

Enhancing Timetable Solutions with Local Search Methods

E.K. Burke¹ and J.P. Newall²

¹ School of Computer Science and Information Technology,
University of Nottingham, Jubilee Campus, Wollaton Road,
Nottingham, NG8 1BB, UK

`ekb@cs.nott.ac.uk`

² eventMAP Ltd,

21 Stranmillis Road Belfast BT9 5AF, Northern Ireland,

`Jim.Newall@eventmaponline.com`

Abstract. It is well known that domain-specific heuristics can produce good-quality solutions for timetabling problems in a short amount of time. However, they often lack the ability to do any thorough optimisation. In this paper we will study the effects of applying local search techniques to improve good-quality initial solutions generated using a heuristic construction method. While the same rules should apply to any heuristic construction, we use here an adaptive approach to timetabling problems. The focus of the experiments is how parameters to the local search methods affect quality when started on already good solutions. We present experimental results which show that this combined approach produces the best published results on several benchmark problems and we briefly discuss the implications for future work in the area.

1 Introduction

1.1 The Examination Timetabling Problem

The examination timetabling problem consists of allocating a number of exams to a limited number of periods or timeslots subject to certain constraints. These constraints may relate to operational limitations (such as no student having to attend two exams at the same time, limitations on seating, etc.) which cannot be overcome, or they may be regarded as being sufficiently important that they should always be observed. In either case we call these hard constraints. We also call a timetable that obeys all hard constraints a feasible timetable. The remaining constraints are those that are considered to be desirable to satisfy, but not essential, such as allowing students study time between exams. These constraints are termed soft constraints. The level of satisfaction of these soft constraints can be considered to be a measure of the quality of a timetable. More information on the various constraints that can exist for the problem can be found in a survey of UK universities [2]. Surveys of practical applications of examination timetabling can be found in [5,6].

1.2 Local Search Methods

It is possible to think of local search techniques as methods which typically function by iteratively applying simple moves to a solution. Before a move is applied the effects it has are evaluated and a decision is made to accept or reject the move based on some criteria. The criteria for accepting moves is generally what distinguishes the various varieties of local search.

The simplest form of local search approaches are Descent-Based methods (or Hill Climbing, depending on your point of view). Here a move is only accepted if it produces a solution at least as good as the current solution. For mainly comparative purposes we consider this simple form of local search in this paper.

A more sophisticated approach, which we also consider in this paper, is Simulated Annealing [11]. This differs from descent methods in that it has a stochastic component whereby worsening moves can be accepted. The probability p of accepting a worsening move is given in (1), where Δ represents the change in quality and T represents the current temperature of the system, this being derived from thermodynamics' Boltzmann's distribution. The temperature of the system is periodically lowered according to a cooling schedule. This results in worsening moves being less likely to be accepted as the process continues. The performance of Simulated Annealing is generally regarded to be highly dependent on the choice of parameters such as starting temperature, terminating temperature and the cooling schedule:

$$p = e^{-\frac{\Delta}{T}}. \quad (1)$$

An alternative that is also investigated in this paper, and that has been shown to work particularly well on the examination timetabling problem, is the Great Deluge algorithm proposed by Dueck [9]. This approach functions in a similar fashion to Simulated Annealing and its variant, Threshold Acceptance [10]. However, worsening moves are accepted or rejected according to whether solution quality falls below or above a specified ceiling, which is reduced linearly throughout the process. The method was successfully investigated for timetabling problems by Burke et al. [1], along with an appropriately configured Simulated Annealing process. The main advantage over Simulated Annealing is the lower number of parameters, and that those required can easily be guessed or worked out from a parameter describing the desired run time.

2 Applying Local Search Techniques to Good Initial Solutions

When applying local search techniques such as the Great Deluge or Simulated Annealing to good-quality initial solutions there are factors we need to take into account. Firstly, these methods usually perform a far-ranging search of the solution space before terminating, which may not be desirable when starting from a good initial point, as we could quickly lose any benefit from the initial quality. Fortunately, we can control how far-ranging a search we wish to carry out

```

sinceLastMove := 0
WHILE sinceLastMove < 1,000,000 DO
choose exam e and period t at random s.t.  $t \neq \text{period}(e)$ 
  IF  $\text{penalty}(e, t) \leq \text{penalty}(e, \text{period}(e))$  THEN
    move exam e to period t
    sinceLastMove := 0
  ELSE
    sinceLastMove+ = 1
  ENDIF
DONE

```

Fig. 1. Pseudo-code describing the Hill Climbing procedure

through parameters for two of the three local search methods that we investigate. It should be noted that the actual method of generating initial solutions is irrelevant here as we are concerned exclusively with establishing how much, and under what conditions, solutions can be enhanced using local search methods. In all the proposed methods we begin with a feasible solution and never accept moves that would produce an infeasible timetable.

2.1 Adaptive Initial Solution Generation

An adaptive heuristic approach is used to generate initial solutions. This basically functions by repeatedly trying to construct a timetable based on an ordering, while promoting any exams that cannot be scheduled to an acceptable level. This in itself produces very competitive results relatively quickly. As the experiments here are largely concerned with the improvements attainable with local search we will not discuss the generation method at length here. For a detailed explanation of the method and experimental results see [3].

As mentioned above, we are not concerned with the actual construction method used. It is likely that the same techniques could be applied just as effectively to other heuristic algorithms such as those presented by Carter et al. [7]. We will now briefly discuss the three local search approaches, and how, where possible, they can be configured to take account of already good solutions.

2.2 Hill Climbing

As Hill Climbing, by its nature, will not accept worsening moves it can only explore a very limited portion of the search space. It is, however, very fast compared to the other methods and will never produce a solution that is worse than the original. For these reasons Hill Climbing provides a good baseline for comparison with the other more thorough search methods.

For these experiments, a simple randomised Hill Climbing method was used, where moves are generated at random (in the same way as for the other methods) and accepted if they do not lead to a worse solution. The process is aborted after

1000 000 unsuccessful successive attempts at generating a move. Figure 1 gives the Hill Climbing procedure used in the tests, where $period(e)$ gives the current period of exam e , and $penalty(e, t)$ gives the penalty arising from scheduling exam e in period t .

2.3 Simulated Annealing

The inherent controlling factor in Simulated Annealing is the temperature. A higher initial temperature will result in increased acceptance of worsening moves, and therefore the search tends to move further away from the starting point. Alternatively, a lower temperature will search more in the vicinity of the initial solution, at the expense that it may not be able to reach other better, though still relatively near, areas of the search space. Figure 2 shows the process used for Simulated Annealing, where T_0 is the initial temperature, T_{min} is the minimum temperature and α represents the cooling schedule.

As selecting a suitable initial temperature T_0 can be a problem for Simulated Annealing we use an approach where we specify a desired average probability of accepting worsening moves. A temperature is then calculated by generating a number of random moves and evaluating the average probability of acceptance. If a temperature of 1 does not provide at least the desired average probability it is doubled and the process repeated. This continues until the generated average probability matches or exceeds the desired average probability. This allows us to specify the temperature indirectly in a less problem-specific way.

Values for T_{min} and α are somewhat easier to determine. As our penalty function has a granularity of 1, a uniform value of $T_{min} = 0.05$ is used throughout to guarantee a Hill Climbing-like phase at the end of the process. Instead of specifying α as a parameter, we give the desired number of moves N as a parameter and calculate α according to

$$\alpha = 1 - \frac{\ln(T_0) - \ln(T_{min})}{N}. \quad (2)$$

2.4 The Great Deluge

The Great Deluge method provides a much simpler mechanism than Simulated Annealing and is known to be effective on exam timetabling problems [1]. Here the controlling factor is the value of an initial ceiling on solution quality, where with a higher ceiling the search is less restricted. We can easily calculate an initial ceiling by multiplying the penalty of the initial solution by some chosen factor.

Figure 3 shows the Great Deluge procedure where b is the amount we wish to reduce the ceiling by at each iteration, and is calculated based on the user specified desired number of iterations N . As it is likely that the ceiling will fall below the current penalty as the solution nears a local optimum, 1000 000 iterations without improvement are allowed in order to accommodate a Hill Climbing-type phase towards the end of the process.

```

T := T0
WHILE T > Tmin DO
  choose exam e and period t at random s.t. t ≠ period(e)
  IF penalty(e, t) ≤ penalty(e, period(e)) THEN
    move exam e to period t
  ELSE
    move exam e with probability
    exp ( (penalty(e, period(e)) - (penalty(e, t)) / T )
  ENDIF

  T := T
DONE

```

Fig. 2. Pseudo-code for the Simulated Annealing process

```

P := initial penalty of solution
B := P * user provided factor
b := B / N
sinceLastMove := 0
WHILE (B > 0 AND P < B) OR sinceLastMove < 1,000,000 DO
  choose exam e and period t at random s.t. t ≠ period(e)
  delta := penalty(e, t) - penalty(e, period(p))
  IF P + delta < B OR delta < 0 THEN
    move exam e to period p
    P := P + delta
    sinceLastMove := 0
  ELSE
    sinceLastMove += 1
  ENDIF

  B := B - b
DONE

```

Fig. 3. Pseudo-code for the Great Deluge approach

3 Results

3.1 Experimental Setup

Experiments were performed on a range of real-world examination timetabling problems available from <ftp://ftp.mie.utoronto.ca/pub/carter/testprob> (see Table 1).

The objectives of these problems are to firstly create a conflict-free timetable, and secondly to minimise the number of cases where a student has exams s periods apart. The weight w_s for a student taking two exams s periods apart is given by: $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$. These are summed up

Table 1. The benchmark problems used

| Data | Institution | Periods | Number of Exams | Number of Students | Density of Conflict Matrix |
|----------|---|---------|-----------------|--------------------|----------------------------|
| car-f-92 | Carleton University, Ottawa | 32 | 543 | 18 419 | 0.14 |
| car-s-91 | Carleton University, Ottawa | 35 | 682 | 16 925 | 0.13 |
| ear-f-83 | Earl Haig Collegiate Institute, Toronto | 24 | 190 | 1 125 | 0.29 |
| hec-s-93 | École des Hautes Études Commerciales, Montreal | 18 | 81 | 2 823 | 0.20 |
| kfu-s-93 | King Fahd University, Dharan | 20 | 461 | 5 349 | 0.06 |
| lse-f-91 | London School of Economics | 18 | 381 | 2 726 | 0.06 |
| sta-f-83 | St Andrew's Junior High School, Toronto | 13 | 139 | 611 | 0.14 |
| tre-s-92 | Trent University, Peterborough, Ontario | 23 | 261 | 4 360 | 0.18 |
| uta-s-93 | Faculty of Arts and Sciences, University of Toronto | 35 | 622 | 21 267 | 0.13 |
| ute-s-92 | Faculty of Engineering, University of Toronto | 10 | 184 | 2 750 | 0.08 |
| yor-f-83 | York Mills Collegiate Institute, Toronto | 21 | 181 | 941 | 0.27 |

and divided by the number of students in the problem to give a measure of the average conflicts per student.

The Great Deluge was applied to the heuristically generated solutions with various initial ceiling values (given in the form of a function of the initial quality). Similarly, Simulated Annealing was applied to the same problems with varying desired initial average probabilities (as described above). Both algorithms were given a desired number of iterations of 20 000 000 and 200 000 000. Each individual experiment was run five times with varying random number seeds. An index of improvement was calculated by summing the average percentage improvement for each of the problems. For example, if five problems improved by 2% and six problems improved by 5% the improvement index will be $(5 \times 2) + (6 \times 5) = 40$.

Often in local search methods it is prudent to keep a copy of the best solution found so far, as the final resting point is not necessarily the best solution encountered. In this case, however, it is more revealing to take the final resting point of the search, as we can then perceive actual reductions in solution quality.

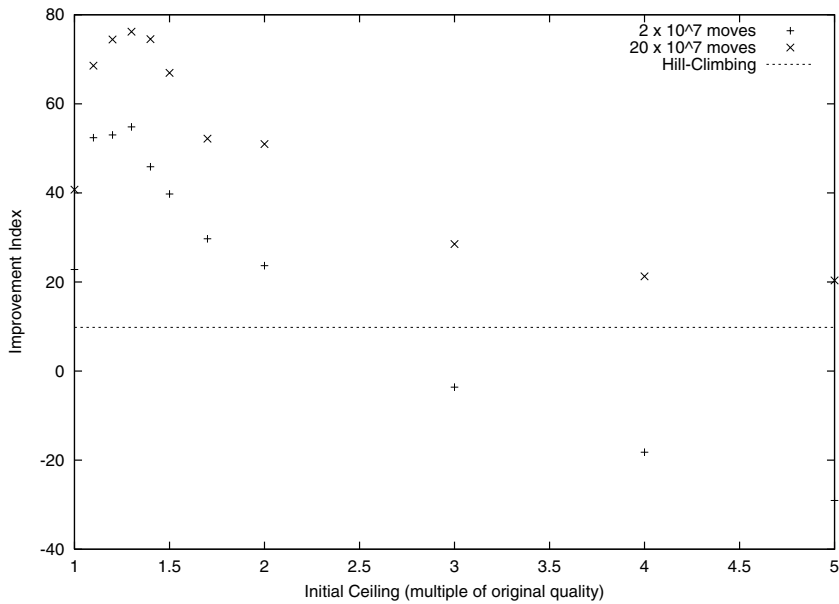


Fig. 4. Results when using Great Deluge to improve solutions

3.2 Results When Using Great Deluge Local Search

The Great Deluge algorithm was run on the generated solution with the initial ceiling set at various values based on the original quality. Figure 4 shows the total improvement index, with the horizontal bar indicating the improvements found by basic stochastic Hill Climbing. It is clear that we need to give the search procedure some extra scope to manoeuvre, though too much scope results in overall degradation of the results. It is even the case that too much scope (a ceiling factor of 5) produces results that are noticeably worse than the original unless given a sufficient amount of time. This said, it seems sensible to set the initial ceiling at the quality of the initial solution, multiplied by a factor of 1.3. Launching the algorithm for 10 times the number of moves results in slightly better improvements. However, the same rules on the initial ceiling still apply. It is worth noting that all reasonable configurations perform considerably better than stochastic Hill Climbing, yielding nearly eight times the improvement of Hill Climbing in the best case.

3.3 Results When Using Simulated Annealing

When experimenting with Simulated Annealing we test a range of temperatures, based on the average probability of accepting worsening moves. The probability is varied between 0.1 and 0.9 and shows the variation in improvement when

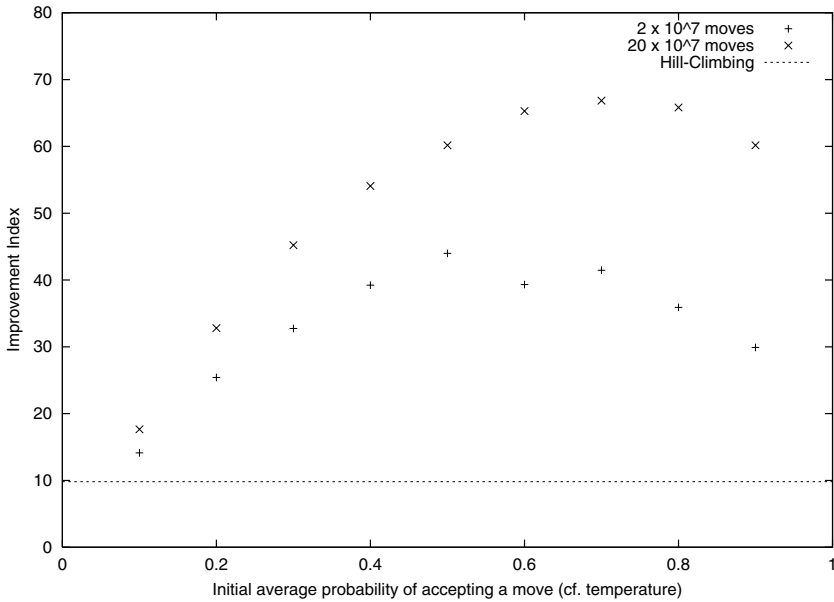


Fig. 5. Results when using Simulated Annealing to improve solutions

using different starting temperatures. The results of this are shown in Figure 5. Simulated Annealing exhibits a similar behaviour to the Great Deluge approach in that there is a “sweet spot” that balances out the need for a far-ranging search against the desire to preserve the quality of the original solution. The differences here though are less pronounced, with none of the tested scenarios producing solutions worse than the originals or, for that matter, than those produced with Hill Climbing. We would, however, expect this to change as the probability approaches one and becomes more like a non-seeded application of Simulated Annealing.

3.4 Analysis

The results indicate a slight advantage in favour of the Great Deluge method when working on improving solutions. The simpler mechanism of the Great Deluge also makes the method more attractive than Simulated Annealing. The results show that for both experiments the value of the parameters can have a dramatic effect on solution quality. Tables 2 and 3 show a comparison between Simulated Annealing (SA) and Great Deluge (GD) when started with and without good solutions, for 20 000 000 and 200 000 000 iterations. It should be noted that the initial

solution for the basic approach was also generated using a similar heuristic method except that soft constraints were not considered. This was necessary as

Table 2. Comparison of both methods with and without heuristic initialisation (20 000 000 iterations)

| Data | SA (basic) init prob = 0.999 | | SA (heuristic) init prob = 0.7 | | GD (basic) init ceil = 1.0 | | GD (heuristic) init ceil = 1.3 | |
|----------|---------------------------------|----------|-----------------------------------|----------|-------------------------------|----------|-----------------------------------|----------|
| | Cost | Time (s) | Cost | Time (s) | Cost | Time (s) | Cost | Time (s) |
| car-f-92 | 5.27 | 31 | 4.19 | 61 | 4.92 | 90 | 4.14 | 81 |
| car-s-91 | 6.48 | 35 | 4.92 | 81 | 5.48 | 108 | 4.84 | 109 |
| ear-f-83 | 41.18 | 16 | 37.62 | 19 | 39.43 | 46 | 37.39 | 36 |
| hec-s-93 | 12.43 | 11 | 12.02 | 11 | 12.34 | 94 | 11.94 | 51 |
| kfu-s-93 | 16.00 | 49 | 15.31 | 74 | 15.95 | 76 | 15.51 | 64 |
| lse-f-91 | 16.16 | 40 | 10.77 | 56 | 15.80 | 55 | 10.75 | 50 |
| sta-f-83 | 176.34 | 14 | 165.70 | 16 | 176.30 | 32 | 166.47 | 42 |
| tre-s-92 | 9.05 | 20 | 8.44 | 26 | 8.95 | 45 | 8.42 | 38 |
| uta-s-93 | 5.10 | 36 | 3.29 | 75 | 4.79 | 74 | 3.26 | 90 |
| ute-s-92 | 28.30 | 23 | 26.30 | 26 | 27.98 | 52 | 26.74 | 37 |
| yor-f-83 | 40.94 | 13 | 39.44 | 15 | 39.13 | 72 | 39.31 | 71 |

Table 3. Comparison of both methods with and without heuristic initialisation (200 000 000 iterations)

| Data | SA (basic) init prob = 0.999 | | SA (heuristic) init prob = 0.7 | | GD (basic) init ceil = 1.0 | | GD (heuristic) init ceil = 1.3 | |
|----------|---------------------------------|----------|-----------------------------------|----------|-------------------------------|----------|-----------------------------------|----------|
| | Cost | Time (s) | Cost | Time (s) | Cost | Time (s) | Cost | Time (s) |
| car-f-92 | 4.86 | 307 | 4.15 | 340 | 4.52 | 916 | 4.10 | 416 |
| car-s-91 | 5.23 | 353 | 4.73 | 409 | 4.87 | 1156 | 4.65 | 681 |
| ear-f-83 | 39.69 | 163 | 36.57 | 167 | 38.45 | 451 | 37.05 | 377 |
| hec-s-93 | 11.81 | 108 | 11.71 | 98 | 11.65 | 966 | 11.54 | 516 |
| kfu-s-93 | 15.18 | 486 | 14.34 | 506 | 15.17 | 749 | 13.90 | 449 |
| lse-f-91 | 13.53 | 396 | 10.90 | 411 | 12.91 | 643 | 10.82 | 341 |
| sta-f-83 | 176.10 | 141 | 168.37 | 132 | 176.44 | 311 | 168.73 | 418 |
| tre-s-92 | 8.67 | 196 | 8.52 | 192 | 8.54 | 473 | 8.35 | 304 |
| uta-s-93 | 4.60 | 351 | 3.19 | 385 | 4.00 | 881 | 3.20 | 517 |
| ute-s-92 | 26.46 | 231 | 25.88 | 226 | 26.06 | 544 | 25.83 | 324 |
| yor-f-83 | 40.00 | 125 | 37.82 | 121 | 37.16 | 786 | 37.28 | 695 |

both Simulated Annealing and the Great Deluge have great difficulty finding feasible solutions for some of these problems. For longer runs of 200 000 000 iterations in all but one case the solution produced by the basic approach is worse than the solution produced using a well configured combination approach. In general, the Great Deluge emerges as the better method, obtaining the best results on seven of the 11 problems. As we would expect though, the benefits of initial solution quality are much more noticeable when dealing with shorter

Table 4. Comparison of results

| Data | Tabu Solver (av.) | Carter et al. | Adaptive with Great | |
|----------|------------------------|---------------|---------------------|----------|
| | Di Gaspero and Schaerf | | Deluge (1.3) (av.) | |
| | Cost | Cost | Cost | Time (s) |
| car-f-92 | 5.6 | 6.2–7.6 | 4.10 | 416 |
| car-s-91 | 6.5 | 7.1–7.9 | 4.65 | 681 |
| ear-f-83 | 46.7 | 36.4–46.5 | 37.05 | 377 |
| hec-s-93 | 12.6 | 10.8–15.9 | 11.54 | 516 |
| kfu-s-93 | 19.5 | 14.0–20.8 | 13.90 | 449 |
| lse-f-91 | 15.9 | 10.5–13.1 | 10.82 | 341 |
| sta-f-83 | 166.8 | 161.5–165.7 | 168.73 | 418 |
| tre-s-92 | 10.5 | 9.6–11.0 | 8.35 | 304 |
| uta-s-93 | 4.5 | 3.5–4.5 | 3.20 | 517 |
| ute-s-92 | 31.3 | 25.8–38.3 | 25.83 | 324 |
| yor-f-83 | 42.1 | 41.7–49.9 | 37.28 | 695 |

20 000 000 iteration runs. Table 4 shows the average individual results for the Great Deluge when given a ceiling factor of 1.3 and allowed 200 000 000 moves, and also compares them with the results of two other methods on the same problem, namely the tabu search method proposed by di Gaspero and Schaerf [8] and the heuristic backtracking approach used by Carter et al. [7]. The times given are in CPU seconds on a 900Mhz Athlon.

On some problems there is substantial improvement on the previously reported results with the others being comparable. The exception is the sta-f-83 problems, on which our proposed method performs the worst (though not by a large margin) of all the techniques. It is also worth noting that the method also outperforms the tabu search approach proposed by White and Xie [12] on the two problems they used for testing.

It would be impractical to display all the produced results here. However, the keen reader can obtain all individual results, together with the produced solutions, from <http://www.asap.cs.nott.ac.uk/misc/jpn/patat2002>

4 Conclusions

It is clear that under the right circumstances both Simulated Annealing and the Great Deluge method can be used to improve on already good-quality solutions. While obtaining maximal improvement requires a delicate balance of the parameters, using guideline values for these parameters should provide near-maximal improvement. This combination of heuristic construction and a subsequent phase of local search produces very competitive results on the benchmark problems, introducing new best solutions for some of the problems. A possible issue that was not clearly reflected in the experiments is the relationship between the amount of time the search is allowed and how much the search process is limited. It seems

logical that if the search process is given more time, then the limitation on the search space can be relaxed a little. There is a suggestion of this in Figure 5, where the best desired average probability is a little higher when run for 10 times the number of moves. Determining this relationship more formally, however, will require substantial experimentation and will be a focus of further work.

Another area in which this work could have a serious impact is in the development of hyper-heuristics (heuristics to choose heuristics). As an aspect of hyper-heuristics deals with what methods are best to use, given a set time frame in which to work, this approach could form part of a larger hyper-heuristic framework, where the framework decides what parameters to use and for how long to perform local searching.

Acknowledgements. The research described in this paper was supported by EPSRC grant no GR/N36837/01. The authors would also like to acknowledge helpful advice given by Yuri Bykov.

References

1. Burke, E.K., Bykov, Y., Newall, J. P., Petrovic, S.: An Investigation of Time Pre-defined Search for the Examination Timetabling Problem. Technical Report NOTTCS-TR-2002-1. School of Computer Science and IT, University of Nottingham (2002)
2. Burke, E.K., Elliman, D.G., Ford, P.H., Weare, R.F.: Examination Timetabling in British Universities – a Survey. In: Burke, E, Ross, P. (Eds.): Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers). Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg New York (1996) 76–90
3. Burke, E.K., Newall, J. P.: A New Adaptive Heuristic Framework for Examination Timetabling Problems. Technical Report NOTTCS-TR-2001-5. School of Computer Science and IT, University of Nottingham (2002)(to appear in Ann. Oper. Res.)
4. Burke, E., Petrovic, S.: Recent Research Directions in Automated Timetabling. *Eur. J. Oper. Res.* **140** (2002) 266–280
5. Carter, M.W.: A Survey of Practical Applications of Examination Timetabling. *Oper. Res.* **34** (1986) 193–202
6. Carter, M.W., Laporte, G.: Recent Developments in Practical Examination Timetabling. In: Burke, E, Ross, P. (Eds.): Practice and Theory of Automated Timetabling I (PATAT 1995, Edinburgh, Aug/Sept, selected papers). Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg New York (1996) 3–21
7. Carter, M.W., Laporte, G., Lee, S.: Examination Timetabling: Algorithmic Strategies and Applications. *J. Oper. Res. Soc.* **47** (1996) 373–383
8. Di Gaspero, L., Schaerf, A.: Tabu Search Techniques for Examination Timetabling. In: Burke, E, Erben W. (Eds.): Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz, Germany, August, selected papers). Lecture Notes in Computer Science, Vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2001) 104–117

9. G. Dueck. New optimization Heuristics: the Great Deluge and the Record-to-Record Travel. *J. Comput. Phys.* **104** (1993)
10. Dueck, G., Scheuer, T.: Threshold Accepting: a General Purpose Algorithm Appearing Superior to Simulated Annealing. *J. Comput. Phys.* **90** (1990) 161–175
11. Kirkpatrick, S., Gelatt Jr, C.D., Vecchi, M.P.: Optimization by Simulated Annealing. *Science* **220**(4598) (1983) 671–680
12. G. M. White and B. S. Xie. Examination Timetables and Tabu Search with Longer-Term Memory. In: Burke, E, Erben W. (Eds.): Practice and Theory of Automated Timetabling III (PATAT 2000, Konstanz, Germany, August, selected papers). *Lecture Notes in Computer Science*, Vol. 2079. Springer-Verlag, Berlin Heidelberg New York (2001) 85–103