International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12– No.19, February 2019 – www.ijais.org

Automatic Timetable Generation using Genetic Algorithm

Williams Kehinde Oladipo
Federal Polytechnic Ile-Oluji, Ondo State, Nigeria

Ajayi Olutayo Bamidele
ICT Resource Centre,
Federal University of Agriculture Abeokuta Ogun State, Nigeria

Ajinaja Micheal Olalekan
Federal Polytechnic Ile-Oluji, Ondo State, Nigeria

ABSTRACT
The generation of timetables has always been tedious right from time and apart from being tedious, the timetable created has always been filled with series of errors and mistakes. So many techniques have been put forward to solving this problem. In this paper, genetic Algorithm was used by creating a group of time series randomly from a given time and courses in other to find a solution to the timetable problems. The courses thus formed are evaluated with the help of the evaluation function. system administrator logs into the system and then the administrator input the courses with their codes and the unit. At that point, the admin will keep adding until the number of courses needed has been inputted. The admin can remove a course that has been inputted in the case of error. After inputting the courses, it moves to the next page where all the lecture halls or rooms that will be used will be inputted. After inputting these, the system then generates the timetable system. This technique (genetic algorithm) used helps in reducing to barest minimum, errors and mistakes in encountered in developing an automatic timetable.

General Terms
Automatic, timetable, generation, genetic, algorithm;

Keywords
Automatic; timetable; generation; genetic; algorithm;

1. INTRODUCTION
The class timetabling problem is a scheduling algorithm with great interest and implications in the fields of operational research and artificial intelligence. The problem was first studied by Gotlieb, who formulated a class-teacher timetabling problem by considering that each lecture contained one group of students and one teacher, such that the combination of teacher and students can be chosen freely. The class timetabling problem is a typical scheduling problem that appears to be a tedious job in every academic institute once or twice a year (Datta *et al*., 2005).

Timetabling is known to be a non-polynomial complete problem i.e. there is no known efficient way to locate a solution. Also, the most striking characteristic of NP-complete problems is that, no best solution to them is known. Hence in order to find the solution to a timetabling problem, a heuristic approach is chosen. This heuristic approach, therein, leads to a set of good solutions.

In a general educational timetabling problem, a set of events (such as courses and exams) are assigned into a certain number of timeslots (time periods) subject to a set of constraint, which often makes the problem very difficult to solve in real-world circumstances (Burke *et al*., 2007 pg 177-192). Timetabling is the task of creating a timetable while satisfying some constraint.

The aim of this paper is to present the generation of course schedules while demonstrating the possibility of building schedules automatically through the use of computers in such a way that there are optimal and complete with little or no redundancy through the development of a viable lecture timetabling software. The main objective is to be able to optimize the algorithm used in recent timetabling systems in order to generate the best timetables with fewer or no clashes. The second objective is to expand the scope of timetable automation systems by making it generic thereby bringing about uniformity in the creation of timetable as it applies to different universities or educational institutions. The software possess an attractive graphical interface which will help interaction between the user and the computer and also improve productivity.

2. RELATED WORKS
Srinivasan, Tian and Jian (2002) worked on an automated time table generation using multiple context reasoning for university modules. The paper presented an evolutionary algorithm (EA) based approach to solving a heavily constrained university timetabling problem. The approach uses a problem-specific chromosome representation. Heuristics and context-based reasoning were also used for obtaining feasible timetables in a reasonable computing time. An intelligent adaptive mutation scheme was employed for speeding up the convergence. The comprehensive course timetabling system presented in the paper was been validated, tested and also discussed using real world data from a large university.

Tahir et al (2004) worked on dynamic Timetable Generation Conforming Constraints. In the paper, the authors proposed an approach that solves the timetabling problem. The approach takes into account many constraints including allocation of room, teacher, course, time slot etc. the algorithm builds the timetable in an incremental manner, dynamically adjusting resources in order of complexity. The algorithm proposed is dynamic in nature. It deals with managing certain constraints as input, then using heuristic approach to scanning all the constraints on priority basis. The sequence of checking of constraints is also dynamic in nature. Though this sequence of constraints can also be altered manually. The proposed algorithm is based on heuristic algorithm. The algorithm takes values as input and manages the constraints and resource scheduling one by one. The main features of the algorithm are as generation of intermediate level as well many final reports including weekly time table, teacher timetable, room wise timetable, student time table, department level time table and

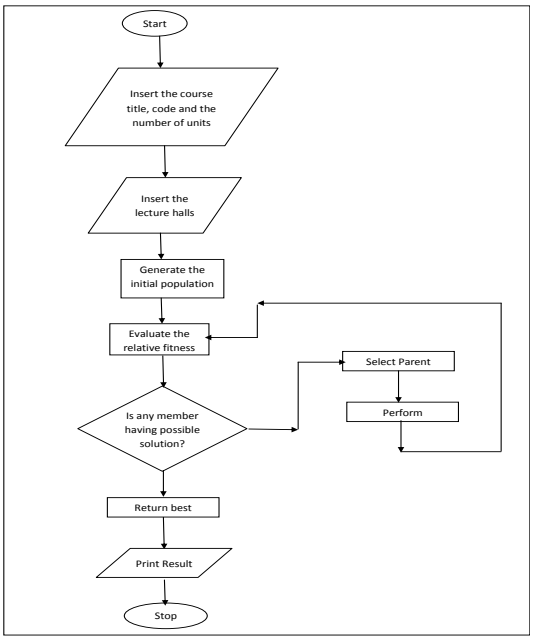
distributing workload of lectures equally among all the specified time slots.

Barkha Narang, Ambika Gupta and Rashmi Bansal (2012) worked on use of active rules and genetic algorithm to genetic automatic timetable. In this paper, the authors proposed an optimized technique to automate timetabling generation system. The proposed technique filters out the best of active rules and uses Genetic algorithm to generate an optimized solution that accommodates various complex constraints such as that for faculties, classrooms, labs etc. The proposed paper takes four parameters as input: person – name of lecturers, subjects – name of courses in the class, room – name of classes and capacity of each and time interval – starting time and the duration.

The genetic algorithm selects the action to be executed. When an event occurs, the system has several actions that it can choose to perform. For each possible event, there is another ordered set of possible actions that can be performed when that event occurs that needs to be maintained. The genetic algorithm always selects the first action initially, but a genetic algorithm running in parallel may dynamically change the order of the actions. The paper explained how the authors have used a set of active rules to express the knowledge of intelligence and how a genetic algorithm can be used to dynamically prioritize rules. The advantages of this approach are: distributed solution, load balancing, and fault situations. They help in optimizing the timetable generation solution.

3. METHODOLOGY

The tools used for the design and methodology include Algorithm and Flow chart. From the flow chart below, the steps carried out by the system administrator are as follow: The system administrator logs into the system. The administrator then input the courses with their codes and the unit. At this point, the admin will keep adding until the number of courses needed has been inputted. The admin can remove a course that has been inputted in the case of error. After inputting the courses, it moves to the next page where all the lecture halls or rooms that will be used will be inputted. After inputting these, the system then generates the timetable system. Figure 1 shows the flow chart;



3.1 Constraints used in generating the genetic algorithm

- Only one lecture is attached to one lecture room.
- Every class must be scheduled exactly once.
- Classes of students must not have two bookings simultaneously.
- A space should be left after the first three periods for break or recess.

3.2 Implementation of design

The following are the hardware requirement of the system that the design can be implemented upon.

- Hard disk space of not less than 500 megabyte
- Pentium II intel processor.
- RAM of not less than 125 megabyte

The design can be implemented on any operating system starting from windows vista. XAMPP server should be installed. Internet browser such as Google Chrome, Internet Explorer, Microsoft Edge, Opera Mini, UC Browser and so on. The following tools were used in the course of developing the system.

- Sublime test 3
- XAMPP

The programming languages used include

- HTML
- PHP
- MYSQL

3.3 Program Modules and Interfaces

Login section: This is the section whereby the administrator logs in in other to be able to have access to inputting the details needed for the timetable generation. This details include the course title, course code, course unit, lecture hall and so on.

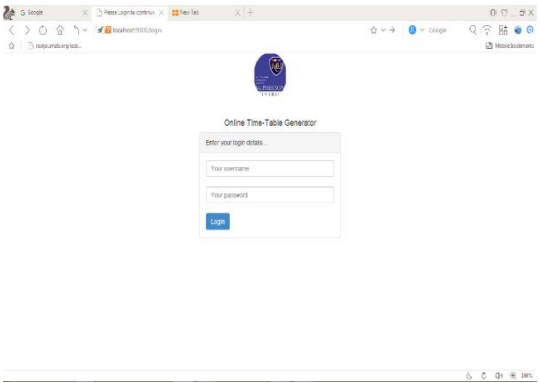


Fig 2: the login interface of the system

Course Input Section: This is the point where the administrator inputs the courses to be generated in the timetable. In this section, the course title is being typed, then the course code and the unit. After inputting all of these, the add button is being clicked. The add button helps to accumulate all the courses to be inputted.

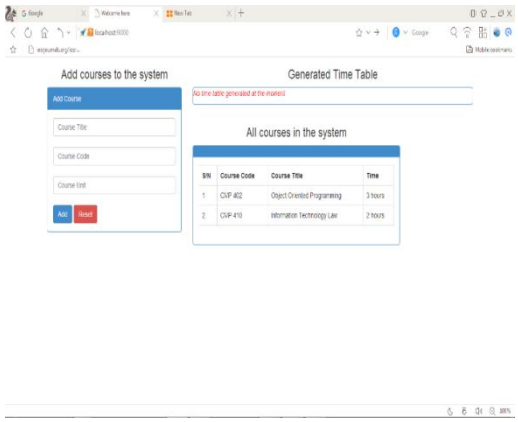


Fig 3: Course input section

4. CONCLUSION

Genetic Algorithm is one of the most effective ways of generating timetables although it does not give a 100% optimal timetable. Its degree of optimality depends on the constraints used and also the fitness function of the parameters. Using Genetic Algorithm is a little slower due to the steps it has to undergo. Generating random numbers, mutating and crossing over parameters before the final result is gotten. Notwithstanding, it still remains the best way to solve timetable issues.

5. RECOMMENDATION

I recommend that in order for Genetic Algorithm to be implemented, the constraints to be used should be chosen carefully in order to get an optimal solution. Although there are research publications concerning genetic algorithm on the internet, the implementation is difficult. Due to its complexity, Genetic Algorithm should be taught as a course for Computer Science students. This will enable students have better understanding of Genetic Algorithm.

6. REFERENCES

[1] Abramson, D. (1991). Constructing Schools Timetables Using Simulated Annealing: Sequential and Parallel Algorithms Management Science.

[2] Adejuwon, O. S. (2012). Development of a University timetable automation system.

[3] Anisha, J., Ganapathy, S. C., Harshita, G., & Rishabh, B. (2015). A Literature Review on Timetable generation algorithms based on Genetic Algorithm and Heuristic approach . International Journal of Advanced Research in COmputer and Communication Engineering , 159-163.

[4] Back, T. (1995). Evolutionary Algorithms in theory and practice.

[5] Baricelli Nils Aall. (1957). Symbiogenetic evolution preocesses realized by artificial methods. 143-182.

[6] Barkha, N., Ambika, G., & Rashmi, B. (2013, July). Use of Active Rules and Genetic Algorithm to Generate the Automatic Timetable. International Journal of Advances in Engineering Sciences Vol. 3 .

[7] Barricelli, N. A. (1963). Numerical testing of evolution theories. Part II. Preliminary test of performance, symboigenesis and terrestial life , 99-126.

[8] Bentley, L. D. Systems Analysis and Design for the Global Enterprise (7th Edition ed.).

[9] Buckles, B. P., & Petry, F. E. (1992). Genetic Algorithms.

[10] Bunday, B. D. (1984). Basic Optimization Methods Edward Arnold.

[11] Chamber, L. (1995). Practical Handbook on Genetic Algorithms: Appplications.

[12] Crosby, J. (1973). Computer Simulation in Genetics. London: John Wiley & Sons.

[13] Dikmann, R., Luling, R., & Simeon, J. (1993). PProblem independent distributed simulated annealing and its applications.

[14] Feiring, B. R. (1986). Linear Programming 60: An Introduction Publication .

[15] Forgel, D. (1998). Evolutionary Computation: the Fossiil Record . New York: IEEE Press.

[16] Fraser, A., & Burnell, D. (1970). Computer Models in Genetics. New York: McGraw-Hill.

[17] Glover, F. (1997). A Template for Scatter Search and Path Relinking. Lecture Notes in Computer Science .

[18] Glover, F., & Laguna, M. (1997). Tabu Search.

[19] Goldberg. (1989). Genetic Algorithm in Search, Optimization and Machine Learning.

[20] Makeower, M. S., & Williamson, E. (1975). Operational Research-Teach Yourself Books.

[21] Paulli, J. (1993). Information utilization in simulated annealing and tabu search.

[22] Rechenberg, I. (1973). Evolution Strategie. Stuttgart: Holzmann-Froboog.

[23] Sanchez, A., Shibata, T., & Zadeh, L. (1997). Genetic Algorithms and Fuzzy Logic Systems.

[24] Sharma, D., & Chandra, N. (1999). An evolutionary approach to constraint-based timetabling.

[25] Taha, T. R. (1987). Numerical schemes for nonlinear evolution equations. The College Jornal of Science and Technology (Jerusalem) .

[26] Tahir, A. M., Hikmat, U. K., & Sajjad, S. (2012). Dynamic Time Table Generation Conforming Constraints a Novel Approach. ICCIT .

[27] Turing, A. (1950). Computing machinery and Intelligence. 433-460.

[28] Ulrich, & Eppinger. (2016, May 27). System Design. Retrieved May 27, 2016, from wikipedia: https://en.wikipedia.org/wiki/Systems_design

[29] Zoints, S. (1974). Linear and Integer Programming.