

Automated Academic Schedule Builder for University's Faculties

Mohamed Abdelfattah and Ahmed Shawish

Abstract—Scheduling has been defined as the allocation of resources to a certain object in a given time and space under a given set of constraints. Scheduling is based on more than one dimension including the availability of instructors, teaching assistants, students, special labs, and others at a certain time. That expansion turns the scheduling process to a very hard and complex problem. Research has gone into creating an automated timetable scheduler using different algorithms, mainly the Graph coloring and Genetic algorithm, and has managed to succeed but under certain conditions. This problem requires a robust solution especially for the university's faculties that share the same resources as it is a redundant process that takes place every semester. This process has an increasing complexity due to the natural growth of the students' population. In this paper, we introduce a new automated schedule builder that utilizes the genetic algorithm to produce an optimal timetable for each faculty within a university. The timetable is treated as a chromosome with a 3D view (time, day and session), where a uniform crossover method is adopted to find the best combination and mutation to be applied in order to produce a good population. In our fitness function, we pass the new population to each generation in order to select the best chromosome at the time. The proposed application produced a nearly optimal timetable that is conflict free and only requires minimum modification to satisfy all requirements.

Index Terms— Genetic Algorithm, Graph Coloring, Timetabling, Scheduling, Automated Scheduling

I. INTRODUCTION

Scheduling is the allocation of resources to objects in a given space and time with certain constraints to minimize cost and to fulfill a group of constraints [6]. Scheduling has become a large aspect in our everyday life and is applied in various fields in order to organize the process of everyday work.

Large institutions face a great amount of difficulty in preparing schedules due to the rapid increase of students and courses, which is not directly proportional to the available resources; hence scheduling becomes a great issue that is time consuming as well as challenging with respect to adhering to all relevant institutions. In the early days, univer-

Manuscript received March 05, 2016; revised March 26, 2016. This work is supported by the faculty of Informatics and Computer Science, The British University in Egypt.

Mohamed Abdelfattah is with the faculty of Informatics and Computer Science, The British University in Egypt, El-Sherouk, Egypt (e-mail: mohamed.abdelfattah@bue.edu.eg)

Ahmed Shawish is with the faculty of Informatics and Computer Science, The British University in Egypt, El-Sherouk, Egypt (e-mail: ahmed.gawish@bue.edu.eg)

-sites could schedule timetables manually with ease because the number of students was yet small and the resources were at the disposal of any number of students; and hence able to avoid any overlap in terms of lecturers, teaching assistants, classrooms and courses. Now with the increasing number of students and courses, resources have become limited and manually creating timetables has become a long and difficult process as a consequence of the need to satisfy all constraints with regards to faculty members and students. Hence, this process becomes a Nondeterministic Polynomial-time (NP)-hard problem.

Previous work was dedicated to creating conflict free timetables that ended up fitting the environment in question by using their own campuses as test cases. In that sense, each university faculty had a system or a way of organization that needed to be taken into consideration before scheduling the timetables. Most solutions did not take that into account.

Adopting the graph coloring algorithm proved to be slow on large scaled inputs [2] and did not cover the constraints for all parties [8]. On the other hand, the genetic algorithm provided a better performance rate, although not all dimensions were taken into consideration [12]. An additional mechanism was added to the genetic algorithm to aid in failure recovery by resetting the algorithm in case it fails, but that could lead to indefinite results on a large scale [5].

Scheduling a conflict free timetable that also gives the students, lecturers, and teaching assistants a comfortable week is highly needed. In order to achieve a timetable that is both conflict free and provides comfort, the timetable has to satisfy a set of constraints which can be categorized as either hard or soft in nature.

In this paper, we introduce an automated schedule builder that utilizes the genetic algorithm to generate an optimal timetable for each faculty within a university. In our solution, the timetable is treated as a chromosome with a 3D view (time, day and session). A uniform crossover method is adopted to find the best combination and mutation to be applied in order to produce a good population. In our fitness function, that tests the fitness of each chromosome, we pass the new population from each generation in order to select the best chromosome at the time. The proposed application produced a nearly optimal timetable that is conflict free and only requires minimum modification to satisfy all requirements.

The rest of the paper is organized as follows; Section 2 covers the background and related work. Section 3 addresses the proposed automated scheduler, while its implementation

details are described in Section 4. The experimental results are illustrated and discussed in Section 5. Finally, the paper is concluded and the future work is noted.

II. BACKGROUND AND RELATED WORK

Automated timetable scheduling is quite an interesting notion with a vast amount work. The majority of proposed solutions were based on two algorithms: the Graph Coloring Algorithm (GCA) and the Genetic Algorithm (GA). These two algorithms compete to solve this problem.

In 2006, Timothy A. Redl published a paper on creating a conflict free timetable using graph coloring algorithm with essential and preferential constraints [2]. Timetabling, being an NP problem, is sometimes needed to explore all possible solutions to achieve an optimal result. However with an exhaustive search and a large scale population, it would take a vast amount of time to compute a result that meets the required constraints. Therefore a near optimal solution could be achieved with the graph coloring algorithm, taking the constraints into consideration. The constraints are divided in two categories respectively essential and preferable constraints, which are further discussed [2]. Redl concluded that with a conflict graph being inputted in the graph coloring algorithm, it will produce a conflict free course timetable where after each session in the timetable could be assigned to a room using a FIRST FIT DECREASING room assignment (FFDRA) algorithm or BEST FIT DECREASING room assignment (BFDRA) algorithm. [2] These algorithms will ensure rooms are big enough to hold the class [9].

Another paper on creating a student timetable generator using color algorithm was also published in Ankara by Baki Koyuncu and Mahmut Seçir in 2006 [8]. As timetabling was a major problem for the majority of universities, Koyuncu and Seçir decided to apply the graph coloring algorithm and see the results based on a large scale population of students, courses and lecturers. Their results concluded that the algorithm proved successful in creating a conflict free timetable in a short period of time and the only time consuming process was the data input. Also, their work only took into consideration student constraints and ignored other types of constraints [8].

Ahmed Abu Absa and Sana'a Al-Sayegh published a paper for a timetable generator using genetic algorithm. The genetic algorithm shows great benefit and effectiveness in creating timetables. But the more conflicts the algorithm encounters, the longer it will take to compute a schedule. For example, if the number of conflicts was to be 5, then the number of iterations to completely produce the timetable would be 3. Therefore two aspects were taken into consideration; firstly the size of the population (or initial population) should be taken into account to avoid deadlock; secondly, the probability of mutation should be taken into account so as to increase the speed of computing the timetable; it is to be noted that the higher the probability of mutation is, the better the results. It was concluded that the genetic algorithm would be a good choice to solve the timetabling problem for universities, but they had to further

test the notion on problems with a larger scale to guarantee accuracy. [12]

In Jaipur, Bharkha Narang, Ambika Gupta and Rashmi Bansal published a paper in 2013 on adding active rules alongside the genetic algorithm to aid the generation of the timetables. They managed to deduce that active rules could be a set for the knowledge of intelligence and the genetic algorithm to help with the dynamic environment, a space that consistently changes its constraints and rules regularly. The combination of these two algorithms had proved to be quite efficient. Each of the two algorithms plays a certain role, where the genetic algorithm reproduces the tables, crosses over the tables and mutates them until it finds the fittest table (that is the optimum solution). It is possible that in that case the genetic algorithm fails and not produces a good enough timetable; in that case, the active rules are right behind it in order to restart the algorithm and set some new parameters in order to find an optimal timetable. [7]

Looking at the different algorithms used above, the Genetic Algorithm and the Graph Coloring Algorithm, and taking into consideration the final paper that was published at the Midwest Artificial Intelligence and Cognitive Science Conference in 2012 [5] in comparing both the algorithms on the same problem at hand, the Genetic Algorithm excelled in producing the results and solving some of the instances the Graph Coloring Algorithm could not solve. Thereafter, the proposed algorithm to be used for our solution is that of the Genetic Algorithm.

III. PROPOSED AUTOMATED SCHEDULE BUILDER

Based on the research done on automated timetable scheduling, there are various solutions published but they were mainly based on a certain environment. The proposed solution is to create a generic timetable scheduler that can adapt to any university and adjust timetables given any set of constraints. The solution is based on two important components: the data module and the engine. The data module covers the essential input data and its representation, while the engine works to create the timetables. The given set of constraints will help the engine to come out with the optimal timetables [11]. In this section, we describe the constraints as well as the two components of our proposed automated schedule builder.

A. Constraints

The genetic algorithm will need to take decisions in order to modify or rate the current timetable at hand. These decisions will be measured by applying the below constraints which are defined as two categories: hard constraints and soft constraints [6].

1) *Hard Constraints*: they are conditions that must be met in order to satisfy a conflict free timetable. The hard constraints that must be fulfilled are as follows:

- Lecture halls, classrooms, or laboratories must not be double booked in the same time period.
- Lecture halls, classrooms, or laboratories must be big enough to hold the class.
- Students must not have two modules at the same time.

- Lecturer/Teaching Assistant must not give two different classes in the same time
- Lecturer/Teaching Assistant must not have a class in a time he/she is unavailable in i.e. off campus
- Each module must be assigned to its appropriate room
- Each module must have a number of consecutive time slots respectively to the length of the session
- Two modules cannot be assigned to one room at the same time

2) *Soft Constraints:* they are conditions that do not have to be met, but are recommended in order to produce a more satisfying timetable for the faculty members and/or students:

- Lecturers should not have consecutive classes in a day.
- Preferred time for lecturers to give class should be met.
- Classes should have their schedule in one building or one section on campus to avoid a lot of movement that could lead to exhaustion.
- Reduce gaps between courses to reduce the time for students on campus.

B. Data Module

The genetic algorithm requires data to work with, which is represented in a specific format, in order to be easily manipulated within the engine. Below are the sub-components of the data module:

1) *Input Data:* The automated scheduler is designed to manage a full university campus; hence all data related to the university shall be used. The data we are looking at here includes the faculties and resources. A faculty will consist of lecturers, teaching assistants, classes and courses. The resources are generally the buildings and rooms in which the scheduled slots will be assigned. This data will be used in the application's engine after data entry to produce the schedule for the selected faculty.

2) *Representation of Data (Chromosome):* once the data is captured as input, it needs to be structured in a specific format which the engine can process to produce the optimal timetable. The structure in which the data will be formatted will be in the shape of a timetable, with multiple slots available for each (day, time). A selected slot based on day, time and slot number will contain the information as follows:

- Day – day of the week in which the session will be given
- Time – time of the session that will be given
- Course – the course that will be given at the specified time
- Lecturer / Teaching Assistant – the individual that will be giving the session
- Class – the class that will be attending the session
- Room – the room of where the session will be held

C. Engine

Genetic algorithms are based on Darwin's theory of evolution in terms of genetics; it is an adaptive heuristic search algorithm first initiated by John Holland in 1975 [4]. The genetic algorithm here has a survival of the fittest approach, where a number of timetables are generated and the best one will surely be produced. Fig. 1 illustrates the

flow of the algorithms operation both diagrammatically and as pseudo code.

```

GenerateInitialPopulation P0;
EvaluatePopulation P0;
Generation counter g = 0;
While g<100 repeat
  Select elements from Pg to copy into Pg+1;
  Crossover some elements of Pg and put into Pg+1;
  Mutate some elements of Pg and put into Pg+1;
  Evaluate best elements of Pg and put into Pg+1;
  Increment generation counter: g ← g+1;
End While;

```

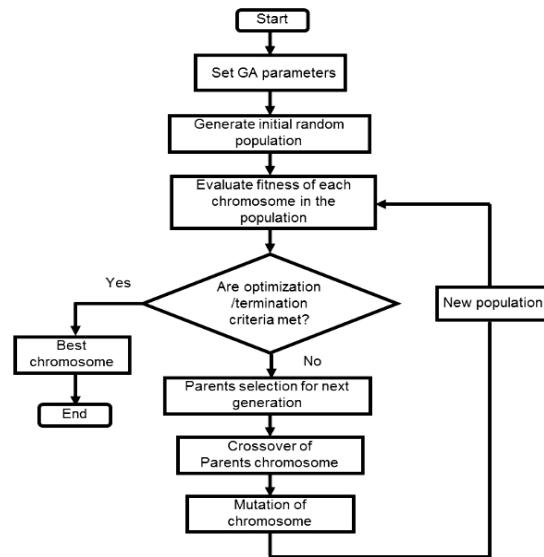


Fig. 1 the Process of the Genetic Algorithm

Each component in Fig 1 is described in the coming categories in the nature of how each component works, and the specific mechanism the component is applying (if any).

1) *Initial Population:* Seeing as there must be an initial population to start with, in our case the population is the timetables (chromosomes) available, so a random generator should be initiated to create random schedules for each class within the selected faculty and that is by randomly placing a course at different times for each group within a class; 10 randomly generated timetables will be produced for each class.

2) *Fitness Module:* After generating our population all individuals will need to be evaluated to see how fit they are, by identifying how many conflicts exist. The evaluation process will be based on the afore mentioned hard constraints.

3) *Selection Module:* The selection operator can be implemented through various ways as stated by Razali and Geraghty [10]. The selection strategy chosen for this project is the Tournament selection process which is implemented as follows: firstly a set number of competitors is identified, *n*; secondly, *n* number of participants are selected from the

current population which would then compete against each other with focus on their fitness levels; Finally, the fittest participant wins and is selected for the crossover or mutation stage.

4) *Crossover Module:* After chromosomes have been evaluated and the selection module has selected the best amongst them, they are passed on to crossover traits to generate a new chromosome, which will have a higher fitness rate than the original chromosomes since best traits of both chromosomes will be selected. There are a few crossover strategies that one could choose from as Jorge explained in his paper[1], the choice will be made depending on the design of the chromosome taking into consideration how its properties will be distributed amongst its offspring's as well as the performance. Unlike the uniform operator, the uniform crossover operator is not a good choice as the simple single point crossover will not shuffle around the required data well. The uniform operator works by setting a probability or percentage (p) and a random percentage is generated (r). If r is greater than p then the single gene that is being processed of parent 1 will go to offspring 2 where the gene of parent 2 will go to offspring 1; if r is smaller than p then the process is reversed.[1]

5) *Mutation Module:* This module will select some other parents in order to alter the chromosome to produce a better offspring; but this module is not operated at all times; after the crossover module executed, a random probability variable is generated. If it is within the set range, the mutation module will execute. The mutation module is executed by shuffling some of the slots in the chromosome and enhancing a few of them. As a result, a better chromosome is produced.

IV. EXPERIMENT AND RESULTS

A. Case Study

The subject of this case study is one of the recently established universities that contain a group of faculties that share the same resources. It is important to note that there is an increasing number of students each year, while the resources (Professors, Teaching Assistants, Lecture halls, Classrooms, and labs) do not increase at the same rate. It is quite clear how difficult the process is especially with the urgent need to produce a conflict free timetable for all faculties with their different classes each academic semester.

We choose to test our solution on a faculty that contains 18 lecturers and 22 teaching assistants (TAs). The staff is teaching 34 modules for 4 classes in the faculty. Each module is composed of a lecture time and laboratory time; the whole class takes the lecture at the same time in a lecture hall, while they are divided into groups ranging from 25 – 30 students to take laboratory and/or tutorials with the teaching assistant. The population size of the 4 grades starting from grade 1 up to grade 4 is 200,170,110 and 80, respectively. This faculty shared their resources with another faculty as follows: 8 lecture halls, 16 classrooms, and 7 Computer Laboratories which can be utilized during the term.

The above data are the required parameters for the Genetic Algorithm. The constraints in which the Genetic Algorithm will work on will be based on the user's input.

B. Results and Discussion

After developing the application, the above case study was applied to test the output of the application; the produced results were acceptable. Figures ranging from Fig. 2 to Fig. 6 demonstrate the application. Fig.2 defines the input of each faculty at the university with its respected staff (lecturers and teaching assistants), modules and number of classes. In Fig.3, the university buildings are inputted along with the lecture halls, laboratories, and classrooms for each building. Fig.4 is the starting point in creating a timetable for a selected faculty for a specific semester; and in Fig.5, the generated timetables can be selected for viewing. Finally in Fig. 6, shows the output of the application, a timetable for a selected faculty and class.

The results of the algorithm will depend on three main aspects; First, the number of generations the algorithm will proceed for, provided that the bigger the number of generations is, the better the results will be. That is because the algorithm takes more time to manipulate and create new timetables in order to reach a better timetable. Secondly, the initial population (or set of timetables) generated will have an impact on reaching a better chromosome, or individual, in the shortest time possible as well as more accurate results. Third and last is the fitness function and the core of the whole schedule builder. All decisions made on a chromosome are made after measuring its fitness. The fitness value produced will affect the selection, crossover and mutation functions.

To better simplify the notion behind the schedule builder, it is safe to say that the algorithm is only as good as the fitness function. In other applications, the fitness function was designed to fit the environment that it would be working in. Nevertheless, in our implementation, it was designed to be of a generic nature that perfectly fit the purpose of selection.

With respect to response time, the two main operators that can save processing time when calculating the nearly optimum timetable are the selection and crossover operators; there are many strategies to apply for any of the operators it is just a matter of knowing which strategy is best. Razali and Geraghty [10] can help in giving specifics on the selection operator, and Mendes can help in selecting the right Crossover operator depending on the parameters of a problem [1].

V. CONCLUSION

The paper proposed an automated timetable schedule builder. An application is developed to enable a university to automatically create all faculty timetables from one place with the click of a button. The solution reported the ability to meet most of the hard constraints with a nearly optimal solution, which would require a few manual adjustments to meet the university resource requirements and save a lot of time. The algorithm used to manage the resources and create the resulting timetable was the Genetic Algorithm. The solution first creates a chromosome, which is the representation of the timetable. It is created as a 3D form of (time, day, and session). After that, the right fitness function to measure the fitness of each chromosome and produce the optimal timetable after a number of generations. The genetic



Fig. 2, Entering Faculty Resources

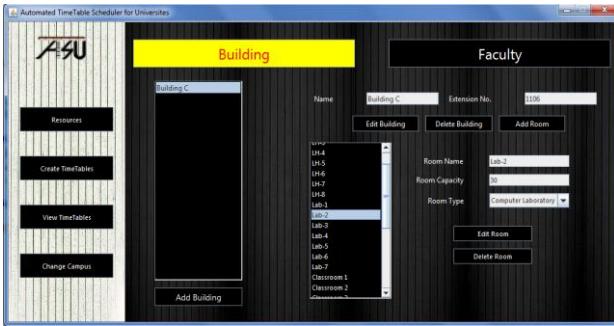


Fig. 3, Entering Building Resources

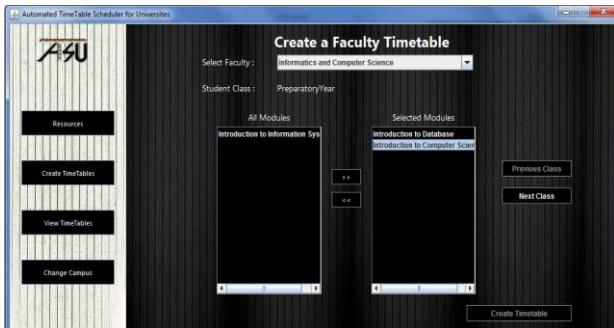


Fig. 4, Creating a Timetable

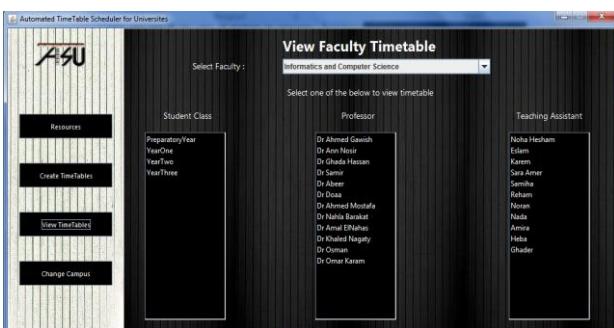


Fig. 5, Viewing a Timetable

Time	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
9 - 10	Free Slot	Java Heba Tutorial	Java Dr Ghada Hassan lecture	Free Slot	Free Slot	Java Heba Tutorial	Free Slot
10 - 11	Free Slot	Digital Design Amira Tutorial	File Organization Samha Tutorial	File Organization Samha Tutorial	Free Slot	Digital Design Dr Samir lecture	Free Slot
11 - 12	Digital Design Amira Tutorial	File Organization Samha Tutorial	Java Heba Tutorial	File Organization Samha Tutorial	Free Slot	Digital Design Amira Tutorial	Free Slot
12 - 13	Digital Design Amira Tutorial	Digital Design Amira Tutorial	Java Heba Tutorial	File Organization Samha Tutorial	Free Slot	Digital Design Amira Tutorial	Free Slot
13 - 14	File Organization Samha Tutorial	Free Slot	Digital Design Amira Tutorial	Java Heba Tutorial	Free Slot	Java Dr Ghada Hassan lecture	Free Slot
14 - 15	Free Slot	Free Slot	File Organization Samha Tutorial	File Organization Samha Tutorial	Free Slot	Java Dr Ghada Hassan lecture	Java Heba Tutorial
15 - 16	Free Slot	Free Slot	Free Slot	Free Slot	Free Slot	Java Heba Tutorial	Digital Design Amira Tutorial

Fig. 6, Timetable view

algorithm can be modified to produce a better generation and this is achieved by amplifying the fitness function and applying a better crossover and mutation method that can be applied on a 3D chromosome.

REFERENCES

- [1] J. Magalhães-mendes, "A Comparative Study of Crossover Operators for Genetic Algorithms to Solve the Job Shop Scheduling Problem," WSEAS Transactions on Computers, vol. 12, no. 4, pp 164-173, 2013.
- [2] T. A. Redl, "On Using Graph Coloring to Create University Timetables with Essential and Preferential Conditions," Proceedings of the 3rd Inter. Conf. on Computational and Information Sciences, University of Houston-Downtown, pp 162- 167, 2006.
- [3] "Scheduling," 17 April 2015. [Online]. Available: <http://dictionary.reference.com/browse/scheduling>.
- [4] P. Weaver, "the origins of bar charting," Mosaic Project Services Pty Ltd, South Melbourne, 2014, [online] available:http://www.mosaicprojects.com.au/PDF_Papers/P182_The_origins_of_bar_charting.pdf.
- [5] M. M. Hindi and R. V. Yampolskiy, "Genetic Algorithm Applied to the Graph Coloring Problem," in Proc. 23rd Midwest Artificial Intelligence and Cognitive Science Conf., pp. 61-66, 2012.
- [6] A. Wren, "Scheduling, timetabling and rostering — A special relationship?" in Proc. 1st Inter. Conf. on Practice and Theory of Automated Timetabling, pp. 46-75, 1995.
- [7] B. Narang, A. Gupta and R. Bansal, "Use of Active Rules and Genetic Algorithm to Generate the Automatic Time-Table," Inter. Jour. on Recent and Innovation Trends in Computing and Communication, vol 3, no. 11, pp 3480 – 3483, 2014.
- [8] B. Koyuncu and M. Seçir, "student time table by using graph coloring algorithm," In 5th Inter. Conf. on Electrical and Electronics Engineering – ELECO, 2007.
- [9] T. A. REDL, "A Study of University Timetabling that Blends Graph Coloring with the Satisfaction," Ph.D. Thesis. Rice University, 2004, [online] available: <https://scholarship.rice.edu/handle/1911/18687>
- [10] N. M. Razali and J. Geraghty, "Genetic Algorithm Performance with Different Selection Strategies in Solving TSP," Proceedings of the World Congress on Engineering, London, 2011.
- [11] A. J. Champandard, "Genetic Algorithms," 2015. [Online]. Available: <http://ai-depot.com/BotNavigation/Avoidance-GeneticAlgorithm.html>.
- [12] A. Ahmed and A. Sana'a, "E-learning Timetable Generator Using Genetic Algorithm," University of Palestine, Gaza.