Volume 4, Issue 9, September 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Development of a University Lecture Timetable using Modified Genetic Algorithms Approach

Alade O. Modupe, Omidiora E. Olusayo, Olabiyisi S. Olatunde
Department of Computer Science and Engineering, Ladoke Akintola University of Technology, Ogbomoso, Nigeria

Abstract: ***Lecture timetabling is a very important process in any educational institutions. It is an open ended problem in which courses must be arranged around a set of timeslots and venues so that some constraints are satisfied. It constitutes a class of difficult-to-solve combinatorial optimization problems that lacks analytical solution methods. The main objective of the research is to solve this problem by using modified Genetic Algorithms approach***
The implementation of the timetabling problem was characterized by crossover, mutation and selection scheme modifications with the introduction of replacement, test and repair strategy. In particular, single crossover, uniform crossover and rank based selection schemes were employed. The timetabling system was subjected to five different mutation rates: 0.02, 0.04, 0.10, 0.25 and 0.50. The results of the experiment revealed that best and feasible timetable was generated at the least considered mutation rate of 0.02 and fitness function of 0.95. None of the hard constraints were violated, so the timetable generated was feasible. Also, it was discovered mutation rate has greater effect on feasible timetable generation; no conflict of courses was experience when mutation rate was reduced to 0.02.

Keywords: Timetable, Lecture, Genetic Algorithms and Genetic Operators

I. INTRODUCTION

Genetic Algorithms (GAs) are adaptive methods which can be used to search, solve an optimization problem. GAs imitates the development of new and better populations among different species during evolution, just as their early biological predecessors [5]. Unlike most standard heuristic algorithms, they use information of a population of individuals (solutions) when they conduct their search for better solutions and not only information from a single individual. Its main application is in solving combinatorial optimization and in the development of parallel computers [1].

Timetabling belongs to the general category of scheduling problems. In scheduling, resources are allocated to certain processes or objects in a certain sequence. Examples of general scheduling problems are making a duty roster, project schedules, bus scheduling, air flight schedules, etc including the well-known Traveling Salesman Problem [2].

Automated timetabling, on the other hand, is a great task that saves a lot of man-hours work, to institutions and companies, and provides optimal solutions with constraint satisfaction within minutes. This has resulted in the improvement in the resource output of any organization in the following ways: boost productivity, quality of services and finally quality of life [5]. However, large-scale timetables, such as university timetables, may need great effort and many hours of work (days) spent, by a qualified person or a team, in order to produce high quality timetables with optimal constraint satisfaction and optimization of the timetable's objectives at the same time.

Timetabling in tertiary institutions can be divided into course and examination timetabling. Our interest is on the lecture timetabling problem which essentially entails the assignment of courses, rooms, students and lecturers to a fixed time period, normally a working week, while satisfying a given number of constraints. These constraints can be divided into hard and soft. Hard constraints must be satisfied, while soft constraints are to be satisfied as much as possible [3].

In the manual lecture timetabling, one of the major problems is dealing with clashes and finding clash free slots. Making a change requires that one has to undo previous lecture allocation and look for a new allocations. This creates a series of backtracks which are difficult to resolve. Timetabling has also been complicated due to many academic programs being run within the University system and insufficient number of lecture halls and spaces. To overcome these problems, the university system needs an automated and feasible timetable generator that satisfies all hard constraints conditions and as much as soft constraints highlighted.

In this work, we present a modified GA based method for solving university course timetabling problems. This method uses modified selection scheme, crossover and mutation strategy that is fully described in the next section. It also uses a replacement strategy, in order to avoid local optima, fulfill constraints and discover optimal solutions efficiently. The standard and the modified GAs are applied to the real-life university course timetabling problem from the Ladoke Akintola University of Technology (LAUTECH) Ogbomoso.

II. UNIVERSITY COURSE TIMETABLING

The timetable format for a single room is a two dimensional array as shown in Figure 1. The timetable for an entire university is therefore a collection of rooms' timetables- one for every room in the university.

In this research work, Faculty of Engineering and Technology, Ladoke Akintola University of Technology was used as a case study. Seven departments, 250 courses and ten lecture halls were used to implement the standard and modified GA algorithms. A timetable is essentially a schedule which must suit a number of constraints. Constraints are almost universally employed by people dealing with timetabling problems. Also, they are almost universally divided into two categories: soft and hard constraints [4].

Hard constraints are constraints that must be satisfied. In any working timetable, there must be no violation. In other words, they are constraints to which a timetable has to adhere strictly in order to be satisfied. For example, a lecturer cannot be scheduled in two places at once. An extensive list of hard constraints considered in this research work is as provided in Table 1

Soft constraints are constraints which may be violated, but of which the violations must be optimally minimized. For example, classes should be booked close to the home department of that class. Also, an extensive list of soft constraints considered in this research work is as provided in Table 2.

Table 1: List of Hard Constraints

S/N	Constraints
C1	No student can attend more than one lecture at the same time
C2	Lecturer cannot teach more than one course at the same time
C3	No room can occupy more than one lecture at the same time
C4	Fridays 13-14 hour and Wednesdays 15-17 hour slots are to be allotted for Muslim prayers and Sport respectively.

Table 2: List of Soft Constraints

S/N	Constraints
C5	As much as possible, minimize the use of early morning (6.00 a.m.) and late evening hours (20 – 22).
C6	Minimize continuous lectures/blocks of the same course in a day. It is preferred to spread them over the week as much as possible.
C7	As much as possible, evening lectures starting from 18 to 20 hours should be assigned to rooms with standby generators so as to minimize loss of lecture hours due to frequent power cuts.
C8	The number of students that attend the course for each lecture, must be less than or equal to the number of seats of all the rooms that host its lectures.
C9	The capacity of classrooms should match with student size.

III. GENETIC ALGORITHMS

Figure 2 represents the flowchart of standard genetic algorithms process. From the experimental work, it was discovered that there was violation of hard constraint (C4) when using standard genetic algorithms. The result was presented in table 4. As a result of this deficiency, a modified version of the standard genetic algorithms was developed to overcome this problem.

In order to solve this problem, modification was incorporated to the standard technique. The first thing considered was the representation method used to encode a timetable solution into an encoded form or chromosome, suitable for applying the genetic operators. According to the literature, 'direct' and 'indirect' approaches exist. A "direct" representation directly encodes all event attributes (day, time slot, teacher, room, etc.) for all events. Thus, in this case, the GA has to decide the appropriate slot for all timetable parameters and deliver a complete and constraint-free schedule. In "indirect" representations the encoded solution (chromosome) usually represents an ordered list of events, which are placed into the timetable according to some predefined method, or "timetable builder". In this work, "direct" representation that encodes 6 fields for each event into the chromosome was chosen: Lecturer, Course, Group, Room, Classroom and Timeslot. The descriptions of each parameter are enlisted below:

1. Lecturer Description: Each lecturer is described by two fields
   (i)identification number (id) (ii) name
2. Course Description: Each course is described by two fields
   (i) identification number (ii) name of the course
3. Room Description: Each room is described by three fields
   (i)name of the room e.g. LH (ii) lab (is it laboratory or not) (iii) size (capacity)
4. Group Description: Each group is described by three fields
   (i) identification number (ii) name (course code) (iii) size (number of students offering that course)
5. Class Description: Each class is described by four fields
   (i)professor number (ii)course no. (iii) duration (hour) (iv) group number
6. Slot Description: Each slot is described by four fields
   (i) Slot code (ii) Start time (iii) end time; and; (iv) name

The next consideration was the blend of genetic operators to incorporate into the GA method, in order to achieve optimal performance. To do this, we first considered standard operators as well as general purpose combinatorial operators. The operators considered in the research work are the single crossover and uniform crossover. Rank based selection scheme - mutation operator and replacement strategy - were also employed. A mutation operator was introduced to simulate the errors that may arise during the copy process. With a given mutation rate, the mutation operator was applied to each bit in the offspring chromosomes.

In figure 3, the flow diagram of the modified genetic algorithms process encompassed the encoding of an initial population created from a random selection of solutions (chromosomes). A fitness value was then assigned to each solution (chromosome); this depends on how close it is to solving the problem. The fitness value was employed to ascertain the extent an individual would be allowed to reproduce to the future generation. Genetic operators (single crossover, uniform crossover, rank based selection scheme) were applied to each population to generate a new population.

Replacement module was introduced to the genetic algorithm process where the newly generated population for a further run of algorithm would be used. The new population was evaluated based on modified selection scheme, crossover and mutation operators. The results produced were tested in test and repair module. If the end condition is satisfied then the program would stop and return the best solution in the current population else the algorithms would go back to replacement step and followed by evaluation step and looping continued until the conditions set were satisfied. A repair strategy was used in order to ensure that all classes appeared at least once per week. For robustness, this is done in two stages. Firstly, any classes which appear more than once in a day are (non deterministically) altered such that they appear at least once per week.

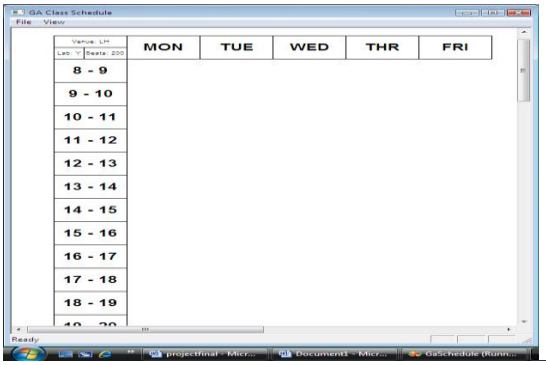

Figure 1 Example of a single room timetable

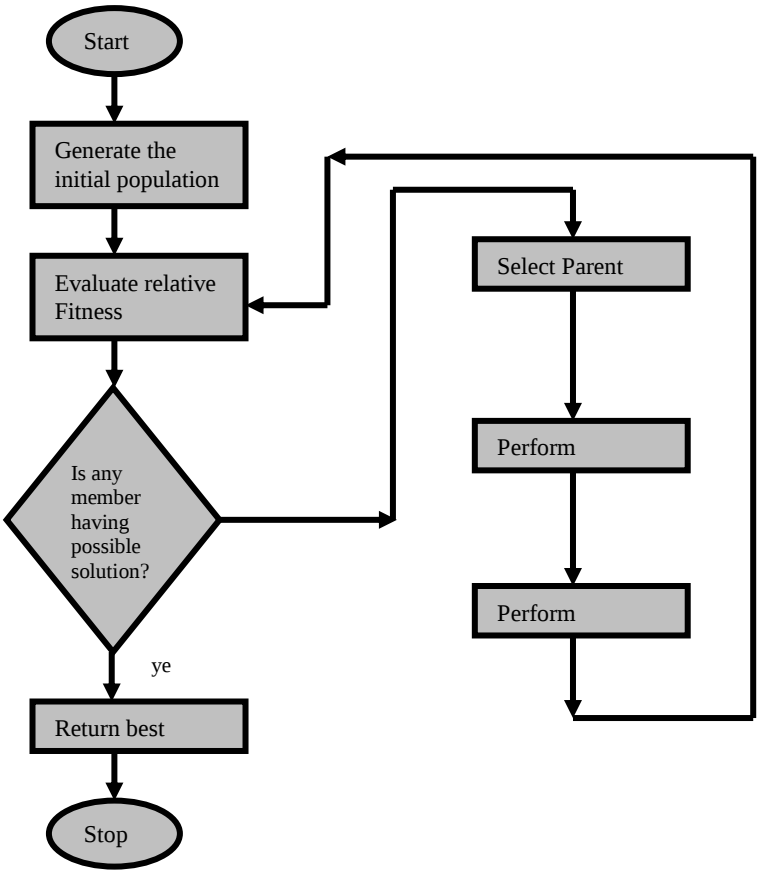

Figure 2: Flow Chart of the Standard Genetic Algorithm

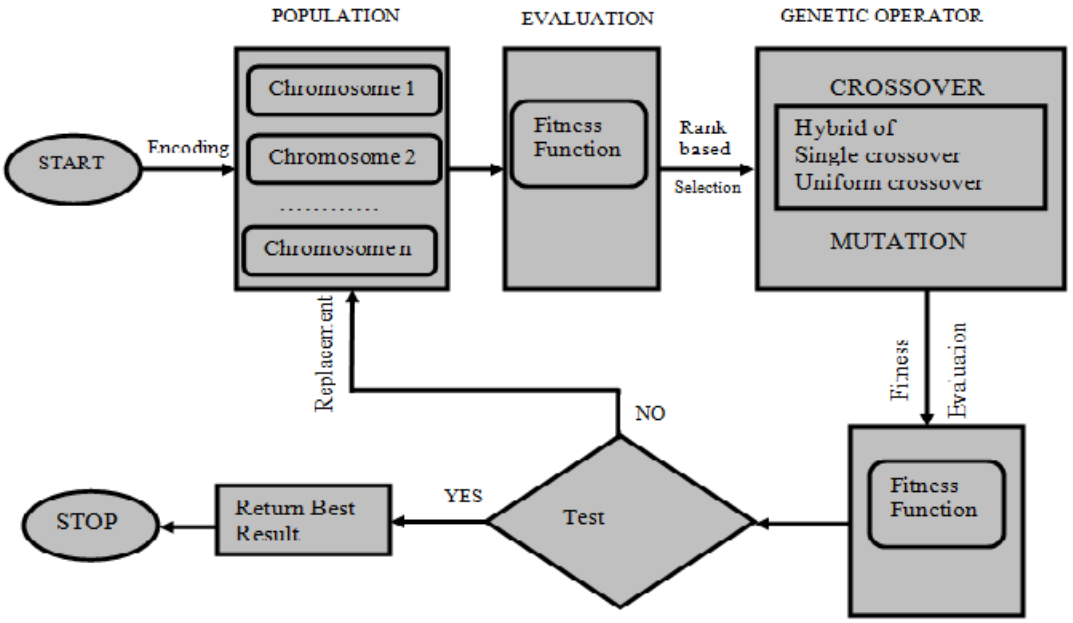

Fig. 3: Flow diagram of the Modified Genetic Algorithms process

IV. RESULTS AND DISCUSSION

The C++ programming language implementation of the two algorithms (Standard Genetic Algorithms and Modified Genetic Algorithms) was carried on two computer systems with different system specifications: Window Vista system with memory of 1015MB and 2.16GHz processor speed and Window7 system with memory of 4.00GB and 2.40GHz. Table 4.1 shows the computer execution of each algorithm on the two computers.

Table 3: Computer Execution Time

	Standard Genetic Algorithms (minutes)	Modified Genetic Algorithms (minutes)
Window Vista 1015MB, 2.16GHz	215.20	189.05
Window 7 4.00GB, 2.40Hz	86.10	61.16

Table 4: Constraints violation of both the standard and modified model

Constraint (C)	Standard model	Modified model
C_1	0	0
C_2	0	0
C_3	0	0
C_4	3	0
C_5	0	2
C_6	0	0
C_7	0	0
C_8	0	5
C_9	2	0

Note: Hard constraints = C_1 to C_4
Soft constraints = C_5 to C_9

Table 5: Calculated Metrics of the two Algorithms.

	n_1	n_2	N_1	N_2	PV	PE
Standard Algorithm	18	46	48	126	1429	57154
Modified Algorithm	25	101	70	135	1044	52200

PV = Program Volume
PE = Programming Effort

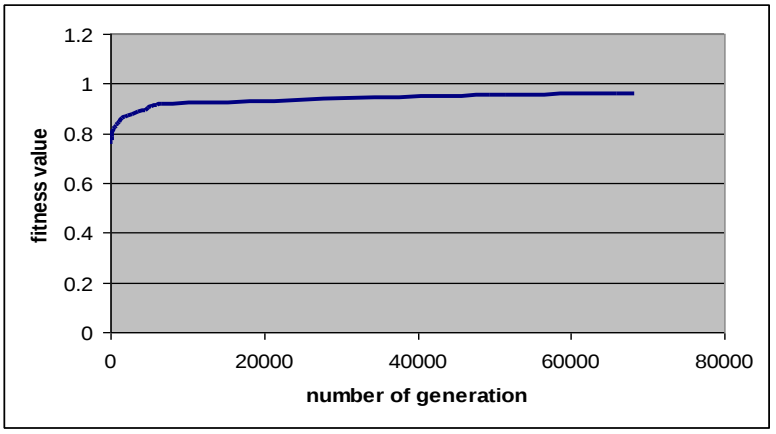

Fig. 3: Fitness value versus Number of Generations

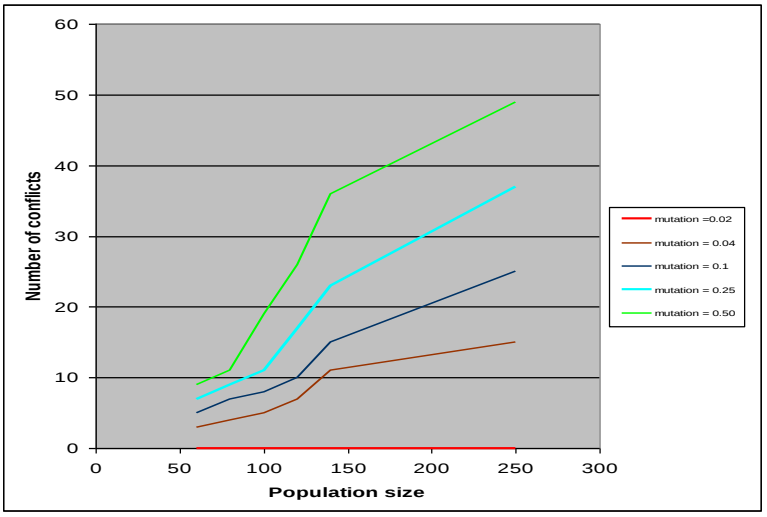

Fig. 4: Effect of different mutation rates on number of conflicts

In tables 3 and 4, the results obtained showed that the modified genetic algorithms are more efficient than the standard genetic algorithms. The performance of both techniques was carried out on two different systems (Window Vista 1015MB, 2.16GHz and Window 7, 4.00GB, 2.40Hz). The experimental results from table 4 showed that Modified genetic algorithms utilized less CPU time than the Standard genetic algorithm. Also, the complexity analysis of both the program volume and programming effort showed that modified genetic algorithms is less in value than standard genetic algorithms- table 5.

In table 4, there was violation of hard constraints. There were three instances of lectures being assigned on the University allotted free periods: Wednesdays (between 15 and 17hours) and Fridays (between 13 and 14hours). Since no hard constraint must be violated at all, the standard genetic technique was found not to be suitable for timetable generation. The modified genetic algorithms did not violate hard constraints at all; only two of the soft constraints were violated. These are two instances of lectures being assigned between 6 and 8 am, there are five instances of lectures where students' population exceeded the required capacity of the lecture room. This makes the modified genetic algorithms a better alternative over standard genetic algorithm.

In figure 4, different course timetables were generated using various mutation rates (0.02, 0.04, 0.10, 0.25, and 0.5). The results obtained (figure 4) revealed that were no conflicts experienced only when the mutation rate was reduced to 0.02. As the number of generation increases, the fitness value also increases until it reaches and stabilizes as the number of generation increases (i.e. 0.95 at a mutation rate of 0.02). From literatures, the closer the fitness value to 1, the better the quality of the timetable would be generated (figure 3).

VI. CONCLUSION

A modified genetic algorithms based technique was successfully developed to generate a conflict free, more efficient and effective lecture timetable with respect to all the considered metrics. The architectural framework of the modified genetic algorithms process could be grouped into three modules: generations of an initial population based on six main entities lecturer, course, group, room, class and timeslots. Evaluation module based on the modifications of roulette selection scheme, single crossover, uniform crossover and mutation rate; replacement, test and repair strategies were also employed. Generations of feasible and optimal lecture timetable were the basic feature of the last module.

REFERENCES

[1] Abdullah, S. and Turabieh, H (2008): Generating university course timetable using Genetic Algorithms and Local Search. The Third International Conference on Convergence and Hybrid Information Technology ICCIT, 1:254-260.

[2] Burke, E.K. and Newall J.P. (1999): "A Multi-Stage Evolutionary Algorithm for the Timetable Problem", the IEEE Transactions on Evolutionary Computation. 3(1):63-74.

[3] Burke, E.K. and Newall J.P. (2003): "Enhancing Timetable Solutions with Local Search Methods", The Practice and Theory of Automated Timetabling, Lecture Notes in Computer Science. Springer. 2740:195-206.

[4] Burke, E., Kingston J. and Werra, D. (2003a): Applications to Timetabling, Handbook of Graph Theory, Chapman Hall/CRC Press. 3: 445-474.

[5] Schaerf, A. (2004): A survey of Automated Timetabling. *Artificial Intelligence Review* 13(2): 87-127.

