lecture Notes in Computer Science vol. 2079. The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000).* Burke, E.K., Erben, W. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 118-131.

Burke, E.K. and Petrovic, S. (2002) Recent Research Directions in Automated Timetabling. *European Journal of Operational Research*, **140**, 266-280.

Carter, M.W. (1986) A Survey of Practical Applications of Examination Timetabling Algorithms. *Operations Research*, **34**(2),193-201.

Carter, M.W., Laporte, G. and Chinneck, J.W. (1994) A General Examination Scheduling System. *Interfaces*, **24**, 109-120.

Carter, M.W. and Laporte, G. (1996) Recent Developments in Practical Examination Timetabling. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 3-21.

Carter, M.W., Laporte, G. and Lee, S.Y. (1996) Examination Timetabling: Algorithmic Strategies and Applications. *Journal of Operational Research Society*, 47(3), 373-383.

Cole, A.J. (1964) The Preparation of Examination Timetables Using a Small-Store Computer. *The Computer Journal*, 7, 117-121.

Corne, D., Fang, H.L. and Mellish, C. (1993) Solving the Module Exam Scheduling Problem with Genetic Algorithms. *Proceedings of the Sixth International Conference in Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*. Chung, P.W./H., Lovergrove, G., Ali, M. (eds). Gordon and Breach Science Publishers, 370-373.

Corne, D., Ross, P. and Fang, H.L. (1994) Fast Practical Evolutionary Timetabling. *Lecture Notes in Computer Science vol. 865. AISB Workshop on Evolutionary Computing*. Fogarty, T.C. (editor). Springer-Verlag, Berlin, Heidelberg, New York, 250-263.

Corne, D. and Ross, P. (1996) Peckish Initialisation Strategies for Evolutionary Timetabling. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 227-240.

David, P. (1998) A Constraint-Based Approach for Examination Timetabling Using Local Repair Techniques. *Lecture Notes in Computer Science vol. 1408. The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97)*, Burke, E.K., Carter, M. (eds). Springer-Verlag, Berlin, Heidelberg, New York, 169-186.

Dueck, G. (1993) New Optimization Heuristics. The Great Deluge Algorithm and the Record-to-Record Travel. *Journal of Computational Physics*, **104**, 86-92.

Di Gaspero, L. and Schaerf, A. (2001) Tabu Search Techniques for Examination Timetabling. *Lecture Notes in Computer Science vol. 2079. The Practice and Theory of Automated*

Timetabling III: Selected Papers (PATAT 2000). Burke, E.K., Erben, W. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 104-117.

Erben, W. (2001) A Grouping Genetic Algorithm for Graph Colouring and Exam Timetabling. *Lecture Notes in Computer Science vol. 2079. The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000).* Burke, E.K., Erben, W. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 132-156.

Ergul, A. (1996) GA-Based Examination Scheduling Experience at Middle East Technical University. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 212-226.

Glover, F. (1986) Future Paths for Integer Programming and Links to Artificial Intelligence. *Computers & Operational Research*, **5**, 533-549.

Hertz, A. (1991) Tabu Search for Large Scale Timetabling Problems. *European Journal of Operational Research*, **54**, 39-47.

Kirkpatrick, S., Gellat, J.C.D. and Vecci, M.P. (1983) Optimization by Simulated Annealing. Science 220, 671-680.

Kirkpatrick, S. (1984) Optimization by Simulated Annealing - Quantitative Studies. *Journal of Statistical Physics*, **34**, 975-986.

Knowles J. and Corne, D. (2002) On Metrics for Comparing Nondominated Sets, in *Proceedings of the 2002 Congress on Evolutionary Computation Conference*, 711-716.

Matula, D.W., Marble, G. and Isaacson, I.D. (1972) Graph Coloring Algorithms. *Graph Theory and Computing*. Read, R.C. (editor). Academic Press, New York, 109-122.

Peck, J.E.L. and Williams, M.R. (1966) Algorithm 286 - Examination Scheduling. *Communication of the ACM*, **9**, 433-434.

Reeves C.R. (1996) Modern Heuristic Techniques. *Modern Heuristic Search Methods*. John Willey & Sons.

Reis, L.P. and Oliveira, E. (2000) Examination Timetabling using Set Variables, in *Proceedings of the 3rd International Conference on the Practice and Theory of Automated Timetabling PATAT2000, Konstanz, Germany,* 181-183.

Ross, P. and Corne, D. (1995) Comparing Genetic Algorithms, Simulated Annealing and Stochastic Hillclimbing on Timetabling Problems. *Lecture Notes in Computer Science vol. 993. AISB Workshop on Evolutionary Computing*. Fogarty, T.C. (editor). Springer-Verlag, Berlin, Heidelberg, New York, 92-102.

Schaerf, A. (1999) A Survey of Automated Timetabling. *Artificial Intelligent Review*, **13**, 87-127.

Terashima-Marin, H., Ross, P.M. and Valenzuela-Rendon, M. (1999) Evolution of Constraint Satisfaction Strategies in Examination Timetabling, in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-99)*. Morgan Kaufmann, 635-642.

Thompson, J.M. and Dowsland, K.A. (1996a) General Cooling Schedules for a Simulated Annealing Based Timetabling System. *Lecture Notes in Computer Science vol. 1153. The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95)*, Burke, E.K., Ross, P. (eds), Springer-Verlag, Berlin, Heidelberg, New York, 345-363.

Thompson, J.M. and Dowsland, K.A. (1996b) Variants of Simulated Annealing for the Examination Timetabling Problem. *Annals of Operations Research*, **63**, 105-128.

Thompson, J.M. and Dowsland, K.A. (1998) A Robust Simulated Annealing Based Examination Timetabling System. *Computers and Operational Research*, 25(7/8), 637-648.

White, G. M. and Xie, B. S. (2001) Examination Timetables and Tabu Search with Longer-Term Memory. *Lecture Notes in Computer Science vol. 2079. The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000).* Burke, E.K., Erben, W. (eds), Springer-Verlag, Berlin, Heidelberg, New York,85-103.

Williams, M.R. (1974) Heuristic Procedures - (if they Work, Leave Them Alone). *Software Practice and Experience*, 4, 237-240.