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

Computer Science Technical Report No. NOTTCS-TR-2002-3

A New Local Search Approach with Execution Time as an Input Parameter

E.K. Burke, Y. Bykov, J.P. Newall and S. Petrovic

First released: May 2002

A NEW LOCAL SEARCH APPROACH WITH EXECUTION TIME AS AN INPUT PARAMETER

Edmund Burke, Yuri Bykov, James Newall and Sanja Petrovic
Automated Scheduling and Planning Group
School of Computer Science and Information Technology, University of Nottingham
Wollaton Road, Nottingham NG8 1BB, UK
{ekb,yxb,jpn,sxp}@cs.nott.ac.uk

ABSTRACT

A common drawback of local search metaheuristics such as simulated annealing in solving combinatorial optimisation problems is the necessity to set a number of uncertain parameters. This makes the algorithm problem-dependent and significantly increases the total time of solving the problem. On the other hand, the methods without any parameters (such as hill-climbing) usually produce results of inferior quality.
In this paper we present a new local search technique. That requires only one parameter (which can be interpreted as search time). Generally, a longer search provides a better result, as long as we can intelligently stop the approach from converging too early. Hence, a user can choose a balance between processing time and the quality of the solution. The proposed method has been tested on real-world university examination timetabling problems and the experiments have confirmed the high effectiveness of the proposed technique. Within an acceptable amount of time, our algorithm produced better results than other published approaches for most benchmark problems.

Keywords: Combinatorial optimisation, Local search, Metaheuristic, Timetabling

1. INTRODUCTION

Local search metaheuristics have been among the most successful approaches to solving combinatorial optimisation problems over the last few years. Local search is the common name for the group of methods which (on the whole) iteratively repeat the replacement of a current solution ***s*** by a new one ***s****, until some stopping condition has been satisfied. The new solution is selected from a neighbourhood ***N(s)*** - the set of candidate solutions into which the current one can be transformed, usually by a single move. The quality of the solution is characterised by its fitness (cost function) ***f(s)***. The goal of the search process is to minimise the cost function.

Variants of this typical basic local search approach differ by their mechanisms of accepting or rejecting the candidate solution from the neighbourhood, definitions of neighbourhood and stopping conditions. A description of the local search methods and their applications to different combinatorial optimisation problems can be found in [1].

1.1 Hill-Climbing

The simplest local search algorithm is hill-climbing. This method was used for the timetabling problem as early as 1960 by Appleby et al. [2]. A candidate solution is accepted only if it has better or equivalent fitness than the current one. Hill-climbing does not require the definition of any parameters and its behaviour is quite stable. It aims to converge very fast but often has a final solution of relatively poor quality as it tends to get trapped in local optima.

1.2 Simulated Annealing

Different extensions of hill-climbing allow the acceptance of worse solutions in order to eventually get better ones. These approaches widen the search space and can improve the quality of the result. One of the most widely studied local search metaheuristics is simulated annealing. It was proposed as a general optimisation technique in 1983 [20] and has been repeatedly applied to solve a wide range of problems. An overview of its different applications is given in [21].

Simulated annealing is similar to hill-climbing but accepts worse solutions with a probability: $P = e^{-\delta/T}$, where $\delta = f(s^*) - f(s)$ and the parameter T denotes the temperature in the process of annealing. Originally it was suggested to start the search from a high temperature and reduce it to the end of a process by progression formula: $T_{i+1} = T_i - T_i * \beta$ (geometric cooling schedule). However, the cooling rate β and initial value of T are usually different for different problems and are often selected empirically. This uncertainty causes problems with the practical use of simulated annealing. This has been indicated in the first application of simulated annealing to the timetabling problem by Davis and Ritter in 1987 [12]. They reported that manual enumeration of parameters took two weeks and therefore they developed a genetic algorithm especially for determination of the best values for the parameters.

Different improvements of the basic simulated annealing algorithm have been suggested, such as: adaptive cooling technique where the temperature is reduced or increased depending on the success of the move [13], launching the algorithm several times starting from a random seed [3], cooling schedules with variable cooling rate and reheating [4], making the temperature dependent on the absolute value of the cost function [22], and the "mean field annealing" technique, where the search space is approximated by the system of thermodynamical differential equations [15]. However, the question about the best values of the parameters still has no definitive answer. Moreover some of the proposed improvements introduce new parameters.

1.3 The Threshold Acceptance Algorithm

The deterministic variant of the simulated annealing, known as the threshold acceptance method, accepts every worse solution, when δ does not exceed some threshold T. It was introduced by Dueck and Scheuer in 1989 [14]. They applied the algorithm to a Travelling Salesman Problem and claimed that their algorithm is superior to classical simulated annealing. Originally the authors suggested to decrease the threshold when the algorithm does not improve the solution for a long time. But it is not clear when to do this and how much to decrease it. Although the acceptance procedure was refined, it still involves a few parameters whose values should be deduced empirically. The adaptive cooling scheme (as it was done for simulated annealing) was also introduced for the threshold acceptance method [19], but as in the previous case, it did not yield sensible practical benefits and this technique is not widely applied.

1.4 Contribution

In this paper we introduce a new variant of a local search metaheuristic for solving combinatorial optimisation problems. Its description is given in Section 2. In Section 3 we define an instance problem and present the investigation of the algorithm's properties and its comparison with other techniques. Section 4 includes the summary of our study and outline of our future work.

2. DEGRADED CEILING ALGORITHM

In our proposed approach we keep the acceptance of better solutions but the procedure for the acceptance of worse ones is different to other published approaches. Our algorithm accepts every solution whose objective function is less than or equal to the upper limit B, which is monotonically decreased during the search. We named this algorithm “degraded ceiling” as ***B*** bounds the search space where the intermediate solutions can be either better or worse than each other. The pseudocode of this algorithm is given in Chart 2.1.

```
Set the initial solution s
Calculate initial cost function f(s)
Initial ceiling B=f(s)
Specify input parameter ΔB = ?
While not some stopping condition do
   Define neighbourhood N(s)
   Randomly select the candidate solution  s* ∈ N(s)
   If ( f(s*) ≤  f(s) ) or ( f(s*) ≤ B )
   Then accept  s*
   Lower the ceiling B = B –ΔB
```

Chart 2.1 Degraded ceiling algorithm

The initial value of ceiling is equal to the initial cost function. This forestalls sharp descents and idle steps in the beginning of the search. Hence, in its basic variant only one input parameter ***ΔB***, the decay rate at each step has to be specified.

3. THE EVALUATION OF DEGRADED CEILING ALGORITHM FOR EXAM TIMETABLING PROBLEM

Exam timetabling problems are known to be difficult real world problems that have been studied in some depth over the last few years or so [24]. A survey of the examination timetabling problem is presented in [5] where the authors analyse the requirements from over 50 British universities.

3.1 Examination Timetabling Problems

The university examination timetabling involves the scheduling of exams within a given number of timeslots (periods) while satisfying certain constraints. Usually constraints are classified as either hard or soft. Satisfaction of the hard constraints is a strict requirement, i.e. in a feasible timetable they should not be violated under any circumstances. Soft constraints can be violated, but it is important to minimise those violations. Thus the cost function of every solution indicates the number of violated soft constraints under assumption that all the hard ones are satisfied or it introduces a very high cost for the violating of a hard constraint.

The prime hard constraint is caused by the common restriction of university examination processes that no one student can sit two exams simultaneously. Therefore any two exams which clash (have common students) must not be placed into the same timeslot. The examination timetabling problem in respect of only hard constraints is analogous to the well-known graph colouring problem [8]. Good quality solutions to these problems can be produced by different kinds of heuristic techniques. The family of so-called sequential graph colouring heuristics has been widely applied to timetabling since 1960, and their detailed description is presented in [9].

Soft constraints differ from university to university [5]. Besides simulated annealing, various metaheuristics have been applied to university examination timetabling: tabu search (eg. [18], [25]), genetic algorithms (eg. [11], [16]) and their hybrids with hill-climbing (memetic algorithms) (eg. [6], [23]). The interested reader can see a more detailed description in the following surveys: [5], [8], [9], [24]. Some future research directions and some new approaches are discussed in [7].

Formally, the examination timetabling problem can be specified by considering a number of exams (***N***), a number of students (***M***) and a given number of timeslots (***P***). In addition the conflicts can be represented by a symmetrical matrix $\boldsymbol{C=[c_{ij}]_{NxN}}$ where every element $\boldsymbol{c_{ij}}$ ***(i,j = 1…N)*** represents the number of students, who take both exams ***i*** and ***j***. The solution to a problem can be considered to be a vector $\boldsymbol{T=[t_k]_N}$, where $\boldsymbol{t_k}$ $\boldsymbol{(1 \le t_k \le P)}$ denotes the timeslot for exam ***k (k = 1…N)***.

The satisfaction of hard constraints can be stated in the following way:

$$\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}\cdot clash(i,j)=0 \qquad \text{where} \qquad clash(i,j)=\begin{cases}1 & if\ t_i=t_j\\ 0 & otherwise\end{cases} \tag{3.1}$$

In order to make a proper comparison with other approaches to exam timetabling problems we use the same cost function, as suggested in [8]. It penalises the average proximity between exams for each student and can be expressed by the following equation.

$$F_C=\frac{\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} c_{ij}\cdot prox(i,j)}{M} \qquad \text{where} \qquad prox(i,j)=\begin{cases}2^{5-|t_i-t_j|} & if\ 1\le |t_i-t_j|\le 4\\ 0 & otherwise\end{cases} \tag{3.2}$$

Every pair of adjacent exams adds 16 to a total penalty, two exams with one period between them generates a cost of value 8, an interval of 2 is penalised by value 4 and so on. The total sum of penalties is divided by the number of students to give a relative measure.

3.2 The Progress Diagram

In our experiments we used publically available examination timetabling data benchmarks. It is located at: ftp://ftp.mie.utoronto.ca/pub/carter/testprob, and comprises 13 datasets from different universities. Their characteristics are given below in section 3.4. We have carried out experiments on all these problems and our method showed practically the same behaviour on all of them.

The investigation of the properties of the degraded ceiling algorithm was started by generating progress diagrams such as that presented in chart 3.1. The algorithm was implemented in Visual C++ 6.0 and launched on PC Athlon 750 MHz with OS Windows 98. Decay rate ***ΔB*** was defined as 0.0005. Every 100 000 moves the current cost $\boldsymbol{F_C}$ and the number of moves $\boldsymbol{N_{mov}}$ were depicted as a point in the "time-cost" space. An example of a resulting diagram for dataset KFU-S-93 is presented on Chart 3.1.

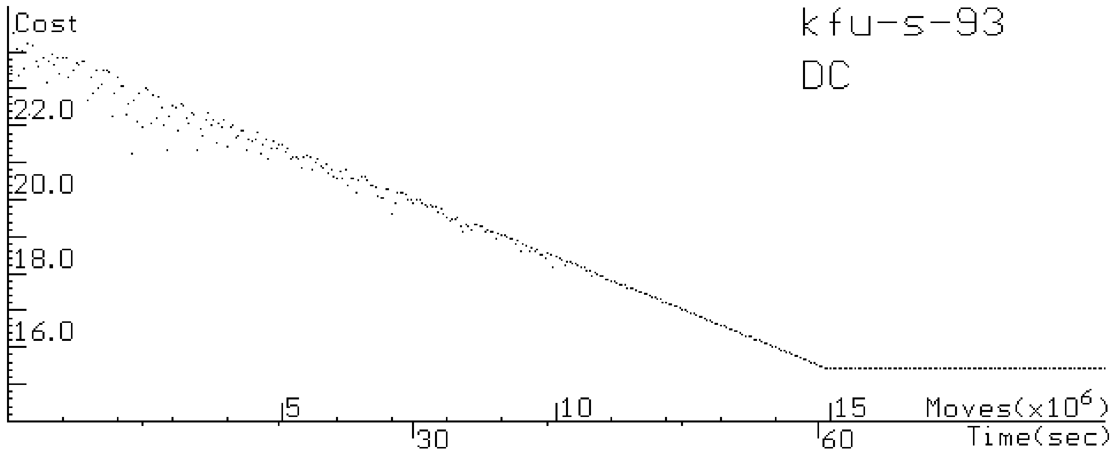

Chart 3.1 The progress of degraded ceiling algorithm

This diagram shows two main properties of the degrading ceiling algorithm:

1. The profile of the process is explicit. The search rigidly follows the degrading of the ceiling. Fluctuations are visible only at the beginning but later all intermediate solutions lie close to the line $\boldsymbol{F_C = B_0 - \Delta B * N_{mov}}$.

2. The point of convergence is quite recognisable. When a current solution reaches the value, where any further improvement is impossible, the search rapidly converges and the diagram becomes level. This moment can be easily detected in order to terminate the search procedure.

These properties of the algorithm allow us to fit the search procedure into a certain time period. Taking into account that the cost function cannot be negative, we can contend that our algorithm *guarantees* producing a result in time ($\boldsymbol{T_P}$) calculated by the following expression:

$$T_P = T_{mov} \cdot N_{mov} = T_{mov} \cdot \frac{B_0}{\Delta B} \qquad \text{where } \boldsymbol{T_{mov}} \text{ is the time of one move} \qquad (3.3)$$

However, in most problems, the algorithm converges before $\boldsymbol{T_P}$ expires. The point of convergence is uncertain and problem-dependent. Therefore if some information about the range of possible results is available, we suggest using it for reducing the number of idle steps. If we estimate the cost function of the future result as $\boldsymbol{f(s')}$ we can calculate $\boldsymbol{\Delta B}$ by formula (3.4).

$$\Delta B = \frac{B_0 - f(s')}{N_{mov}} \qquad (3.4)$$

Usually such approximation is quite possible. For example, in our following experiments we approximated $\boldsymbol{f(s')}$ by employing the results of a simple hill-climbing algorithm.

3.3 Time-Cost Diagrams

The influence of processing time to the performance of the method was investigated while launching the algorithm several times for a different predefined number of moves. The results are presented as “time-cost” diagrams where every point corresponds to the final cost function and processing time of a separate solution. The example of such diagram for the CAR-S-91 problem is shown in Chart 3.2.

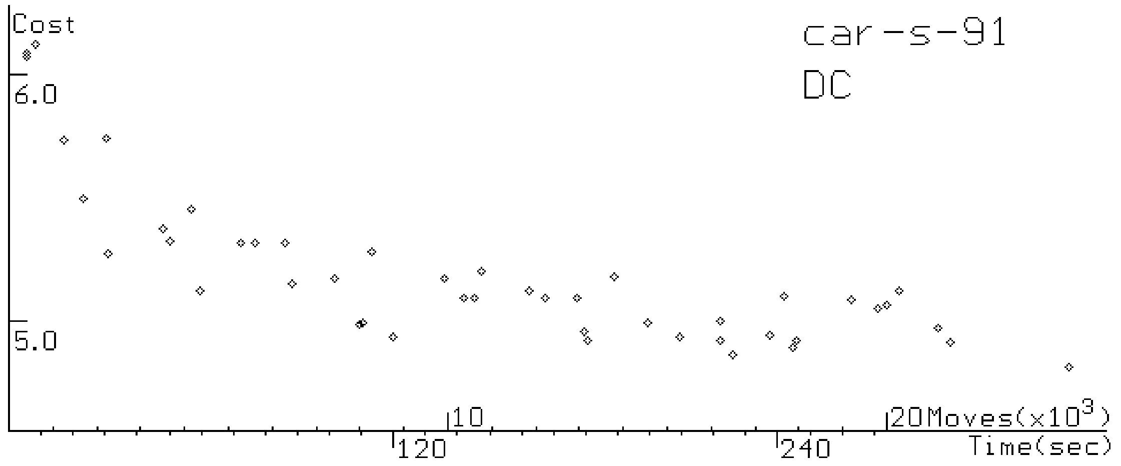

Chart 3.2 Time-cost diagram of degraded ceiling algorithm

Even though the results are relatively scattered (which is not surprising), the clear tendency in this diagram can be observed: *a longer search produces better results.* The algorithm allows a user to improve the quality of the solution but he/she should pay a price for it with an increase in the amount of processing time.

The reasonable balance between time and cost depends on the user's opportunities and preferences. In some cases the user requires results quickly but in other situations he/she agrees to spend much more time to search for a high quality solution. In the case of timetabling, very fast but relatively poor results cannot be considered as a best choice. In most situations the calculation of the solution is just part of the process, which includes the preparation of input data and administering the results of the software. Commonly it takes several days. The renewing of data is infrequent (because an examination timetable is normally only produced once or twice a year). However, the high quality of the solution is very important as the examination process affects a high number of people. In this environment a searching procedure, which can last several hours, seems to be quite acceptable.

3.4 Comparison with Other Techniques

As the quality of the solution depends on the search time, the performance of the degraded ceiling algorithm can be estimated only in respect of the time-cost diagram. Correspondingly, it can be properly compared with any other approach only if a time-cost diagram is also available for the second competitor. Therefore in the following experiments we tried to distribute the outcome of compared techniques evenly in the given time period by varying their parameters.

An example of comparison of our method with simulated annealing for TRE-S-92 dataset is shown in Chart 3.3. The simulated annealing algorithm was launched several times with variations of the initial temperature from 10^{-3} to 10^{5} (our results confirmed that the best value is located inside this interval). In order to get approximately the same launch time in both algorithms, the cooling rate $\boldsymbol{\beta}$ was varied from 10^{-6} to $2*10^{-5}$.

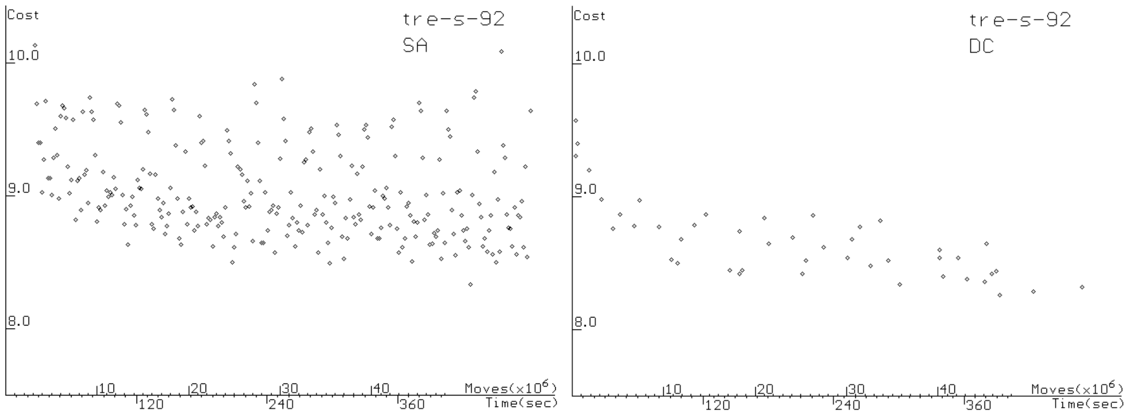

Chart 3.3 Comparison of simulated annealing and degraded ceiling algorithms

The simulated annealing diagram shows a substantially higher scatter of results than the degraded ceiling algorithm. A greater number of poor quality solutions are generated though the use of inappropriate parameter values. The superiority of the degraded ceiling algorithm is obvious from these diagrams. Although both methods have approximately the same values of the cost function for the best results (in the given launch time), simulated annealing *can reach* it only with properly defined parameters, while degraded ceiling *does* it always.

Another advantage of the degrading ceiling algorithm is the effectiveness of the search. The deriving of the best cooling schedule requires several launches of the simulated annealing algorithm. Therefore its total processing time (from input to output) is several times longer than the processing time of a single launch. With degraded ceiling all this time is spent in a single launch (and it gets better results). Hence, with respect to the total processing time, the performance of the degraded ceiling is substantionally better.

The same comparison was carried out with the threshold acceptance algorithm. For the dataset CAR-F-92 the initial threshold was varied in the interval: $5*10^{-5}$ – 2.5 and rate of its decreasing: 10^{-12} - $2*10^{-5}$. The resulting diagrams are presented on Chart 3.4, which shows the same behaviour as for simulated annealing.

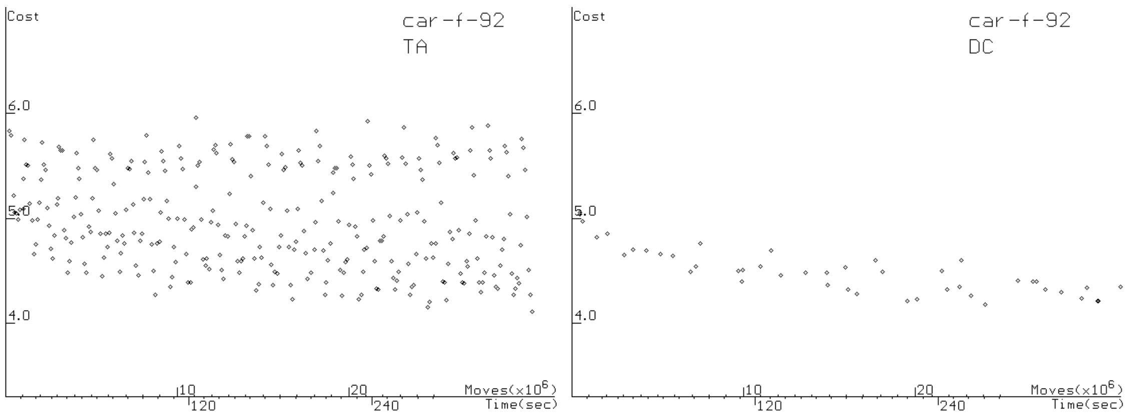

Chart 3.4 Comparison of threshold acceptance and degraded ceiling algorithms

In the experiments on hill-climbing the time-cost diagrams were produced with a very short search time. The search time of hill-climbing depends on the stopping condition. We used as the stopping condition the given number of idle steps and it was varied in the range of 1-10000. The results for YOR-F-83 problem are presented in Chart 3.5.

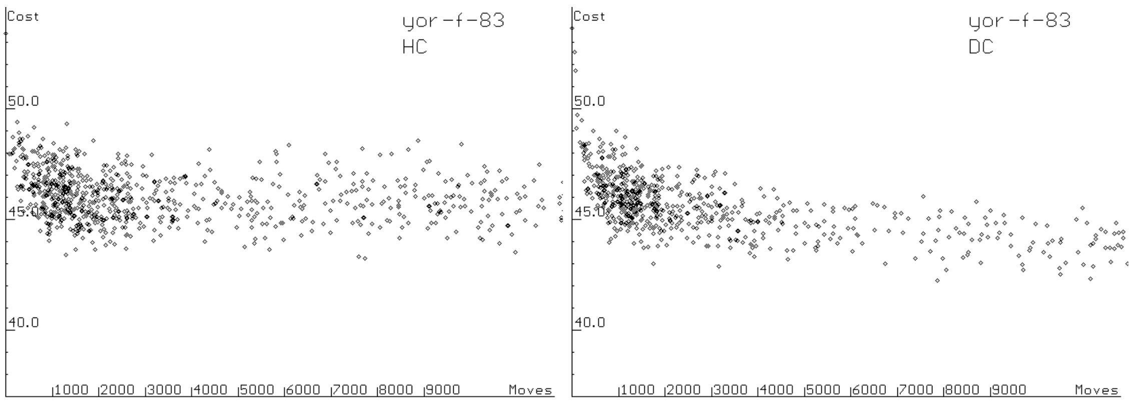

Chart 3.5 Comparison of hill-climbing and degraded ceiling algorithms

Both diagrams have the same distribution of points at the beginning. This means that if the stopping condition of hill-climbing is chosen correctly then both algorithms have the same performance. However the behaviour of those techniques in the right hand sides of diagrams became different. If the chosen number of idle steps is too high – hill-climbing wastes this additional time, but degraded ceiling uses it for improving the solution.

3.5 Comparison with Published Results

It is clear (from section 3.4) that the degrading ceiling algorithm outperforms simulated annealing and hill-climbing. In this section the degraded ceiling algorithm is compared with other published techniques. We consider sequencing heuristics with backtracking (SH) [10] and tabu search with a variable tabu list (TS) [17]. However, an analysis of the time-cost diagrams (as in the previous section) cannot be done because only a few values of the final cost are available and computing times are incomparable due to the use of different hardware. Thus we indicate in our comparison only the best published fitnesses (with reference to the used method). From our time-cost diagrams we picked up the best results (together with their time) and average costs (calculated from five results, placed in the right part of the diagrams). These results are compiled in Table 3.1. The best known results are presented in bold.

Table 3.1 Published and our results for proximity cost

Data set	Exams	Students	Time slots	Published results		Degraded Ceiling Algorithm		
				Best cost	Method	Best cost	Average cost	Time for best (sec)
CAR-F-92	543	18 419	32	5.2	(TS)	**4.2**	4.3	274
CAR-S-91	682	16 925	35	6.2	(TS)	**4.8**	5.0	328
EAR-F-83	189	1 125	24	36.4	(SH)	**35.4**	36.7	134
HEC-S-92	80	2 823	18	**10.8**	(SH)	**10.8**	11.5	278
KFU-S-93	461	5 349	20	14.0	(SH)	**13.7**	14.4	729
LSE-F-91	381	2 726	18	10.5	(SH)	**10.4**	11.0	1030
PUR-S-93	2419	30 032	43	**3.9**	(SH)	4.8	4.9	1412
RYE-S-93	481	11 483	23	**7.3**	(SH)	8.9	9.3	752
STA-F-83	138	611	13	160.8	(TS)	**159.1**	159.4	157
TRE-S-92	261	4 360	23	9.6	(SH)	**8.3**	8.4	392
UTA-S-92	638	21 267	35	3.5	(SH)	**3.4**	3.5	585
UTE-S-92	184	2 750	10	25.8	(SH)	**25.7**	26.2	236
YOR-F-83	180	941	21	41.0	(TS)	**36.7**	37.2	546

The computing time for our best results is presented only to show that the given solutions were produced in quite an acceptable time (10-30 min). Nevertheless, 10 out of 13 of our best results are better than the best published ones. Moreover, our average results are better in 6 out of 13 cases than the best previously achieved. Even if the best published results were produced quicker, the prolongation of time in our algorithm is well justified by the quality of the obtained solutions.

4. CONCLUSIONS AND FUTURE WORK

This paper introduces a variant of local search which we named “degraded ceiling”. It does not exploit the properties of any particular problem and can be considered as a general purpose technique. The advantage of this method is that it requires the definition of only one parameter that corresponds to a search time. This is problem-independent and has an obvious physical meaning, which precludes its inappropriate definition and means that we do not need to launch the algorithm several times to find the best value. Its efficiency can be further improved with the estimation of the desired cost, as idle steps at the end of the process can be drastically reduced.

Our algorithm shows the clear trade-off between search time and the quality of the overall result, which is that a longer search produces better solutions. This property allows the user to choose an acceptable processing time for each particular problem. The experiments with benchmark exam timetabling problems confirm the advantages of the presented technique. For most datasets, the degraded ceiling algorithm achieved results better than any previously published ones.

Our future work will include evaluation of the algorithm in other domains. Additional issues will be investigated: how to choose good initial solutions, how to define non-linear ceiling functions, hybridisation of the degraded ceiling with other metaheuristics, etc.

BIBLIOGRAPHY

[1] Aarts, E. and Lenstra, J. K. (1997), “Local Search in Combinatorial Optimization (Wiley-Interscience Series in Discrete Mathematics and Optimization)”, John Wiley & Sons, Chichester.

[2] Appleby, J. S., Blake, D. V. and Newman, E. A. (1960), "Techniques for Producing School Timetables on a Computer and their Application to other Scheduling Problems", The Computer Journal, Vol. 3, 237-245.

[3] Abramson, D. (1991), “Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms”, Management Science, Vol. 37(1), 98-113.

[4] Abramson, D., Krishnamoorthy, M. and Dang, H. (1999), “Simulated Annealing Cooling Schedules for the School Timetabling Problem”, Asia-Pacific Journal of Operational Research, Vol. 16, 1-22.

[5] Burke, E. K., Elliman, D. G., Ford, P. H. and Weare, R. F. (1996), “Examination Timetabling in British Universities: a Survey”, in Burke E., Ross P. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95). Lecture Notes in Computer Science, Vol. 1153. Springer-Verlag, Berlin Heidelberg, New York, 76-90.

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

[7] Burke, E. K. and Petrovic, S. (2002), “Recent Research Directions in Automated Timetabling”, to appear in European Journal of Operational Research –EJOR.

[8] Carter, M. W. (1986) “A Survey of Practical Applications of Examination Timetabling Algorithms”, Operations Research, Vol. 34(2) , 193-201.

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

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

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

[12] Davis, L. and Rutter, L. (1987), “Schedule Optimization with Probabilistic Search”, Proceedings of the 3rd IEEE Conference on Artificial Intelligence Applications (Orlando, Florida, USA, Feb. 1987), IEEE1987, 231-236.

[13] Dowsland, K. (1993), “Simulated Annealing”, in: Reeves C.R. (ed.) “Modern Heuristic Techniques for Combinatorial Problems”, Blackwell, 20-69.

[14] Dueck, G. and Scheuer, T. (1990), “Threshold Accepting: A General Purpose Optimization Algorithm Appearing Superior to Simulated Annealing”, Journal of Computational Physics, Vol. 90, 161-175.

[15] Elmohamed, M. A. S., Coddington, P. and Fox, G. (1998) “A Comparison of Annealing Techniques for Academic Course Scheduling”, in Burke E., Carter M. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers (PATAT '97). Lecture Notes in Computer Science, Vol. 1408, Springer-Verlag, Berlin Heidelberg, New York, 92-112.

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

[17] Di Gaspero, L. and Schaerf, A. (2001) “Tabu Search Techniques for Examination Timetabling”, in Burke E., Erben W. (eds.): The Practice and Theory of Automated Timetabling III: Selected Papers (PATAT 2000). Lecture Notes in Computer Science, Vol. 2079, Springer-Verlag, Berlin Heidelberg, New York, 104-117.

[18] Hertz, A. (1991), “Tabu Search for Large Scale Timetabling Problems”, European Journal of Operational Research, Vol. 54, 39-47.

[19] Hu, T. C., Kahng, A. B. and Tsao, C. W. (1995), “Old Bachelor Acceptance: A New Class of Non-Monotone Threshold Accepting Methods” ORSA Journal on Computing Vol. 7(4), 417-425.

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

[21] Koulmas, C., Antony, S. R. and Jaen, R. (1994) “A Survey of Simulated Annealing Applications to Operations Research Problems”, Omega International Journal of Management Science Vol. 22, 41-56.

[22] Melicio, F., Caldeira, P. and Rosa, A. (1999), “Solving the Timetabling Problem with Simulated Annealing”, Proceedings of the First International Conference on Enterprise Information Systems (ICEIS'99), 272-279.

[23] Paechter, B., Cumming, A., Norman, M. G. and Lucian, H. (1996), “Extensions to a Memetic Timetabling System”, in Burke E., Ross P. (eds.): The Practice and Theory of Automated Timetabling: Selected Papers (ICPTAT '95). Lecture Notes in Computer Science, Vol. 1153, Springer-Verlag, Berlin Heidelberg, New York, 251-265.

[24] Schaerf A. (1999), “A survey of automated timetabling”, Artificial Intelligent Review, Vol. 13, 87-127.

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