Design and Implementation of An Automatic Examination Timetable Generation and Invigilation Scheduling System Using Genetic Algorithm

Abdulaziz Aminu
Department of Computer Science
Yusuf Maitama Sule University
Kano, Nigeria
abdulazizaminu029@gmail.com

Wahyu Caesarendra
Faculty of Integrated Technologies
Universiti Brunei Darussalam
Brunei Darussalam
wahyu.caesarendra@ubd.edu.bin

Umar S Haruna
Department of Computer Science
Yusuf Maitama Sule University
Kano, Nigeria
shafsonharun@gmail.com

Abubakar Sani
Department of Computer Science
Yusuf Maitama Sule University
Kano, Nigeria

Mansur Sa'id
Department of Computer Science
Yusuf Maitama Sule University
Kano, Nigeria

Daniel S Pamungkas
Mechatronics Departement
Politeknik Negeri Batam
Batam, Indonesia
daniel@polibatam.ac.id

Sumantri R Kurniawan
Mechatronics Departement
Politeknik Negeri Batam
Batam, Indonesia
sumantri@polibatam.ac.id

Endang Kurniawan
Departement of Information System
Pesantren Tinggi Darul 'Ulum Univ.
Jombang, Indonesia
kurniawan1911@gmail.com

***Abstract*— University timetabling is a highly constrained problem that requires considerable time and effort in the process. Stakeholders face a severe dilemma when coming up with a timetable manually as numerous clashes may arise. This necessitates the need for devising an efficient automated means of getting rid of the hectic and heavy tasks involved in examination timetable and Invigilation scheduling. Yusuf Maitama Sule University Kano is of no exception as several clashes were reported to examination office before the commencement of examinations. This paper proposed a solution to the problem of examination timetable and invigilation scheduling in Yusuf Maitama Sule University Kano, Nigeria. The paper employed Genetic Algorithm (GA) that provides an optimal solution by randomly and effectively selects a time slot for a particular course; a venue for such a course; and invigilators that suits particular teacher-student ratio of 1:40. The solution is implemented in Java with its Graphical User Interfaces (GUIs) designed using Java Swing and also use SQLite for database related activities (reading and writing of data).**

Keywords*— *examination timetable, genetic algorithm, invigilation scheduling, java

I. INTRODUCTION

The act of time-wise scheduling of several events for future occurrence is generally referred to as timetable. Furthermore, the timetable can also be defined as an organized and tabulated list of events that provides sufficient information about them. The term scheduling refers to the allocation of several resources to a particular task while considering some certain constraints. On the same note, [1] defined scheduling as the "allocation of resources over time to perform a collection of tasks, while taking all constraints into account, so that input demands are met in a timely and cost-effective manner. Considering this definition, a reliable and standard timetable must meet several requirements and should also satisfy the desires of all parties involved as much as possible, with the view of maintaining its credit as being part of the instruments for ascertaining the success of any given institution.

Timetabling is an NP-hard optimization problem that requires a right and optimized solution among a set of complex variables and constraints. The optimization problem assigned a set of feasible tuples that optimizes a set of metrics and indicators such as: minimizing gaps between time slots; minimizing space utilization; minimizing cost relevant to the use of resources and so on [2]. However, finding an efficient solution to such a problem is quite tricky especially when the number of variables becomes larger. However, [2] asserted that a solution to such problems could be provided by having a clear understanding of hardness and simplicity of the problem.

Further considered timetabling as a constraint satisfaction problem that requires satisfy solution is presented in [3]. These constraints can be categorized into hard and soft ones. Soft constraints are those constraints that if violated, the resulting schedule is still valid but affects the accuracy of the scheduling as a whole, whereas violation of hard constraints leads to invalidation of the entire scheduling process. Hence, hard constraints specification are inevitable to the achievement of a proper timetable, because their violation disqualifies it from being clash-free.

Academic institutions (high schools, colleges, universities) have varieties of a timetabling problem that needs to be scheduled for future occurrence. The resources to be coordinated in a school timetable are students; teachers; rooms, and time slots. Other factors include the subject of the class, the type of available rooms. The organization and the structure of a timetable depend on the nature of the timetabling problem.

Universities, like other academic institutions, need a proper allocation of resources to several entities. These resources that include people, venue, and time require a proper allocation by making a careful decision by gathering accurate information. University timetabling problem as defined by [4], is the "task of assigning several events such as lectures, examination, meeting and so on, to a limited

978-1-7281-2807-8/19/$31.00 ©2019 IEEE

timeslot and venues such as classrooms, exam rooms, and laboratories in accordance with a set of constraints. University timetabling problem has many variations that include meetings, supervision, lectures, examinations and so on. However, the problem can be more generally categorized into examination and lecture problems. Course timetabling problem is the problem of providing a weekly schedule of university courses to a set of lecturers, thereby minimizing overlaps concerning the clash of lectures among students. As earlier stated, the mode of lectures in university is entirely different from high school timetabling because university timetabling requires more human judgment than the others.

Similarly, examination timetabling problem entails scheduling of examination for a set of university courses, thereby minimizing or avoiding overlap for courses having ordinary students, and spreading the exams for the students as much as possible. In many universities, examinations were conducted at the end of each semester for the whole students at a specific interval of time, i.e. when constructing an examination, a specific duration of time is assigned in which the entire courses should be scheduled within that time frame.

Most academic institutions adopt the style of hand operation when it comes to the generation of timetable although most of their activities were automated. The traditional hand operation method of timetabling is a tedious and time-consuming task that ends up with an inaccurate schedule with several clashes in several resources such as time slots and venues. Using the manual timetable creation for a small number of courses and students may create an optimal timetable but as the problem resources increase (students, courses), generating a timetable for such a problem manually will become near impossible. This necessitates the need for devising an efficient automated means of getting rid of the hectic and onerous tasks involved in examination timetable scheduling.

Furthermore, it is worth noting that scheduling can significantly affect students' performances during examinations. Automatic timetable generation system is any web-based or standalone tool that is capable of generating a well-organized timetable based on some certain constraints after feeding it with all the required inputs (e.g. rooms, courses, etc.).

Issue of time allocation to a group or organization has been problematic in nature. Manually justifying time and venue to be allocated to a particular group or person can take considerable amount of time and effort. Still with this effort manual preparation may end off with clashes in the schedule. In this paper, we use Genetic Algorithm (GA) to develop a standalone application for automatic generation of examination timetable that will liberate lecturers from the laborious and difficult tasks involved in coming up with examination timetable manually.

REVIEW OF RELATED WORKS

Many algorithms were used to develop tools that aid various organizations to automate their timetabling scheduling. The following paragraph will review some of the methods that were used to automate the task of timetabling.

Ref. [10] used firefly algorithm which is also a bio-inspired algorithm to develop a university course timetable with the main aim of reducing the inconveniences in terms of speed and effectiveness that are associated with other searching techniques. Their solution is found to be of satisfaction to both teachers and students as an optimal timetable is generated.

Ref. [11] proposed tabu search to provide an automated means of generating examination timetable. Candidate solutions are generated randomly, and these solutions are optimized in such a way that the search space is explored. The system is tested using the datasets of the faculty of engineering at the University of LAUTECH, Nigeria. The datasets consist of 153 courses, 15 venues to be scheduled for about 25 days. It was shown that the timetable was generated and the constraints are satisfied.

Ref. [13] used the combination of genetic and greedy algorithms to automate the task of examination timetabling in the Arab East College for High Education in Saudi Arabia. The two algorithms were combined together to generate an optimal schedule. The genetic algorithm is populated with a set of configurable timetables and in turn, reproduce for an approximate (2000) number of generations to produce the desired optimal timetable.

Ref. [14] studied simulated annealing (SA) and GA and evaluated their performance measures in solving examination timetabling problem. The performance of the two approaches was evaluated based on their computing resources. It has been shown that substantial computing resources were used by GA in contrast to SA. Meanwhile, a hybrid implementation of the two algorithms was proposed to combine their strengths thereby eradicating their weaknesses.

II. GENETIC ALGORITHM

Genetic algorithm (GA) was first described by John Holland in the 1960s and with the help of his students and colleagues at Michigan University, the work is further developed in the between the 1960s and 1970s with the aim of understanding the phenomenon of adaptation as it occurs in nature as well as to develop ways in which mechanisms of natural adaptation might be imported into computer systems. Genetic algorithm is considered to be capable of finding reasonable solutions to complex problems as they are highly capable of providing solutions to optimization problems (i.e., constrained and unconstrained). It makes use of techniques inspired by biology which includes selection, chromosomes, crossover, and mutation to solve a given problem [5]. GA has general steps that are to be followed when implementing it in a particular programming language as shown in Figure 1. The operators of Genetic Algorithm are:

I. Population Initialization: The first step starts by initializing randomly generated population. Each member of this population is an abstraction of a possible solution to a problem. Every member of the population is assigned a fitness value which is rated according to the fitness function. The success of this algorithm depends on the goodness of the initial population and the population size [3].

II. Selection: it selects some chromosomes in the population to undergo recombination using a specified strategy. The chromosome with better fitness value has the highest probability of being selected for reproduction than the others [3].

III. Crossover: After deciding the parent chromosomes to be selected, the next step is to combine two or more chromosomes to produce new offspring's [6]. The simplest way to do this is to choose randomly some crossover point, copy everything before this point from the first parent, and then copy everything after a crossover point from the second parent. There are several version of crossover operators that are generic in nature so that the programmer can choose the desired operator.

IV. Mutation: Mutation may be defined as small as well as a random alteration in an individual after undergoing crossover to produce new offspring [7]. However, the probability mutation is considered to be small (i.e., around 1 to 5%) because diversity is needed to be maintained. As opposed to crossover, the mutation is crucial to the convergence of a genetic algorithm [7].

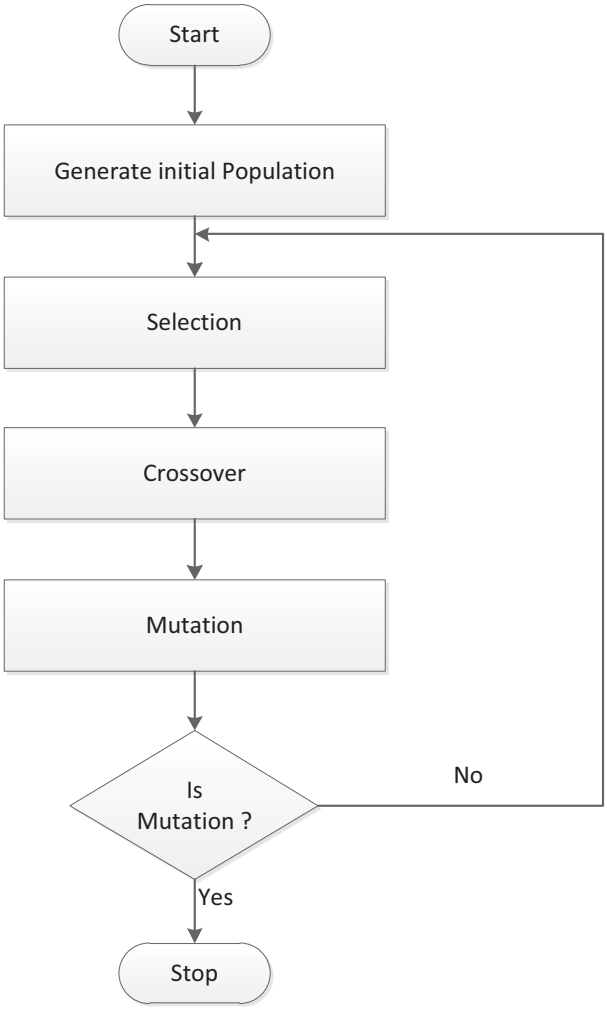

Fig. 1. Working of a genetic algorithm

III. PROBLEM DEFINITION

The subjected timetabling problem can be described with the help of the following interpretation. In the problem, there is a finite set of courses C = $\{c_1, c_2, \ldots, c_k\}$, a finite set of rooms in which an examination can be conducted R = $\{r_1, r_2, \ldots, r_\ell\}$, a set of invigilators I = $\{i_1, i_2, \ldots, i_n\}$ and a finite set of potential time slots for the exam T = $\{t_1, t_2, \ldots, t_m\}$. This timetable schedule is an ordered 4-tuple (a, b, c, d) were a ∈ C, b ∈ R, c ∈ I and d ∈ T with an interpretation of "an examination for course a starts at time d in room b with invigilators c".

The algorithm will use the given set of variables (C, R, I, T) to generate a schedule that satisfies some set of constraints. These constraints can be classified as hard and soft constraints, some of the hard constraints that must be satisfied by the algorithm in this project work include:

I. Each course should be scheduled as a single examination.

II. Each venue should be enough to accommodate all students in a particular course.

III. No two exams should be allocated a particular room at the same time.

IV. An invigilator must not be assigned to two examinations at the same time.

V. A student must not have two examinations at the same time.

The algorithm will generate a population of candidate solutions called chromosomes. Each chromosome will be rated by a fitness function as shown in equation (1) which will take consideration from the above constraints. From there individuals with better fitness values be selected to undergo recombination using crossover and mutation. This process will be repeated until an individual with a fitness value of 1 is evaluated that leads to the generation of a clash-free examination timetable.

$$\boldsymbol{fitness(i) = 1/(n+1)} \tag{1}$$

$$\boldsymbol{where\ i = individual, n = number\ of\ clashes}$$

In this study, the object-oriented design approach has adopted a focus on data which is intended to be manipulated in the program. Object-oriented analysis and design methods are becoming the most rated and widely used method for computer systems design. The Unified Modelling Language (UML) that was proposed by Object Management Group (OMG) in 1997 is considered to be the standard language in object-oriented analysis and design [12]. The use case diagram in Figure 2 shows the basic interaction between the user and the system.

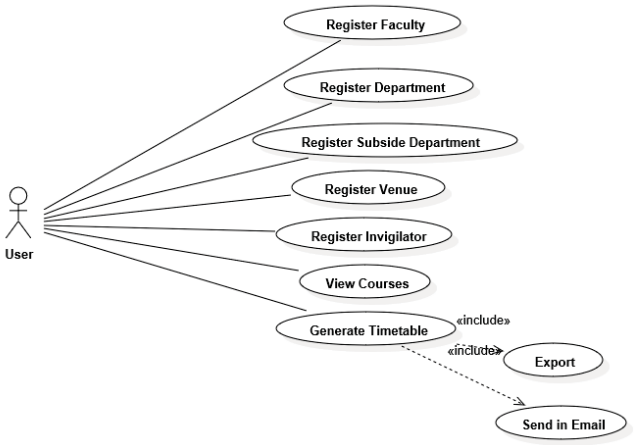

Fig. 2. Interaction between user and the system

IV. Functional Components of the Timetabling Tool

The system comprises of eight modules as shown in Figure 3. In the faculty module, the user registers the available faculties in the university. Also, the user is expected to register departments and courses that belong to a particular faculty. If a course is not only offered by the department that belongs to it, the user registers those departments as subsidiary departments to the course. Examination venues, as well as invigilators, can be registered in their respective module and invigilators are assigned to each exam the teacher to student ratio of 1:40. In the generate timetable module (Figure 4), the user can set the exam duration, choose the desired semester, and click the generate button and wait for a while for the timetable generation.

When the timetable is generated successfully, the user can export the result of the generated timetable in either Excel (Figure 5) or pdf format to any directory in his/her computer. Besides the format timetable can also be sent as email attachment to a specified email address.

The system is tested and implemented in the above-mentioned university using its real data (courses, rooms, and invigilators) and it was proved to be an ideal solution to the problem associated with the manual system. The constraints in which our work is built upon as listed in section IV were efficiently handled as none of it is violated that leads to a generation of a clash-free and customized examination timetable. The sample data that was used comprises of about 106 and 75 courses for first and second semester respectively, and as such, a clash-free examination timetable is generated that will be conducted for a minimum of 18 days with three sessions for morning, afternoon and evening as shown in figure 5.

Fig. 3. System Dashboard showing various modules

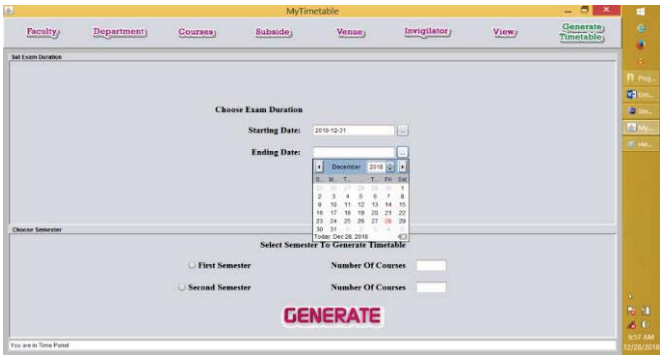

Fig. 4. Module in which timetable is finally generated

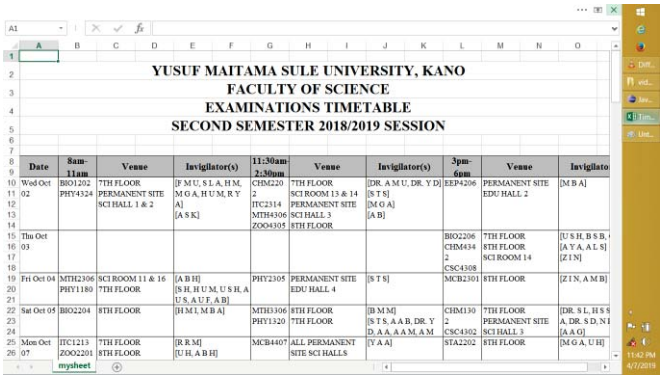

Fig. 5. Generated timetable in excel

V. Conclusion

Timetabling is a core for running any organization that involves more than one staff or members. The schooling system is primarily dependent upon the rotation of teachers for a particular time and place. Several courses designed by a particular system requires acute permutation of the courses and venue. This paper has concentrated on designing a system that solved the timetabling problem of Yusuf Maitama Sule University Kano, Nigeria. The solution is implemented in Java with its Graphical User Interfaces (GUIs) designed using Java Swing. In addition, the application uses SQLite relational database management system for database related activities (reading and writing of data). It has been shown that developed system has remedied the weaknesses of the existing system in the

university in such a way that no clash is reported when the timetable is generated using the developed system.

However, in the future, we envisage coming up with the web and Android versions of it to be able to incorporate the concepts of invigilation reminders to lecturers (invigilators) as well as generating a customized version of invigilation schedules for each lecturer.

REFERENCES

[1] Nashwan A. A & Talal H. A., 2016, *International Journal of advanced Research in Computer and Communication Engineering,* "Solving of Lectures Timetable Problem and Automatic Timetable Generation Using Genetic Algorithm," vol. 5, no. 9, pp. 505-512.

[2] Esraa A. A. & Ghada A. E., 2016 *Alexandria Engineering Journal,* "A Utilization-based Genetic Algorithm for Solving the University Timetabling Problem," vol. 5, no. 5, pp. 1395-1406.

[3] Dipesh M., Rajesh K. S. and Satyabrata D., 2015 *International Journal of Advanced Research in Computer and Communication Engineering,* "Automatic Timetable generation using Genetic Algorithm," vol. 4, no. 2, pp. 245-248.

[4] Akinrinwa, S. P., Olabode, O., Oluwadare, S. A., Adesuyi, A. T., 2015 *2nd International Conference and Exhibition (OESD-FUTA),* "Decision Support System for Examination Timetabling," vol. 8, no. 7, pp. 439-444.

[5] Brian Mc Ginley, 2015 *IEEE Transaction on Evolutionary Computation,* "Maintaining healthy population diversity using adaptive crossover, mutation and selection," vol. 8, no. 5, p. 692.

[6] Stuart Russel & Peter Norvig 2010, Artificial Intelligence a modern approach, Third Edition ed., New Jersey: Prentice-Hall.

[7] TutorialsPoint 2016, Genetic Algorithms Tutorial, 2nd Edition ed., India: Tutorials Point.

[8] Chaya Andrade & Saminda Premaratne 2016, *International Journal of Modern Research in Engineering and Technology,* "Utilization of Timetable Management System to a Medium Scaled University," vol. 1, no. 2, pp. 9-16.

[9] Tutorials Point 2015, Unified Modelling Language Tutorial, India: Tutorials Point pvt Ltd.

[10] Deeptimanta O., Rajesh K. S. and Satyabrata D. 2016, *International Journal of Advanced Research in Computer Science and Software Engineering,* "Automatic Generation of Timetable Using Firefly Algorithm", vol. 6 no. 4, pp. 589-593.

[11] Hamid D. L., Ibrahim A. A., Elijah O. O., Oladiran T. A. and Oluyinka I. O., 2014, *International Journal of Scientific & Engineering Research,* "University Examination Timetabling Using Tabu Search", *vol.* 5 no. 10, pp. 785-788.

[12] Johan J. & Eric N., 2016. *Investigating a Genetic Algorithm-Simulated Annealing Hybrid Applied to University Timetabling Problem,* Stockholm: Kth Royal Institute of Tchnology, SChool of Computer Science and Communication.

[13] Ali S., Mohammad F. and Hadi S. A. 2017, *Ninth International Conference on Advanced Computational Intelligence,* "Exam Scheduling: A Case Study," pp. 137-142.

[14] Oyeleye C. A, Omidiora E.O., Olabiyisi S.O. and Oladosu J.B. 2012, "Performance Evaluation of Simulated Annealing and Genetic Algorithm in Solving Examination Timetabling Problem", vol. 7, no. 17, pp. 1727-1733.