

DOI: <http://dx.doi.org/10.21123/bsj.2022.19.2.0399>

## An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem

*Israa Abdulameer Abduljabbar\**

*Sura Mahmood Abdullah*

Department of Computer Science, University of Technology, Baghdad, Iraq

\*Corresponding author: [110033@uotechnology.edu.iq](mailto:110033@uotechnology.edu.iq), [110050@uotechnology.edu.iq](mailto:110050@uotechnology.edu.iq)

\*ORCID ID: <https://orcid.org/0000-0003-4196-9144>, <https://orcid.org/0000-0002-1396-9086>

Received 2/6/2020, Accepted 14/2/2020, Published Online First 20/9/2021, Published 1/4/2022



This work is licensed under a [Creative Commons Attribution 4.0 International License](https://creativecommons.org/licenses/by/4.0/).

### Abstract:

Scheduling Timetables for courses in the big departments in the universities is a very hard problem and is often be solved by many previous works although results are partially optimal. This work implements the principle of an evolutionary algorithm by using genetic theories to solve the timetabling problem to get a random and full optimal timetable with the ability to generate a multi-solution timetable for each stage in the collage. The major idea is to generate course timetables automatically while discovering the area of constraints to get an optimal and flexible schedule with no redundancy through the change of a viable course timetable. The main contribution in this work is indicated by increasing the flexibility of generating optimal timetable schedules with different copies by increasing the probability of giving the best schedule for each stage in the campus with the ability to replace the timetable when needed. The Evolutionary Algorithm (EA) utilized in this paper is the Genetic Algorithm (GA) which is a common multi-solution metaheuristic search based on the evolutionary population that can be applied to solve complex combinatorial problems like timetabling problems. In this work, all inputs: courses, teachers, and time acted by one array to achieve local search and combined this acting of the timetable by using the heuristic crossover to ensure that the essential conditions are not broken. The result of this work is a flexible scheduling system, which shows the diversity of all possible timetables that can be created depending on user conditions and needs.

**Key words:** Constraints, Evolutionary Algorithm (EA), Fitness function, Genetic Algorithms (GA), Timetable Schedule (TTS).

### Introduction:

Artificial Intelligence (AI) is one of the most important branches of computer sciences that has been created for machines supporting to help in taking decisions and solving complex problems as humans do. This generally consists of taking characteristics from human individual behavior and applying them as computer procedures<sup>1</sup>.

A flexible approach can be taken depending on the user's demands, which affects how to make intelligent behavior appears artificially<sup>2</sup>. GA is a heuristic technique that depends on natural evolution. It deals with a population named chromosomes as a candidate for problem-solving. The algorithm picked one chromosome from the population according to fitness values established by the fitness function; this performs the main role in the Genetic Algorithm. The major idea behind this work is the generation of course timetables automatically while discovering the area of

constraints like the availability of classrooms, courses and times since these timetables are generated using GA to get an optimal and flexible schedule with no redundancy through the change of a viable course timetable<sup>3,4</sup>.

The aim can be divided into two objectives: The primary objective is to be able to optimize the algorithm used to generate timetable systems and to get optimal timetable with no clashes. The minor objective is to expand the area of course scheduling systems by making it universal thereby bringing about uniformity in the creation of timetables as it applies to different campuses this is to create timetables that agree with the requirements of any educational institution.

There are many gaps and limitations in the previous systems as a traditional generation of timetables, which may include the following<sup>5,6</sup>.

- Processing of a specific course timetable needs long-time because of data duplicated.

- Many administrative errors occurred due to conflicts in time requirements.

- Creation of timetable by staff is slow.

- The resulted timetable is not ideal due to clashing course requirements and allocations.

- Lots of work papers is needed.

- Updating the timetable is more difficult.

The main contribution in this work is indicated by increasing the flexibility of generating optimal timetable schedules with different copies (probabilities) by increasing the probability of giving the optimal schedule for each stage in the campus with the ability to replace the timetable when needed.

### Related Work

Timetabling is a process concerned with making a timetable having events arranged according to a time when they take place must be subject to the timing constraints of each entity placed in the Table<sup>7</sup>. These courses are usually taught by different lecturers in different classrooms to specify some timing conditions in their lectures. Given all courses and course details for each course, the classical manual timetabling system is time-consuming, heavy resources involve many steps and needs re-processing the same data several times<sup>8,9</sup>.

Genetic algorithm is a random and probabilistic search, it produces the child population by comparing it to their parent population because it picked the parents are the fittest among the whole population set, and the bad parents die off in the next generation<sup>9,10</sup>. This algorithm is continued until some termination criteria<sup>7</sup> are agreed upon by the user<sup>11,12</sup>.

In 2014, Asif A., and Sachin B.<sup>13</sup> built a model that was utilized to generate an agreeable timetable using the probabilistic factors. Therefore, the authors used the genetic algorithm with the heuristic approach to build and improve the academic timetable. The goal of this work is to effectively use the infrastructure when the lecturers and the students are the stakeholders through the improvement process. For execution, each of the crossover, mutation, and the fitness function is to be computed. In a genetic algorithm, each chromosome has a fitness function. After analysis, if the fitness value is high then it refers to the better solution and after that, the parents are chosen according to their fitness to increase progeny for a new generation where fitter chromosomes have a greater opportunity to increase, the result of this work gave only one good solution for solving timetable problem.

In 2015, Sandesh Timilsina, Rohit Negi, and Yashika Khurana<sup>14</sup> implemented different genetic algorithmic methods in the scope of timetable scheduling. The first of this method is the 3D cutting method which has been tested in small and large timetable scheduling problems. The problem of small size was fixed easily without collides. This method was stopped at about 95 collides when trying to solve the large problem, so intelligent operators were added to improve the method which decreased the collides to about 20 collides in the least amount of runtime. The second method was set evaluation which has been tested on the final exam scheduling for all courses. The outcomes of fitness function were spotted for likelihoods of crossover that ranging from 0.00, 0.30, 0.40, 0.50, 0.60, 0.65, 0.70, 0.75, and 1.00. The last method is the Advanced-GA method which was used to fix the timetable-scheduling problem and finally, the solution from manmade was compared to the result of the last method. The result displayed that the man-made solution was fit to face all the uphill constraints but the genetically developed solution was not fit to be faced with all the uphill constraints. Furthermore, the genetically developed solution was fit to be faced with smooth constraints better than the man-made solution.

In 2016, Esraa A. Abdelhalim and Ghada A. El-Khayat<sup>15</sup> introduced a work that used some heuristics to generate a good quality timetable based genetic algorithm, the proposed algorithm was tested by Alexandria University in Egypt and also tested against two difficult benchmark problems. Testing proved that the proposed work produced a good tool for managing university timetables.

In 2017, Jumoke Soyemi, John Akinode, and Samson Olorunoba<sup>16</sup> used the genetic algorithm to design an Electronic Lecture Timetable Scheduler (ELTS) for the learning institutions to conquer the limitations for the manual system of the timetable by making the manual procedure simpler, guarantee optimal distribution of resources, decrease the danger of collision of classrooms and lecturers. The ELTS facilitated the tedious tasks of the manual timetable procedure by improving the matters related to the manual procedure to get better the activities of the academy within the institution.

In 2018, M F Syahputra and others<sup>17</sup> implemented the genetic algorithm to organize the timetable procedure for the theoretical and practical category in university. The algorithm is used to process data which include the schedules of lecturers, courses, and classrooms, gained from university. The result of the organizing timetable using a genetic algorithm is related to a random function, which will influence the best result gained

from the experiment. In each experiment, the final population is related to the final fitness value, which affects the result of the organizing timetable procedure created by the algorithm. Finally, the organizing timetable procedure is the optimal timetable that corresponds with accessible periods, classrooms, courses, and lecturer schedules.

In 2018, Izah R. Ahmad and others<sup>18</sup>, created a timetable using genetic algorithm (GA) by Java programming languages to solve many restrictions such as the class size with student's ability, irritability time to each class and lecturer who is a responsible for each class, the number of classes in one day, and the subject of participation and lecturer. The genetic algorithm with Java programming languages managed to decrease the conflicts and to improve the fitness function.

In (2019), Deeba Kannan, Kuntal Bajpayee, and Samriddho Roy<sup>19</sup> used genetic algorithm to solve the timetable scheduling problem complexity as type of solving NP-hard problems. Therefore, this work focus on the minimization of the complexity of time, decreasing the conflict ratio, and minifying the encoding of the search space are the major points to solve the timetable-scheduling problem. Thus, the (GA) has many features managed through them to execute the above points and solve the problem.

In May 2020, Ashis Kumar Mandal and others<sup>20</sup> investigated an assignment approach for a partial exam for solving the problem of examination timetabling; the approach called partial graph heuristic organized with a modified great deluge algorithm (PGH-MGD), the result of this approach only produced good solutions that are competitive with those of the previous methods.

Many modern works were produced utilized different optimization algorithms and heuristics techniques for solving timetabling schedule problems like particle swarm optimization and local search algorithm<sup>21</sup>, Prey-predator algorithm<sup>22</sup>, adaptive evolutionary memetic algorithm<sup>23</sup>, cellular memetic algorithm<sup>24</sup> and constructive ordering heuristics<sup>25</sup>. All works proved that the scheduling timetable for education purposes still a complex problem and needs more and more studies.

The main gap in the previous studies as we found that all the data presented as a constant and the resulted timetable is restricted to the number of courses. In our work, all data presented as factors, where more than one timetable can be created, with the possibility of adding or removing some courses. This is useful in some situations, for example, in the case of generating a timetable for students who have special cases, such as those who have met in some

courses, and other uncommon and crisis cases that may encounter the educational process in universities at present.

### The Proposed System Environment

This work suggested a solution for the timetabling problem that many universities faced with the beginning of each academic year and this work aimed to reduce the high cost and the slowness of manual procedure that involved in the generation of optimal timetables.

The system input consists of the various courses, halls of lectures, departments, programs, classrooms, lecturers, and the specification of conditions from which the timetable is constructed. The proposed scheduling system for this work generates optimal timetables using the genetic algorithm principle mainly the selection and the crossover.

Figure 1 shows the requirement of the system working environment as the first step to prepare the inputs to the designed website page, then implements the genetic algorithm program to get the optimal timetable.

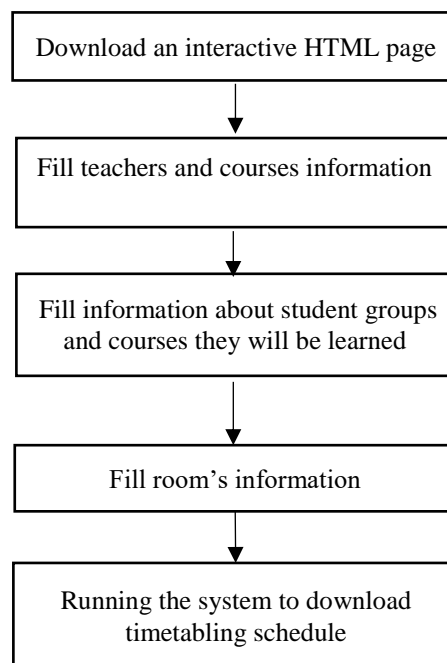


Figure 1. The operating environment of the system

### Design/Implementation of Constraints HARD CONSTRAINS

These are constraints that must be pursued during the search and without meeting these constraints, the solution is not true. These constraints are:

- One teacher cannot give two courses at the same time.

- A set of students cannot have two classes at the same time.
- Two courses cannot be grouped in one classroom at the same time.
- A teacher cannot teach no more than maximum allotted hours.
- Classes Cannot be on off days (Friday and Saturday).
- Maximum number of hours allotted for a day cannot be exceeded.

### Soft Constraints

These constraints that are not mandatory for a solution must be pursued, but the quality of the courses schedule is decided by following these constraints:

- In the timetable of teacher, no more than two lectures a day, each one 1 hours and 30 minutes, breaks are preferred 30 minutes or more.
- In the students' timetable, we have to put three lectures a day at maximum with 30 minutes breaks between lectures so that they have more time for self-study at home.
- We must ensure that a group of students does not have to attend college only to attend one class, there has to be a maximum of three classes in a day and a minimum of two classes so that it will be worth it to come to college on a particular day.

### Assumptions and Dependencies

- At max 4 variables for which computation is required.
- Number of lectures for batches is not greater than available time for the week.
- Number of rooms provided are sufficient enough to hold the classes.
- No ambiguity in the information provided, that is repetition won't be handled by the software.
- Input is done correctly, a bug due to spelling mistake won't be handled, that is for example, at one place its "prolog" at another is "prolog", so these two will be taken as two subjects even might user intended to write the same thing.
- No preferences are allowed for any teacher.

### Functional Requirements

- The page should be available on the internet for the user.
- The page should provide a query if the user wants to make the timetable.
- Option about exiting should always be there.
- The system should be able to take input about teacher and store it.
- The system should be able to check if the number of teachers doesn't overflow and if they do then it should show error message accordingly.
- Then it should be able to provide options about reducing batches or increasing the teachers.
- The system should be able to check if the number of teachers is sufficient for the number of batches by using information about the maximum working hours of the teacher.
- The system should be able to take input about rooms and store it.
- Then the system should be able to give options about adding new rooms.

### Proposed System Block Diagram

The system block diagram is shown in Figure 2 and displays how our proposed system works from the observer's point of view. The use cases in the diagram will display what a system does rather than how and will display the interactions between the system and the user. Use cases act many users named "actors" and they can interact with the system in different ways.

### ACTORS

- The user of the system
- Courses Scheduling System

### USE CASES

- Define the inputs: departments, courses, teachers, and classrooms.
- Fixing the constraints
- Stratify and prove Constraints
- Crossover Course distribution
- Mutate Course distribution
- Generation of random timetable

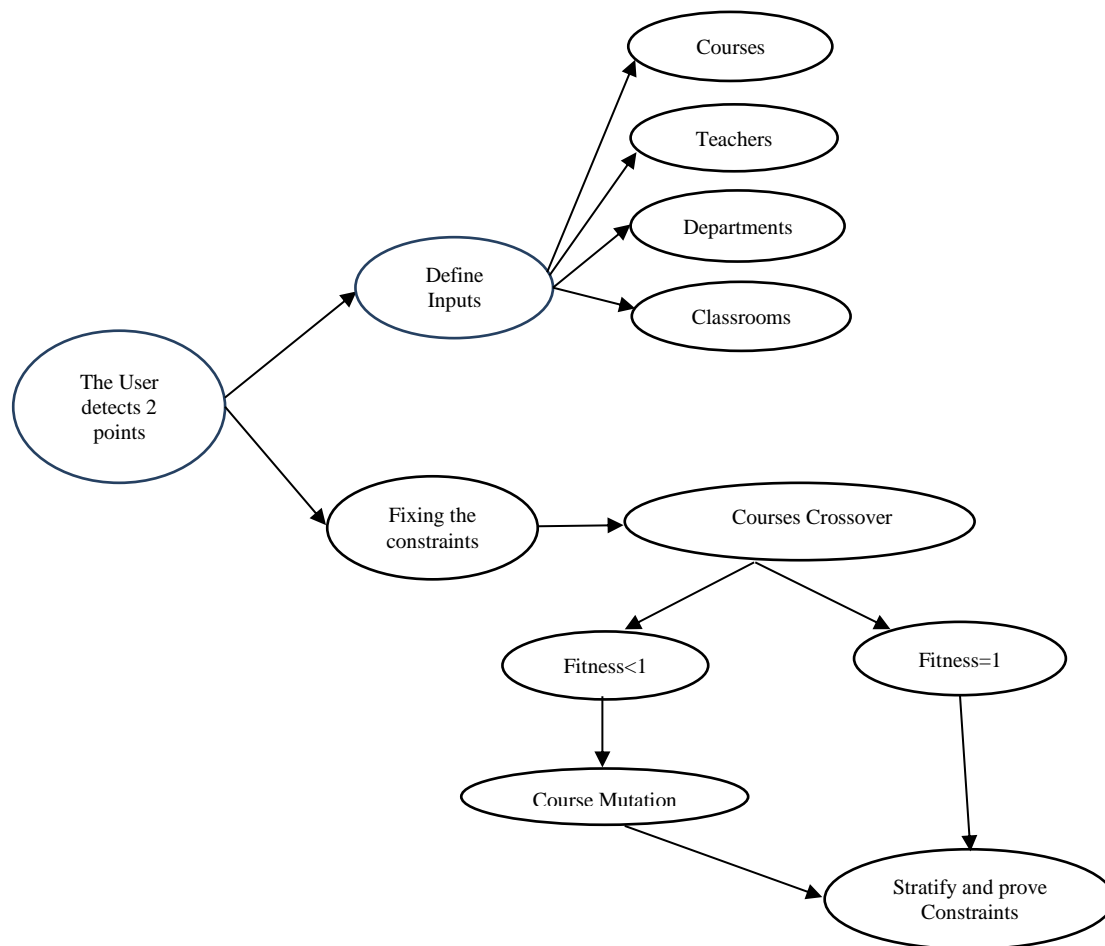


Figure 2. System Block Diagram

**The Proposed EA for Timetable Problem**

A genetic algorithm (GA) is a metaheuristic search optimization technique that shapes the basis of biological genetic and development. The algorithm forms a model of biological development and executions the related accounts. The GA can realize the universal optimization and parallel treating to optimize the configuration of different resources <sup>20</sup>.

**Initial Population: -**

The population contains a group of chromosomes. Each one of these chromosomes is made of a set of genes and the chromosome meets the given hard and soft constraints such as the number of lectures wanted, the time, the available classrooms and labs, lecturers. In this work, the proposed chromosome generation characterized in our case study by the college of computer science in the University of Technology-Baghdad, Iraq which has six branches: Software Engineering (SW), Information System (IS), Artificial Intelligence (AI), Data Security (DS), Computer Networks (NW), and Multimedia (MM), these branches are represented by the

following binary representation as indicated in the Table 1:

**Table 1. Branches Binary Code Representation**

Branches	Proposed binary code
SW	001
IS	010
AI	011
DS	100
NW	101
MM	110

SW: Software Engineering, IS: Information System, AI: Artificial Intelligence, DS: Data Security, NW: Computer Networks, and MM: Multimedia.

All branches have students of 4 classes from class 1 to class 4 with the following binary representation as indicated in the Table 2:

**Table 2. Classes' Binary Code Representation**

class	Proposed binary code
first	001
second	010
third	011
fourth	100

This configures the first chromosome part indicated by branch and class and the generated code.

The department building consists of 18 classrooms and 7 labs, numbered from (0 to 24) with the following binary representation as indicated in the Table 3:

**Table 3. classrooms Binary code representation**

classrooms	Proposed binary code
R1	000000
R2	000001
-	-
-	-
R24	011000
R25	011001

The maximum number of teachers is 64 lecturers represented with the following binary code as indicated in the Table 4:

**Table 4. Lectures Binary Code Representation**

lectures	Proposed binary code
Lec1	000000
Lec2	000001
-	-
-	-
Lec63	111110
Lec64	111111

The lectures time is indicated with a maximum of 3 lectures a day started in three different times (8:30,10:30 and 12:30) are coded by the binary representation of numbers (1, 2 and 3) respectively, each lecture has 1 hour and thirty minutes (1:30) only, the three times are coded as in the following as indicated in the Table 5 :

**Table 5. Lecture Time Binary Code Representation**

Time	Proposed binary code
8:30-10	001
10:30-12	010
12:30-2	011

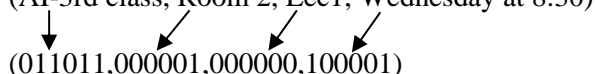
The days when the students and lecturers can meet are five days: Sunday, Monday, Tuesday, Wednesday and, Thursday, these days are coded as described below in Table 6:

**Table 6. Day Binary code representation**

Days	Proposed binary code
Sunday	001
Monday	010
Tuesday	011
Wednesday	100
Thursday	101

In this work, the proposed chromosome is generated from four parts (department code and class, classrooms, lectures, days, and times), each part coded with 6-bit and the total chromosome length is 24-bit.

Example: to represent the chromosome of the students in the artificial intelligence/ third class whose lec1 teach them in room 2 in Wednesday at 8:30 am will be represented as follows:

(AI-3rd class, Room 2, Lec1, Wednesday at 8:30)  
  
 (011011,000001,000000,100001)

The generated chromosome =0110110000010000000100001 (24-bit length).

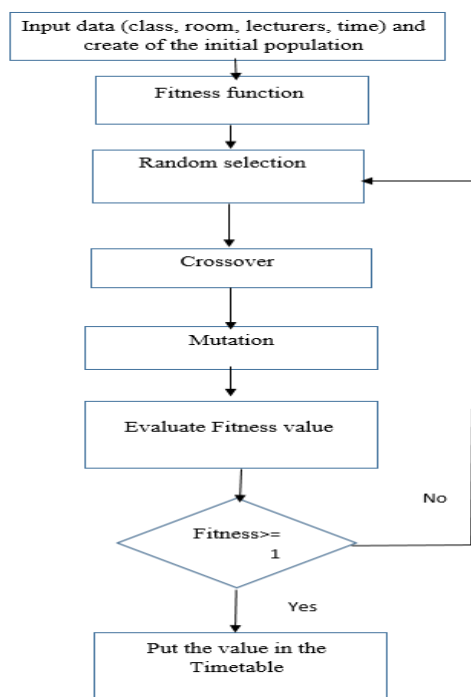
2. Fitness function:-the function tests the fitness value of each chromosome and chooses chromosomes with the highest fitness value for transit. The suggested fitness function is described in equation (1):

$$\text{Fitness} = 1 / (1 - (\text{soft cons.} + \text{hard cons.})) \dots\dots\dots (1)$$

3.Selection:- for selection method, Roulette method is used to select the chromosomes with the highest probability since if the chromosomes have greater fitness values will have more opportunities to pick from the mating pool and participate in the crossover function.

4. Crossover:- the goal is to crossover genes of two actively chromosomes. There are different crossover methods but we have selected the single-crossover in different positions. In the single crossover, we choose a specific point of the two chromosomes and exchange them to obtain the new generation for the next process. The proposed crossover exchanged the first gene of the branch class code, the second gene of the classroom code, the third gene of the lecture name code, and the fourth gene of the day\time code.

5.Mutation:- In the mutation, the best two chromosomes combine to build a random population once more and then we verify it's optimization through evaluation function. Figure 3 shows the steps of the proposed genetic algorithm for the timetable schedule.



**Figure 3. The Proposed Genetic Algorithm for TTS**

Assume  $P_i$  is the parent population and  $Chi$  are children that are generated from  $P_i$  in current iteration  $i$ .  $C$  is the number of chromosomes in  $P_i$ .

**Algorithm (2): Proposed Genetic Algorithm for solving TTS**

```

Begin
i ← 0;
Initialize the first population  $P_i$ 
Evaluate  $P_i$ 
While termination criteria not met do
Apply genetic operators to generate  $Chi$ 
1: Reproduction and random generation of( $P_i$ ) ;
For each generation
Begin
Choose good solutions to generate a new population
Create from parents new solutions
Compute the fitness value for new solutions
Substitute old population with new ones
End for
2: Apply Crossover ( $P_i$ );
for each gene in ( $p_1, p_2$ )
Begin
if( $p_1(\text{gene}) \neq p_2(\text{gene})$ )
Begin
ch(gene)= $p_1(\text{gene})$ ;
End if
else
Begin
ch(gene)=random( $p_1, p_2$ )(gene);
End else
End for
Step3: Apply mutation ( $P_i$ ) if needed;
for each gene in individual {if ( $p(\text{Random}) < p_i$ )
Begin
gene = picking a value randomly from the list of possible values;
Step4: Evaluate fitness;
End loop
pick  $P(i+1)$  from  $P_i$  and  $Chi$  ;
i=i+1;
End while
End
    
```

```

ch(gene)= $p_1(\text{gene})$ ;
End if
else
Begin
ch(gene)=random( $p_1, p_2$ )(gene);
End else
End for
Step3: Apply mutation ( $P_i$ ) if needed;
for each gene in individual {if ( $p(\text{Random}) < p_i$ )
Begin
gene = picking a value randomly from the list of possible values;
Step4: Evaluate fitness;
End loop
pick  $P(i+1)$  from  $P_i$  and  $Chi$  ;
i=i+1;
End while
End
    
```

**Results:**

To enter the schedule of the materials for each stage, we can press the "create table for class 1" or "create table for class 2" and so on for the other of classes. We can also edit or update the schedule we already entered. Figure 4 shows the main interface of the timetable system and Figures 5 and 6 show the generated timetable schedule for courses of artificial intelligent branch for the first stage, second stage, third stage and fourth stage respectively.



**Figure 4. The Main System Interface**

Table for class One	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
08:30 10:00	Structured Programming	Discrete Structures	Mathematics	Software Development Fundamentals	Software Engineering	
10:30 12:00		Logic Design	Probability Theory	Discrete Structures		Introduction to Statistics
12:30 02:00	Mathematics	Structured Programming			Computer Organization	English Language

Table for class Two	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
08:30 10:00	Object Oriented Programming	Numeric Analysis		Mathematics	Software Engineering	Database Foundation
10:30 12:00	Sorting and Searching Algorithms	Databases Design	Democracy	Database Foundation	Human Rights	Mathematics
12:30 02:00	Object Oriented Programming	Analysis and Design of Algorithms	Data Structures	Software Engineering	Computational Complexity	

Figure 5. The Generated Timetable Schedule for 1<sup>st</sup> class on the left and 2<sup>nd</sup> class on the right

Table for class Three	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
08:30 10:00	Microprocessor	Computer Graphics and Visualization	Information Retrieval Techniques	Compiler Design		
10:30 12:00	Computation Theory	Parallel Programming Paradigms		Computer Networks	Computer Graphics and Visualization	Mobile Application Design
12:30 02:00	Machine Learning	Software Modelling and Analysis	Computer Architecture	Data Mining and Data Warehousing	Software Design	Computer Graphics and Visualization

Table for class Four	Saturday	Sunday	Monday	Tuesday	Wednesday	Thursday
08:30 10:00	Computer and Data Security	Windows Programming		Intelligent Applications		Graduation Project
10:30 12:00	Operating Systems	Intelligent Applications	Operating Systems		Web Programming	Image Processing
12:30 02:00		Image Processing	Computer and Data Security	Web Programming	Modeling and Simulation	

Figure 6. The Generated Timetable Schedule for 3<sup>rd</sup> class on the left and 4<sup>th</sup> class on the right

## Discussion:

Scheduling timetables for courses for the big departments in the universities is a very hard problem and is often be solved by many previous works although results are partially optimal because some works were taken in their consideration that the input data are constants as shown in <sup>16, 17, 21</sup>. Other works depended on fitness value to organize the timetable as indicated in (18) and the proposed partially exam timetable presented in <sup>20, 22</sup>. Also, <sup>15</sup> dealt with the scheduling problem on a narrow and specific scale. While our work implements the principle of an evolutionary algorithm by using genetic theories to solve the timetabling problem as trying to get a random and optimal timetable with the ability to generate a multi optimal timetable for each stage in the collage. The ability of generated multi-solution gives our work more flexibility since there is no conflict in any condition or event with our results.

The execution of our algorithm is very promising and comes to steady-state after a few cycles, so the algorithm will stop in a case when no change on the timetable. Table 7 illustrates the parameter used in our algorithm.

Table 7. The proposed Parameter Used

Parameter	value
Population size	100
Crossover rate	0.8
Mutation rate	0.02
Crossover type	Single point
Selection	Roulette method
Chromosome length	24-bit

Some advantages and characteristics can be discussed from our proposed work; these can be summarized as follows:

## The Proposed System Advantages

- Flexibility of the system since GA generated many solutions and the user can ask for change every time needed.
- Power with minimal processing.
- Decreases the time demanded to create optimal timetables.
- It produces an easy transmit for data entry and revision.
- High productivity.
- Eliminates the need for paperwork.



### User Characteristics

- Input and Storage: the system will take all the input from the user and will save it in its database for all the future computation.
- Creating Random Solutions: using the above input by using the appropriate algorithm will generate some random solutions.
- Calculating Fitness Function: by using the already decided penalty, the system will calculate the fitness functions of the solutions and will sort them in increasing.
- Applying Genetic Algorithm through Mutation: By using, a fitness score system will apply genetic algorithm or its improved version to create the best possible solution in minimum possible time.
- Efficiency: the system will try to take minimum time and resources.
- User-Friendly Interface: no knowledge about the software will be required to use it, very straight design

### Conclusions:

The Genetic algorithm in the timetabling structure has proven to be effective in many virtual problems. The genetic algorithm has proven that it plays a good role in discovering the interest space even in a hard and realistic world outlook. This work clarifies how the set of active constraints is used to inspire intelligent knowledge and how the GA can be implemented to determine the priority in the constraints in the dynamical scope of the developing environments. This work confirms that GA can locate the local optimum then stop. This is constantly a risk with GA, but it relies on the search area.

GA applies the proposed course timetable since there are several reasonable solutions and the GA will return one of them. In some situations, the GA may fail when there is a single reasonable solution, but once more it can be reactivated the search by the active constraints with numerous solutions to locate the best one of them. Nothing can bar this structure from being a part of an expensive and strong event/action algorithm. In such a case, it can be applied to select the constraint to be worked when there are not another criteria for selecting the constraint. In another case, instead of having several constraints, which will make the active constraints allocations complicate and increased the design, consequently by increasing the periods for test and maintenance. This approach has many advantages that are the easy design with the decreasing of the time needed for the development and the

maintenance of the system constraints in the scene of dynamically evolving environments.

### Authors' declaration:

- Conflicts of Interest: None.
- We hereby confirm that all the Figures and Tables in the manuscript are mine ours. Besides, the Figures and images, which are not mine ours, have been given the permission for re-publication attached with the manuscript.
- Ethical Clearance: The project was approved by the local ethical committee in University of Technology.

### Authors' contributions statement:

AA Israa proposed the work, designed the figures and the tables, analyzed and compared the results. MA Sura collected the data related to the previous studies, discussed and compared the results. Both authors read the manuscript carefully and approve the final manuscript.

### References:

1. Marczyk A. Genetic algorithms and evolutionary computation. The Talk Origins Archive: <http://www.talkorigins/faqs/genalg/genalg.html>. 2004 Apr.
2. Coello CA. An updated survey of GA-based multiobjective optimization techniques. *ACM Computing Surveys (CSUR)*. 2000 Jun 1;32(2):109-43.
3. Bhattacharjya RK. Introduction to genetic algorithms. IIT Guwahati. 2012 Oct 19;12.
4. Hammood MM. Application of data mining algorithm with genetic algorithm. *TJPS*. 2008;13(3):9-16.
5. Affenzeller M, Wagner S, Winkler S, Beham A. Genetic algorithms and genetic programming: modern concepts and practical applications. Crc Press; 2009 Apr 9.
6. Willemen RJ. School timetable construction--Algorithms and complexity;(2002).
7. Sigl B, Golub M, Mornar V. Solving timetable scheduling problem using genetic algorithms. In *Proceedings of the 25th International Conference on Information Technology Interfaces*, 2003. ITI 2003. 2003 Jun 19 (pp. 519-524). IEEE.
8. Bäck T, Fogel DB, Michalewicz Z, editors. *Evolutionary computation 1: Basic algorithms and operators*. CRC press; 2018 Oct 3.
9. Burke EK, Kendall G. *Search methodologies*. Springer Science+ Business Media, Incorporated; 2005.
10. Reeves C, Rowe JE. *Genetic algorithms: principles and perspectives: a guide to GA theory*. Springer Science & Business Media; 2002 Dec 31.
11. Koza JR, Keane MA, Streeter MJ, Mydlowec W, Yu J, Lanza G. *Genetic programming IV: Routine human-competitive machine intelligence*. Springer Science & Business Media; 2006 Mar 4.

12. Chambers LD, editor. Practical handbook of genetic algorithms: complex coding systems. CRC press; 2019 Jul 17.
13. Ansari A, Bojewar S. Genetic Algorithm to Generate the Automatic Time-Table–An Over View. IJRITCC. 2014;2(11):3480-3.
14. Timilsina S, Negi R, Khurana Y, Seth J. Genetically Evolved Solution to Timetable Scheduling Problem. Int J Comput Appl. 2015 Jan 1;114(18).
15. Abdelhalim EA, El Khayat GA. A utilization-based genetic algorithm for solving the university timetabling problem (uga). AEJ. 2016 Jun 1;55(2):1395-409..
16. Soyemi J, Akinode J, Oloruntoba S. Electronic Lecture Time-Table Scheduler Using Genetic Algorithm. In 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech) 2017 Nov 6 (pp. 710-715). IEEE.
17. Syahputra MF, Apriani R, Abdullah D, Albra W, Heikal M, Abdurrahman A, et al Genetic algorithm to solve the problems of lectures and practicums scheduling. In IOP Conference Series: Materials Science and Engineering 2018 Feb (Vol. 308, No. 1, p. 012046). IOP Publishing..
18. Ahmad IR, Sufahani S, Ali M, Razali SN. A Heuristics Approach for Classroom Scheduling Using Genetic Algorithm Technique. In Journal of Physics (IOP Science): Conference Series 2018 Apr (Vol. 995, No. 1, p. 012050). IOP Publishing..
19. Deeba K, Kuntal B, Samriddho R. Solving Timetable Scheduling Problems Using Genetic Algorithm. International IJRTE. February 2019; 7(5C):168-170.
20. Mandal AK, Kahar MN, Kendall G. Addressing Examination Timetabling Problem Using a Partial Exams Approach in Constructive and Improvement. Computation. 2020 Jun;8(2):46.
21. Abayomi-Alli O, Abayomi-Alli A, Misra S, Damasevicius R, Maskeliunas R. Automatic examination timetable scheduling using particle swarm optimization and local search algorithm. In Data, Engineering and Applications 2019 (pp. 119-130). Springer, Singapore.
22. Tilahun SL. Prey-predator algorithm for discrete problems: a case for examination timetabling problem. Turk J Elec Eng Comp Sci. 2019 Mar 1;27(2):950-60.
23. Lei Y, Gong M, Jiao L, Shi J, Zhou Y. An adaptive coevolutionary memetic algorithm for examination timetabling problems. IJBIC. 2017;10(4):248-57.
24. Leite N, Fernandes CM, Melicio F, Rosa AC. A cellular memetic algorithm for the examination timetabling problem. COR. 2018 Jun 1;94:118-38.
25. Pillay N, Özcan E. Automated generation of constructive ordering heuristics for educational timetabling. Ann. Oper. Res. 2019 Apr 1;275(1):181-208.

## خوارزمية تطويرية لحل مشكلة جدولة توقيتات المقررات الأكاديمية

سرى محمود عبدالله

اسراء عبد الامير عبد الجبار

قسم علوم الحاسوب، الجامعة التكنولوجية، بغداد، العراق.

### الخلاصة:

جدولة أوقات الدروس في الأقسام الكبيرة في الجامعات تعتبر مشكلة صعبة للغاية وغالبًا ما يتم حلها من قبل الموظفين على الرغم من أن النتائج مثالية بشكل جزئي. لقد حددت للغاية مشكلة جدولة الوقت من مشكلة التحسين التوافقي، يهدف هذا العمل تطبيق مبدأ الخوارزمية التطويرية باستخدام النظريات الوراثة لحل مشكلة الجدولة الزمنية في محاولة الحصول على جدول زمني عشوائي ومثالي مع القدرة على إنشاء جدول زمني متعدد الاحتمالات ومثالي بشكل كامل لكل مرحلة دراسية في القسم المعني وبما يتلاءم مع القيود التي يفرضها الطلبة والكادر التدريسي وضمن ساعات عمل محددة مسبقًا. تتمثل الفكرة الرئيسية في إمكانية إنشاء جداول زمنية للدروس بطريقة تلقائية بعد تحديد الشروط المجدية للحصول على جدول زمني مرن ومثالي بدون تكرار من خلال تقديم جدول زمني قابل للتبديل والتدوير. تكمن المساهمة الرئيسية في هذا العمل من خلال زيادة مرونة توليد جداول زمنية مثالية بنسخ مختلفة من خلال زيادة احتمال إعطاء أفضل جدول زمني لكل مرحلة في الحرم الجامعي مع القدرة على استبدال الجدول الزمني عند الحاجة. الخوارزمية التطويرية (EA) المستخدمة في هذه الورقة هي الخوارزمية الجينية (GA) التي هي عبارة عن بحث متعدد الحلول يعتمد على عدد المجتمع التطويري الذي يمكن تطبيقه لحل مشاكل معقدة مثل مشاكل الجدول الزمني. في هذا العمل، جميع المدخلات: الدروس والكادر التدريسي والوقت قد تمثلت بمجموعة واحدة لتحقيق البحث المحلي ودمج هذا التمثيل للجدول الزمني باستخدام التبادل الموجه لضمان عدم خرق الشروط الأساسية التي تم تحديدها مسبقًا كدالة تطابق. قدمت النتائج نظام جدولة مرن حيث أظهرت نتائج الاختبار تنوع جميع الجداول الزمنية الممكنة التي يمكن إنشاؤها بما يتلاءم مع شروط المستخدم وحاجاته.

الكلمات المفتاحية: القيود، الخوارزميات التطويرية، دالة التطابق، الخوارزمية الجينية، الجدولة الزمنية.