A SIMULATED ANNEALING HEURISTIC FOR TANZANIAN HIGH SCHOOLS TIMETABLING PROBLEM

Allen R. Mushi[#1], Peter Batho[*2]

[#]*Department of Mathematics, University of Dar es salaam, Box 35062, Dar es salaam, Tanzania*
[*]*Department of Mathematics, Azania Secondary School, Box 9074, Dar es salaam, Tanzania*

***Abstract*— High School Course Timetabling Problem involves scheduling of lessons and teachers to timeslots within a week while satisfying a set of constraints. Like many other variants of timetabling problems, it is known to be NP- Hard, and therefore no optimal solution procedure is known to solve the problem in a reasonable time scale. Currently, very little work has been done in automating the high schools timetabling in Tanzania. Given the growing needs for educational resources in the country, optimization techniques for these resources cannot be avoided. In this paper, we present a heuristic algorithm based on Simulated Annealing. The algorithm has been tested with success on real data obtained from three high schools. A thorough experimentation has been done on six of cooling schedules. The results compare well with the previous work on the same data set and perform much better than manual system.**

***Keywords*— School Timetabling Problems, Combinatorial Optimization, Simulated Annealing, Cooling Schedules.**

I. INTRODUCTION

Academic course timetabling is a hard task that educational centers have to perform regularly. The process involves scheduling of lessons of each subject and the associated teachers to a fixed number of timeslots usually within a week. The schedules must adhere to some constraints which are normally divided into hard and soft. Hard constraints must be adhered to, while soft constraints have to be satisfied as much as possible. Many variants of timetabling problems exist ranging from industrial to academic timetables. Academic timetables are categorized into University and High School timetables. Academic timetables are grouped into course and examinations timetables. For further details on timetabling classifications, see Schaerf [1] and Burke [2]. This paper focuses on High School Course Timetabling Problem in Tanzania. Although many contributions to this problem exist, the characteristics of the problem vary depending on the school policies, the country laws and other particular variables.

Optimal solution strategies have been attempted basing on integer programming [3], [1] and network flow [5]. Some successes have been recorded on optimal solution methods but limited to small size instances. For instance, Birbas [6], presented a novel two-stage approach for the school timetabling problem. Integer programming models were successfully used for both stages for an application to a school with only 23 teachers and 95 courses.

The complexity of the problem makes it difficult to find an optimal solution, so heuristic techniques are traditionally used in practice [7]. Early heuristic techniques were based on a successive augmentation, that is, a partial timetable is filled in, lesson by lesson, until either all lessons have been scheduled or no lesson can be scheduled without violating the constraints. Obviously such methods can easily miss a feasible timetable in the search space. More recently, some approaches based on global search techniques appeared in the literature; among others; we have Simulated Annealing [8],[9], Tabu Search [10],[11], Genetic Algorithms [12], Constraints Satisfaction [13], and a combination of different methods [14]. Graph colouring algorithms have been attempted on high school algorithms including a paper by Abraham and Kingston [15] on the use of bipartite edge coloring. New approaches for high schools timetabling includes the work by Cerdeira-Pena [7]. They presented a Random Non Ascent (RNA) algorithm in combination with genetic algorithms to provide a stable solution. However, their algorithm was tested mostly on synthetic data and a few real data from a small size school (11 teachers and 33 classes). Only one paper is known by authors of this paper which attempts to develop algorithms for Tanzanian High Schools Timetables; it uses Non-Linear Great Deluge Algorithm [16]. This is therefore a new contribution to the problem by adapting Simulated Annealing. Simulated Annealing has been widely used in solving combinatorial optimization problems with great success. It is the interest of this paper to compare performances with the previous work and provide an alternative approach to the previously developed algorithm.

The next sections are organised as follows; first a general look of the high school timetabling in Tanzania is provided, followed by discussion on Simulated Annealing. An implementation of the algorithm to Tanzanian high school follows before presenting a summary of results. The last section provides conclusion and gives a direction for future research.

II. High School Course Timetabling Problem (HSCTP) in Tanzania

Tanzania educational system includes seven years of primary schools, followed by four years of ordinary secondary school, then two years of high schools (Advanced level). The performance from high schools is the main criteria for admission into tertiary level education. Recently, major efforts have been made to improve secondary school education which has resulted into huge expansion of ordinary level secondary schools throughout the country. The move has also resulted into stiff competition for the few available high school positions, prompting the interest on this research.

High school structure is organized in such a way that a student will have to pick three subjects for specialization. For instance, a student may be assigned Physics, Chemistry, and Mathematics (PCM) as subjects of specialization, the so called 'combination'. A high school may offer several combinations, catering for different student backgrounds. Each combination is divided into streams depending on the year of study and number of students required per classroom. Thus, a group PCMVA would stand for PCM combination for form V students in stream A - known as a student group. Each student group uses a fixed classroom throughout their studies. Different student groups however, may share the same subject and the same teacher. A teacher may teach more than one subject (at most two), where each subject has a pre-determined number of lessons to be taught every week. There are compulsory courses depending on the student combination. For instance, a student doing science or business related combinations, which do not include mathematics, will have to take a Basic Applied Mathematics (BAM) course. There are periods, which need to be set aside every week for common subjects such as religious studies and civics.

The schools have the following hard constraints in common;

1. A teacher cannot be assigned more than one lesson at the same time
2. A class cannot be assigned more than one lesson at the same time
3. Different lessons of the same subject group cannot be assigned to the same timeslot
4. The religion and civics timeslots must be respected and must have common times to all student groups
5. A timetable must be complete i.e. all lessons must be assigned to timeslots

Soft constraints considered in this implementation are as follows;

1. Lessons of the same subject must be spread as much as possible throughout the week.
2. Minimize the use of early morning timeslots as some teachers needs time to commute to work due to traffic jams; minimize the use of late afternoon hours for similar reasons.
3. Some teachers prefers to have free slots at some times of the week due to other commitments
4. Some lessons have preferences for free periods during the week e.g. Biology classes needs some time to collect samples in the field for dissection

III. Simulated Annealing Algorithm

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 the physical objects cools slowly down by following a particular cooling schedule. A temperature value T is defined initially, and cools slowly until an equilibrium point is achieved. The work by Reeves [17], gives a comprehensive description of the technique. The general algorithm is as presented by the pseudo code in Figure 1.

Initialize parameters (Temperature T, Freezing point F)
Get initial solution S_0
While temperature T > freezing point F {
Get the solution in the neighbourhood of S ($S \in N(S_0)$);
$\sigma = S - S_0$;
if ($\sigma < 0$);
Accept new solution ($S_0 = S$);
else{
Generate a random value *x* between 0 and 1;
if ($x < e^{\frac{-\sigma}{T}}$);
Accept new solution ($S_0 = S$;
Else Reject new solution;
}
Update temperature ($T = f(T)$);
}

Figure 1: Simulated Annealing Algorithm

IV. Adapting Simulated Annealing to the HSCTP

To apply Simulated Annealing to HSCTP, one needs to make a number of decisions with respect to the general algorithm; these are the kind of solution structure, neighbourhood structure, cooling schedules, initial solution strategy and stopping criteria.

A solution: is represented by a 0-1 variable *x* such that

$$x_{ijk}\begin{cases}1 \text{ if lesson } i \text{ of subject } j \text{ is in slot } k\\ 0 \quad \text{Otherwise}\end{cases}$$

The number of allowable periods per week differs from one school to another; in the case study, the number

of periods varies from 40 to 50. These are numbered consecutively from 1 on Monday to the last period on Friday.

Teachers' allocation to subjects is represented by an array T such that $t_j \in T$ is a teacher of subject *j* for a given stream.

Neighbourhood structure: We have considered swapping move in which a solution $x' \in X$ is a neighbour of solution $x \in X$ if it can be obtained by interchanging two timeslots k_1 and k_2 assigned to two lessons in *x*. The swapping move selects randomly two lessons and swaps their timeslots.

An example of a swap move is as shown in Table 1, where the contents of slots 11 and 14 are swapped.

TABLE 1: THE SWAP MOVE

MONDAY	**1**	**2**	**3**	**4**	**…**
TUESDAY	**11** ←	**12**	**13**	**14** →	**…**
	slot k_1			**slot k_2**	
WEDNESDAY	**21**	**22**	**23**	**24**	**…**
THURSDAY	**31**	**32**	**33**	**34**	**…**
FRIDAY	**41**	**42**	**43**	**44**	**…**

This type of move does not guarantee feasibility of the resulting neighbor but it generates a quick initial solution.

Objective function: is a linear combination of all constraints, both hard and soft; more weights are assigned to the hard constraints to discourage infeasibilities.

Thus, objective function is of the form

$$f(x) = \sum_{i=1}^{8} \lambda_i f_i(x)$$

Where λ_i= weight given to constraint *i* and $f_i(x_i)$ is the function representing constraint *i*.

The constraint functions are similar to those defined in Mushi [16]; and hereby summarized as follows;

Suppose there are *n* lessons, *y* subjects, *z* classes and *m* timeslots and;

$H = \{l_1, l_2, ...l_n\}$: set of all lessons

$K = \{k_1, k_2, ...k_m\}$: set of all timeslots (periods)

$J = \{j_1, j_2 ... j_y\}$: set of all subjects groups

$C = \{c_1, c_2, ...c_z\}$: set of all classes

L_j = Number of lessons of subject j

T_j = A teacher who teaches subject group j

G_j = A class of a subject group j

S_M = A set of early morning timeslots

S_A = A set of late afternoon timeslots

S_R = A set of common religion timeslots

S_P = A set of slots which have restrictions due to other preferences

Then;

- $f_1(x) = \lambda_1 v$; $f_1(x)$ counts the number of teacher collisions *v*,
  where
  $$v = \begin{cases} 1 \text{ if } \sum_{(i,j)\in H, k\in K'} \sum_{(p,q)\ni T_q = T_q} (x_{ipk} + x_{jqk} > 1) \\ 0 \quad \text{Otherwise} \end{cases}$$
- $f_2(x) = \lambda_2 w$; $f_2(x)$ counts the number of times *w* a class has been assigned more than one lesson at the same timeslot, where
  $$w = \begin{cases} 1 \text{ if } \sum_{k\in K'} \sum_{i\in H} \sum_{j\in J \ni G_j = c} x_{ijk} > 1 \\ 0 \quad \text{Otherwise} \end{cases}$$
- $f_3(x) = \lambda_3 u$; $f_3(x)$ counts the number of lesson collisions *u*, Where
  $$u = \begin{cases} 1 \text{if } \sum_{j\in J, k\in K'} \sum_{(u,r)\ni u\neq r} (x_{ujk} + x_{rjk} > 1) \\ 0 \quad \text{Otherwise} \end{cases}$$
- $f_4(x) = \lambda_4 \sum_{k\in S_P} x_{ijk}$; $f_4(x)$ counts the number of times that non-preference timeslots have been used.
- $f_5(x) = \lambda_5 \left| \sum_{j\in J} (\sum_{i\in H} \sum_{k\in K'} x_{ijk} - L_j) \right|$; $f_5(x)$ sums up the cases where assigned lessons of a subject are incomplete.
- $f_6(x) = \lambda_6 \sum_{(i_1,i_2)\in H} \sum_{(k_1,k_2)\in K' \ni k_1 \neq k_2} \sum_{j\in J \ni x_{i_1} j_{k_1} = x_{i_2} j_{k_2} = 1} \frac{1}{(k_1 - k_2)^2}$; $f_6(x)$ spreads lessons of a subject over the week.
- $f_7(x) = \lambda_7 \sum_{k\in S_M} x_{ijk}$; $f_7(x)$ minimizes the use of early morning hours,
- $f_8(x) = \lambda_8 \sum_{k\in S_A} x_{ijk}$; $f_8(x)$ minimizes the use of late afternoon hours.

A solution is feasible only if the sum of objective values of the hard constraints is zero.

A. *Initial* solution

Normally, it is better to start with an initial solution which satisfies all the hard constraints to give an initial feasible solution. Then the next task is to satisfy the soft constraints as much as possible. However, it is not very effective when the process of finding initial solution

spends too much time in finding initial feasible solution. An alternative is to start with a simple initial solution which is infeasible and penalise infeasibilities in the objective function. We follow the later option due to its simplicity in constructing initial solution. Going through the subjects, we assign all lessons of the subject to the timeslots, starting from 1 to the last timeslot until all lessons have been allocated. Once at end, the process continues from the first timeslot and continues to the next timeslot until all subject lessons have been assigned. In this way, we are reducing the possibility of lesson and teacher collisions but without guarantee of a feasible solution.

B. Cooling schedules

Most features in simulated annealing – solution space, move types, stopping criteria and objective functions are pre-determined and fixed during execution of the algorithm. The only feature which is variable during execution is the temperature. Thus, one of the most important features in simulated annealing is the way the temperature cools which depends on the type of cooling schedule. Many attempts have been made to suggest good schedules. This paper considers the following cooling schedules;

Geometric Cooling: $f(T) = \alpha T$ where $0<\alpha<1$ denotes the cooling factor. Typically the value of $\propto$ is chosen in the range 0.90 to 0.99. This cooling schedule is simple to implement and has a good theoretical foundation; it is also the most widely used [18].

Exponential Schedule: The annealing procedure in this schedule involves repeatedly lowering the temperature by a factor $0<\propto<1$, taking enough steps at each temperature to keep the system close to equilibrium until the system approaches ground state [19]. The exponential schedule is given by $f(T) = T_o \alpha^T$, where T is the temperature count and T_0 is the initial temperature.

Martin and Sierra [20] performed a comparison of Simulated Annealing cooling schedules. The following cooling schedules have been selected from their extensive list due their simplicity of implementation and varying cooling properties;

Logarithmic schedule: $f(T_k) = \dfrac{T_0}{1+\log(1+k)}$, where k=0,1, … indicates the temperature cycle i.e. iterations of the algorithm. This schedule does not depend on the previous temperature value but only on the iteration step.

Logarithmic Multiplicative cooling:

$f(T_k) = \dfrac{T_0}{1+\alpha \log(1+k)}$, ($\alpha$>1). This is a modification of the logarithmic schedule by introduction of a parameter α. The higher the values of α, the slower is the cooling and vive versa.

Linear multiplicative cooling: $f(T_k) = \dfrac{T_0}{1+\alpha k}$, ($\alpha$>0). Cooling in this schedule is inversely proportional to the value of α and iteration *k*. Thus cooling depends from both iteration step and a chosen value of control parameter α.

Quadratic multiplicative cooling: $f(T_k) = \dfrac{T_0}{1+\alpha k^2)}$, ($\alpha$>0). This modifies the previous linear multiplicative cooling by replacing *k* with k^2. The schedule is therefore more sensitive to the iteration step than the linear multiplicative cooling.

C. Stopping Criteria

The chosen stopping criteria is to use a minimum value for the freezing point and observe results with the best freezing point. It is common knowledge that sometimes an algorithm may end up spending most of the time in searching unnecessarily on the lower levels while the best value has already been achieved. Aspiration criteria are used to break this search. Through experimentation it was found necessary to have an aspiration criteria whereby the search is stopped when no changes happens in the cost function after 4000 iterations.

V. SUMMARY OF RESULTS

The algorithm was implemented in C++ language using an Intel(R) Core(TM) 2.93GHz processor, 2.00GB of RAM machine. Table 2 shows properties of the three high school problems – Jangwani, Azania and Tambaza, where the largest school is Tambaza with a total of 118 subjects, 50 teachers and 919 students.

TABLE 2: THE SUMMARY OF DATA FROM THE THREE HIGH SCHOOLS

Problem	Number of Students	Number of Subjects	Classes	Total Timeslots	Teachers
Jangwani	280	46	10	40	26
Azania	750	62	14	45	31
Tambaza	919	118	26	50	50

For standard evaluation of the schedules, the initial temperature (T_0) was set at 100,000 for both cases. This value was found through experimentation and seemed to work well to the tested problems. Several runs were performed on each problem and the suitable set of parameters was obtained as shown in Table 3. Note that, given a fixed value of T_0, the only parameter that needed turning was α, except for logarithmic schedule which does not require α.

TABLE 3: PARAMETERS USED IN EACH PROBLEM AND COOLING SCHEDULE

Cooling Schedule To=100,000	Azania (α)	Jangwani (α)	Tambaza (α)
Exponential	0.995	0.995	0.995
Logarithmic	-	-	-
Geometric	0.99	0.995	0.99
Logarithmic Multiplicative	100	100	100
Linear Multiplicative	50	100	100
Quadratic multiplicative	0.3	0.3	0.5

The objective function coefficients (λ) are as shown in Table 4 below; they are the same as those used in Mushi [16].

TABLE 4: VALUES OF λ IN THE COST FUNCTION

Weight	Value	Description
λ_1	20	Teacher collision
λ_2	10	Class collision
λ_3	10	Lesson collision
λ_4	3	Non-preference timeslots
λ_5	10	Lesson completeness
λ_6	5	Lesson spread
λ_7	3	Early morning hours
λ_8	3	Late afternoon hours

Experimental results are as shown in Table 5 for all cooling schedules. The last row shows a comparison with previous results from Non-Linear Great Deluge algorithm [16] on the same data set. Exponential and geometric schedules have outperformed other schedules in terms of cost reduction in both cases. They also differ in processing time, where exponential schedule was significantly faster than the rest of schedules. Both schedules performed slightly better in terms of solution value than the Non-Linear Great Deluge, although the later was faster in most of the cases.

TABLE 5: EXPERIMENTAL RESULTS VS COOLING SCHEDULES

Cooling schedule	Azania Initial solution = 630.67		Jangwani Initial solution = 500.67		Tambaza Initial solution=2190.67	
	Final solution	Time (Sec)	Final solution	Time (Sec)	Final solution	Time (Sec)
Exponential	303.003	182	249.003	63	549.003	557
Logarithmic	303.003	199	249.005	123	549.006	589
Geometric	303.003	220	249.003	127	549.003	649
Logarithmic Multip	303.003	468	249.003	283	589.003	566
Linear Multip	303.003	197	252.003	129	549.003	600
Quadratic multip	303.003	191	252.003	134	549.003	601
Non-Linear Great Deluge	306.002	169	249.005	110	549.006	526

It was also interesting to note the behaviour of cooling curves of the tested schedules as shown in Figure 2; a case taken from Azania dataset. Exponential and Geometric schedules follow a similar curve which slopes smoothly to the minimum temperature value of zero. Logarithmic schedule starts by dropping sharply and then follows a very slow cooling path which did not reach freezing point within reasonable time; the algorithm had to be stopped by the aspiration criteria. Surprisingly, the cost function still converged to the best possible value, indicating that at the lower stages of this schedule, cost reduction does not vary with the cooling schedule but depends on the length of time spent at the lower levels.

Both multiplicative schedules (logarithmic, linear and quadratic) shows a sharp drop to almost freezing point and thereby spend most of the searching time near the freezing point. This approach may skip most of the bad solution moves and tends to be like greedy search algorithms. Clearly, the curve representing exponential and geometric functions relates well with their reported better performances. A good cooling schedule must therefore be able to spend significant time on upper levels before approaching the lower temperature levels. This allows for selection of some bad solutions at higher temperatures and thereby improves the chance for better convergence.

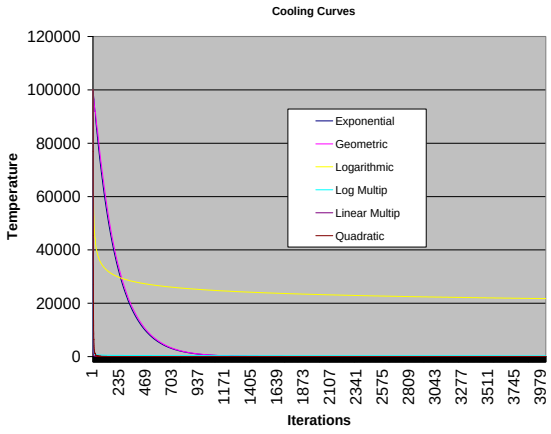


Figure 2: Cooling Curves for Azania Case

The behaviour of cost reduction in the objective function was also observed and both cases showed a similar step-drop structure as shown in Figure 3. This sample from Azania dataset shows quick drops with some prolonged intervals of search within a cost level, before moving to lower level. This shows the importance of having a cooling schedule which can spend some time in each level of cost reduction before moving into lower level.

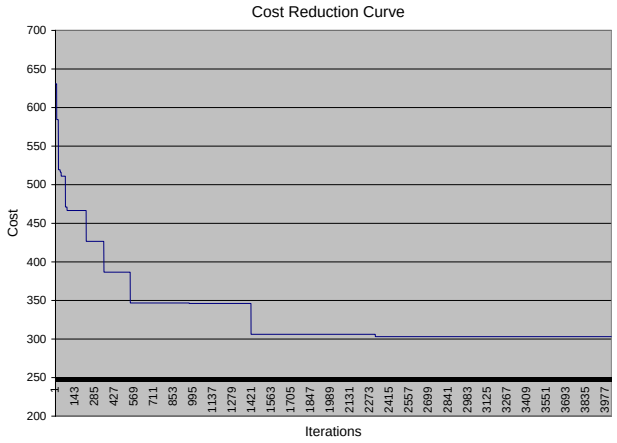


Figure 3: Cost Reduction Function for Azania data

The best automated solution was then compared with manually generated solution. The manual solution was found by converting the properties of the manually generated solution into the solution structure x_{ijk}; then run this solution into the objective function to obtain the resulting cost. Table 6 shows the resulting comparison; clearly automatic solution far outweighs the quality of the manually generated solution. This is shown by the percentage improvement column which presents the percentage of further reduction compared to the manual solution.

TABLE 6: MANUAL VS AUTOMATIC TIMETABLE SOLUTIONS

Problem	Final Cost (Manual)	Final Cost (Automatic)	% Improvement
Azania	339.108	303.003	10.64
Jangwani	311.014	249.003	19.93
Tambaza	611.571	549.003	10.23

VI. CONCLUSION

In this paper, a Simulated Annealing algorithm has been developed for the high schools timetabling problem in Tanzanian environment. Since the solution quality very much depends on the quality of the cooling schedule, a total of six cooling schedules were tested as selected from various literatures. The cooling schedules were selected on the criteria of their simplicity of implementation, reported success and dependence on a single parameter $\propto$. A number of parameters have been proposed in the summary of results including a table of values of α and a unique initial temperature value of T_0 = 100,000. It has been concluded that the best cooling schedule in this implementation is exponential, followed by geometric. These cooling schedules have also shown a smooth temperature reduction which allows for better exploration of neighbourhood at each temperature value.

Furthermore, it has been observed that the implementation performs better than the previous work on Non-Linear Great Deluge algorithm as applied in the same dataset. The automatic methods have shown better performance than manual solutions, paving the way for further research for the Tanzanian high schools problems.

D. Future Research Directions

We have considered high schools at advanced level only i.e. form V and form VI. Some schools have a mixture of both ordinary and advanced level at the same school and shares the same resources. It would be interesting to extend the algorithm and cover both levels. Furthermore, the study was based on the three available datasets; more schools needs to be brought into the study which may reveal further challenging features.

Six schedules were considered in this study, however many variants of the simulated annealing exists, including Mean-Field Annealing and Adaptive cooling [18]. There are many global heuristics which have been successfully applied to timetabling problems, including Tabu Search, Genetic Algorithms, Tilling Algorithm [21] and many others. Their adaptation and comparison of performances could be worth investing, for Tanzanian high school timetabling problems. A variant of the problem is that of examinations timetabling; so far no work is known, which has addressed the examinations timetabling problems to Tanzanian high schools. That is another avenue for exploration.

REFERENCES

[1] Schaerf, A., "A survey of Automated Timetabling", Journal of Artificial Intelligence Review, vol. 13, pp 87-127, 1999.

[2] Burke, E. K., Kingston, J and De Werra, D, "Application to timetabling", section 5.6 of the Handbook of Graph Theory: pp. 445–474, Chapman Hall, 2004.

[3] Tripathy, A., "Computerized Decision Aid for Timetabling: A case Analysis", Discrete Applied Mathematics, vol. 35(3), pp 313–323, 1992.

[4] Birbas T, Daskalaki S, Housos E., "Timetabling for Greek high schools", Journal of the Operational Research Society, vol. 48, pp. 1191-1200, 1997.
[5] De Werra, D, "An introduction to Timetabling", European Journal of Operational Research, vol. 19, pp. 151–162, 1985.
[6] Birbas T, Daskalaki S, Housos E., "Timetabling for Greek high schools", Journal of the Operational Research Society, vol. 48, pp. 1191-1200, 1997.
[7] Cerdeira – Pena, A., Carpente, L., Fari, L and Seco, D, "New Approach for the Timetabling problem", Seventh Mexican International Conference on Artificial Intelligence: pp. 261 – 267, 2008.
[8] Abramson, D, "Constructing School Timetables using Simulated Annealing: Sequential and Parallel Algorithms", Management Science, vol. 37(1), pp. 98–11, 1991.
[9] Defu Zhang, Yongkai Liu, Rym M'Hallah, Stephen C. H. Leung, "A Simulated Annealing with a new neighbourhood structure based algorithm for high school timetabling problems", European Journal of Operational Research, vol. 203, Elsevier, pp. 550-558, 2010.
[10] Costa, D, "A Tabu Search Algorithm for Computing an Operational Timetable", European Journal of operational Research, vol. 76: pp. 98-110, 2003
[11] Schaerf A., "Tabu Search Techniques for Large High School Timetabling Problems", AAAI-96 Proceedings, pp. 363-368, 1996.
[12] Darigo, M., Colorni, A and Maniezo, V, "Genetic Algorithms: A New Approach for the timetable Problem", NATO ASI series F 82, pp. 235–239, Berlin: Springer, 1990.
[13] Yoshikawa, M., Kaneko. and Watanabe, M, "A constraint-Based High school Scheduling System", IEEE Experts, vol. 11(1), pp. 63–72, 1996.
[14] Cooper, T.B and Kingston, J. H, The "Solution of Real Instances of the timetabling Problem", The computer journal, vol. 33(6): 645–653, 1993.
[15] Abraham D, Kingston J, "Generalized Bipartite Edge colouring to solve Real Instances of the Timetabling Problem", PATAT-02, 4th Conference of Principles and Practice of Automated Timetabling, Gent, Belgium, pp. 141-143, 2002.
[16] Mushi A., "Non-Linear Great Deluge Algorithm for Tanzanian High Schools Timetabling", International Journal of Advanced Research in Computer Science, vol. 2, pp. 584-590, 2011.
[17] Reeves, C. V., "Improving the Efficiency of the Tabu search for Machine Sequencing, Problems", The Journal of the Operational Research Society, vol. 44(4), pp 375 – 382, 1993.
[18] Elmohamed M. A. S, "A Comparison of Annealing Techniques for Academic Course Scheduling", Northeast Parallel Architecture Centre, Paper 8, Syracuse University, SURface, 1997.
[19] Nourani Yaghout, Andresen Bjarne, "A comparison of simulated annealing cooling strategies", Journal of Physics A-Mathematical and General, vol. 31, pp. 8373-8385, 1997.
[20] Martin J. F. D, Sierra J. M. R., "A Comparison of Cooling Schedules for Simulated Annealing (Artificial Intelligence)", Encyclopaedia of Artificial Intelligence, IGI Global, pp. 344-352, 2009.
[21] Kingston J H., "A Tilling Algorithm for High School Timetabling", PATAT V, Pittsburg USA, pp. 233-249, 2004.