International Journal of Scientific & Engineering Research Volume 3, Issue 8, August-2012 1
ISSN 2229-5518

Hybrid MetaHeuristic Feature Extraction Technique for Solving Timetabling Problem

Olabiyisi Stephen O., Fagbola Temitayo M., Omidiora Elijah O. and Oyeleye Akin C.

(Only author names, for other information use the space provided at the bottom (left side) of first page or last page. Don't superscript numbers for authors)

Abstract— In this paper, a hybrid algorithm that combines extracted features of two well known metaheuristics, Simulated Annealing (SA) and Genetic Algorithm (GA) is proposed. In the proposed algorithm, useful features of each metaheuristic are exploited to obtain better solutions for the automatic generation of examination timetable. A performance evaluation of the proposed algorithm was carried out. The computational results illustrate the ability of the hybrid algorithm to provide good quality solutions to the examination problem instances within reasonable computation time.

Index Terms— Feature Extraction, Genetic Algorithm, Simulated Annealing, Examination Timetabling Problem.

1 Introduction

AN educational timetabling is a multi-dimensional and highly constrained problem. Examination timetabling problem can be defined to be the problem of assigning a number of events into a limited number of time periods. Burke [1] defines timetabling as follows. "*Timetabling* is the allocation, subject to constraints, of given resources to objects being placed in space time, in such a way as to satisfy as nearly as possible a set of desirable objectives."

University Examination Timetable Problem is NP complete problem and has received tremendous attention from disciplines like Operations Research and Artificial Intelligence during past few years given its wide use in universities [2]. 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. However, the more soft constraints are satisfied the better the quality of the timetable.

Examinations must be scheduled so that no student has more than one examination at a time. Generating educational timetables manually often involves numerous rounds of changes before they can be satisfactory. The problem becomes even more difficult when student number rises which makes automated system a necessary tool.

The examination timetabling problems can be classified in terms of the specific solution techniques used. The common solution techniques used in timetabling research are graph coloring heuristics, mathematical programming, tabu search, simulated annealing, genetic algorithms, network flow models, and constraint programming [3].

However, Genetic Algorithms (GA) and Simulated Annealing (SA) have emerged as the leading methodologies for search and optimization problems in high dimensional spaces. Previous attempts at hybridizing these two algorithms have been cumbersome and required major changes to both.

Over the last decade, *Genetic Algorithms* (GA) have emerged as a leading tool for optimization of arbi*trary* functions and for guided search problems in high dimensional spaces. GA's are typically comprised of two types of operations: *mutation* and *crossover* which are repeatedly applied to a population of *chromosomes*, each of which encodes a possible solution to the given problem. GA's have been successfully applied to many theoretical optimization problems and several industrial applications.

GA is not very efficient because of its need to maintain a large population of solutions and this may consume several megabytes of memory for the encoding of a single solution and thus not as good as SA in this regard. It would be impractical to manipulate a large population of candidate solutions using GA. Another problem frequently found in GA optimization is premature convergence. This is typically the result of the extreme reliance on crossover. The dominance of crossover can result in stagnation as the population becomes more homogeneous, and the mutation rate is too low to move the search to other areas.

Simulated Annealing (SA) is another algorithm which is popular in heuristic optimization. SA belongs to a class of algorithms called *probabilistic hill-climbing* which dynamically alter the probability of accepting inferior solutions. The SA algorithm is especially popular in the field of VLSI design where it has been successfully applied to the optimization of extremely high-dimensional problems which contain *tens* or *hundreds of thousands* of parameters to be optimized.

Simulated annealing is a search strategy which keeps track of one feasible timetable. On each iteration, a neighbour is generated – another feasible timetable, slightly altered at random from the current one. This neighbour is accepted as the current timetable if it has a lower penalty. If the neighbour has a higher penalty, it may be accepted according to a probability which is related to a control parameter called temperature. The temperature, and thus the probability of inferior neighbours being accepted, is decreased each iteration or (more usually) after a particular number of iterations (this number may be constant or it can increase as the temperature decreases). The process is analogous to the cooling process in actual annealing. One drawback with simulated annealing is that the cooling process can take a long time in order to achieve good results.

However, SA obtains very good solutions, only if its parameters are well tuned. SA requires an initial solution for solving NP complete, combinatorial and optimization problems for the resultant solution to be satisfactory and this is a major limitation of the SA algorithm. Since neither of the two algorithms seems to be universally preferred for all problems, researchers have often resorted to building a large battery of optimization algorithms and finding, through experimentation, which tool satisfactorily fits the problem at hand. This provides the basic motivation for trying to merge GA and SA into a single algorithm or module, which can be designed and configured as the hybrid mode of GA and SA.

In this paper, a new method that hybridizes genetic algorithm and simulated annealing algorithm (GA-SA Hybrid) is proposed, designed and implemented for the automatic generation of tertiary institution examination time-tabling structure.

2 REVIEW OF RELATED WORKS

Several works have approached the timetabling problem. Oyeleye [4] developed a SAGA hybrid algorithm by using the initial temperature and cooling rate of SA to control the operations of the GA. The hybrid system was evaluated using certain evaluation metrics including the program size, simulation time, program volume, program level, program effort and lines of code. The researcher concluded that the hybrid system developed returned feasible examination timetable that resulted in best performance when compared with GA and SA models. Mushi [5] worked on simulated annealing algorithm for the examinations timetabling problem at University of Dar es salaam, based on a Simulated Annealing heuristic. Mushi was able to solve an existing problem and show that the automated system performs better and faster than the manually generated solution.

Dimopoulou & Milliotis [6] reported a system which combines both Integer Programming and heuristic procedures for Athens University of Economics and Business. Several researchers have attempted this problem using simulated annealing including [7], [8]. Tabu search methods have also been used by many researchers such as [9], [10] and [11]. There are also researches on the use of evolutionary algorithms [12] and constraint satisfaction methods [13].

A more thorough survey of Examinations timetabling problems is provided by [14]. 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 [7], [9] and [15].

Analyzing the results obtained by the various works published, we can say that the automatic generation of schedules is capable of achieving. Some works showed that when compared with the manually scheduled examination timetables in institutions of learning, the time tables obtained by the algorithms for solving the examination timetabling problem are of better quality using some function of evaluation.

3 MATERIALS AND METHOD

3.1 Framework for the Examination Timetable

The examination timetabling problem can be seen as consisting of two subproblems:

(1) Assigning timeslots to an examination
(2) Assigning an examination to appropriate venues or theatres.

The examination timetabling problem is subject to a variety of hard and soft constraints. Hard constraints need to be satisfied in order to produce a *feasible* solution.

In this problem, in order for a timetable to be *feasible*, it is necessary that every exam event $e_1,\ldots,e_n$ is assigned to exactly one room $r_1,\ldots,r_m$ and exactly one of t timeslots (where in all cases $t \leq 36$, which is to be interpreted as twelve days of three timeslots), such that the following three hard constraints are satisfied: Constraints that will be considered include:

3.1.1 Hard Constraints

i.) No student is required to attend more than one event at any one time (or, in other words, conflicting exam events should not be assigned to the same timeslot);
ii.) All exam events are to be assigned to *suitable* rooms. That is, all of the features required by an exam event are satisfied by its room, which must also have an adequate seating capacity;
iii.) Only one exam event is assigned to any one room in any timeslot (i.e. no *double-booking* of rooms is allowed).

3.1.2 Soft Constraints

i.) Candidates prefer to have at least one gap between. In general, we would like to spread each candi date 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.

3.2 Representation Model for the Exam Timetable

Definition

H - Set of all the periods of time within which examinations can occur. $H = \{h_1, h_2, \ldots h_m\}$ where m corresponds to the maximum number of periods of time.

Definition

D - Set of all subjects, in a given season, that will be under examination.

$D = \{d_1, d_2, \ldots d_k\}$ where k is the maximum number of subjects, in a given season will be under examination.

3.3 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 minimize the function;

$$f(x) = \sum_{i=1}^{h} \lambda i \quad fi \qquad (x)$$

(1)
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.

3.4 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) ϵ E with $i < j$ such that s_i=s_j, and define $f_i(s) = \sum_{(i,j)\in A} mij$, then minimize $\lambda_i f_i(s)$ f_i (s) gives the total number of candidate clashes associated with the current solution, and therefore f_i (s) = 0 is a necessary condition for a feasible solution. Since this is a hard constraint, λ_i must be a sufficiently large value.

ii.) *Room capacities must not be violated;*

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

$$r_i - \sum_{j\epsilon Bn} C_f$$

(2)

For feasibility, the capacity of room i must be ≥ 0.
Also let

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

for some room i.
Then the function is

$$f_2(s) = \sum_{i\in E} \sum_{t} d_{it},$$

(3)
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 $f_3(S) = \max_{i\epsilon E}\{|k_i|\}$ is minimized to $\lambda_3 f_3(s)$ where λ_3 is a weight value.

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 uϵU
Assign portions of u into the emptiest space until all is scheduled.
If infeasible assign to the closest feasible timeslot.
Next u
End_Initial_Timetable

3.5 A Framework for the Hybrid GA-SA Algorithm

At the point of convergence of the evaluation function of both the GA and SA comes the integration and model design of the hybrid using certain features. Two solutions are selected with a decreasing probability of selecting less-fitted feature solutions. In order to decrease the probability of selecting less-fitted features, the fitness evaluation function is changed to the following:

$$f_i = (D_{MAX} - D_i)^{\alpha\times t} \qquad (4)$$

where t is the number of iterations or generations. As the number of generation increases, the fitness value would increase and induce the algorithm to choose better-fitted solutions. The mutation rate would decrease as the number of generations grows.
This is formulated as follows:

$$C_m = 1 - t / t_{max} \qquad (5)$$

where t is the number of iterations or generations that the algorithm has gone through and t_{max} is the maximum number of generations.
If the produced offspring is less-fitted than the worst solution in the population, it would replace the worst solution only when the probability $\delta \le e^{(-\Delta E / T)}$ is met.

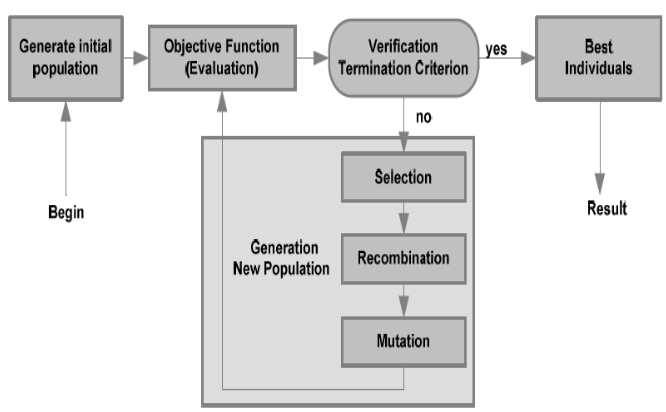

Fig. 1. Flow of GA

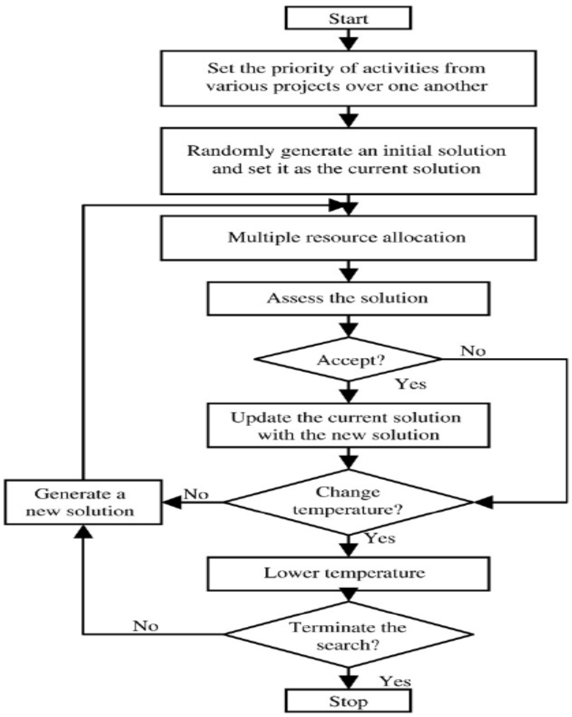

Fig. 2. Flow of SA

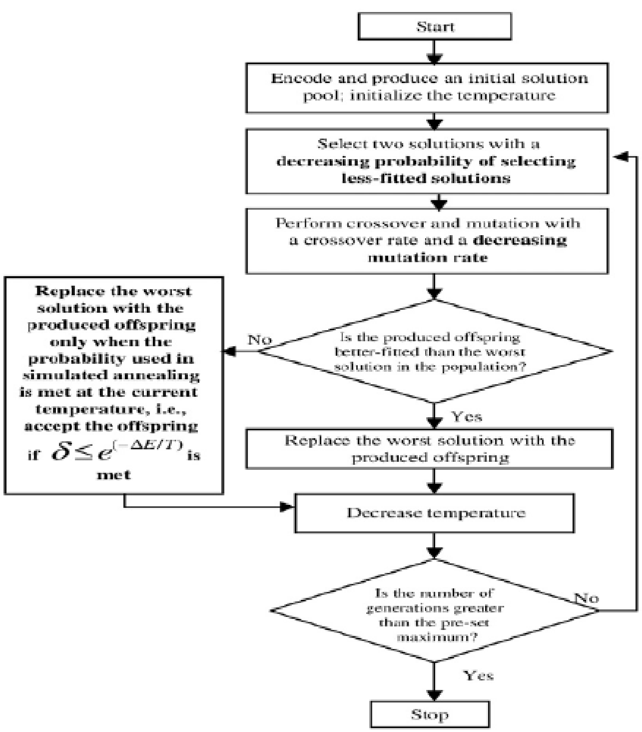

Fig. 3. Flow of hybrid of GA and SA

3.6 Simulation Parameters

3.6.1 Violations of the Examination Timetabling

Course clashing, venue capacity, venue with lab equipment, list of courses, total number of students per course, list of venues and capacity of venues

3.6.2 Parameters for GA-SA

Number of generations = 1000, size of population = 100, chromosome length = 54 (54 timeslots in a week), mutation probability = 0.1, crossover probability = 0.7, Length of markov chain = 10000, maximum temperature = 100, alpha = 0.95, freezing point = 0.1.

3.7 Implementation Tool

The programming tool used to implement the algorithms is MATLAB. This is because MATLAB is a very powerful computing system for handling the calculations involved in scientific and engineering problems. The name MATLAB stands for MATrix LABoratory. With MATLAB, computational and graphical tools to solve relatively complex science and engineering problems can be designed, developed and implemented. The timetabling problem follows LAUTECH timetable dataset format and will be used to evaluate the performance of the developed hybrid GA-SA system.

4 Results and Discussion

From the summary of the results obtained from the simulation, simulated annealing algorithm performs better than both the genetic algorithm and the hybrid GA-SA algorithm in terms of optimality of output generated.

However, simulation results showed that simulated annealing algorithm spends a more considerable time to generate the timetable than the other two algorithms which accounts for the optimality of the timetable generated as almost all hard constraints are satisfied.

The genetic algorithm on the other hand spends a lesser time than the simulated annealing algorithm during the generation process but does that by violating some hard constraints.

As computing resource is very expensive, there occurs a need to reduce the time and space complexities inherent in the use of algorithms, hence a need for a more time-enhanced algorithm which came as the hybrid GA-SA algorithm. The hybrid GA-SA, though violated some hard constraints as observed in the GA result also, executes with reduced time for generating the output. It is the most efficient algorithm in terms of time and space complexities and computing resource management though optimality of result is not guaranteed. However, the simulated annealing algorithm consumes a lot of computing resource but ensures optimality of output generated.

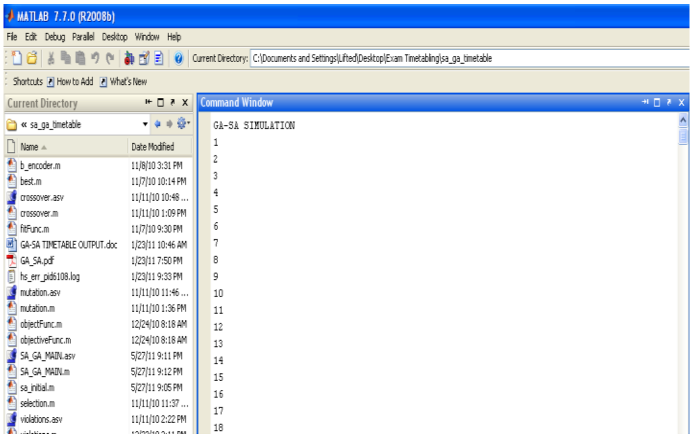

Fig. 4. Hybrid GA-SA during execution

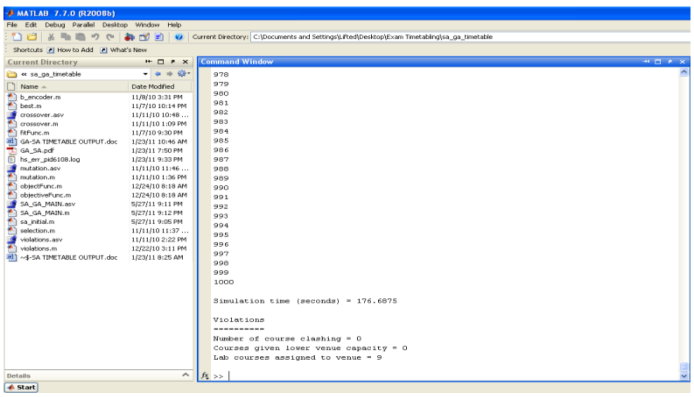

Fig. 5. Hybrid GA-SA Execution Completion

LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY, OGBOMOSO

2009/2010 RAIN SEMESTER EXAMINATION TIMETABLE

DATE		8:00 – 11:00AM		12:00-3:00PM			4:00-7:00PM		
DAY 1		COURSES	VENUES		COURSES	VENUES		COURSES	VENUES
	Hr 1	EEE 202	OBEL (90) 80	Hr 4	CSE 312	PL (90) 90	Hr 7	FSE 302	OBEL(90)90
	Hr 2	EEE 202	NBEL (95) 80	Hr 5	CSE 312	MICOM (90) 90	Hr 8	MEE 208	LH(100) 80
	Hr 3	EEE 202	NBEL (95) 80	Hr 6	EEE524	MICOM (90) 75	Hr 9	AGE 516	NLT(250) 75
DAY 2	Hr 10	CVE 526	MKO (750) 45	Hr 13	CSE 306	PL(85)90	Hr 16	MEE 232	MICOM(90)850
	Hr 11	CHE 310	LH(100)90	Hr 14	CSE 512	OBEL(90)30	Hr 17	AGE 204	LH(100)120
	Hr 12	CHE 310	OBEL(90)90	Hr 15	FSE 308	LH(100)90	Hr 18	CVE 332	PL(85)350
DAY 3	Hr 19	CHE 206	NBEL(95)80	Hr 22	FSE 508	LH (100)75	Hr 25	CVE 500	MKO(750)60
	Hr 20	EEE 200	MKO(750)65	Hr 23	MEE 506	LH(100)75	Hr 26	CHE 210	NBEL(95)80
	Hr 21	CSE 306	OBEL (90)90	Hr 24	AGE 304	BEL(85)90	Hr 27	CSE 314	MICOM(90)90
DAY 4	Hr 28	EEE 316	PL(85)90	Hr 31	CHE 310	PL(85)90	Hr 34	CSE 314	NLT(250)90
	Hr 29	CHE 504	PL(85)45	Hr 32	CHE 310	1200LT(1200)90	Hr 35	EEE 200	BEL(85)65
	HR 30	MEE 232	MKO(750)850	Hr 33	MEE 506	OBEL(90)75	Hr 36	CVE 500	OBEL(90)60
DAY 5	Hr 37	CVE 332	MKO(750)350	Hr 40	CSE 512	MICOM(90)30	Hr 43	CHE 210	LH(100)80
	Hr 38	CSE 512	NBEL(95)30	Hr 41	CHE 504	MICOM(90)45	Hr 44	FSE 302	OBEL(90)90
	Hr 39	FSE 508	1200LT(1200) 75	Hr 42	CVE 332	NBEL(95)350	Hr 45	CSE 312	OBEL(90)90
DAY 6	Hr 46	EEE 524	NLT(250)75	Hr 49	AGE 516	OBEL(90)75	Hr 52	EEE 200	1200LT(1200)6 5
	Hr 47	EEE 316	NBEL(95)90	Hr 50	CVE 526	NLT(250)45	Hr 53	MEE 232	NBEL(95)850
	Hr 48	MEE 208	NLT(250)80	Hr 51	CVE 526	BEL(85)45	Hr 54	CHE 310	BEL(85)90

****

HINT: XYZ (1)2 → COURSE (XYZ) AND CAPACITY (1) AND STUDENT_ASSIGNED (2)

****

Fig. 6. GA Timetable Generated

LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY, OGBOMOSO

2009/2010 RAIN SEMESTER EXAMINATION TIMETABLE

DATE		8:00 – 11:00AM		12:00-3:00PM			4:00-7:00PM		
DAY 1		COURSES	VENUES		COURSES	VENUES		COURSES	VENUES
	Hr 1	CSE 312	MICOM (90) 90	Hr 4	CHE 310	PL (90) 90	Hr 7	CSE 306	OBEL(90)90
	Hr 2	CHE 210	PL (85) 80	Hr 5	MEE 208	MICOM (90) 90	Hr 8	CSE 512	LH(100) 80
	Hr 3	CSE 314	1200LT (1200) 90	Hr 6	CVE 526	MICOM (90) 75	Hr 9	AGE 304	NLT(250) 75
DAY 2	Hr 10	EEE 524	MKO (750) 45	Hr 13	EEE 202	PL(85)90	Hr 16	CHE 210	MICOM(90)850
	Hr 11	CVE 500	LH(100)90	Hr 14	MEE 208	OBEL(90)30	Hr 17	CHE 206	LH(100)120
	Hr 12	CVE 332	OBEL(90)90	Hr 15	AGE 204	LH(100)90	Hr 18	CHE 206	PL(85)350
DAY 3	Hr 19	FSE 308	NBEL(95)80	Hr 22	AGE 516	LH (100)75	Hr 25	AGE 304	MKO(750)60
	Hr 20	EEE 200	MKO(750)65	Hr 23	EEE 524	LH(100)75	Hr 26	CHE 504	NBEL(95)80
	Hr 21	CSE 306	OBEL (90)90	Hr 24	EEE 202	BEL(85)90	Hr 27	AGE 204	MICOM(90)90
DAY 4	Hr 28	CVE 332	PL(85)90	Hr 31	FSE 302	PL(85)90	Hr 34	MEE 506	NLT(250)90
	Hr 29	CVE 500	PL(85)45	Hr 32	MEE 506	1200LT(1200)90	Hr 35	EEE 200	BEL(85)65
	HR 30	MEE 232	MKO(750)850	Hr 33	EEE 316	OBEL(90)75	Hr 36	CVE 500	OBEL(90)60
DAY 5	Hr 37	CVE 332	MKO(750)350	Hr 40	CSE 512	MICOM(90)30	Hr 43	CHE 210	LH(100)80
	Hr 38	CSE 512	NBEL(95)30	Hr 41	CHE 504	MICOM(90)45	Hr 44	FSE 302	OBEL(90)90
	Hr 39	MEE 232	1200LT(1200) 75	Hr 42	CVE 332	NBEL(95)350	Hr 45	CSE 312	OBEL(90)90
DAY 6	Hr 46	CVE 526	NLT(250)75	Hr 49	CSE 306	OBEL(90)75	Hr 52	CVE 526	1200LT(1200)6 5
	Hr 47	CSE 512	NBEL(95)90	Hr 50	MEE 232	NLT(250)45	Hr 53	CSE 512	NBEL(95)850
	Hr 48	EEE 316	NLT(250)80	Hr 51	EEE 200	BEL(85)45	Hr 54	EEE 316	BEL(85)90

****

HINT: XYZ (1)2 → COURSE (XYZ) AND CAPACITY (1) AND STUDENT_ASSIGNED (2)

****

Fig. 7. SA Timetable Generated

LADOKE AKINTOLA UNIVERSITY OF TECHNOLOGY, OGBOMOSO

2009/2010 RAIN SEMESTER EXAMINATION TIMETABLE

DATE		8:00 – 11:00AM		12:00-3:00PM			4:00-7:00PM		
DAY 1		COURSES	VENUES		COURSES	VENUES		COURSES	VENUES
	Hr 1	EEE 202	OBEL (90) 80	Hr 4	CSE 312	PL (90) 90	Hr 7	FSE 302	OBEL(90)90
	Hr 2	EEE 202	NBEL (95) 80	Hr 5	CSE 312	MICOM (90) 90	Hr 8	MEE 208	LH(100) 80
	Hr 3	EEE 202	NBEL (95) 80	Hr 6	EEE524	MICOM (90) 75	Hr 9	AGE 516	NLT(250) 75
DAY 2	Hr 10	CVE 526	MKO (750) 45	Hr 13	CSE 306	PL(85)90	Hr 16	MEE 232	MICOM(90)850
	Hr 11	CHE 310	LH(100)90	Hr 14	CSE 512	OBEL(90)30	Hr 17	AGE 204	LH(100)120
	Hr 12	CHE 310	OBEL(90)90	Hr 15	FSE 308	LH(100)90	Hr 18	CVE 332	PL(85)350
DAY 3	Hr 19	CHE 206	NBEL(95)80	Hr 22	FSE 508	LH (100)75	Hr 25	CVE 500	MKO(750)60
	Hr 20	EEE 200	MKO(750)65	Hr 23	MEE 506	LH(100)75	Hr 26	CHE 210	NBEL(95)80
	Hr 21	CSE 306	OBEL (90)90	Hr 24	AGE 304	BEL(85)90	Hr 27	CSE 314	MICOM(90)90
DAY 4	Hr 28	EEE 316	PL(85)90	Hr 31	CHE 310	PL(85)90	Hr 34	CSE 314	NLT(250)90
	Hr 29	CHE 504	PL(85)45	Hr 32	CHE 310	1200LT(1200)90	Hr 35	EEE 200	BEL(85)65
	HR 30	MEE 232	MKO(750)850	Hr 33	MEE 506	OBEL(90)75	Hr 36	CVE 500	OBEL(90)60
DAY 5	Hr 37	CVE 332	MKO(750)350	Hr 40	CSE 512	MICOM(90)30	Hr 43	CHE 210	LH(100)80
	Hr 38	CSE 512	NBEL(95)30	Hr 41	CHE 504	MICOM(90)45	Hr 44	FSE 302	OBEL(90)90
	Hr 39	FSE 508	1200LT(1200) 75	Hr 42	CVE 332	NBEL(95)350	Hr 45	CSE 312	OBEL(90)90
DAY 6	Hr 46	EEE 524	NLT(250)75	Hr 49	AGE 516	OBEL(90)75	Hr 52	EEE 200	1200LT(1200)6 5
	Hr 47	EEE 316	NBEL(95)90	Hr 50	CVE 526	NLT(250)45	Hr 53	MEE 232	NBEL(95)850
	Hr 48	MEE 208	NLT(250)80	Hr 51	CVE 526	BEL(85)45	Hr 54	CHE 310	BEL(85)90

****

HINT: XYZ (1)2 → COURSE (XYZ) AND CAPACITY (1) AND STUDENT_ASSIGNED (2)

Fig. 8. Hybrid GA-SA Timetable Generated

TABLE 1
SUMMARY OF RESULTS OBTAINED

	GA ALGORITHM	SA ALGORITHM	GA – SA ALGORITHM
Simulation time(seconds)	197.3438	561.6094	176.6875
Nos of Courses Clashing	0	0	0
Courses given lower venue capacity	9	0	0
Lab courses assigned to venues	7	7	9

5 Conclusion

As computing resources become very expensive, there arises a need to reduce the time and space complexities inherent in the use of algorithms, hence a need for a more time and spaced-enhanced algorithm which came as the hybrid GA-SA algorithm as proposed in this study. Simulated annealing and genetic algorithm have been successfully used for solving the examination timetabling problem. However, the results generated indicates a very high consumption of computing resources by simulated annealing but with high optimality while genetic algorithm results showed that though the consumption of computing resources is reduced yet the two algorithms still consume a considerable part of the computing resources.

This study designed a hybrid GA-SA algorithm which presents an output with a well minimized utilization of computing resources. A performance evaluation was carried out among the three algorithms. The result of the evaluation revealed that in terms of optimality of result without taking cognizance of the time and space complexities, simulated annealing is the best of the three. In addition, based on computing resource management, the hybrid GA-SA algorithm is the best of the three algorithms under such consideration.

References

[1] E. K. Burke, B. McCollum, A. Meisels, S. Petrovic, and R. Qu (2007), "A Graph-Based Hyper-Heuristic for Educational Timetabling Problems," European Journal of Operational Research, 176, 177-192

[2] Abdullah, S., Ahmadi, S., Burke, E. K., and Dror, M., (2007), "Investigating Ahuja-Orlin's Large Neighborhood Search Approach for Examination Timetabling," OR Spectrum, 29, 351-372.

[3] M. Henz, and J. Würtz (1996), "Constraint-Based Timetabling- A Case Study," Applied Artificial Intelligence, 10, 439-453

[4] C. A. Oyeleye (2012), "Development of a Hybrid Model for Solving Examination Timetabling Problem," Unpublished PhD Thesis, Department of Computer Science, LAUTECH, Ogbomoso, Nigeria.

[5] Mushi A. R., (2007), "Simulated Annealing Algorithm for the Examination Timetabling Problem," AJST, Vol. 8, No. 2: December 2007.

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

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

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

[9] G. Kendall, N. Hussin (2004), "Tabu Search Hyperheuristic 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.

[10] G. White, B. Xie, S. Zonjic (2004), "Using Tabu Search with longer-term memory and relaxation to create examination timetables," *European Journal of Operational Research*, 153 pp. 80-91.

[11] L. Gaspero (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.

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

[13] H. Terashima, P. Ross, M. Valenzuela (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.

[14] P. Cowling, G. Kendall, N. Hussin (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.

[15] M. Carter, G. Laporte, J. Chinneck (1994), "Features for Contraint Programming," *Interfaces*, Vol. 24, 3, May-June, pp. 109-120.