Volume 4, No. 2, Jan-Feb 2013
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
CASE STUDY AND REPORT
Available Online at www.ijarcs.info

Late Acceptance Heuristic for University Course Timetabling Problem

Yohana Marwa
Department of Mathematics, Mwenge University College of Education, Box 1226, Moshi, Tanzania
yohanamarwa@yahoo.com

Allen Rangia Mushi*
Department of Mathematics, University of Dar es salaam, Box 35062, DSM, Tanzania
allen.mushi@gmail.com

Abstract: This paper describes a Late Acceptance Heuristic for University Course Timetabling Problem, using a case study of a University College in Tanzania. Late Acceptance is one of relatively new heuristic procedures that try to improve searching by delaying acceptance of latest solutions. The results are compared with an implementation on Simulated Annealing heuristic, which is a well documented and successful heuristic procedure for similar problems. It is shown that Late Acceptance Procedure is a good procedure for Course timetabling problem and compares well with Simulated Annealing.

Keywords— Late Acceptance, Simulated Annealing, Course Timetabling, Combinatorial Optimization, heuristics

I. INTRODUCTION

Timetabling Problem is a Combinatorial Optimisation Problem which has captured the interest of many researchers in Operations Research and Artificial Intelligence domains. University Course Timetabling Problem (UCTP) is the problem of allocating resources such as courses, teachers and teaching space over time (usually a week) while satisfying a number of constraints over those resources. The constraints are normally divided into hard and soft. While hard constraints must be satisfied, soft constraints are to be satisfied as much as possible but can be tolerated when necessary. Over the years, constructing a timetable at a university has become more and more complex, mainly due to the growing number of students and courses under limited resources. No optimal algorithm is known for their solutions within reasonable time [1]. However, several solution techniques have been employed in trying to find good solutions. These techniques include graph colouring, Integer Programming, Global heuristics such as Tabu search, Simulated Annealing, Genetic Algorithms [2] and Fuzzy Logic [3].

A good number of papers have been published on UCTP for specific Universities. Finding good quality solutions to these problems depends on the technique itself and the structure employed during the search. Artificial Intelligence has been used with some success. One of the recent works include Asaju et al [4] who applied Artificial Bee Colony on Course timetabling and Socha et al [5] who applied a variant of Ant-Colony called min-max Ant system. Hybrid methods are also common in the literature such as the work by Chiarandini et al [6]. The most popular and traditional heuristic techniques includes Simulated Annealing and Tabu Search. Mushi applied Simulated Annealing [7] and Tabu Search [8] for a case study of the University of Dar es salaam with recorded success. However, it was found that the quality of solution depends on the selection of the set of parameters to be used.

One of the challenges in heuristic algorithms therefore is the dependency on selection of parameters for their performances. The more the number of parameters required the more unstable is the technique. Consequently, there is demand for designs of good algorithms which requires less number of parameters and therefore more stable. Burke and Bykov [9] introduced a time-predefined approach for course timetabling. This method requires the definition of only two parameters which are; search time and an estimation of desired solution quality. A work by Orbit et al [10], introduces a Great Deluge algorithm which is another stable procedure. This method tries to avoid the fall into local solution and increase the chance of reaching a global optimal solution by using a 'water level' parameter which is slowly increased from lower levels. Recently, a new approach on timetabling problems based on basic local search had been proposed by Burke and Bykov [11]. This works by delaying acceptance of solution during local search and therefore named "Late Acceptance".

The technique requires only one parameter, making it more stable. A few papers have been reported on the performance of this approach in different case studies despite of its stability structure. Abuhamdah and Ayob [12] extended Late Acceptance, by introducing the so called, Average Late Acceptance Randomized Descent (ALARD) to solve university course timetabling problem. In their work, they only dealt with students' satisfaction and did not consider the lecturers allocation. This paper reports on a case study of the Late Acceptance heuristic as implemented to Mwenge University College of Education (MWUCE) in Tanzania.

The paper starts by describing the course timetabling problem at MWUCE, followed by mathematical formulations. Late Acceptance Heuristic is then described with its adaptation to MWUCE problem. A summary of results is presented before giving a conclusion and suggestions for future research.

II. COURSE TIMETABLING AT MWUCE

MWUCE is a young growing institution, which continues to grow and expand in her structures and programmes of study. The expansion of student enrolment currently stands at approximately 1,700 students. There are two semesters per each academic year with approximately 121 courses per semester, 22 rooms which includes classrooms and laboratories, 91 lecturers, and 1,700 students to be scheduled on a five days week. Each day is made up of

© 2010, IJARCS All Rights Reserved

10 one- hour timeslots starting from 8.00 a.m. to 6.00 p.m., giving a total of 50 timeslots for the whole timetabling period. The necessary data includes students-course registrations, lecturer-course assignments, course sizes, room sizes and room types. Timetabling constraints are normally divided into two, namely hard and soft. Hard constraints must be satisfied for a feasible timetable, while soft constraints are to be satisfied as much as possible although their violation is tolerable. The following is the situation at MWUCE in terms of hard and soft constraints;

A. Hard Constraints:

a. No student can attend more than one lecture at the same time,
b. No lecturer can teach more than one lecture at the same time,
c. No room can occupy more than one lecture at the same time,
d. No room can be assigned a lecture with more students than its capacity.

B. Soft Constraints:

a. Minimize the use of Friday midday and evening timeslots to allow for Muslim prayers and Adventists Sabbath Day (SDA) respectively.
b. Minimize the use of evening timeslots throughout the week. These times are not very popular because of shortage of accommodation in the college for both students and lecturers, where most of them live far away from the college.
c. Spread lectures of the same course over the week. It is not desirable to have all lectures on the same day or close days.

C. Model Formulation:

Consider the following definitions;
A= {1,…,K} is a set of courses
B= {1,…,T} is a set of timeslots
C= {1,…,R} is a set of rooms
x_k = Timeslot of course k
y_k = Room of course k
C_r = Capacity of room r
a_k = Number of students in course k
And collision matrix;

$$m_{k_1k_2} = \begin{cases} 1 & \text{if course } k_1 \text{ collides with course } k_2 \\ 0 & \text{otherwise} \end{cases},$$

Hard constraints are then formulated as follows;

a. Course collision:

We say that courses k_1 and k_2 collides if they have at least one student or lecturer in common. Note that when the two courses collide then collision matrix associated with the two courses will have a value of 1 (i.e. $m_{k_1k_2}$ =1). To constrain this case, we must make sure that when the two courses collide then they are not slotted in the same timeslot i.e.

$$m_{k_1k_2}x_{k_1} + m_{k_1k_2}x_{k_2} \le 1, \quad \forall k_1 \ne k_2$$

b. Room collision:

No room can have more than one course at the same timeslot i.e. given any two courses k_1 and k_2, then

$$(x_{k_1} = x_{k_2}) \Rightarrow y_{k_1} \ne y_{k_2}$$

And

$$(y_{k_1} = y_{k_2}) \Rightarrow x_{k_1} \ne x_{k_2} \ \forall k_1, k_2 \ni k_1 \ne k_2$$

Let $\alpha_1 = \begin{cases} 1 & \text{if } y_{k_1} \ne y_{k_2} \\ 0 & \text{otherwise} \end{cases}$

$\alpha_2 = \begin{cases} 1 & \text{if } x_{k_1} = x_{k_2} \\ 0 & \text{otherwise} \end{cases}$

Then we have $\alpha_1 - \alpha_2 \le 0$

c. Room capacity:

No room can have more students than its capacity i.e. $a_k \le C_{y_k}$ for all rooms y_k

Soft constraints in this application make up the objective function. This is found by minimizing the cost associated with penalties on violations of soft constraints. The formulation of each soft constraint is explained as follows;

d. Minimize the use of religion and evening periods;

Let $u_{ki} = \begin{cases} 1 & \text{if } x_k = i \\ 0 & \text{Otherwise} \end{cases}$

then

$f_1(x) = \lambda_1 \sum_k \sum_{i \in G} u_{ki}$ and

$f_2(x) = \lambda_2 \sum_k \sum_{i \in H} u_{ki}$

Where λ_1 = weight given to evening timeslots,
λ_2 = weight given to religion timeslots,
G= timeslots of the evening periods of the week,
H= timeslots of the religion periods of the Fridays

e. Spread consecutive course lectures over the week:

If $(x_{k_1} = a)$ and $(x_{k_2} = b)$ then

$|b-a| = |x_{k_2} - x_{k_1}|$ should be as large as possible.

And $t = \frac{1}{(x_{k_2} - x_{k_1})^2}$ should be as small as possible

Hence, $t = \frac{1}{(x_{k_2} - x_{k_1})^2}$ is to be minimized

The resulting objective function is therefore;

$$Minimize \ z = \lambda_1 \sum_k \sum_{i \in G} u_{ki} + \lambda_2 \sum_k \sum_{i \in H} u_{ki} + \lambda_6 \sum_{\substack{(x_{k_1}, x_{k_2}) \in T \\ x_{k_1} \ne x_{k_2}}} \frac{1}{(x_{k_2} - x_{k_1})^2}$$

Where λ_6= weight given to the constraint on distance between events.

Since we are using heuristics in the implementation, we have decided to have all constraints in the objective function and penalize higher on the hard constraints. The objective function is therefore presented by combining all equations as follows;

$$Minimize \quad f(x,y,u) = \lambda_1 \sum_k \sum_{i \in G} u_{ki} + \lambda_2 \sum_k \sum_{i \in H} u_{ki} + \lambda_3 \sum_{(k_1,k_2) \in K} m_{k_1k_2}(x_{k_1} + x_{k_2} - 1)$$

© 2010, IJARCS All Rights Reserved

$$+\lambda_4 \sum_k \sum_{i=(1,2)} y_{k_i} + \lambda_5 \sum_k \left(c_{y_k} - a_k\right) + \lambda_6 \sum_k \sum_{\substack{(x_{k_1}, x_{k_2}) \in T \\ x_{k_1} \neq x_{k_2}}} \frac{1}{\left(x_{k_2} - x_{k_1}\right)^2}$$

Where $f_1(x,y,u)$ is the use of special times for evening timeslots, $f_2(x,y,u)$ is the use of special times for religion timeslots, $f_3(x,y,u)$ represents students lecturer collisions, $f_4(x,y,u)$ is for room clashes, $f_5(x,y,u)$ is for room size violations and $f_6(x,y,u)$ is the distance between events of the same course.

III. LATE ACCEPTANCE HEURISTIC

Late Acceptance heuristic tries to avoid falling into local optima by instituting various strategies which involves careful acceptance of bad solution (moves) in anticipation for better results in future. It accepts a candidate solution if it is not worse than that solution which was "current'' several steps before. The work by [11]; gives a comprehensive description of the technique. The algorithm only requires one parameter L and is as presented in Fig.1.

Given an initial solution S_0 and $f(S_0))$

Set $S_{best} = S_0$, $f(S_{best}) = f(S_0)$

Set all initial values in L= $f(S_0))$

I=0;

for n=0 to N

Generate some feasible neighbour solutions of S_0 and choose the best neighbour (S^*)

If $f(S^*))< f(S_{best}))$

$S_{best} = S^*$, $f(S_{best}))= f(S^*))$;

$S_0 = S^*$;

Else

V = I mod L

If $f(S^*) \leq f(S^v))$

$S_0 = S^*$; $f(S^v))= f(S^*)$; I = I+1;

End if

End if

End for

Return the best obtained feasible solution

Figure. 1 Late Acceptance Algorithm

A. *Adapting Late Acceptance to MWUCE:*

To apply Late acceptance to MWUCE Timetabling, one needs to make a number of decisions with respect to the general algorithm; these are the kind of initial solution, neighbourhood structure, length of the list L, initial values, cooling function and stopping criteria.

a. *Initial Solution:*

It is preferable to start with an initial solution which satisfies all the hard constraints to give an initial feasible solution. This can be used to ensure that the selected solution neighbourhood is always feasible. However, finding an initial feasible solution may be expensive in terms of time. It may not be a very good idea to spend most of the reasonable time in finding initial feasible solution; the whole process may be too time-consuming. The chosen strategy is to start with the simple initial solution which is infeasible and penalise all infeasibilities in the objective function. The following simple algorithm was therefore applied to obtain an initial solution;

Sort the courses and rooms in descending order of capacity

For all courses starting from the beginning;

Assign a course to the latest available timeslot and room

If timeslots or rooms have been exhausted;

go to the beginning of the list

End if

End for

Figure. 2: Initial Solution Algorithm

Though simple but it is helpful since it reduces the possibility of course collisions; neighbouring courses are more likely to be assigned into different timeslots.

b. *Neighbourhood Structure:*

In this work, we have used different neighbourhood structures as follows:

NS1: Randomly select two courses and swap their timeslots if feasible (and swap their rooms if necessary).

NS2: Pick a course randomly; find a neighbourhood to the left and right then change timeslots randomly for courses in the neighbourhood and check for both course collision and room capacity collision.

NS3: Pick a course randomly and replace its timeslot with a randomly selected timeslot.

NS4: Randomly select four courses and swap their timeslots if feasible (and swap their rooms if necessary).

c. *Length of the list L:*

We have applied different values of L in anticipation for better results. From experiments with the algorithm, the best values of L were found to be between 6 and 12. Note that this is the only parameter required in the Late Acceptance heuristic.

d. *Stopping criteria :*

The stopping criteria used in this study is a pre-determined value N. The best values for this are obtained experimentally and found to be in the range from 100,000 to 100,000,000. It is worth noting that this criterion does not have to be pre-determined as can be related to the quality of solution.

During experimentation, the algorithm output was compared to that of Simulated Annealing. We present here the Simulated Annealing algorithm and show how it was implemented in the study.

e. *Simulated Annealing Algorithm:*

The algorithm is a simulation of the physical cooling process called "annealing", where the physical objects cools slowly down by following a particular cooling schedule. A temperature value is defined initially, and cools slowly until an equilibrium point is achieved. The work by Collin [13], gives a comprehensive description of the technique. The algorithm is as presented below;

© 2010, IJARCS All Rights Reserved

Simulated_Annealing
Select an initial solution so;
Select an initial temperature $t_0 > 0$;
Select a temperature reduction function f;
While (Not Freezing_Condition)
Randomly select $S \in N(S_0)$;
$\sigma = f(S) - f(S_0)$;
If($\sigma < 0$) $S_0 = S$;
else
generate random x uniformly in range (0, 1)
if(x < exp($-\sigma/t$) $S_0 = S$; end if
end if
Update temperature $t = f(t)$;
Check_Freezing_Condition (t);
End while
S_0 is the approximate solution
End Simulated Annealing

Figure. 3: Simulated Annealing Algorithm

Same initial solution strategy and neighbourhood structures were used in Simulated Annealing. The most common temperature reduction function is geometric function; $f(t) = \alpha t$ where $0.8 \le \alpha \le 0.99$.

IV. SUMMARY OF RESULTS

In this chapter, we present and discuss results of the algorithms using data from MWUCE whose properties are summarised in Table I.

Table 1: Summary of Data

Number of students	Number of courses	Number of rooms	Number of lecturers
1700	121	22	91

The algorithm was implemented on C++ compiler and test run on Intel(R) core(TM) i3 CPU M370@2.40GHz machine. The results are as presented in Table II with varying values of L, iteration J and using different neighbourhood structures. 'Cc' is the total number of course collisions in the solution, while 'Rc' is the total number of room collisions in the solution and, 'Cp' is the total number of Capacity collisions.

Table 2: Results obtained using Late Acceptance at iteration j=10,000,000

Neighborhood	Length L	Cc	Rc	Cp	Final solution	Time(s)
NS1	7	0	0	9	7914	4185.75
	8	0	0	9	7914	4185.34
	10	0	0	9	7914	4186.34
	12	0	0	9	7914	4189.07
NS4	7	4	0	4	7834	4199.57
	12	4	0	4	7834	4178.11
NS2	7	19	0	1	8797	4214.04
	8	19	0	1	8797	4135.72
	9	23	0	1	9090	4198.36
	10	11	12	1	9068	4203.43
	11	13	2	1	8459	4188.8
NS3	7	8	2	1	8034	4198.84
	8	8	2	1	8034	4198.66
	10	8	2	1	8034	4180.32

After experimentation with varying values of neighbourhood, L and J, the best results were obtained when J=10,000,000. The best value is obtained at neighbourhood NS4 with L=12. However, this corresponds to 4 course collisions and 4 capacity collisions. Due to the importance of having course collision-free timetable, it is preferable to have a solution which has no course or room collisions. The most preferable timetable is therefore corresponding to NS1 with L=8, where Cc=0, Rc=0 and Cp=9.

A comparison was done with Simulated Annealing solution where the best results are presented in Table III with NBR4.

Table 3: Results obtained after using Geometric Cooling Function

Temperature	α	Cc	Rc	Cp	Final solution	Time(sec)
100000	0.995	19	0	3	8954	1.451
100000	0.998	21	0	4	9194	3.682
100000	0.999	18	0	5	9034	7.878
1000000	0.995	19	0	3	8934	1.591
1000000	0.998	18	0	3	8874	4.025
1000000	0.999	12	0	6	8634	8.58
10000000	0.995	17	0	3	8794	1.81
10000000	0.998	18	0	5	9034	4.461
10000000	0.999	16	0	3	8714	9.235
100000000	0.9999	7	0	9	8474	117.85

The parameters freezing point 0.00005, alpha (α) =0.9999 and temperature of 100,000,000 produced the solution of 8,474 at running time of 117.85. This set gives the best solution for Simulated Annealing with Cc=7, Rc-0 and Cp=9. The quality of solution in this case is lower than that of Late Acceptance.

V. CONCLUSION

The study experimented on Late Acceptance heuristic which is one of recent strategies and compared results with those of Simulated Annealing which is a well known efficient technique. This was done purposely because this is the first study on the MWUCE data set and therefore no previous results are available for comparison.

It has been concluded that Late Acceptance is a good strategy for MWUCE course timetabling. Although no feasible solution was found, the best solution greatly reduced infeasibilities in the original problem. In the best solution, both values of L gave a solution with zero values of course collisions and room collisions with a neighbourhood of NS1. In this case, a feasible solution can be easily obtained by fixing other resources and just expand slightly the room capacities. Although neighbourhood NS4 provided the lowest value of final solution, it failed to achieve collision-free courses which are more complex constraint. With these results it has also been observed that the neighbourhood NS1 outperforms all other neighbourhood structures. Furthermore, Late Acceptance demonstrated better results than Simulated Annealing technique for the tested data set.

© 2010, IJARCS All Rights Reserved

The results indicate that the current timetabling system at MWUCE operates under constraints on room capacities. Some candidates may be studying under unfavourable conditions.

VI. FUTURE RESEARCH DIRECTIONS

There is a need to search for better stable algorithms which do not greatly depend on the choice of parameters. Other soft constraints were not considered in this paper such as such lecturer's preferences and academic institution's regulations. Exploration of more neighbourhood structures might improve performance. Furthermore, it would be interesting to simulate room sizes and test the algorithm so as to advise management on the extent of room extension required that would lead to good feasible solution.

VII. REFERENCES

[1] Cooper T., Kingston J. "The Complexity of Timetable Construction Problems", 1996, Springer Lecture Notes in Computer Science 1153, pages 283-295

[2] A. Schaerf, "A survey of Automated Timetabling", *Technical Report CS - R9567*, CW1, Amsterdam, NL, 1995.

[3] A. Chaudhuri1 and Kajal De, "Fuzzy Genetic Heuristic for University Course Timetable Problem", Int. J. Advance. Soft Comput. Appl., Vol. 2, No. 1, March 2010, pages 100-123.

[4] L. Asaju, T. Ahamad, and A. Mohammed, "Tackling University Course Timetabling Problem using Artificial Bee Colony", *Frontiers in information technology* by Al-Dahoud Ali, Masaum Network 2012 chapter twelve.

[5] K. Socha, J.D. Knowles, and M. Samples, "A max - min ant system for the university course timetabling problems", In: Ant Algorithms: Proceedings of the Third International Workshop (ANTS 2002), LNCS 2463, pp. 1 - 13, Springer

[6] M. Chiarandini, M. Birattari, and K. Socha "An effective hybrid algorithm for university course timetabling", Journal of Scheduling, Vol. 9, Issue 5, pp. 403- 432, 2006.

[7] A. R. Mushi, "Two-phase Heuristic algorithm for University Course Timetabling problem: The case of University of Dar es salaam", Tanzania Journal of Science. Vol.37, pp 73-83, 2011.

[8] A. R. Mushi, "Tabu Search Heuristics for University Course Timetabling problem", African Journal of Science and Technology.Vol.7, No.1, pp 34-40, 2006.

[9] E. K. Burke and Y. Bykov, A time-predefined approach to course timetabling, Yugoslav J. Operations Res., (YUJOR), 13(2):139-151, 2003.

[10] J. H. Orbit, D.L. Silva, D. Ouelhadj, and M. Sevaux, "Non-Linear Great Deluge with learning Mechanism for Solving the Course timetabling problem", MIC 2009: The VIII Meta - heuristics International Conference, Hamburg, Germany, pp. Id1 - id10.

[11] E. K. Burke and Y. Bykov, "A late acceptance strategy in Hill-climbing for Examinations Timetabling", In PATAT '08 Proceedings of the 7th International Conference on the Practice and Theory of Automated Timetabling, 2008.

[12] A. Abuhamdah and M. Ayob, "Average Late Acceptance Randomized Descent Algorithm for solving course timetabling problems", Proceeding in 4th International symposium on information technology, Selangor, Malaysia, IEEE, 2(15-7), 748-753, 2010.

[13] Colin R. Reeves (Ed.) "Modern Heuristic Techniques for Combinatorial Problems", Blackwell Scientific Publications, Oxford, 1993.

© 2010, IJARCS All Rights Reserved