IMPLEMENTATION OF A TABU SEARCH HEURISTIC FOR THE EXAMINATIONS TIMETABLING PROBLEM

AR Mushi
Department of Mathematics, University of Dar es salaam, Box 35062, Dar es salaam, Tanzania
allen.mushi@gmail.com

ABSTRACT

This paper reports on the design and implementation of an algorithm for the construction of an examinations timetable. The Examinations Timetabling Problem is the problem of assigning examinations and candidates to time periods and examination rooms while satisfying a set of specific constraints. Every University has a different set of constraints and structure of examinations. Generally, timetabling problems are NP-Hard and therefore very difficult to solve. However, they are of great interest due to their important practical application in educational institutions. This paper discusses a heuristic algorithm basing on the examinations timetabling at the University of Dar es salaam. The algorithm uses a Tabu Search technique, which has been successfully applied to other variations of the problem. Real life instance of the problem has been solved within reasonable time and compares with results of the previous work which used a Simulated Annealing Algorithm. It is concluded that the algorithm gives a better performance than manual system and compares well with Simulated Annealing results; Tabu Search is therefore applicable and a good approach to the problem of Examinations timetabling at UDSM.

Keywords: Timetabling, Tabu Search, Combinatorial Optimization, Scheduling

INTRODUCTION

The Examination-Timetabling Problem (ETP) is the problem of assigning examinations and candidates to time periods and examination rooms while satisfying a set of specific constraints. Constraints are normally divided into hard and soft (Elmohamed et al, 1998, Slim and Marte, 1998), and where hard constraints must be satisfied while soft constraints may be tolerable but must be satisfied as much as possible. ETP is NP-Hard (Cooper and Kingston 1996, de Werra 1985) and therefore no optimal algorithm is known which can give a solution within reasonable time. Due to its practical importance, many researchers have attempted to solve different variations of the problem.

A good number of papers have been published on ETP for specific Universities. A survey on automated timetabling is found on Schaef (1995). De Werra (1985) gives a good introduction into timetabling problems. Previous attempts include the use of graph colouring concepts, (see the work done by Welsh and Powell, 1967), by reducing ETP into the chromatic number problem which is also NP-Hard (Karp, 1972). Some attempts have also been made to use mathematical programming techniques (Dimopolou and Milliotis, 2001) with success only on small size instances.

Due to the complexity of the problem, most of the literature concentrates on the heuristic algorithms, including Genetic Algorithms (Corne *et al*. 1993), Tabu Search (Mooney 1991), Simulated Annealing (Cooper and Kingston 1996), and Scatter Search (Marti *et al*. 2001). Some researchers have recently paid attention to the Logic Programming techniques, the so-called Constraint Satisfaction Problems, because they have a natural way of representing constraints, such as a paper by Stamapoloulos *et al*. (1998). However, such methods have some

limitations including unstable performance as pointed out by Bartak (1999).

Articles describing practical case studies are relatively few compared to theoretical work on the timetabling problems. These include Carter et al (1994) who described a system called EXAMINE, and applied to Universities of Toronto and Carleton, and a case study by Thomson and Dowsland (1995) on a system called TISSUE. Kendall & Hussin (2005) describes a Tabu Search hyper heuristic specifically for ETP at University of Technology MARA; Ayob et al (2007) describes a formulation of an ETP for Universiti Kebangsaan Malaysia by proposing a useful objective function. Ayob & Hamdan (2009) proposed a multistage Tabu Search for ETP and compared their results on benchmark tests where they reported some promising results requiring improvement.

This paper is a contribution to the case studies with a focus to UDSM. Mushi (2007) presented a first attempt to automate examinations timetabling at UDSM and used Simulated Annealing. This is another attempt to improve on the previous work by applying Tabu Search and compare the results.

The Examinations Timetabling Problem differs significantly from the Course Timetabling Problem. The major differences include;

- An examination requires only one timeslot throughout the planning period, while course timetable may require several timeslots and with different categories such as lectures, tutorials and practical sessions.
- An examination timetable may span several weeks depending on the University's policy, while a course timetable must fit within a week and to be repeated for a full semester.
- An examination room can occupy more than one examination at the same time, while a course timetable can only have one lecture in a room at a time.
- An examination may span several rooms at the same time depending on the number of candidates and rooms availability, while a course timetable can only take place in a single room.
- Some constraints differ in weight between the two types of problems. For instance, distance between examination rooms is not a problem in examinations timetable, but it is an important issue for consideration when planning a course timetable.
-

In this paper a heuristic algorithm is discussed based on Tabu Search as applied to the examinations timetabling problem at the University of Dar es salaam (UDSM) in Tanzania, East Africa. Firstly an overview of the examinations timetabling process at UDSM is presented, then follows the discussion on the design and implementation issues of the algorithm. Lastly, summary of results and conclusions are presented.

Examinations timetable at the University of Dar es salaam

The examinations period is fixed to 2 weeks, with 3 examination sessions per day except on Fridays where we have 2 sessions to allow time for Muslim and Seventh Day Adventist prayers. An examination week is made up of five days, from Monday to Friday. Some examinations have more candidates than a single room can hold, thus some examinations are scheduled in more than one room. On the other hand, a room can have more than one examination scheduled in it if sufficient room space is available. To optimize examination space, lecture theatres are also used for examinations. However, since lecture theatres have close and slanted seats, the examination capacity is significantly small compared to lecture capacity. Cafeterias are also used for examinations but only during the morning session of the day since they are later used for serving food. During the research period, most examinations were 3 hours long but there were a few deviations

between 2_ and 3_ hours, and majority of the examinations were closed book.
Because the university campus is very large (more than 6000 hectares), there is a gap of at least 1_ hours between examination sessions, so that the distance between examination rooms is not a problem to candidates. Examination invigilators are normally scheduled by individual departments after the release of the main timetable. The central timetable is therefore not involved in scheduling invigilators. The problem implemented in this algorithm includes all hard constraints and the most important soft constraints as follows:

Hard Constraints

i) A candidate cannot do more than one examination at the same time
ii) A room cannot be assigned more candidates than its capacity
iii) Only rooms with standby generators can be used for the evening session of the day. This is to avoid the risk of power cuts, which is a possibility.
iv) Cafeterias can only be used during the first timeslot of the day.

Soft Constraints

i) Discourage continuous examinations for a candidate in a day. Since there are three sessions in a day, we prefer to have at least one gap between examinations per candidate in a day. In general, we would like to spread each candidate examinations as much as possible within the planning horizon.
ii) Minimize the split of examinations into rooms. The ideal situation is when each examination is conducted in a single room. When this is not possible, then a set of rooms for each examination should be minimal, so as to help departments in planning for the scarce invigilators.
iii) Large examinations should be scheduled earlier, to allow more time for examiners to mark the large number of scripts associated with the examination.
iv) There are some special requests by both students and departments in the order of examinations due to mostly personal reasons.

Tabu Search Algorithm

Tabu Search (TS) is a global heuristic technique which tries to avoid falling into local optima by creating a special list called Tabu (Reeves, 1993). Any solution which has been recently selected is put into a Tabu list so that it becomes 'taboo' for a short period of time, depending on the length of the list. This minimizes the chance of cycling in the same solution, and therefore creates more chances of improvement by moving into unexplored areas of the search space. The work by Glover (1990) and Glover & Laguna (1997) gives a comprehensive description of the technique. TS is selected for this project because of its success in other timetabling scenarios (including Ab Malik et al 2009; Kendall and Hussin 2005; Mooney 1991). The algorithm applied in this project is;

Tabu Search {
Initialize parameters;
Get Initial Solution (S_o);
Converged = false;
While Not Converged {
Get a set of solutions S in neighborhood of S_o($S \in N(S_o)$)
Moved = false;
While Not Moved {
if S Not Tabu {
Given an objective function f, find $\sigma = f(S) - f(s_o)$;
Push S into Tabu list
Moved = true;
if($\sigma < 0$) Accept solution ($S_o = S$);}
Next S $\in N(S_o)$ }
Converged = Test convergence; }
return S_o as best solution; }

The most challenging part in this general heuristic lies in the design and implementation of variables specific to UDSM. The following section describes the

representation of the Tabu Search heuristic to UDSM examination timetabling problem.

Representation of the problem

The discussion in this section is also described in Mushi (2007), and summarised here for clarity of presentation. Since there are 3 examination sessions per day from Monday to Thursday and two sessions on Fridays, there are 28 examination sessions for the two examination weeks. In this project, each of these time sessions is called a timeslot and define T = total number of timeslots.

Given a total of n rooms and m examinations, lists R_n and C_n are defined such that;
R_i = capacity of room i,
C_i = Number of candidates for examination i
A timetable is represented by an integer-valued list of events S_m such that;
s_i = Timeslot assigned to examination i

Since an examination can span more than one room, an adjacency list K_m is defined, where each node of the list stores all rooms assigned to the examination i.e. k_i = A list of rooms allocated to examination i.

On the other hand, since a room can be assigned to more than one examination at the same time, there is a need to keep track of all remaining space in a room after allocation, so that the room can be used for further examination assignment. To achieve this, we declare a matrix P_{nxT} such that,
P_{ij} = remaining capacity of room i at timeslot j.

Initially, P stores the maximum capacity of each room, but capacities are decremented by the size of the examinations assigned to the room, so that the remaining capacity can be used by other suitable examinations. Thus a fully utilized room has a capacity of zero.
In this setting, hard constraints (iii) and (iv) can be enforced by simply assigning values of zero to all restricted timeslots in rooms with no standby generators and cafeterias respectively.
A conflict matrix M_{mxm} is defined such that;

$$M_{ij} = \begin{cases} 1 & \text{if exam } i \text{ clashes with exam } j \\ 0 & \text{Otherwise} \end{cases}$$

Examinations i and j clashes if they have at least one student in common.

Cost function

Given a solution s, and a set of v constraints,

$$\text{Minimize } f(s) = \sum_{i=1}^{v} \lambda_i f_i(s),$$

Where f_i = cost function associated with constraint i and
λ_i = weight given to constraint i which represents the importance of the constraint to the overall performance measure of the solution.
Higher penalties are assigned to hard constraints to discourage them from appearing in the best solution.

Modelling Constraints

Let E = set of all examinations
F = set of all rooms

Then the components which makes up the cost function are defined as follows;

i) *No candidate can have more than one examination at the same time*;
Two examinations i and j have a candidate collision if they belong to the same time slot (i.e. $s_i = s_j$) and $M_{ij} = 1$. Thus it only suffices to minimize $\lambda_1 f_1(s)$, where

$$f_1(s) = \sum_{\substack{(i,j) \in E \ni S_i = S_j \\ i<j}} M_{ij},$$ and λ_1 is a large value.

$f_1(s)$ is the total number of candidate collisions associated with the current solution.

ii) *No room can be assigned to examinations with more candidates than the room capacity*;
The number of times that a room has been assigned more students than its capacity is calculated as follows;

Let L_i = a set of examinations assigned to room *i*

The remaining capacity of room *i* is then given by; $Cap\ (i) = R_i - \sum_{j \in L_i} c_j$.

For feasibility, the condition $Cap\ (i) \geq 0$ must be satisfied.

So let $b_i = \begin{cases} 1 & if\ Cap(i) < 0 \\ 0 & Otherwise \end{cases}$, for some room *i,* and minimize $\lambda_2 f_2(s)$, where;

$f_2(s) = \sum_{i \in E} b_i$, with λ_2 a large value.

The function $f_2(s)$ gives the total number of rooms which have been assigned examinations with higher number of candidates than the room capacity. For a feasible solution, the condition $f_2(s) = 0$ must be satisfied.

iii) *Minimize the number of splits of an examination into rooms;*
The procedure is required to minimize the number of rooms assigned to an examination, which is achieved by minimizing the maximum number of rooms per examination. Thus we minimize $\lambda_3 f_3(s)$, where;

$f_3(s) = \max_{i \in F} \{|K_i|\}$

iv) *Minimize continuous examinations per each candidate*;
Since there are many candidates with complex course selections, this constraint may have many conflicting constraints between candidates. The number of examinations with common students which are scheduled in consecutive timeslots is to be minimized. Note that two examinations *i* and *j* are scheduled in consecutive timeslots if $|s_i - s_j| = 1$, and have candidates in common if $M_{ij} = 1$. Thus, the problem is to minimize $\lambda_4 f_4(s)$ where;

$$f_4(s) = \sum_{\substack{(i,j) \in E \ni |S_i - S_j| = 1 \\ i<j}} M_{ij}$$

v) *Large examinations should be scheduled first;*
There is a need for a relation which will give priority to large examinations on the lower-numbered timeslots. Experimental results shows that the following function provides a good measure; t^2/z , where *t* is the timeslot and *z* is the size of an examination. Thus, the cost function involves minimizing $\lambda_5 f_5(s)$, where;

$$f_5(s) = \sum_{i \in E} \frac{s_i^2}{c_i}$$

vi) *Examinations scheduled for evening sessions must be assigned to rooms with standby generators*;
Let G = set of all rooms fitted with standby generators and W = set of all evening timeslots of the planning horizon i.e. V = {3, 6, 9, 12, 17, 20, 23, 26}. We simply assign zeros to all rooms $r \in$ G at all timeslots $t \in$V in P i.e. $P_{rt} = 0\ \forall r \in G, \forall t \in V$.

vii) *Cafeterias can only be used during the first morning session of the day*;
Like the previous constraint we simply assign zeros to the respective positions in P i.e. $P_{rt} = 0\ \forall r \in X, \forall t \in Y$
where X = set of all cafeterias, and Y = set of all afternoon and evening timeslots.

Initial timetable

It is important to have an easy and quick way of generating an initial feasible timetable. Each examination is assigned to the earliest possible feasible timeslot and earliest feasible room. To achieve the assignment of large examinations first, both examinations and rooms are sorted in descending order of their sizes. The initial solution is to be feasible only by satisfying all the hard constraints. It is not necessary though that the initial solution should be completely feasible, since we penalize higher on hard constraints in the cost function. An initial solution is therefore preferred to be as feasible as possible, and any infeasibility

can be tolerated in anticipation of improvement in the Tabu Search process. Note that, during assignment of rooms to an examination, it is necessary to search for the available room space even in rooms which have already been used, as they may still have free space.

Tabu Search implementation

One of the major challenges in Tabu Search is the selection of types of moves. Several types of moves are possible, but two types in particular have shown promising results.

The first type is the change of a timeslot by the randomly generated one as follows;

4. Select a random examination *e* in the set of all possible examinations
5. Select randomly a new timeslot *t* in the set of all possible timeslots.
6. Assign the new timeslot *t* to the examination at position *e*.

Similar moves have been described by, Thomson and Dowsland (1996), Saleh and Coddington (1998) and Reeves (1999). The size of the neighbourhood associated with this kind of move is

$|N(s)| = |e| \times (|t|-1)$, where $|e|$ = total number of examinations, and $|t|$ = total number of timeslots.

The second type of move is the swap of two timeslots as follows;

1. Select randomly two examinations e_1, e_2 in the set of all possible examinations
2. Swap time slots of the two examinations

The size of the neighbourhood associated with this kind of move is $|N(s)| = |e|x(|e|-1)$.

The algorithm is tested by using these two move types.

Aspiration criteria

Sometimes a candidate solution could be in the Tabu list, but would bring large improvement in the solution if accepted. In this case an aspiration criteria is used, where a candidate solution with large improvement in the solution is accepted regardless of its Tabu status. We accept any solution which brings an improvement of a value with $\sigma \leq -100$.

Stopping criteria

Fixing the number of iterations has a disadvantage that the algorithm can run for a long time without improvement just to complete the set of iterations. A stopping criterion has been applied, which considers the number of iterations without a change in solution value. The algorithm stops after running 1000 iterations without solution change. This is an experimental value that has been found to be sufficient to show lack of improvement.

SUMMARY OF RESULTS

The algorithm was tested on an examination timetabling problem previously solved by manual methods for semester 1 of the 2003/2004 academic year. A program written in C++ is tested by running on a 2.4 GHz, Pentium 4 processor. Table 1 shows the data for the specific problem that we have tested;

Data	Value
Candidates	8161
Rooms	74
Examinations	729
Total timeslots	28

Table 1: Data for the tested problem at UDSM

Table 2 shows the weights used in the cost function for each type of constraint. These weights have been assigned according to our experience on the user needs. In this case, avoiding continuous examinations is more important to candidates than the number of examination splits into rooms. Likewise, scheduling large examinations early in the timetable is more important to examiners than examination splits. A value of 10 was found to be sufficiently large to prevent infeasibilities in the solution space.

Weight	Value	Description
λ_1 - λ_2	10	Hard constraints
λ_3	1	Number of examination splits
λ_4	2	Continuous examinations
λ_5	2	Large examinations first

Table 2: Weights used in the objective function

These choices of λ have performed very well as shown in Table 3 which presents a summary of results and performance of the algorithm. The presented values are the average of values obtained by runs made for varying random number seeds. The two columns show the performance, using the time change and time swap moves respectively. Time change performs slightly better than time swap. This may be caused by the restrictions imposed by the time swap on the possible timeslots for swapping. The time change moves allows random generation of timeslots which entail the introduction of new timeslots which may not be used by the current solution. The best solution was found after 2,965.69 seconds which is approximately 50 minutes. This time is tolerable in this timetable application compared to three weeks required for the manual timetable generation.

	Time change	Time swap
Initial cost	249.975	249.975
Final Cost	10.865	14.475
Student/Lecturer Collisions	0	0
Room size	0	0
Continous exams	0	0
Cafeterias violation	0	0
Large exams first	5.865	9.475
Exam splits	5	5
Standby generators violation	0	10
Iterations	100,000	100,000
Time	2,790.82	2,965.69
% Cost improvement	96%	94%

Table 3: Performance of the Algorithm

Figure 1 compares the performance of time move and time swaps in improving the cost value by iterations. The performances of the two cases are very close to each other, but time moves perform slightly better on the lower cost values. In both cases, the algorithm was stopped after 100,000 iterations which show a sufficient convergence to a fixed value. The best Tabu list size used was 24.

Table 4 represents a comparison of the performances of the manual and automatic systems in terms of constraint violations. Both timetables are feasible by satisfying the hard constraints but the automatic systems clearly outperform the manual system in achieving the soft constraints. The performances of Tabu Search and Simulated Annealing are very close (96% and 96.08 respectively). Simulated Annealing is relatively faster, with 37 minutes compared to 50 minutes of the Tabu Search. In general, the two algorithms do not differ significantly.

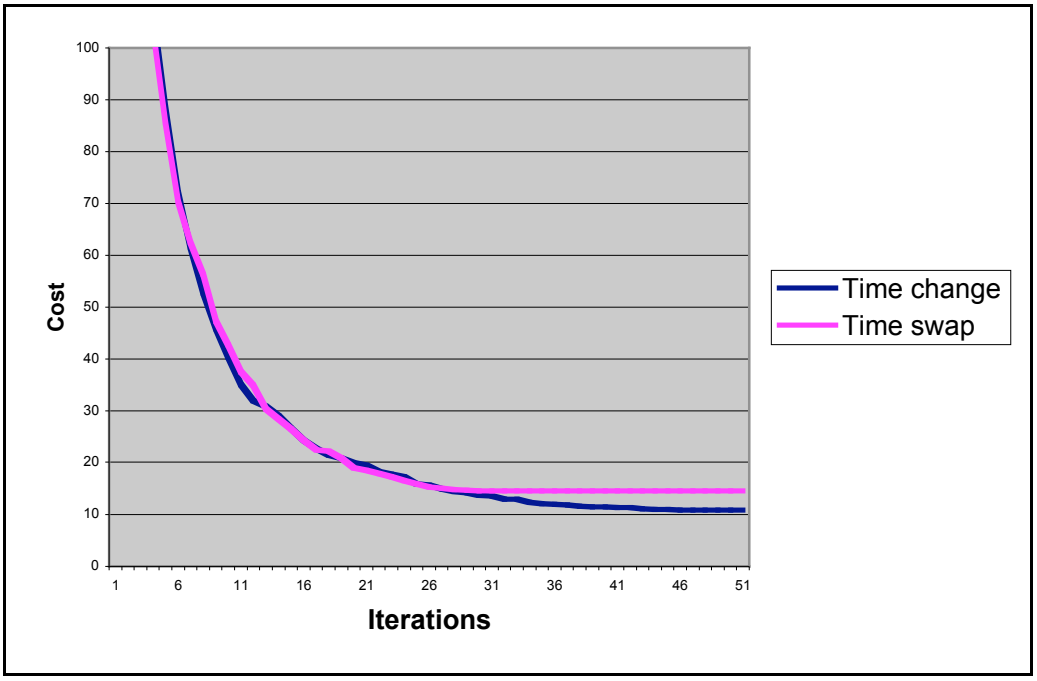


Figure 1: Performance improvement by iterations

Constraints	Violations in Manual	Violations in Tabu Search	Violations in Simulated Annealing
Candidate clashes	0	0	0
Room size violations	0	0	0
Continuous exams	20	0	0
Large exams first	29.97	5.86	4.81
Exam splits	8	5	5
Standby generators	0	0	0
Cafeterias violation	0	0	0
Total violation cost	57.97	10.86	9.81
% Improvement from Cost	77%	96%	96.08%

Table 4: Comparison of Performances

SUMMARY AND CONCLUSION

The aim of the project was to design and implement an algorithm to solve the examinations timetabling problem at UDSM using Tabu Search technique. This has been achieved by solving a real instance of the problem within reasonable time of 50 minutes. The algorithm has been compared with the manually generated case and shows great improvement in satisfying the soft constraints without violating the hard constraints. Comparison with Simulated Annealing shows that the performances of the two approaches are close to each other.

Tabu Search is therefore a viable and good heuristic for the UDSM examinations timetable with a random time change as the best move. Fine tuning of parameters may however bring better results. This algorithm also sets a benchmark for comparison with other heuristic techniques to be developed for this UDSM application.

Further Research

Examinations timetabling problems are very specific to institutions, since each institution has its own structure and policies. This paper presents only the second attempt in development of algorithms which can help in the automation of examinations timetables at UDSM. There is therefore a room for further development of algorithms and compare performances. Other variants of timetabling problems exist, including course and high school timetabling. Very limited work has been done in Tanzanian institutions, development of algorithms for automation of these timetabling problems in almost unexplored area of further research. The presented algorithm may be improved by designing variable Tabu list structures and creating different aspiration criteria. Limited effort is made on exact methods to timetabling due to the fact that the complexity of NP-Hard problems. However, with the current improvement in exact solvers, it is worth developing efficient mathematical programming models and use the results as benchmarks for heuristic algorithms.

REFERENCES

Ab Malik, A, Ayob M and Hamdan A 2009 Iterated Two-stage Multi-neighborhood Tabu Search Approach for Examination Timetabling Problem, 2nd Conference on Data Mining and Optimization 27-28 October 2009 Selangor Malaysia

Ayob M, Abdullah S and Ab Malik A 2007 A Practical Examination Timetabling Problem at the Universiti Kebangsaan Malaysia, *International Journal of Computer Science and Network Security* **7**(9): 198-204

Bartak R 1999 Constraint Programming: In pursuit of the Holy Grail, Proceedings of WDS99 (Invited lecture) Prague

Carter W Laporte G Chinneck J 1994 A General Examination Scheduling System, *Interfaces* **24**: 109-120

Cooper T B Kingston J H 1996 The Complexity of Timetable Construction Problems, Springer Lecture Notes in Computer Science **1153**: 283-295

Corne D Fang H S Mellish C 1993 Solving the Modular Examination Scheduling Problem with Genetic Algorithms, Proceedings of the sixth International Conference of Industrial and Engineering Applications of Artificial Intelligence and Expert Systems Edinburgh

de Werra D 1985 An Introduction to Timetabling, *Eur. J. Operat. Res* **19**: 151-162

Dimopoulou M Milliotis P 2001 Implementation of a University course and examinations timetabling system, *Eur. J. Operat. Res.* **1(**130): 202-213

Elmohamed S Fox G Coddington P 1998 A Comparison of Annealing techniques for Academic Course Scheduling, Lecture Notes in Computer Science **1408** 92-114

Glover F 1990 Tabu Search: A tutorial, *Interfaces* **20(**4): 74-94

Glover F Laguna M 1997 Tabu Search, Kluwer Academic Publishers

Karp K M 1972 Reducibility among Combinatorial Problems, In Complexity of Computer Computations, Plenum Press New York

Kendall G and Hussin N M 2005 Tabu Search Hyper-Heuristic Approach to the Examination Timetabling Problem at University Technology MARA In: Burke E K Trick M (eds), PATAT 2004 LNCS **3616** 199-218 Springer Heidelberg

Marti R Lourenco H Laguna M 2001 Assigning Proctors to Examinations with Scatter Search In: Laguna M González-Velarde J L (Eds), Computing Tools for Modelling Optimization and Simulation: Interfaces in Computer Science and Operations Research Kluwer

Academic Publishers Boston 215-227

Mooney L M 1991 Tabu Search Heuristics for Resource Scheduling with Course Scheduling Applications, PhD Dissertation Purdue University

Mushi A R 2007 Simulated Annealing Algorithm for the Examinations Timetabling Problem, *Afr. J. Sci. Technol. (AJST)* Science and Engineering Series **8**(2): 24 – 32

Reeves C R (E.) 1993 Modern Heuristic Techniques for Combinatorial Problems, Blackwell Scientific Publications Oxford

Reeves C R 1999 Landscapes, operators and heuristics search, *Annals of Operations Research* **86** 473-490

Saleh E Coddington P Mihala B 1998 A Comparison of Annealing Techniques for Academic Course Scheduling, Lecture Notes in Computer Science **1408**, 92-114

Schaef A 1995 A survey of Automated Timetabling, *Artificial Intelligence Review* **13(**2): 87-127

Slim A Marte M 1998 University Timetabling using Constraint Handling Rules, *Journees Phrancophones de Programmation par contraintes* Nates France

Stamapoloulos P Viglas E Karaboyas S 1998 Nearly Optimum Timetable construction through CLP and Intelligent Search, *Int. J. Artific. Intellig. Tools* **7(**4): 415-442

Thompson J Dowslan K 1995 TISSUE wipes away exam time tears, *OR Insight* **8(**4) 28-32

Thompson J Downsland K 1996 Variants of simulated annealing for the examination timetabling problem, *Ann. Operat. Res.* **63** 105-128

Welsh D J Powell M B 1967 An Upper Bound for the Chromatic Number of a Graph and Its Applications to Timetabling Problems, *Comp. J.* **10** 85-86