2021 International Conference of Technology, Science and Administration (ICTSA)

Genetic Algorithm For Solving University Course Timetabling Problem Using Dynamic Chromosomes

Ghazi Alnowaini
Department of Mechatronic Engineering
Taiz University
Taiz, Yemen
Dr.ghazi.aln@gmail.com

Amjad Abdullah Aljomai
Information Technology Departement
Taiz University
Taiz, Yemen
amjadsam202095@gmail.com

2021 International Conference of Technology, Science and Administration (ICTSA) | 978-1-6654-2862-0/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICTSA52017.2021.9406539

***Abstract*—Building of university timetable is one of the problems that are difficult to be solved because of the large number of lectures and conflicts between them. This makes it difficult to introduce schedule-building restrictions that consume time and effort for table production. Many methods have been suggested that the computer is used to solve this problem, including a Genetic Algorithm (GA) where the main purpose of the algorithm is to reduce the number of conflicts in the timesheet and to reduce the search space encoding. This paper proposes an automated system to build a faculty timetable using a genetic algorithm. A genetic algorithm had been used to schedule the timetable of the faculty of engineering and information technology with a dynamic chromosome size that is flexible with the course numbers of each department. This algorithm can be applied in different institutions (i.e. faculties, or institutes) According to their limitations. The proposed system achieved great results during the evaluation phase of around 93% compared to manual scheduling or the systems available .**

Keywords— Genetic Algorithm, dynamic chromosome, timetable, university timetable, optimization

I. INTRODUCTION

Timetable scheduling problems have received tremendous attention from disciplines like Operations Research and Artificial Intelligence throughout the years. Several optimization techniques are used to solve them and produce optimal or near-optimal solutions instead of the exact solution.

In universities, however, the semester timetable scheduling process has become very complex since there are four or five batches of students, and many student groups and some student groups take some courses together. There are some courses in a specific batch that are followed by all the students. You have to make sure that there are no conflicts between student groups, lecturers, and lecture halls when planning the timetable. Large-scale schedules such as university schedules may require many hours of work spent by a qualified person or team to produce high-quality schedules with optimum restriction satisfaction.

The timetable should satisfy the following conditions:

- A particular can attend only one class at one time.
- One instructor can teach only one class at one time.
- In-room, only one class can be taught at one time.

GAs are used in optimized searching processes and scheduling process. This algorithm is generally applied to the problem of where optimization is the key goal. A timetable problem is related to finding the exact time allocation within a limited time of a set of events (courses-lectures) and assigning to them a set of resources (teachers, students, and Lecture Halls) while satisfying some constraints. The constraints are classified into two types namely hard Constraints and Soft constraints. Hard constraints are those that must be strictly applied and worked on with, while soft Constraints can be violated if necessary.

In this paper, a genetic algorithm had been used to schedule the timetable of the faculty of engineering and information technology with a dynamic chromosome size that is flexible with the course numbers of each department. This algorithm can be applied in different institutions (i.e. faculties, or institutes) According to their limitations.

II. PREVIOUS STUDIES

In 2013, Ahmed and Osman [15] presented an automated system to construct the University of Al Jouf timetable using genetic algorithms. They depended on three elements in constructing timetable classrooms available in the college, academic staff members, and courses taught in the college. These elements were linked together using matrices. They used Java, XML, and C# in the practical application of the genetic algorithm. The system has succeeded in building a timetable of lectures to community college (70 courses) free of any conflict or duplication. They used the hard constraints in the solution, and they ignored the soft constraints for the lack of clarity and disagreement upon a university.

An algorithm is designed to fit the rigid constraints of the college and the rigid constraints of the college have been collected and restricted, matrices are then used in the initial design that relates data, and then an algorithm is designed to further the constraints we have encountered in building the table [1].

In 2017, Gore et al published the Study of Timetable Generation Using Ant Colony Optimization Algorithm [2]. Evolutionary techniques have been used to solve the

978-1-6654-2862-0/21/$31.00 ©2021 IEEE

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.

timetable scheduling problem. The ant colony optimization met heuristic algorithm has already proved that it can be used in simplified artificial instances of College course timetabling problems. Till now limited work has been done on practical timetabling scheduling problems. They Focused on applying the optimization of the ant colony to a highly restricted real-world instance of the timetabling issue of college courses. Discussion of the memory is an efficient design of the construction graph and the sophisticated construction processes of the solution.

Most real-world timetabling problems are NP-complete due to inherent complications and variations of the problem. Thus, heuristic solutions that are generally good enough for practical use (although they do not guarantee an optimal solution) are used, and if implemented then, due to their manageability and good performance, heuristic techniques have proven to be particularly suitable for solving these types of problems.

University Timetable Scheduling Using Genetic Algorithm was presented by Premasiril et al [3] in 2019. The main objective of the study is to develop an algorithm to optimize the resources at hand while meeting all the difficult constraints and most of the feasible soft constraints using the strategy of particle swarm optimization by satisfying hard and soft constraints.

The algorithm has been validated using real data for university courses' timetabling problem. The Particle Swarm Optimization (PSO) approach is also an evolutionary technique but it differs from the Genetic Algorithm approach. In PSO there is no DNA related genetic evolution but instead, the process is based on the experience gained by the particles. Abdullah et al [4] investigated an optimization algorithm (genetic algorithm) combined with a sequential local search for the syllabus' courses timetabling problem in "An Investigation of a Genetic Algorithm and Sequential Local Search Approach for Curriculum-based Course Timetabling Problems".

The algorithm consists of two stages as the initial construction stage and the improvement stage. The construction stage consists of three phases as the largest degree heuristic, neighborhood search, and tabu search. In "Fuzzy Genetic Heuristic for University Course Timetable Problem", Arindam Chaudhuri and Kajal De present a Fuzzy Genetic Heuristic Algorithm to solve the problem [5].

The method incorporates Genetic Algorithms using indirect representation. The Fuzzy set method deals with soft constraints violation. This approach has a significant improvement in satisfying soft constraints. "Solving the class timetable problem by using genetic algorithm and tabu search algorithm" [5] Premlata A. Sonawane, Leena Ragha discusses a combination of two approaches; genetic algorithms and Tabu search.

First, in the algorithm, they use the Genetic Algorithm approach to select the best chromosome with a better fitness value. Then the Taboo search algorithm is used to enhance the quality of the solution. Students could not go back and forth to these two places. It is time-wasting and not fair to students. Timetable scheduling was a huge burden because of all the above-mentioned problems.

Manual Scheduling took so much time and the clashes occurred more often. In timetable scheduling, there are some constraints to be satisfied. There are hard constraints that should be satisfied no matter what to come up with a feasible solution and there are soft constraints that are not that essential to satisfy better to satisfy as much as possible. The following constraints have been identified to find a feasible solution. The proposed algorithm was programmed using C# programming language.

Population Size: 80 Crossover Probability: 0.8 Mutation Probability: 0.02Crossover Type: Two Point The algorithm was tested using the real data of the Department of ICT of the Faculty of Technology. The algorithm satisfied all the hard constraints and 60% of the soft constraints.

Unlike previous studies, a genetic algorithm had been used to schedule the timetable of the faculty of engineering and information technology with a dynamic chromosome size that is flexible with the course numbers of each department and each term. This algorithm can be applied in different institutions (i.e. faculties or institutes) According to their limitations.

III. METHODOLOGY

A. *Hard constraints*

1. No student can attend more than one event at the same time.
2. The lecture hall should have the capacity to hold the number of students
3. Students should follow the scheduled course.
4. Only one event can be scheduled in a room at any timeslot.
5. A lecturer cannot be teaching two courses at the same time.
6. Two practical sessions cannot be scheduled in one laboratory at the same time.

B. *Soft constraints*

1. The lecture should not go for more than 3 hours.
2. Students should have some free periods between lectures.
3. Each department has one day without lectures.
4. Some instructors do not wish their lectures to be consecutive.
5. Students want their lectures to be close together.
6. Some instructors prefer certain hours for their lectures.
7. Most students and instructors do not prefer to have large space (gaps) between their lectures.
8. There should be special halls for the lecturers.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.

9. The instructors do not take more than five hours a day.
10. Instructors may prefer to have free days.
11. No instructors should be scheduled in more than two consecutive time slots.

C. Data collection

Data were collected such as the number of lecturers in the department, information about lecturers (i.e., their qualifications, experiences, etc.), the number of classes, the number of courses and their units, the days of the lectures, etc.

D. Elements of a timetable

Basic data in the system (courses - teaching staff - halls - periods – departments - levels).

1) Academic courses:

The course name, its code, the number of theoretical and practical hours for each course were entered.

2) Teaching staff:

The name of the lecturer and his academic degree was entered.

3) Halls:

The names and numbers of the halls were entered. Also, their type (hall - amphitheater - laboratory - studio) and the carrying capacity.

4) Periods:

It includes the number of days and periods (since there are six days and three-time periods every day).

5) Departments:

The College of Engineering has six departments (Communications Engineering, Software Engineering, Information Technology, Manufacturing, and Industrial Engineering, Mechatronics and Robotics Engineering, and Computer Networks & Distributed Systems Engineering).

6) Levels:

Each department of the College of Engineering contains five levels.

E. Timetable development using the genetic algorithm

The idea starts by linking each of (courses - teaching staff - departments - levels) using a flexible interface in which the user links each department and level to the course and the staff who will teach this course.

This data will be saved in a separate table we called (block), and figure 1 explains this idea.

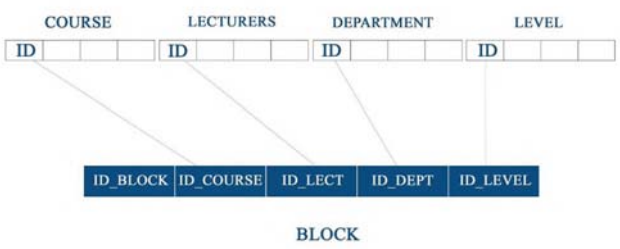

Fig. 1. The structure of block table of the Faculty of Engineering.

The size of the chromosome had to be determined, so the idea was to rely on the block table in determining the size because the block table contains the official courses in each term [11, 14]. The method is done by knowing the number of records in this table and multiplying them by three. Timetabling Chromosomes applied by the genetic algorithm were constructed into tuples or groups of elements (E) which were spaces for the selected input. In this study, elements are composed of three types of data; which are BLOCK (B), TIME (T), and ROOM (R).

Our idea is that we will rely on the positions in the chromosome, where we start from the zero position until the end of the size of the chromosome minus one, and that all three positions represent a lecture as shown in the following figure.

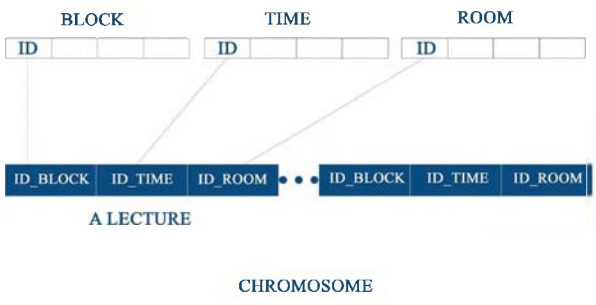

Fig. 2. The structure of the chromosome that has been used in this study.

In this study, a dynamic size of the chromosome was used to fit every semester courses for each department. In other words, the size of the chromosome on semester one can be different in the size of the second semester depending on the course number. This change makes the proposed model flexible with various faculties.

F. Fitness function

The fitness function was used to measure the appropriateness of timetabling and search to find the best one, which can raise problems. Fitness function constructed from timetabling conditions. It can be shown using the following equations:

total_max=max_constraint-Violated_constraint

IV. PROPOSED MODEL

A genetic algorithm [6] is a field of computer science that is a subset of evolutionary computation. They are very often based on computational models in adaptive environments to work effectively. For example, a self-adaptive approach is needed to increase the level of generalization in timetabling. Another example is the dynamic actions needed by a robot to manage its surroundings. A genetic algorithm (GA) is a search tool used to find real or approximate solutions to optimization and search issues in computation [4, 7, 8]. A genetic algorithm is a strategy for solving problems of restricted and unconstrained optimizations. [11]. Figure 3 shows the representation of chromosome, gene, and population.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.

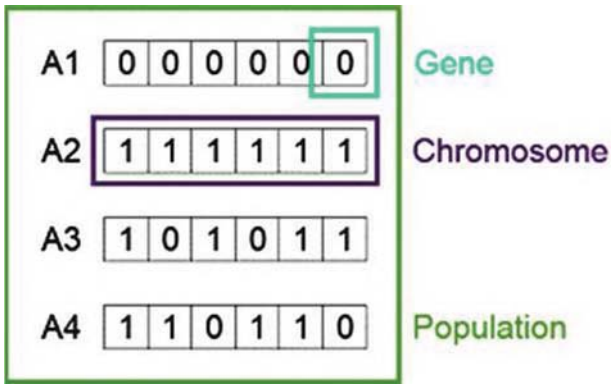

Fig. 3. Chromosome Structure in Genetic Algorithm.

In a genetic algorithm[2, 11-13], as shown in Figure 4, a chromosome population consisting of a given random set of genes is initiated according to the steps outlined in Fig 4.

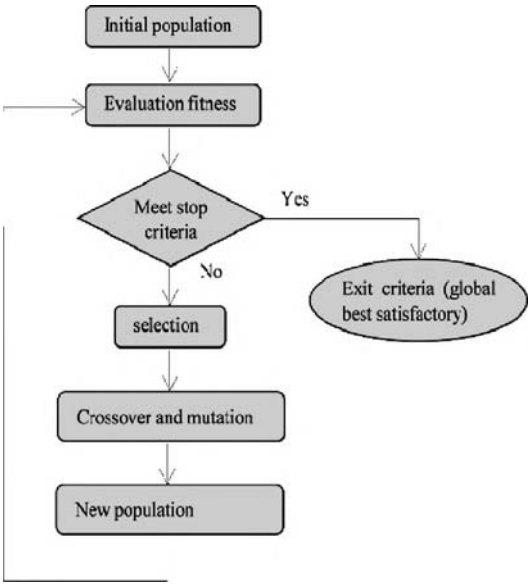

Fig. 4. Genetic Algorithms Steps.

A genetic algorithm is a heuristic search that is inspired by the genetic evolutionary theory of Charles Darwin. This algorithm illustrates the natural selection mechanism where the fittest individuals are chosen for selection to produce next-generation offspring.

With each chromosome (organism) representing a complete solution to a given problem, a genetic algorithm generates an initial chromosome population. Random values are initialized by the genes that make up the chromosomes. Depending on a feature unique to a given problem, the solution quality of each chromosome [12].

After the initial population, the whole process of 'GA' revolves around three operations called selection, crossover, and mutation as Figure 5.

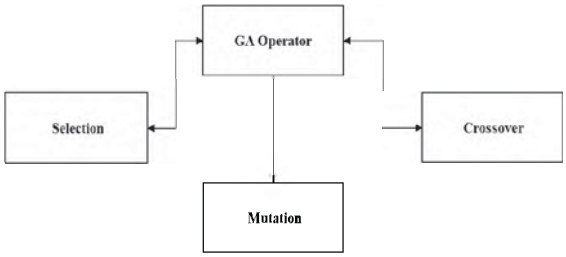

Fig. 5. The three operations are called selection, crossover, and mutation.

The first question to ask before beginning to solve a genetic algorithm problem is the encoding of chromosomes. Genetic algorithms run on alternatively encoded space and solution space.

Two basic criteria for choosing the encoding form are followed by genetic algorithms::

A. The principle of meaningful building blocks:

The schemata should be short, of a low order, and relatively unrelated to schemata over other fixed positions. The principle of minimal alphabets:

The alphabet of the encoding should be as small as possible while still allowing a natural representation of solutions[5].

Reproduction (or selection) is an operator in a new population that allows more copies of better strings. The first operator applied to a population is typically replication. A proportion of the current population is chosen to breed a new generation during each successive generation. Via a fitness-based process, individual solutions are chosen, where fitter solutions are usually more likely to be picked. The system of selection scores each individual's fitness and selects the best answer preferentially [6].

Common Methods of Selection are:

- Roulette wheel approach.
- Tournament selection.
- Stochastic remainder selection.

Herein, the Roulette wheel method had been used in this work.

Other selection algorithms do not consider all people to be chosen, but only those with a higher fitness value than a given (arbitrary) constant. Other algorithms choose from a restricted pool, where, based on fitness value, only a certain percentage of individuals are allowed.

This is conceptually described as a game of roulette. As seen in Figure 6, each person gets one part of a wheel, but the greater fit is less compared to smaller lines as depicted in figure 6.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.

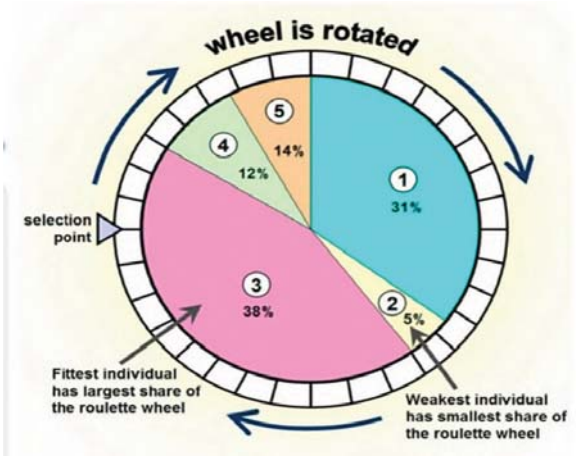

Fig. 6. Roulette wheel.

The criterion is that there is a better probability of success for the fittest chromosomes than poorer ones. This replicates evolution in that fitter chromosomes appear to have a greater chance of surviving and will step on into the next generation to form the mating pool. Weaker chromosomes are not without an incentive. These people may have a genetic code of nature that may prove beneficial to future generations [14].

The parents used in the crossover to produce a new child are determined by the use of the selection procedure. Crossover is introduced by choosing a random point on the chromosome where the swap of parents' pieces occurs. Then, the crossover sets up a new offspring depending on the selected exchanging point with specific pieces of the parents. Dependent on the selected exchanging point with specific pieces of the parents, the new offspring is created.

To obtain a new solution, a mutation may be described as a small random tweak in the chromosome. It is used in the genetic pool to preserve and incorporate variation and is typically introduced with a low probability. After the crossover is over, mutation takes place. This operator randomly adds the modifications to one or more "genes" to create new offspring. Typically, after the junction is made, the mutation happens.

V. ANALYSIS

A. *Current system*

Scheduling is the process of creating a schedule that contains events arranged according to the time at which they occur, which must be subject to timing constraints for each entity that is placed in the schedule. The university timetabling in this context highlight to the strict task performed by the assigned staff to set timetables for all the courses, which have to satisfy all the mandatory requirements and reach a finalized timetable.

These courses are usually taught by various lecturers in different departments who may also want to specify some timing limitations in their lectures. Therefore, university staff is responsible for creating a timetable that satisfies all the requirements and constraints. The current system is a manual timetabling system, which is time-consuming, resource-intensive, requires many steps, and requires multiple reprocessing for the same data.

The process of preparing the timetable by university staff is autonomous and can be improved through cooperation with the various entities involved. Manual scheduling faces many problems, the most important of which are the following:

- Repeated time may be allocated to certain courses, which leads to duplicated data.
- Many management errors can occur due to confusing time requirements.
- A manual timetabling process by university staff is slow.
- The final timeline created may not be close to the optimal timetable due to material requirements and provisions.
- It generates lots of papers and assignments.
- It is not flexible as changes may not be made easily.

B. *The proposed system*

The proposed systems were developed to solve the timetabling problem that universities face in each academic year and reduce the high cost and slow turnaround involved in creating near-optimal timetables. The system has the capabilities to enter various courses, lecture halls, departments, levels, buildings, and lecturers, and define some restrictions through which the timetable is set.

The proposed timetabling system in this project seeks to create near-optimal timelines using the principles of a genetic algorithm.

1) Advantages of the proposed system:

- Provides flexibility, unlike the manual scheduling system.
- Uses minimal processing/computing power.
- Significantly reduces the time required to create near-optimal schedules.
- Provides an easy way to enter and review data through an easy-to-use interface.
- Increases productivity.
- The optimality of the generated timelines is 93%.
- Removes, almost all the paperwork.
- Simplifies the timetabling process.

2) Limitations of the proposed system:

- The proposed system can only create timelines based on some difficult constraints.
- The generated timetable by the system is still subject to review.
- Not all principles of the genetic algorithm have been implemented in the system.

VI. RESULT AND DISCUSSION

The proposed system was evaluated and tested on solving a timetable-scheduling problem in the Faculty of Engineering and Information Technology.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.

We used a Core i7 CPU, 20 RAM, and 2 graphics cards while testing our system.

The obtained timetable scheduling in the 100th generation is shown in the following figure 7.

Fig. 7. Part of the obtained faculty timetable schedule.

The timetable of the faculty had been scheduled efficiently and on time. Although, the complexity of constraints that includes time, classroom, and lecturer, the proposed system obtained promising results. Part of the obtained results is shown in the above figure (Fig.7).

VII. Conclusions and Future Work

This paper discussed the use of a genetic algorithm for scheduling faculty timetables in an efficient way. Where the Fitness function was 24.09 at a time of 5 minutes. The conclusions can be redrawn in the following points:

1. Part of the main objective of the project has been accomplished, which is to find the perfect or near-optimal solution for the lecture schedule for the College of Engineering for three departments (Information Technology, Communication, Computer Networks, and Distribution Systems).
2. The scheduling time is greatly reduced, and the effort involved in designing the schedule is reduced and preventing conflicts in times and halls.
3. The increase in the number of generations reduces the number of conflicts for the courses.
4. The more the number of courses exceeds 82, the more the number of conflicts increases dramatically.
5. The time spent in producing the timetable is 5m.
6. The increase in mutation rate increases the number of clashes. As a result, the mutation rate is optimal at 0.01.

For further improvement or future studies, the system should consider the following points;

1. Schedule all the sections of the Faculty of Engineering.
2. Make the possibility to request reservation of free times for the lecture halls
3. Design the test table.
4. Create a lecture schedule for various colleges.

References

[1] A. K. Herath, "Genetic Algorithm For University Course Timetabling Problem," University of Mississippi, 2017.

[2] P. Gore, P. Sonawane, and S. J. I. J. I. R. C. C. E. Potdar, "Timetable generation using ant colony optimization algorithm," vol. 5, no. 3, pp. 6033-6039, 2017.

[3] H. Alghamdi, T. Alsubait, H. Alhakami, A. J. E. Baz, Technology, and A. S. Research, "A Review of Optimization Algorithms for University Timetable Scheduling," vol. 10, no. 6, pp. 6410-6417, 2020.

[4] S. Abdullah, H. Turabieh, B. McCollum, and E. K. Burke, "An investigation of a genetic algorithm and sequential local search approach for curriculum-based course timetabling problems," in Proc. Multidisciplinary International Conference on Scheduling: Theory and Applications (MISTA 2009), Dublin, Ireland, 2009, pp. 727-731.

[5] A. Chaudhuri and K. J. I. J. A. S. C. A. De, "Fuzzy genetic heuristic for university course timetable problem," vol. 2, no. 1, pp. 100-123, 2010.

[6] S.-C. Wang, "Genetic algorithm," in Interdisciplinary Computing in Java Programming: Springer, 2003, pp. 101-116.

[7] A. Jaiswal, G. J. I. J. o. E. S. Dubey, and Engineering, "Identifying best association rules and their optimization using genetic algorithm," vol. 1, no. 7, 2013.

[8] M. Al-Ashhab and A. Alghamdi, "Two-Stage Multi-Objective University Courses Timetabling Using Genetic Algorithms."

[9] E. Yu and K. S. J. I. t. i. o. r. Sung, "A genetic algorithm for a university weekly courses timetabling problem," vol. 9, no. 6, pp. 703-717, 2002.

[10] O. Chohan, "University Scheduling using Genetic Algorithm," ed, 2009.

[11] F. A. Omara and M. M. Arafa, "Genetic algorithms for task scheduling problem," in Foundations of Computational Intelligence Volume 3: Springer, 2009, pp. 479-507.

[12] J. Dean, "Staff scheduling by a genetic algorithm with a two-dimensional chromosome structure," in Proc of the 7th Conference on the Practice and Theory of Automated Timetabling, 2008.

[13] D. Dasgupta and D. R. McGregor, "Designing application-specific neural networks using the structured genetic algorithm," in [Proceedings] COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks, 1992, pp. 87-96: IEEE.

[14] S. Navlakha and Z. J. M. s. b. Bar - Joseph, "Algorithms in nature: the convergence of systems biology and computational thinking," vol. 7, no. 1, p. 546, 2011.

[15] A. Ahmed, M. Osman," Building Al Jouf University timetable using Genetic Algorithm,", 2013.

Authorized licensed use limited to: Robert Gordon University. Downloaded on May 29,2021 at 16:23:20 UTC from IEEE Xplore. Restrictions apply.