African Journal of Science and Technology (AJST)
Science and Engineering Series Vol. 8, No. 2, pp. 24 - 32

SIMULATED ANNEALING ALGORITHM FOR THE EXAMINATIONS TIMETABLING PROBLEM

Mushi, A. R.

Department of Mathematics, University of Dar es salaam, Box 35062, Tanzania

ABSTRACT:- *Examination scheduling is a very important process in Educational Institutions. The main challenge is to assign examinations to timeslots and rooms over a fixed period of time while satisfying a set of constraints on both candidates and invigilators. The problem becomes even more difficult when student number rises and student choice widens which makes automated system a necessary tool. The problem varies in detail from Institution to Institution. Thus any generic approach may not be able to capture sufficiently the characteristics of specific problems. This paper reports the design of an algorithm for the Examinations Timetabling Problem at University of Dar es salaam, based on a Simulated Annealing heuristic. We have been able to solve an existing problem and show that the automated system performs better and faster than the manually generated solution.*

Keywords: *Timetabling, Simulated Annealing, Optimization, Scheduling*

INTRODUCTION

Examination scheduling is a very important process in Educational Institutions. The main challenge is to schedule examinations to timeslots and rooms over a specific time period while satisfying a set of constraints. These constraints are normally divided into hard and soft. Hard constraints must be satisfied for the timetable to be feasible, such as candidate examination clashes. Soft constraints must be satisfied as much as possible, such as satisfying special request by candidates or invigilators. Since there are many soft constraints depending on requests (some even contradicts each other) we may not be able to satisfy all of them. However, the more soft constraints are satisfied the better the quality of the timetable. Many researchers have presented their work on variations of the examinations timetabling problem, mostly heuristic approaches, since this problem is N-Hard. Dimopoulou and Milliotis (2001) reported a system which combines both Integer Programming and heuristic procedures for Athens University of Economics and Business. Carter et al (1994) developed a system called EXAMINE which was successfully applied to Toronto and Carleton Universities. Also Carter et al (1996) compares the performance of several algorithmic strategies as applied to EXAMINE. Several researchers have attempted this problem using simulated annealing including Thompson and Dowsland (1998), and Duong and Lam (2004). Tabu search methods have also been used by many researchers such as Kendall and Hussin (2004), White et al (2004), and Gaspero (2002). There are also researches on the use of evolutionary algorithms (Paquete and Fonseca, 2001) and constraint satisfaction methods (Terashima et al, 1999). A more thorough survey of Examinations timetabling problems is provided by Cowling et al (2002). Most of the papers however, are theoretical and only few present a practical implementation of the Examination timetable for specific Universities. Some of these few case studies include Thompson and Dowsland (1995), Lim et al (2000), Kendall and Muhsin (2004), and Carter et al (1994). This research reports the design of an algorithm for the Examinations Timetabling Problem at the University of Dar es salaam.

Examination timetable at UDSM

UDSM has approximately 15,000 students and the number is increasing annually. There are about 2,400 courses, 700 lecturers and 74 rooms which can be used for examinations. Currently, UDSM is using a software package which only helps to detect collisions, but the timetable has to be created manually. It takes several weeks to prepare an examination timetable using the current setting, and still we can not be sure that the resulting timetable is the best

possible. Our aim is to provide an algorithm which will shorten this period by automatically generating a timetable which can then be imported into the timetable software for other changes in case previously unknown soft constraints appears after release of the timetable.

The University has two semesters per each academic year. Each semester is made up of fifteen weeks of teaching, followed by two weeks of examinations. There are three examination sessions per day except on Fridays where there are two sessions. There are only two sessions on Fridays so as to allow for Muslim Prayers and start of Sabbath for Seventh day Adventists. Examinations are mostly three hours long with a few exceptions which deviates for half an hour. There are an increasing number of courses which cut across faculties and University-wide courses which are offered to more than a thousand students at the same time. The problem is also complicated by the freedom of choices by students on optional courses, where students have a wide range of choices which cut across departments and faculties. Examination room space is limited and therefore various measures have been taken to improve the situation, including;

- Cafeterias are also used for examinations, but only during the morning session of the day, since they are used afterwards for serving food.
- Since lecture theatres are slanted, they have traditionally been used for only a quarter of their capacity so that candidates can be dispersed as far as possible from each other. However, we have designed a colouring scheme which allows candidates from four different examinations to use the same room at the same time. The four colours are arranged in such a way that, no neighbouring candidates come from the same examination.
- Some of the library space is also used for examinations.

A room can be assigned more than one examination during the same session depending on the space availability. On the other hand, an examination can be split into more than one room depending on the size of the course and capacity of available space. Distance between examination rooms is not important since there is a gap of at least one hour between examination sessions. Invigilators are specified by departments after the release of the timetable so that invigilator planning is not part of this automatic process. The constraints implemented in this algorithm are as follows;

Hard Constraints

i.) Candidate clashes must be avoided i.e. a candidate can only do one examination at a time
ii.) Room capacities must be respected
iii.) To avoid the risk of power cuts, evening session examinations can only be allocated to rooms which have standby generators.
iv.) Restriction on the use of cafeterias must be respected i.e. they can only be used during the first session of the day.

Soft Constraints

i.) Candidates prefer to have at least one gap between examinations. In general, we would like to spread each candidate examinations as much as possible within the planning horizon.
ii.) Splitting of examinations into rooms must be minimized as much as possible. This is done in order to help departments in planning for invigilators who are also scarce.

The algorithm based on Simulated Annealing

Simulated Annealing is a global heuristic technique which tries to avoid falling into local optima by accepting bad solutions when specific function conditions have been satisfied. The algorithm is a simulation of the physical cooling process called 'annealing', where a physical object cools down by following a particular cooling schedule. A temperature value is defined initially, and cools slowly until an equilibrium point is achieved. Reeves, (1993) give a good description of the algorithm, which is as summarised here;

ETP_SA Algorithm
Initialize parameters (Temperature T, freezing point F);
Get Initial Solution S_o;
While temperature T > freezing point F {
  Get Solution in the neighbourhood of S ($S \in N(S_o)$);
  $\sigma = S - S_o$;
  If ($\sigma < 0$)
    Accept new solution ($S_o = S$);
  Else {Generate a random value *x* between 0 and 1;
    If($x < e^{-\frac{\sigma}{T}}$)
      Accept new solution ($S_o = S$);
    Else
      Reject new solution
    }
  Update temperature (T = *f*(T));
  }
End ETP_SA;

Problem representation

Let
T = total number of timeslots (there are 28 timeslots in the two examination weeks).
n = total number of examinations
E = set of all examinations
r_i = capacity of room i
c_i = Number of candidates for examination i
p_{ij} = Remaining capacity of room i at timeslot j. This structure is updated whenever an examination is allocated to a room, so that the remaining space can be used for other examinations whenever the need arises. Initially, p_{ij} stores the maximum capacity of room i in all timeslots j. Thus a fully utilized room has capacity of zero.
Since candidate clashes are intolerable, we represent them by a conflict matrix M_{nxn} such that m_{ij} = $\begin{cases}1 & \text{if exam } i \text{ clashes with exam } j \\ 0 & \textit{Otherwise}\end{cases}$ for all $m_{ij} \in M$.

In this case examinations i and j clashes if they have at least one candidate in common and represented by $m_{ij} = 1$.

The solution is represented by two variables s and k such that;

s_i = Timeslot assigned to examination i
k_i = A list of rooms allocated to examination i

Objective function

This is represented as a weighted linear combination of functions associated with all constraints in the problem. For faster execution, it has been observed that it is better to include hard constraints as well in the objective function and assign higher weight to their functions. Thus, given a solution x, and a set of h constraints, we minimise the function;

$$f(x) = \sum_{i=1}^{h} \lambda_i f_i(x),$$

where f_i = 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. Weights are assigned according to the order of importance with higher weights given to higher ranked priorities. The priorities have been designed from experience on the user needs. Next section describes the designed functions for each of the constraints.

Constraint functions

i.) A candidate can have at most one examination at the same timeslot;

Two examinations i and j have a candidate clash if they have been allocated to the same timeslot (i.e. $s_i = s_j$) and $m_{ij} = 1$. We need a function to count the number of these clashes whenever they appear in a particular solution and aim at minimizing them.

Let A = Set of all pairs of examinations $(i,j) \in E$ with $i<j$ such that $s_i=s_j$, and define $f_1(s) = \sum_{(i,j)\in A} m_{ij}$, then minimize $\lambda_1 f_1(s)$. $f_1(s)$ gives the total number of candidate clashes associated with the current solution, and therefore $f_1(s)=0$ is a necessary condition for a feasible solution. Since this is a hard constraint, λ_1 must be a sufficiently large value.

ii.) Room capacities must be respected;
A room cannot be allocated to more candidates than its capacity in any time slot. In this case we need a function to count the number of times that the constraint is violated. That is, calculate the number of times that a room has been assigned more candidates than its capacity. Let B_{it} = a set of examinations assigned to room i at timeslot t, then the remaining capacity of room i in time t is given by $r_i - \sum_{j\in B_{ti}} c_j$. For feasibility, the capacity of room i must be ≥ 0. Also let

$$d_{it} = \begin{cases}1 & \textit{if Capacity of room } i < 0 \textit{ at time } t \\ 0 & \textit{Otherwise}\end{cases}, \text{ for some}$$

room i. Then the function is $f_2(s) = \sum_{i\in E}\sum_{t} d_{it}$, and minimize $\lambda_2 f_2(s)$, where λ_2 is a large value. Since $f_2(s)$ is the total number of rooms with an overflow of candidates, then the condition $f_2(s) = 0$ must be satisfied for a feasible solution.

iii.) Minimal number of examination splits into separate rooms;
This is achieved by simply minimizing the maximum size of k_i. Thus the function is $f_3(s) = \max_{i\in E}\{|k_i|\}$ and minimize $\lambda_3 f_3(s)$, where λ_3 is a weight value.
iv.) Minimal continuous examinations per candidate;

We consider examinations which have been scheduled in consecutive timeslots. To minimize continuous examinations, it is sufficient to make sure that consecutive examinations with common candidates are minimal. Two examinations *i* and *j* are scheduled in consecutive timeslots if $|s_i - s_j| = 1$, and have candidates in common if $m_{ij}=1$.
Let F = set of all pairs of examinations $(i,j) \in E$ such that $|s_i - s_j|=1$.
Then, the task designed in this case is to minimize the number of consecutive examinations with common students, which is represented by the function $f_4(s) = \sum_{\substack{(i,j)\in F \\ i<j}} m_{ij}$, and minimize $\lambda_4 f_4(s)$.

v.) Scheduling large examinations as early as possible;

The relation t^2/z is proposed between timeslot t and examination size z. The output of this relation gives lower values for large examinations when assigned to lower numbered timeslots. Experimental results therefore shows that the function $f_5(s) = \sum_{i \in E} \frac{s_i^2}{c_i}$ gives a good measure of priority, and we minimize $\lambda_5 f_5(s)$.

vi.) Only Rooms with Standby generators can be used for evening examinations

Let
H = set of all rooms fitted with standby generators and
L = set of all evening timeslots of the planning horizon i.e. $L = \{3, 6, 9, 12, 17, 20, 23, 26\}$.

Simply assign zeros to all rooms $i \in H$ at all timeslots $j \in L$ in room capacity structure i.e. $p_{ij} = 0 \; \forall i \in H, \forall j \in L$.

vii.) Restricting the use of Cafeterias to only the first session of the day;

Given,
V = set of all cafeterias
W = set of all afternoon and evening timeslots
Simply assign zeros to positions $i \in V$, $j \in W$ in the room capacities p_{ij} i.e. $p_{ij} = 0 \; \forall i \in V, \forall j \in W$

Generating an initial timetable

Since we include all constraints into the objective function, it is not necessary to have a feasible initial solution. However, a feasible solution simplifies a lot of initial searching and may speed up the process significantly. We therefore aim at creating an initial feasible solution, but using a simple procedure which does not spend too much running time. Each examination is assigned the earliest possible feasible timeslot and earliest feasible room. To allow early assignment of large examinations, both examinations and rooms are sorted in descending order of sizes. Note that, solution is feasible if it satisfies all the hard constraints. During scheduling of rooms to an examination, it might be necessary to search on rooms which have already been assigned examinations as they may still have free space that can be used for other examinations. Therefore, the algorithm is in two phases, where phase I tries to find suitable timeslots without splitting an examination into different rooms. The unsuccessful cases are then put into phase II which splits the examinations into as few rooms as possible. This is done by assigning the examination to the most unused seats in the non-clashing timeslots. If a course does not find a feasible timeslot, it is assigned to the less infeasible timeslot i.e either time-feasible or room-feasible. The general algorithm is demonstrated by the following pseudocode;

```
Initial_Examination_Timetable
Phase 1
For each examination c
    slotted = Assign timeslot and room to examination c
    if Not slotted
        Put in the list U of unslotted examinations
Next c
Phase 2
For each unslotted examination u∈ U
    Assign portions of u into the most empty space until
    all is scheduled. If infeasible assign to the closest
    feasible timeslot.
Next u
End_Initial_Timetable
```

Cooling schedules

It is important that the cooling function allows sufficient time to explore many possible solutions in one level before moving into lower temperature and ultimately freezing point. It is also important not to spend too much time on

high temperatures, where most neighbourhood moves are accepted as this can lead to a wastage of running time. A good schedule is expected to spend more time on lower temperatures so as to allow for convergence. However, it is not advisable to spend too much time on low temperatures where most neighbourhood moves are rejected (Eglese, 1990). The most commonly used schedule involves geometric reduction function;

$$\alpha(t) = at, \ \ where \ \ a < 1$$

For comparison purposes we also test another cooling schedule as suggested by Lundy and Mees (1986) which involves a much slower cooling rate; $\alpha(t) = \frac{t}{(1+\beta t)}$

where $\beta < 1$.

Move structure

Choice of the structure for a move is also an important decision as it affects the cooling process and ultimately quality of the solution. Many choices are possible, but the following two options were found to perform better;

i.) Changing the time of an examination by a random time as follows;
   a. Select a random examination $e \in$ E
   b. Select randomly a new timeslot $t \in \{1,2\ldots28\}$
   c. Assign the new timeslot t to the examination e.

   Such move selections have been previously suggested by Thomson et al (1996) ElMohamed (1998) and Reeves (1999) and have worked well for similar problems. It is a simple move, yet it allows a well balanced mix of events among all timeslots. The size of the neighbourhood associated with this kind of move is $|N(s)| = n$ x (T-1).

ii.) Change of a timeslot followed by a swap of two randomly selected timeslots;
   a. Select two examinations e_1, e_2 at random in E
   b. Select a timeslot t at random in $\{1,2,\ldots28\}$
   c. Replace a timeslot of examination e_1 by t
   d. Swap timeslots of e_1 and e_2

   This move affects two examinations at a time and is therefore a longer move than the previous case. The reason for this kind of move is basically to create a longer pattern of changes in a move in anticipation of better exploration of the solution space. The size of this neighbourhood is $|N(s)| = T$ x n x (n-1), which is larger than the previous case.

Experimental Results

The algorithm was written in C++ and tested on a 2.4 GHz Pentium machine. The test problem is a previous manual solution for semester 1 of the 2003/2004 academic year. The problem data are as specified in Table 1.

Table 1: Data for Semester 1 2003/04 at UDSM

Data	Value
Number of candidates	8161
Number of Invigilators	607
Number of Rooms	74
Number of Examinations	729
Total number of timeslots	28

The weights used in the objective function for each constraint type are as shown in Table 2. These weights have been assigned according to experience obtained from the users of the current system. For instance, it is more important to candidates that examinations should be spaced apart than minimizing number of examination splits. It is also more important to schedule large examinations early in the timetable than minimizing examination splits. A value of 10 was found to be sufficiently large to prevent infeasibilities in the solution space. It is worth noting that, these values can be changed as the users demand changes, but the used vales in Table 2 are sufficient to demonstrate the performance of the algorithm.

Table 2: Weights used in the objective function

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 3 presents a summary of results and performance of the algorithm. Since moves were selected randomly, the values presented are an average of values obtained after several runs on different random number seeds. The program was tested using the two cooling schedules and move types as shown in Table 3. Type 2 moves with Geometric cooling perform slightly better than other cases. The best solution was found after 2,232.05 seconds which

Table 3: Performance of the Algorithm

	Time change (Type1)		Time change/swap (Type2)	
	Geometric	**Lundy&Mees**	**Geometric**	**Lundy&Mees**
Initial cost	249.98	249.98	249.98	249.98
Final Cost	11.30	10.86	9.81	10.22
Student/Lecturer Collisions	0.00	0.00	0.00	0.00
Room size	0.00	0.00	0.00	0.00
Continous exams	0.00	0.00	0.00	0.00
Cafeterias violation	0.00	0.00	0.00	0.00
Large exams first	6.30	5.86	4.81	5.22
Exam splits	5.00	5.00	5.00	5.00
Standby generators violation	0.00	0.00	0.00	0.00
Iterations	91000.00	99000.00	91000.00	99000.00
Time	2643.33	3221.54	2232.05	2807.59
% Cost improvement	95.48%	95.66%	96.08%	95.91%

is approximately 37 minutes which is tolerable in this application compared to manual system which takes more than three weeks. The value of cost improvement shown is a percentage of improvement in relation to the initial solution. In both cases, all hard constraints were satisfied and most of the soft constraints were completely satisfied. In general, moves of type 2 perform better than those of type 1 since the best final cost is lower.

This may be caused by the fact that, type 2 moves performs a wider exploration on the neighbourhood at each temperature. In this experiment, the best value of α in Geometric function is 0.995 and the value of β in the Lundy & Mees is 0.01. In both cases, the best freezing point employed is 0.001 with the best initial temperature value of 100.

Figure 1 shows the cooling process with respect to iterations for all four tested cases. Iteration values shown are multiples of a thousand.

It is interesting to note that, both cases of type 2 moves started by accepting many bad moves, resulting into climbing high hills, but later drops sharply into better cost values. Unlike type 2, both type 1 moves show a relatively slow convergence into solution without long jumps. This means that moves of type 2 performs better because they allow higher exploration of solution space on high temperatures resulting into a better chance of falling into global solution.

Figure 2 is a closer look on the performance of the four cases in terms of cost improvement. Clearly, moves of type 2 with Geometric cooling perfumes better as shown in the figure, despite of the fact that they give high cost values on high temperatures.

Table 4 shows the quality of the automatic system compared to the manual one. In addition to the fact that automatic system is faster, it shows better performance in satisfying soft constraints. Although both timetables are feasible by satisfying the hard constraints, the automatic system clearly outperforms manual system in achieving

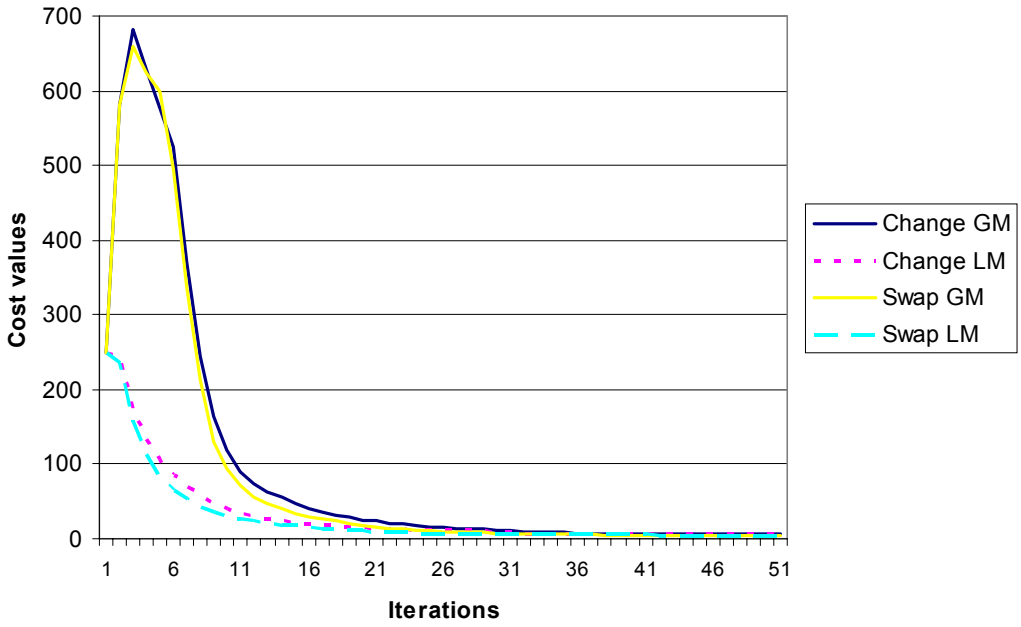


Figure 1: Cooling process by iterations

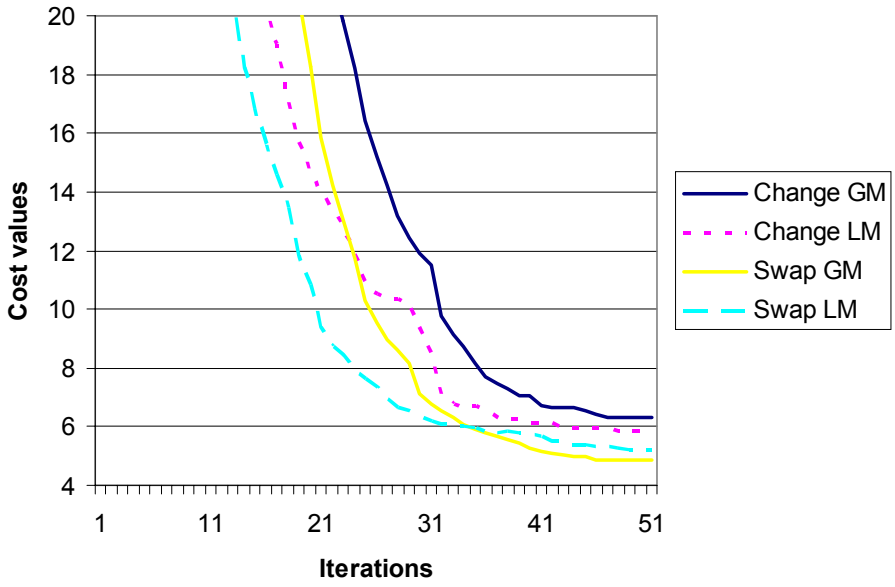


Figure 2: Performance by Iterations

Table 4: Manual vs Automatic Performances

Constraints	Violations in Manual	Violations in automatic
Candidate clashes	0.00	0.00
Room capacity violations	0.00	0.00
Continuous exams	20.00	0.00
Early large exams	29.97	4.81
Exam splits	8.00	5.00
Standby generators	0.00	0.00
Cafeterias violation	0.00	0.00
Total violation cost	57.97	9.81
% Improvement from Cost	76.81%	96.08%

soft constraints as shown in the table for the cases of 'Continuous exams', 'Early large exams', and 'Exam splits'.

It should be noted that this performance is only with respect to all constraints which have been included in the algorithm. We acknowledge that, other constraints may involve some manual modifications into the current solution, but it is not expected to bring significant changes to the current best solution.

CONCLUSION AND SUGGESTED FUTURE RESEARCH

We have designed and implement an algorithm to tackle the examinations timetabling problem at UDSM using Simulated Annealing. A practical problem was used as a test case and solved with very high performance in terms both constraint satisfaction and time. Geometric cooling schedule with moves of type 2 have shown better performance than other tested cases. The algorithm performs significantly better than manual system. In this light, we can conclude that Simulated Annealing is a viable option for examinations timetabling at UDSM and performs better when using geometric cooling schedule and type 2 moves.

Due to the nature of the problem at UDSM, many of the soft constraints may not be known during the time of timetable preparation. Furthermore, sheer number of soft constraints, together with contradictory nature of some of them may not make it possible to include them all into the algorithm. We have included only those we have considered to be more important at this stage. However, it will be interesting to add as many soft constraints as possible and thereby improve further the quality of the automated procedure.

REFERENCES

1. Carter M., Laporte G., Chinneck J. (1994), *Interfaces*, Vol. 24, 3, May-June, pp. 109-120.

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

3. Cowling P., Kendall G., Hussin N. (2002). A survey and case study of Practical Examination Timetabling Problems, *Proceedings of the 4th International Conference on Practice and Theory of Automated Timetabling* (PATAT 2002), Gent, pp. 258-261.

4. Dimopoulou M., Milliotis P. (2001). Implementation of a University Course and Examination Timetabling System, *European Journal of Operational Research*, Vol. 130, pp. 202-213.

5. Duong T., Lam K. (2004). Combining Constraint Programming and Simulated Annealing o University Exam Timetabling, *2nd International Conference, Associating Enquiring Vietnamese and French speaking people in Data Processing* (RIV'04), February 2-5, Hanoi, Vietnam.

6. Eglese R., W. (1990). Simulated Annealing: A tool for Operational Research, *European Journal of Operational Research*, Vol. 46, pp. 271-281.

7. ElMohamed s., Coddington P. (1998). A Comparison of Annealing Techniques for Academic Course Scheduling, *Lecture Notes in Computer Science*, Vol. 1408, pp. 92-114.

8. Gaspero L. (2002). Recolor, Shake and Kik: a recipe for the Examination Timetabling Problem, In Burke E., Causmaecker P. (Eds.): *Proceedings of the 4th International Conference on the Practice and Theory of Automated Timetabling*, August, pp. 404-407.

9. Kendall G., Hussin N. (2004). Tabu Search Hyper-heuristic Approach to the Examination Timetabling Problem at University Technology MARA. *Proceedings of the 5th International Conference on Practice and Theory of Automated Timetabling* (PATAT'04), Pittsburg, PA, USA, August.

10. Lim A., Ang J., Ho W., Oon W. (2000). UTTSExam: A Campus-Wide University Exam-Timetabling System, *Innovative Approaches in Artificial Intelligence*, pp. 1020-1025.

11. Paquete L., Fonseca C. (2001). A Study of Examination Timetabling with Multiobjective Evolutionary Algorithms, MIC'2001 – *4th Metaheuristics International Conference*, Porto, Portugal, July 16-20, pp. 149-153.

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

13. Reeves C., (1999). Landscapes, Operators and Heuristics Search, *Annals of Operations Research*, Vol. 86, pp. 473-490.

14. Terashima H., Ross P., Valenzuela M. (1999). Evolution of Constraint Satisfaction Strategies in Examination Timetabling, In W. Banzhaf et al (Eds.) *Proceedings of the GECCOE-99, Genetic and Evolutionary Computation Conference*, pp. 635-642, Morgan Kaufmann.

15. Thomspson J., Dowsland K. (1995). TISSUE wipes away exam time tears, *OR Insight*, Vol. 8, Issue 4, October-December, pp. 28-32.

16. Thompson J., Dowsland K. (1996). Variants of Simulated Annealing for the Examination Timetabling Problem, *Annals of Operations Research*, Vol. 63, pp. 105-128.

17. Thomson J., Dowsland K. (1998). A Robust Simulated Annealing Based Examination Timetabling System, *Computers & Operations Research*, Vol. 25, No. 7/8, pp. 637-648.

18. White G.,Xie B., Zonjic S. (4002). Using Tabu Search with longer-term memory and relaxation to create examination timetables. *European Journal of Operational Research*, 153 pp. 80-91.