الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

University of Badji Mokhtar - Annaba　　　　جامعة باجي مختار – عنابـــة

Faculty of Technology

Department : Computer science

Domain : Mathematics-Computer science

Sector : Computer science

Speciality: « systèmes informatiques »

Thesis

Presented in order to obtain the License 's degree

Theme

Genetic Algorithm for solving timetable problem

Presented by : Fakir Siyaf

Supervisor : ATIL F.	Prof.	Badji Mokhtar Annaba University
Co-Suervisor : GHEDIR L.	Ph.D Student	Badji Mokhtar Annaba University

Academic year : 2022/202

Acknowledgments

First of all, we must praise Allah for giving us the power and courage to finish this project.

I would like to express my deepest gratitude to my supervisors Pr ATIL Fadilla and Dr GHEDIR Lina whose direction and ideas helped me throughout this project.

I would like to thank all of our professors from my past 3 years at the University of Badji Mokhtar Annaba.

Lastly, thanks to everyone who supported me and played a role in our project, including family and friends.

Thanks to you all.

Dedication

To my family, this is yet another milestone for me. I could not have made it without your support. I love you and appreciate you all.

To all my friends in all over Algeria and beyond who supported me during my studies, I will forever be grateful.

Table of contents

Tables of figures

Introduction

Context of the project:

Timetable problem solving is an important and challenging task that involves the allocations of resources such as rooms, teachers, and time slots of the set of events, while satisfying various constraints.

This problem occurs in many real-world scenarios, such as universities, schools, and organizations where a large number of courses or events need to be scheduled efficiently.

Problem statement:

These constraints can include the availability of resources, the preferences of teachers and students, and the need to avoid conflicts between events.

Some challenges in timetable problem solving can include dealing with large and complex problem instances, handling uncertainty and changes in the availability of resources, and finding solutions that are both feasible and optimal.

Objectives:

The main objective of this final year project is to design and implement a genetic algorithm approach for solving timetable problems. The proposed approach will involve the development of a fitness function that evaluates the quality of candidate solutions based on various criteria.

Question:

1-How can genetic algorithm be applied to solve timetable scheduling problems, and what are the key steps involved in this approach?

Thesis structure: this thesis is composed of four chapters that progress from the general scope of timetable problem to the implementation of our solution.

Chapter 1: provide an overview of the genetic algorithm.

Chapter 2: introduces the generalities of timetable problems and solutions.

Chapter3: explore the major steps of applying genetic algorithm to solve timetable problems

Chapter4: provide a detailed description of the implementation of GAschedules.

Chapter 1: the genetic algorithm

1. Introduction:

In this chapter will provide an overview of the genetic algorithm a powerful optimization technique inspired by the principles of natural selection.

2. Bioinspired algorithms:

Bioinspired optimization algorithms are techniques that effectively address optimization issues in a wide range of application domains by drawing on physical laws, evolutionary theory, and specific traits of living things.

The first optimization model based on Charles Darwin's theory of evolution, known as the Genetic Algorithm (GA), was proposed by John Holland et al. In 1975. Due to the advantages that genetic algorithm, offer over conventional optimization techniques and the increase in computing power of technology, a large number of bioinspired algorithm have been developed from the year 2000, ranging from those that simulate the alignment of practices during the cooling of materials (Simulated Annealing) to those that model the movement of galaxies (Galaxy-based Search Algorithm) to those that are based on the group behavior of specific animal species. Some of the most significant algorithms are shown in Figure 1 [3].

Figure 1: Examples of bio-inspired optimization algorithms [3].

3. Natural selection:

Natural selection in is the process through which population of living organisms adapt and change. Individuals in population are naturally variable, meaning that they are all different in some ways. This variation means that some individuals have traits better suited to the environment than others. Individuals with adaptive traits that give them some advantage are likely to survive and reproduce. These individuals then pass the adaptive traits on to their offspring. Over time, these advantageous traits become common in the population. Through this process of natural selection, favorable traits are transmitted through generations.

Natural selection can lead to speciation, where one species gives rise to a new and distinctly different species. It is one of the processes that drives evolution and helps to explain the diversity of life on earth [4].

4. Genetic algorithm:

Genetic Algorithm is a search heurist that is modeled after Charles Darwin’s theory of naturel selection, which holds that the fit individuals are selected for reproduction in order to produce offspring of the next generation [5].

The algorithm establishes a model of biological evolution and implements the relevant calculation. The genetic algorithm can realize global optimization and parallel processing to optimize the configuration of various resources to make our task easier [2].

Generally, there is five phases in a genetic algorithm:

4.1 Initial population :

The process begins with a set of individuals, which is called a Population. Each individual is characterized by a set of parameters (variables) known as Genes, which are joined into a string to form a Chromosome (solution) like shown below (Figure 2).

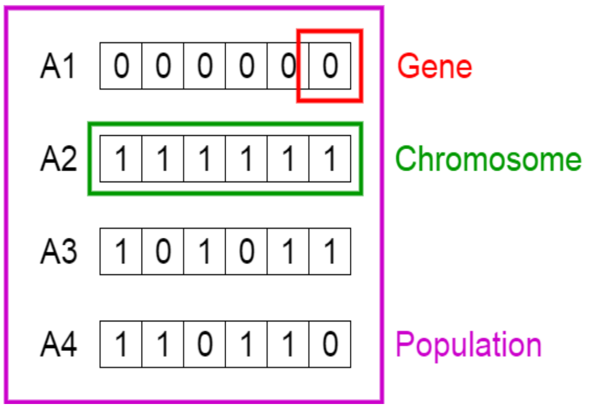


Figure 2: Population, Chromosomes and Genes [5].

In a genetic algorithm, the set of genes of an individual is represented using a string, in terms of an alphabet. Typically, binary values are used (string of 1s and 0s) [5].

4.2 Fitness function :

The fitness function assigns each individual a fitness score based on how fit they are (their capacity to compete with other individuals), and the probability that they will be chosen for reproduction is based on their fitness score [5].

4.3 Selection :

The chromosome with higher fitness functions will have a higher chance of being chosen to participate in the crossover function, which is done using the roulette method or tournament selection [2]

4.4 Crossover :

There are various methods of crossover but we have chosen the single crossover.

In single crossover, we choose a random point on each chromosome, the right sides of both chromosomes swap, and we produce a new offspring that is prepared for further processing. The goal is to crossover genes of two actively participating chromosomes [2].

4.5 Mutation:

This implies that some of the bits in the bit string can be flipped in certain new offspring formed, where some of their genes can be subjected to a mutation with low random probability (Figure 3) [5].

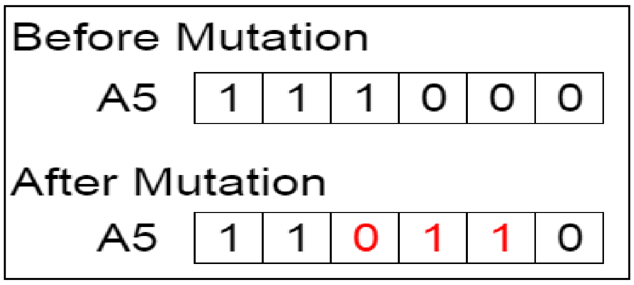
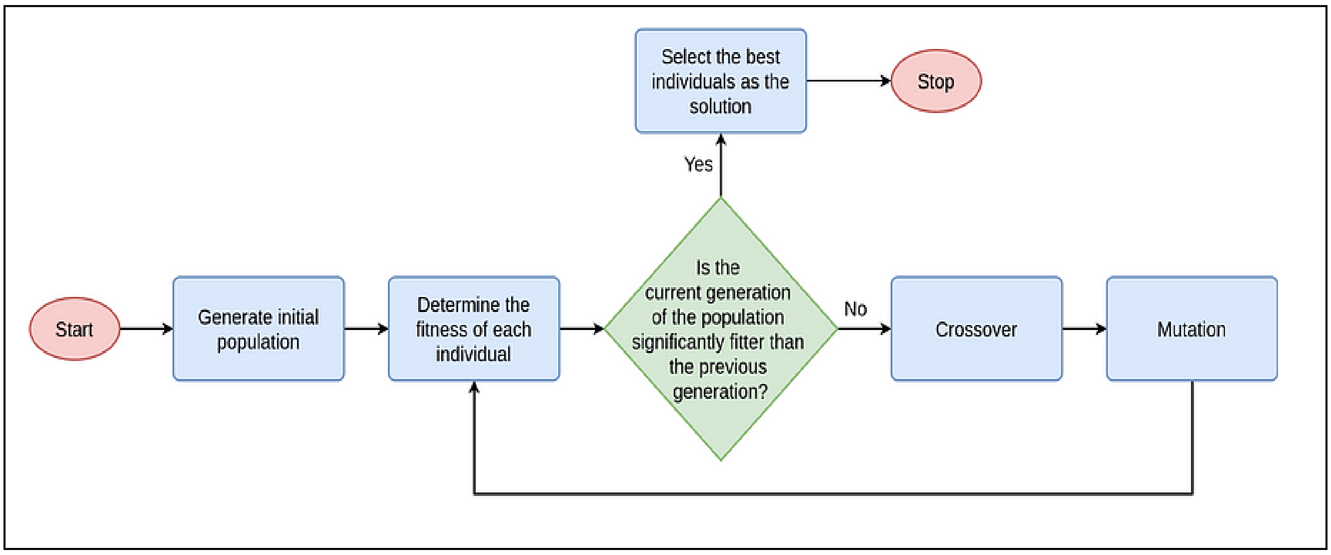


Figure 3: Mutation: before and after [5].

To summarise, the following figure denotes the flow chart of how a general algorithm works (Figure 4).



Start
Generate initial population
Determine the fitness of each individual
Is the current generation of the population significantly fitter than the previous generation?
Yes
Select the best individuals as the solution
Stop
No
Crossover
Mutation

Figure 4: Flow chart of how a general genetic algorithm works [5].

5. Conclusion:

Throughout this chapter, we have delved in the generalities of genetic algorithm by exploring all the phases. The upcoming chapter will provide a detailed description of our approach, outlining how genetic algorithm can be applied to solve the timetable scheduling problems.

Chapter 2: Timetable scheduling

1. Introduction:

Scheduling problem can be classified into various types based on their characteristics and application domains; one specific type of scheduling problem is the timetable scheduling. In this chapter, we will approach the generalities of the timetabling problem by presenting the necessary definitions derived for our solution.

2. Scheduling problems:

Scheduling concerns the allocation of limited resources to tasks over time its decision-making process that has a goal the optimization of one or more objectives.

Scheduling problems are encountered in all types of systems, since it is necessary to organize or distribute the work between many entities. Generally, we can distinguish between those timetable-scheduling problems [1].

3. Timetable scheduling problem:

Time table scheduling is a NP hard problem, concern to finding the exact time allocation within limited time of number of events (courses-lectures) and assigning to them number of resources (teachers, students and classrooms) 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 [2].

3.1 NP hard problem:

NP (non deterministic polynomial time) problem is a class of decision problems that cannot be solved in polynomial time on any real computer, unlike problems in the class”p”.

A problem is considered NP-hard if there exists a reduction (i.e., a transformation from one problem to another) to the NP class and it requires at least an exponential algorithm to solve it [6].

3.2 Timetable as an NP hard problem:

Although manual scheduling is time consuming and inaccurate, small universities adapt to generate their schedules. As the complexity of university increases it to become necessary to adopt computer methods to ease the task of timetabling.

It is to be noted that when the student population with diverse interests and requirements increases and the teaching programs get complicated with the growth of university, the number of constraints grows resulting in an exponential rise in the computational time, making it an NP-complete operation [12].

3.3 Timetabling: Soft and hard constraints :

3.3.1 **Soft constraints**:

The soft constraints include assigning "hard courses" to time slots in the morning sessions when students can concentrate on such subjects, avoiding changing rooms after every lecture unless small groups must move to cramped spaces like laboratories, and staff preferences like teaching at particular times and classrooms of choice are other constraints [12].

3.3.2 **Hard constraints**:

- A teacher could not attend two classes at the same time.
- A course could not be taught in two different classes at the same time.
- A teacher teaches only one course in one room at each timeslot.
- At each daily timeslot in one room, only one group of students and one teacher could attend.
- A teacher teaches for only one group of students at each daily timeslot.
- There are some predefined courses, which are scheduled, in a given timeslots.
- The capacity of the classrooms should be proportional to the number of students of the given course [13].

4. Timetable scheduling solutions:

Timetable problems can have various solutions depending on the specific constraints and requirements. Here are some common approaches and techniques used to solve timetable problems:

4.1 Constraint programming (CP):

CP Is an approach for formulating and solving discrete variable constraint satisfaction or constrained optimization problems, which can be used to find feasible solutions to timetable problems.

That systematically employs deductive reasoning to reduce the search space and allows for a wide variety of constraints [7].

4.2 Integer linear programming (ILP):

ILP is a mathematical optimization technique used to solve optimization problems where the decision variables are required to take integer values, which deals with continuous variables [8].

4.3 Graph coloring:

Graph coloring is the procedure of assignment of colors to each vertex of a graph G such that no adjacent vertices get same color. The objective is to minimize the number of colors while coloring a graph. The smallest number of colors required to color a graph G is called its chromatic number of that graph. Graph coloring problem is a NP Complete problem [9].

4.4 Mathematical modeling:

Mathematical modeling refers to the process of creating a mathematical representation of a real-world scenario to make a prediction or provide insight. There is a distinction between applying a formula and the actual creation of a mathematical relationship [10].

4.5 Heuristic algorithms :

Heuristic algorithms are used to solve NP problems and decrease the time complexity of problems by giving quick solutions; it is popularly utilized in artificial intelligence problems [11].

5. Conclusion:

Throughout this chapter, we have explored into the generalities of timetable scheduling problems and solutions, by examining the properties of timetables, and the constraints involved. In our work, we will focus on solving the timetable problem using genetic algorithm. The upcoming chapter will provide a detailed description about the genetic algorithm.

Chapter 3: optimizing timetable problem using genetic algorithm

1. Introduction:

In this chapter, we will provide a comprehensively explore the major steps involved in applying the genetic algorithm to solve timetable problems in our solution.

2. System development:

The system represents a windows application that aims to address the challenges associated with creating efficient timetables.

The development of this system involves utilizing genetic algorithm, by simulating evolutionary processes; the genetic algorithm explores different combinations and variations of timetables to identify the most optimal solution.

The application's interface provides a user-friendly experience, allowing users to input timetable data like departments, rooms, teachers, and time slots. Through an iterative process of generating and evaluating timetables, the system employs genetic operators like selection, crossover, and mutation to improve the solutions overtime.

The resulting timetables not only meet various constraints but also seek to optimize factors such as class schedules, room allocations, and instructor assignments.

By leveraging the capabilities of genetic algorithm, this application offers an effective and automated approach to efficiently solve tome table problems.

2.1 UML modeling:

This activity diagram in figure 5 illustrates the various interactions between the user and the system:

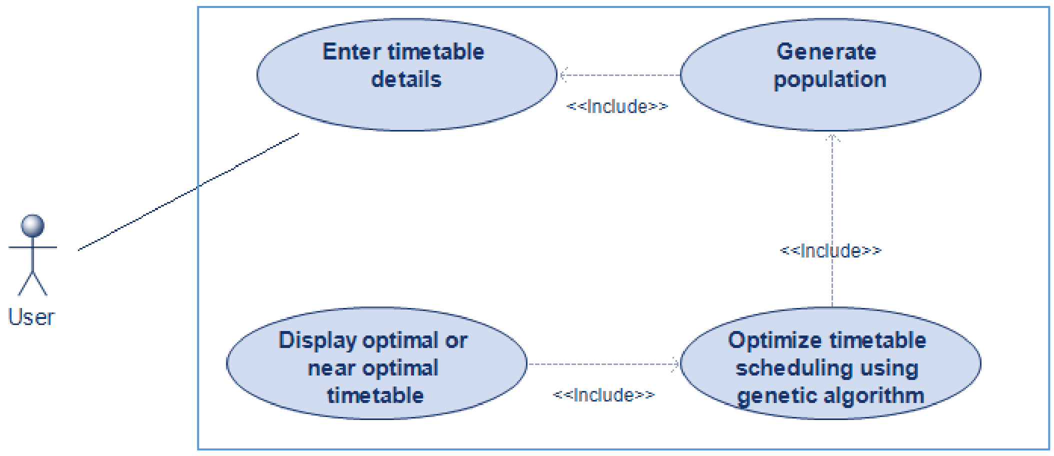

Enter timetable details
Generate population
<<Include>>
<<Include>>
User
Display optimal or near optimal timetable
<<Include>>
Optimize timetable scheduling using genetic algorithm

Figure 5: Use case diagram

This class diagram in figure 6 illustrates the various interactions between the user and the system:

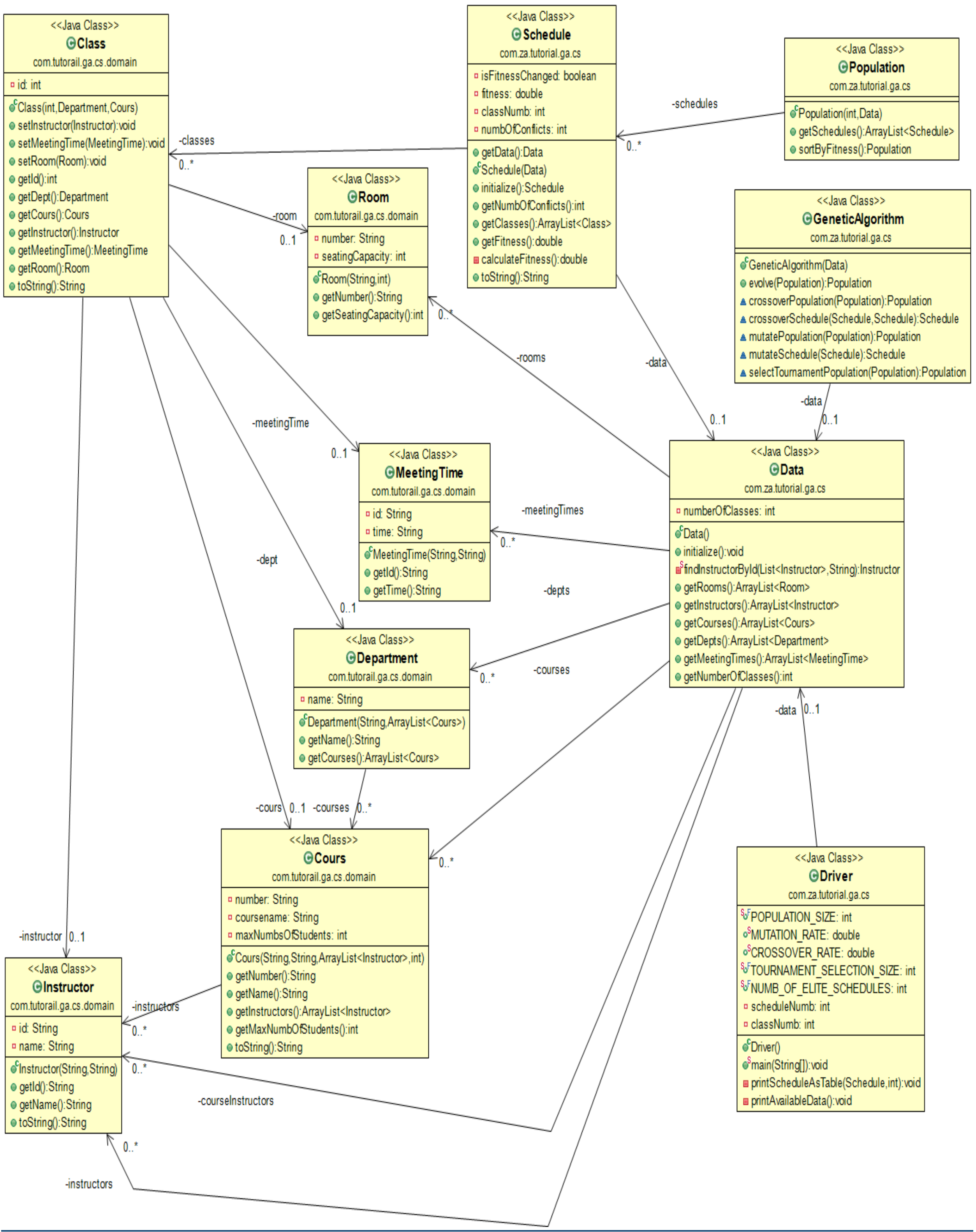
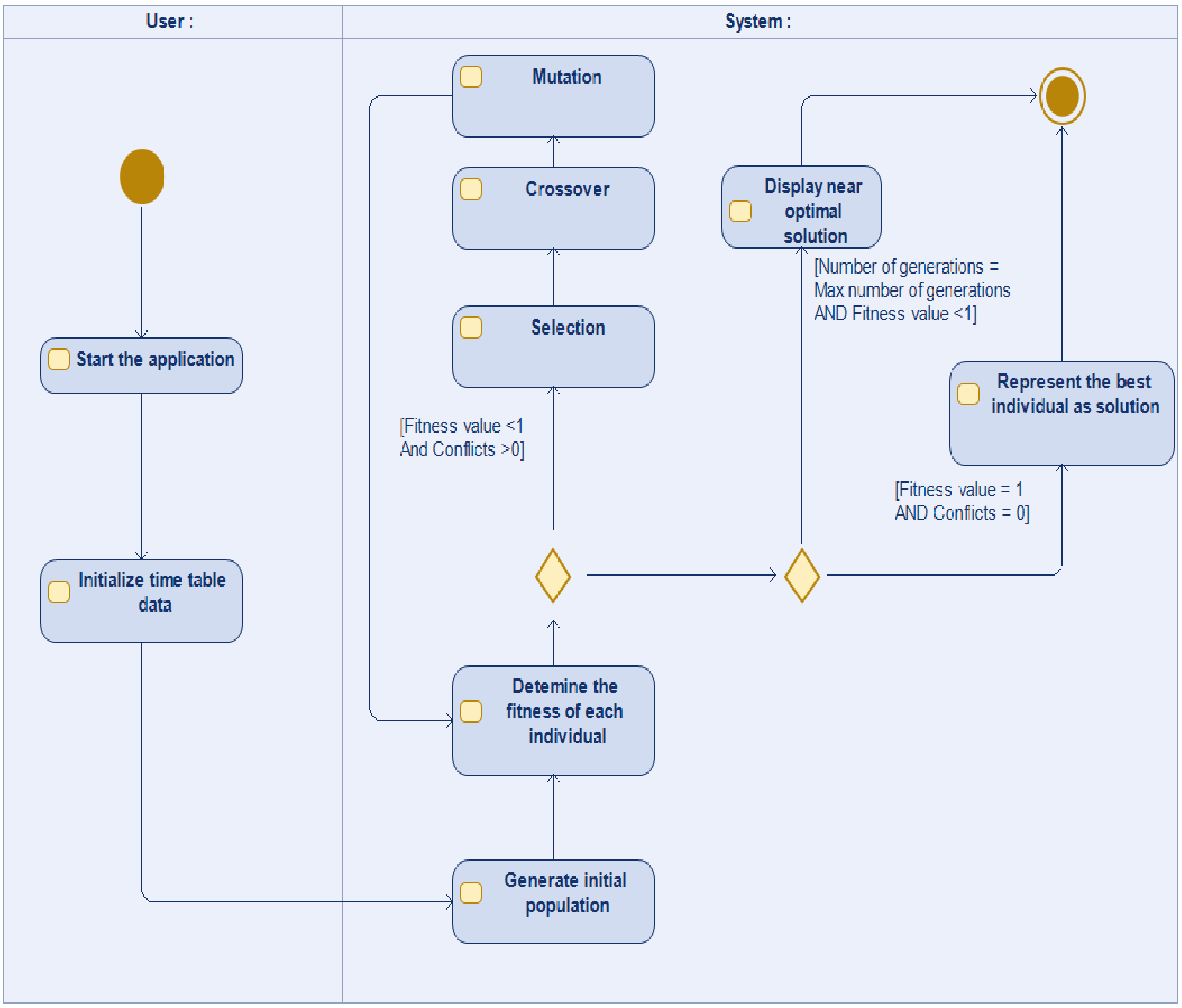


Figure 6: Class diagram

This activity diagram in figure 8 illustrates the various interactions between the user and the system:



User :
System :
Start the application
Initialize time table data
Generate initial population
Detemine the fitness of each individual
[Fitness value <1 And Conflicts >0]
Selection
Crossover
Mutation
Display near optimal solution
[Number of generations = Max number of generations AND Fitness value <1]
Represent the best individual as solution
[Fitness value = 1 AND Conflicts = 0]

Figure 7: Activity diagram

3. Optimizing time table scheduling using genetic algorithm:

Genetic algorithm can be applied to solve time table scheduling problems be imitating the process of natural evolution. The goal is to find an optimal or near-optimal solution that satisfies various constraints, such as class scheduling, room assignments, and teacher availability.

Here is a general concept of how genetic algorithm can be used for a timetable-scheduling problem:

3.1 Encoding:

Define a suitable encoding representation for the timetable that effectively captures the required information about the schedule, including classes, time slots, rooms, and teachers.

Each class will be assigned a department name, course name, room, instructor, and a meeting time by the class scheduler.

This encoding can take the form of a string, and array, or a well-designed data structure in our work we decided to in an array list (figure 8).

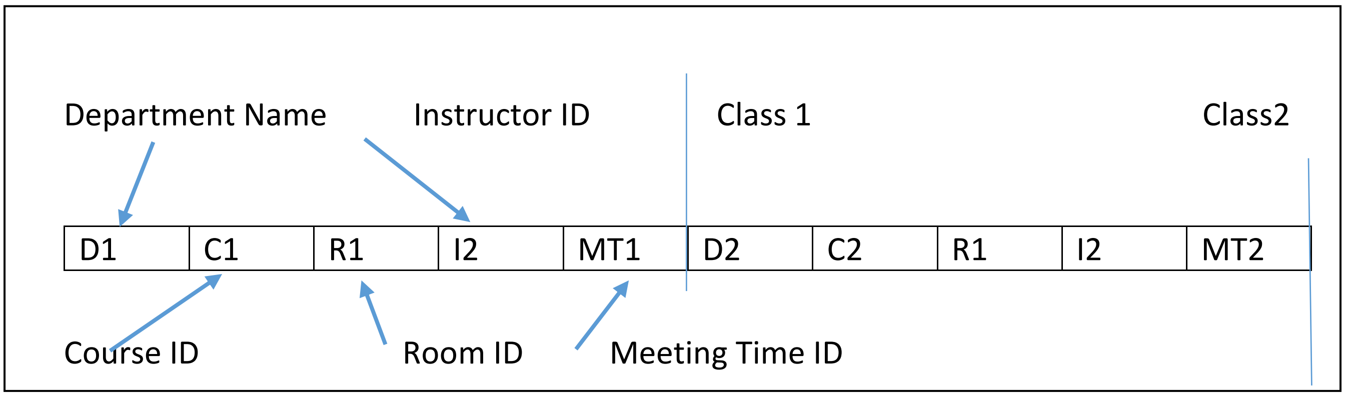


Figure 8: Array is split into five to retrieve information for each class

3.2 Initialization:

Generate an initial population of potential solutions. Each individual in the population represents a possible timetable.

These individuals serve as the starting point for the genetic algorithm.

In our project, a string code can be allocated to each department, course, room, instructor, meetingTime to form a chromosome like shown below:

[Department, Course, Room, Instructor, MeetingTime].

[D1,C1,R1,I1,MT1]

3.3 Fitness evaluation:

Define a fitness function that evaluates the quality of each individual in the population.

The fitness function should consider constraints in the timetable problem, such as:

- preventing any overlapping of classes in the same room
- maximizing room and teachers utilization,
- minimizing gaps in the schedule.

Assign a fitness score to each individual, reflecting how well it satisfies the defined criteria.

3.4 Selection:

Apply selection techniques, in our solution we choose tournament selection to choose individuals from the population for reproduction based on their fitness score (Figure 9).

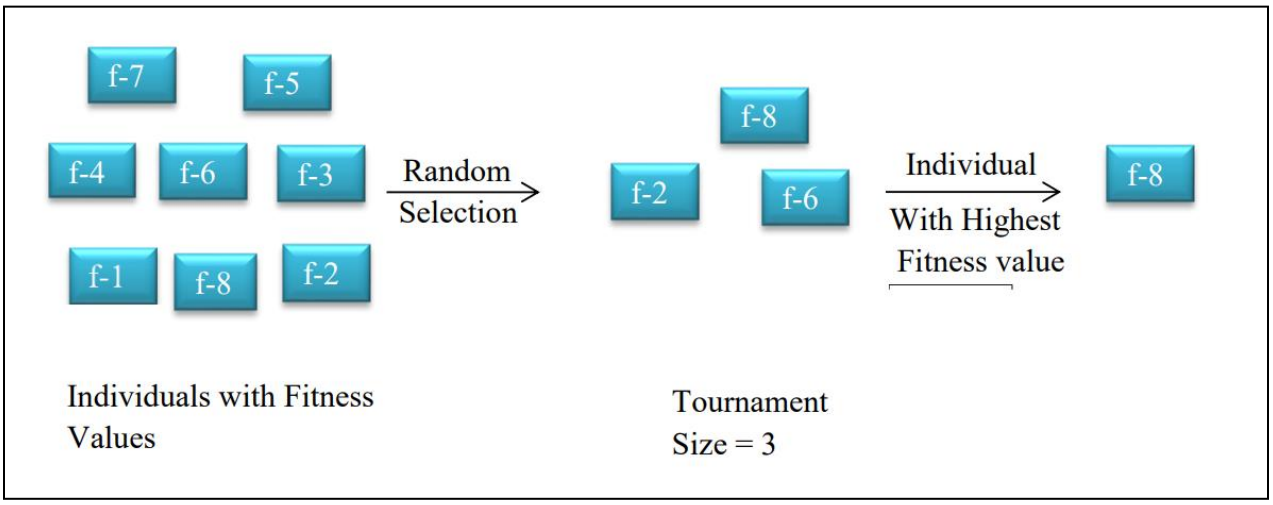


Figure 9: Tournament mechanism [12].

Individuals with higher fitness values are more likely to be selected, as they represents better solutions.

3.5 Crossover:

Preform crossover operations on the selected individuals to generate offspring for the next generations.

Define crossover points that combine genetic information of two parents to create new individuals.

In our solution, crossover could involve exchanging class assignments, meeting times, or rooms between parents, or combining different aspects of their solutions.

3.6 Mutation:

Apply mutation to introduce changes to the offspring’s genetic materials, and to maintain diversity in the population.

Mutations in the timetable context could include swapping class assignments, modifying timeslots or room assignments.

3.7 Method of replacement:

Replace the current population with the newly generated offspring and continue the process of evaluating fitness, selecting parents, preforming crossover, and applying mutation for multiple generations or until an optimal schedule is found.

3.8 Termination:

Decide when to stop the algorithm, in our solution, we have chosen two criteria to terminate the algorithm:

- Maximum number of generations.
- Number of conflicts.
-

3.9 Solution extraction:

Once the algorithm terminates, the best individual (or one of the best individuals) in the final population represents the solution to the timetable problem.

Extract the schedule from the encoded representation and translate it into a format understandable by users.

4. Conclusion:

In this chapter, we have seen that solving timetable problems by the genetic algorithm, involves going through several steps. However, the implementation of the system requires a number of tools, which will be seen in the next chapter.

Chapter 4: Implementation & Presentation

1. Introduction:

After outlining the key aspects of our work, this chapter aims to provide a detailed description of the implementation of our system. We will focus on specifying the development environment and the programming language utilized in the creation of our solution.

2. Technologies & Tools used :

Java: Java is a widely used object-oriented programming language and software platform that runs on billions of devices, including notebook computers, mobile devices, gaming consoles, medical devices and many others. The rules and syntax of Java are based on the C and C++ languages [4].

Eclipse: is an integrated development environment (IDE) used in computer programming, it was the main Code Editor we used to write most of the code for GAschedules.

Vscode: short for Visual Studio Code. Apopular and powerful source-code editor made by Microsoft.

NetBeans IDE: NetBeans is an integrated development environment (IDE) for Java. NetBeans allows applications to be developed from a set of modular software components called modules.

Modelio: Modelio is an open-source UML tool developed by Modeliosoft.

3. System objective:

The objective of our word is to develop a windows application called “GAschedules” that focuses on optimizing timetable problems using genetic algorithm. The purpose of this application is to provide optimal or near optimal solutions for the user who wish to access the system.

4. Usage scenario:

As soon as the application is opened, the user is prompted to enter various details related to the problem domain, the input data is used to initialize the 'Data' object, which stores all the necessary information. The 'Driver' class initializes the genetic algorithm by creating an instance of the 'Genetic algorithm' class and passing the 'Data' as a parameter, a population of schedules is generated using 'Population' class.

The algorithm enters a loop where it involves the population iteratively until a satisfactory solution is found or maximum number of generations is reached.

If a schedule with a fitness value of 1.0 is found, it means optimal solutions has been found, the algorithm stops and prints the optimal solution followed by the number of generations and the execution time it took to find the solution .

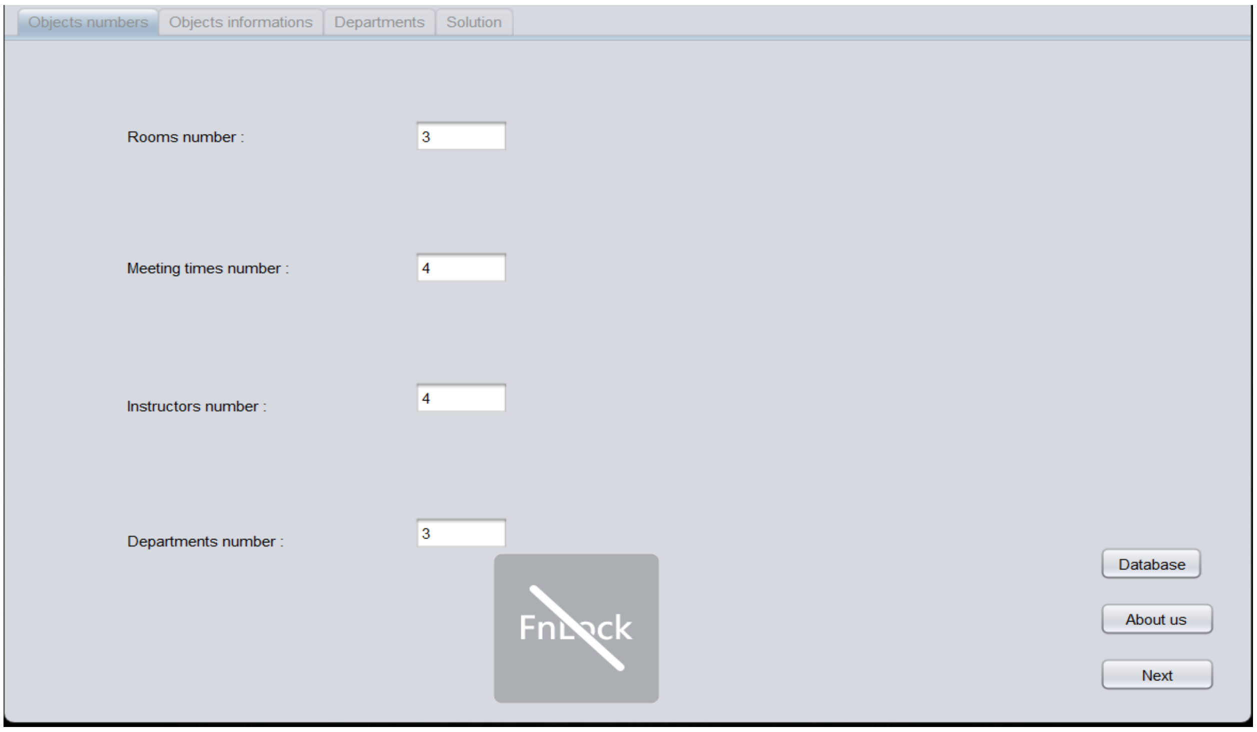
If the maximum number of generations is reached the algorithms stops and prints the near optimal solution followed by the number of generations, execution time, and number of conflicts.

After the algorithm stops, the user is able to extract the schedule.

5. Presentation :

When launching the "GAschedule" application represented by the icon in figure 10, the first interface appears offering the user to enter the number of the following objects: rooms, instructors, meetingtimes, and department. First interface in figure 11

Figure 10: Application icon <<*GAschedules*>>



Rooms number : 3

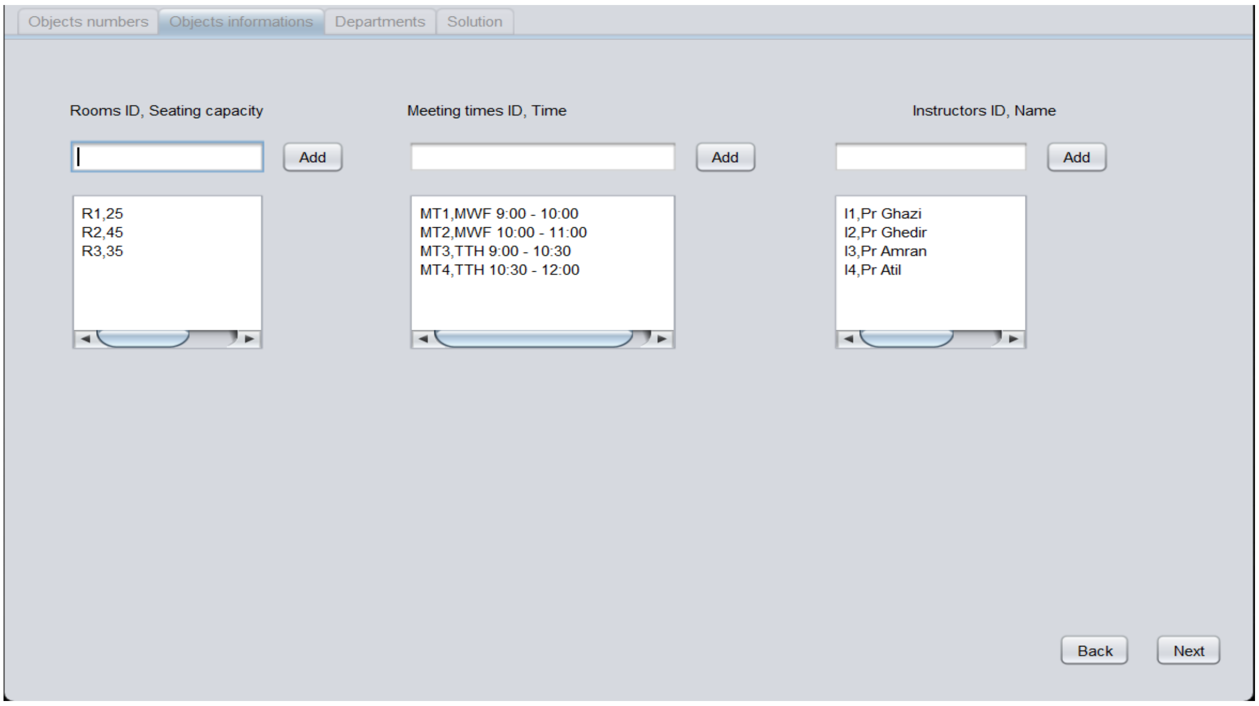
Meeting times number : 4

Instructors number : 4

Departments number : 3

Figure 11: Presentation for the application

After the user enters the number of each objectives, the next step starts by offering the user to provide information related to the timetable, figure 12 an interface for entering data for every object, figure 13 an interface for providing information for each department.



Rooms ID, Seating capacity

R1,25
R2,45
R3,35

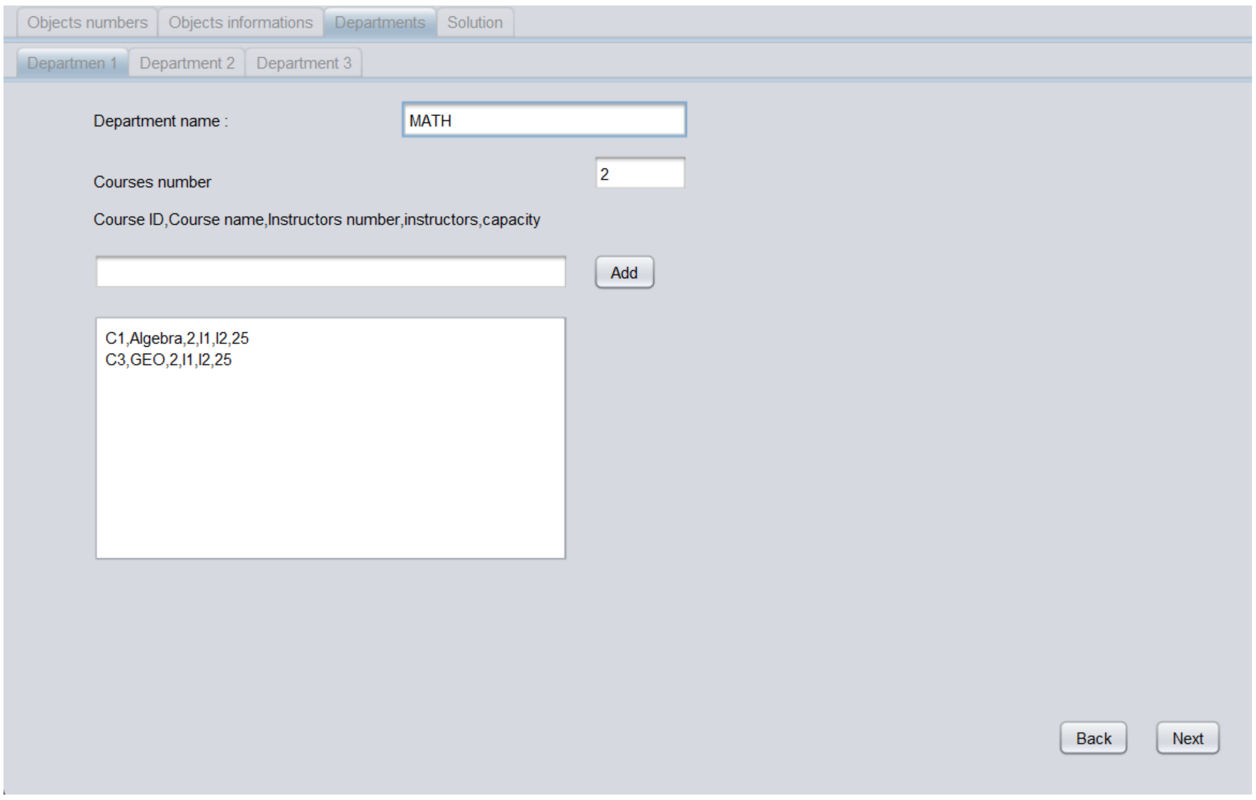
Meeting times ID, Time

MT1,MWF 9:00 - 10:00
MT2,MWF 10:00 - 11:00
MT3,TTH 9:00 - 10:30
MT4,TTH 10:30 - 12:00

Instructors ID, Name

I1,Pr Ghazi
I2,Pr Ghedir
I3,Pr Amran
I4,Pr Atil

Figure 12: Interface for objects information



Department name : MATH

Courses number 2

Course ID,Course name,Instructors number,instructors,capacity

C1,Algebra,2,I1,I2,25
C3,GEO,2,I1,I2,25

Figure 13: interface for entering data for department

In this window, the user get to enter the details for each department based on the number of departments that he declared in the first interface like shown in figure 11.

6. Testing GAschedules on the problem:

All the execution of the test cases is carried out on a computer having the following specifications:

- CPU : intel i5-10300h
- GPU: Nvidia RTX 2060
- RAM: 16 GB 2933 MHz
- OS: Windows 11 Professional 64 bits

The increase in population size shows a steady increase in the number of generations that takes to reach a valid solution (Figure 14).

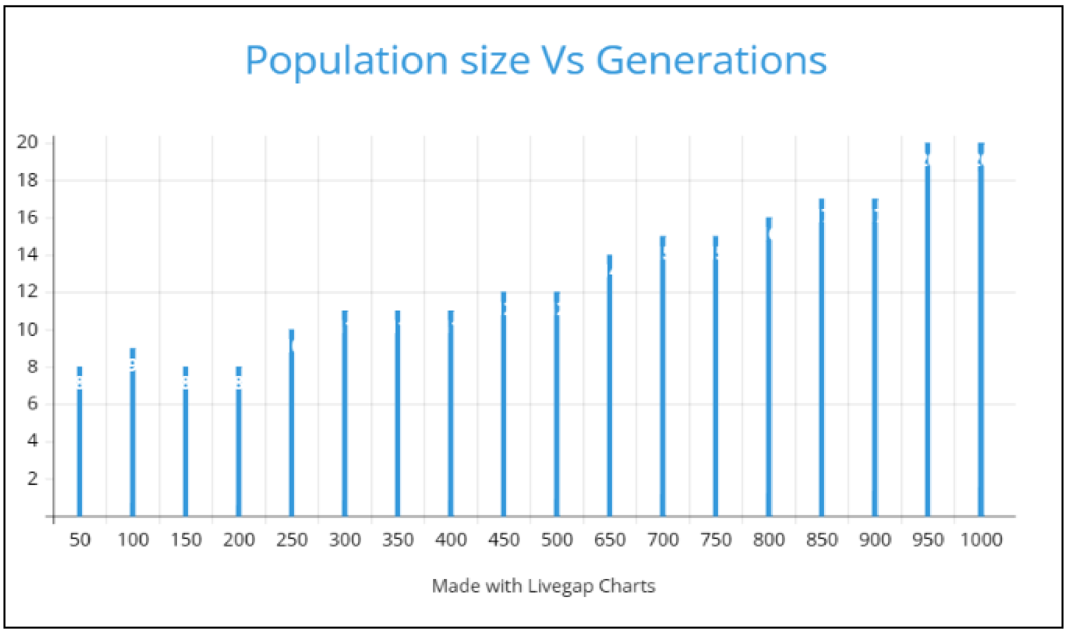
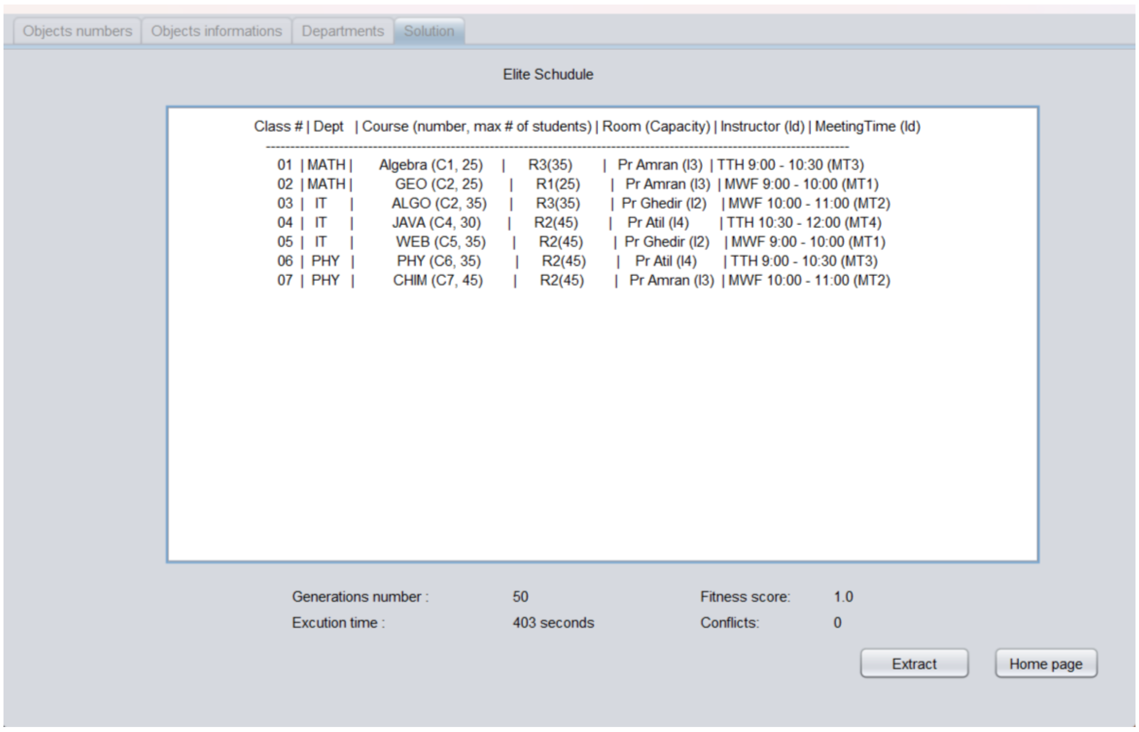


Figure 14: Population size increase vs number of generations to reach a solution.

If the system reach an optimal solution it will show number of generations and the time it took to get the solution like shown in figure 15



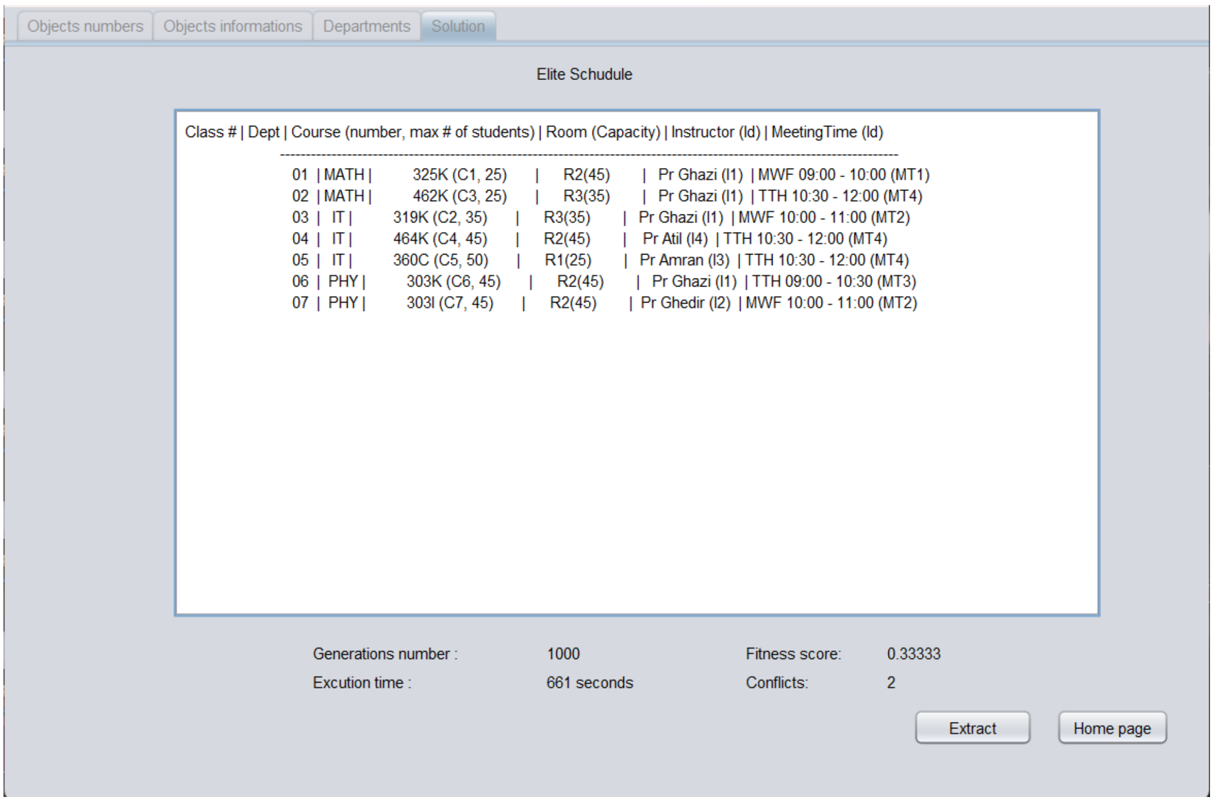
Elite Schudule

```
Class # | Dept   | Course (number, max # of students) | Room (Capacity) | Instructor (Id) | MeetingTime (Id)
--------------------------------------------------------------------------------------------------------------
01 | MATH |   Algebra (C1, 25)   |   R3(35)   | Pr Amran (I3) | TTH 9:00 - 10:30 (MT3)
02 | MATH |   GEO (C2, 25)       |   R1(25)   | Pr Amran (I3) | MWF 9:00 - 10:00 (MT1)
03 |  IT  |   ALGO (C2, 35)      |   R3(35)   | Pr Ghedir (I2) | MWF 10:00 - 11:00 (MT2)
04 |  IT  |   JAVA (C4, 30)      |   R2(45)   | Pr Atil (I4)   | TTH 10:30 - 12:00 (MT4)
05 |  IT  |   WEB (C5, 35)       |   R2(45)   | Pr Ghedir (I2) | MWF 9:00 - 10:00 (MT1)
06 | PHY  |   PHY (C6, 35)       |   R2(45)   | Pr Atil (I4)   | TTH 9:00 - 10:30 (MT3)
07 | PHY  |   CHIM (C7, 45)      |   R2(45)   | Pr Amran (I3) | MWF 10:00 - 11:00 (MT2)
```

Generations number :	50	Fitness score:	1.0
Excution time :	403 seconds	Conflicts:	0

Figure 15: Optimal solution

If the algorithms reach maximum number of generation followed by number of conflicts like in figure 16.



Elite Schudule

```
Class # | Dept | Course (number, max # of students) | Room (Capacity) | Instructor (Id) | MeetingTime (Id)
-----------------------------------------------------------------------------------------------
01  | MATH |     325K (C1, 25)    |    R2(45)    |  Pr Ghazi (I1)  | MWF 09:00 - 10:00 (MT1)
02  | MATH |     462K (C3, 25)    |    R3(35)    |  Pr Ghazi (I1)  | TTH 10:30 - 12:00 (MT4)
03  |  IT |    319K (C2, 35)    |   R3(35)    |  Pr Ghazi (I1)  | MWF 10:00 - 11:00 (MT2)
04  |  IT |    464K (C4, 45)    |   R2(45)    |  Pr Atil (I4)  | TTH 10:30 - 12:00 (MT4)
05  |  IT |    360C (C5, 50)    |   R1(25)    |  Pr Amran (I3)  | TTH 10:30 - 12:00 (MT4)
06  |  PHY |    303K (C6, 45)    |   R2(45)    |  Pr Ghazi (I1)  | TTH 09:00 - 10:30 (MT3)
07  |  PHY |    303I (C7, 45)    |   R2(45)    |  Pr Ghedir (I2)  | MWF 10:00 - 11:00 (MT2)
```

Generations number : 1000

Excution time : 661 seconds

Fitness score: 0.33333

Conflicts: 2

Figure 16: Near optimal solution

7. Conclusion:

In this chapter, we have presented the different interfaces offered by our application GAschedule. In addition, the implemented methods used in order to achieve the solutions for the problem.

Conclusion & Perspective

Scheduling problem is a model of complicated problems. Too many things have to be considered in order to arrange a schedule, such as instructors' availabilities, number classes, and courses.

Many solutions have emerged in the past few years to fix this problem. Still, many of these solutions have failed to design an optimal schedule, yet they have still shown a big impact on timetable problem. Which motivated us to conceive a better solution.

The genetic algorithm can solve the timetable problem by searching through a large space of possible timetables, using the principles of natural selection and genetics to find an optimal or near optimal solution in a reasonable amount of time.

GAschedules has undergone testing by users in the IT departments of different universities, who have expressed their overall satisfaction with the solution.

However, we encountered numerous difficulties while implementing our solution, such as the necessity to develop our own user interface to ensure a consistent design in our application that effectively complemented the algorithm, this process consumed a significant amount of time.

Several future perspective are offered to this work, firstly the addition of a database can enable a larger search space for solutions, allowing for more comprehensive scheduling options. Additionally, improvements can be made for the genetic algorithms itself, this will involve optimizing the selection, crossover, and mutation functions to increase the accuracy of the solutions. These advancements will contribute to the continued development of the GAschedules.

References

[1] T'Kindt, V., & Billaut, J. (2013). *Multicriteria Scheduling: Theory, Models and Algorithms*. Springer Science & Business Media.pp.5-6.

[2] Kannan, D., Bajpayee, K., & Roy, S. (2019). Solving Timetable Scheduling Problems Using Genetic Algorithm. *IJRTE: International Journal of Recent Technology and Engineering, Vol. 7, Issue 5C, no. 41*, pp.168-169.

[3] Berrocal, V. (2023). Bio-inspired optimization algorithms. Oga. https://www.oga.ai/en/blog/bio-inspired-optimization-algorithms/.

[4]*NaturalSelection*.(n.d.b). https://education.nationalgeographic.org/resource/natural-selection/

[5] Mallawaarachchi, V*. (2020, March 1). Introduction to Genetic Algorithms — Including Example Code. Medium. https://towardsdatascience.com/introduction-to-genetic-algorithms-including-example-code-e396e98d8bf3.*

[6] Ghedir L. (2019). Optimisation Du Decoupage De Materiuex : Une Approche Par Algorithmes Genetiques . Mémoire de Master. Universite Badji Mokhtar-Annaba. Faculte Des Sciences De L'ingeniorat. Department D'informatique . pp.12.

[7]Kanet, John J.; Ahire, Sanjay L.; and Gorman, Michael F., "Constraint Programming for Scheduling" (2004). MIS/OM/DS Faculty Publications. pp.2.

[8] Lancia, G., & Serafini, P. (2017). *Integer Linear Programming. Compact Extended Linear Programming Models*, pp.*43.*

[9]*The Graph Coloring*. (n.d.). https://www.tutorialspoint.com/the-graph-coloring.

[10] *What is Math Modeling? | Society for Industrial and Applied Mathematics*. (n.d.). https://m3challenge.siam.org/resources/whatismathmodeling.

[11] Datta, S., & Datta, S. (2023). Greedy vs. Heuristic Algorithm | Baeldung on Computer Science. *Baeldung on Computer Science*. https://www.baeldung.com/cs/greedy-vs-heuristic-algorithm.

[12] Herath, A. K. (n.d.). *Genetic Algorithm For University Course Timetabling Problem*. eGrove. https://egrove.olemiss.edu/etd/443.pp14.

[13] Babaei, H., Karimpour, J., & Hadidi, A. (2015). *A survey of approaches for university course timetabling problem. Computers & Industrial Engineering, 86, pp.43.*

[14] *What is Java? | IBM*. (n.d.). https://www.ibm.com/topics/java.

Abstract

Many educational organizations , including universities, encounter challenges when attempting to create schedules. Some of these problems fall under the category of NP-hard, which requires heuristic algorithms to solve.

This Thesis walks you through our process of research, architecture and implantation of the <<*GAschedules*>> that uses genetic algorithm to solve timetable problems. The algorithms start by representing timetable as a population of individuals from the provided data by the user, each individual is a potential solution. Through the process of selection, crossover, and mutation operators used to explore the search space and improve fitness over generations. Fitness function evaluates timetables based on constraints like rooms availability and course conflicts. Through iterative genetic algorithm, the algorithm refines the population, converging on an optimal timetable solution.

Key words.[Genetic algorithm, Optimization problems, Heuristic methods, Management of timetables, timetables.**]**

Résumé

De nombreuses organisations éducatives, y compris les universités, rencontrent des difficultés pendant la réalisation des emplois du temps. Certains de ces problèmes relèvent de la catégorie NP-difficile, ce qui nécessite des algorithmes heuristiques pour être résolus.

Cette thèse vous guide à travers notre processus de recherche, d'architecture, de conception et d'implantation de notre application « *GAschedules* » qui utilise un algorithme génétique dans le but de résoudre les problèmes de gestion des emplois du temps. Les algorithmes commencent par représenter emplois du temps comme une population d'individus à partir des données fournies par l'utilisateur, chaque individu étant une solution potentielle. Grâce au processus de sélection, de croisement et de mutation utilisés pour explorer l'espace de recherche et améliorer le fitness au fil des générations. La fonction de remise en forme évalue les horaires en fonction de la disponibilité des salles et des horaires de cours. Grâce à un algorithme génétique itératif qui affine les populations d'individus au fur et à mesure des générations, convergeant vers une solution horaire optimale.

Mots clés. [Algorithme génétique, Problèmes d'optimisation, Méthodes heuristiques, Gestion des emplois du temps, emplois du temps.**]**

ملخص

تواجه العديد من المنظمات التعليمية، بما في ذلك الجامعات، تحديات عديدة عند محاولة إنشاء جداول الزمن. بعض هذه المشاكل تندرج تحت فئة NP-hard مما يتطلب استخدام خوارزميات تقريبية للحل. تتناول هذه الأطروحة عملية البحث والتصميم والتنفيذ التي تستخدم خوارزمية وراثية لحل مشاكل جداول الزمن. تبدأ الخوارزمية بتمثيل الجدول الزمني كمجموعة من الأفراد المستمدة من البيانات المقدمة من قبل المستخدم، حيث يعتبر كل فرد حلاً محتملاً من خلال عملية الاختيار والتقاطع وعوامل التغير المستخدمة لاستكشاف فضاء البحث وتحسين اللياقة عبر الأجيال. تقوم دالة اللياقة بتقييم جداول الزمن استنادًا إلى القيود مثل توافر الغرف وتعارض وحدات التدريس. من خلال الخوارزمية الوراثية التكراريةتقوم الخوارزمية بتحسين الأفراد، مع الإتجاه نحو الحل الذي يمثل الجدول الزمني الأمثل أو شبه الأمثل.الخوارزمية الجينية ، تنقح الخوارزمية السكان ، وتتقارب على حل الجدول الزمني الأمثل أو شبه الأمثل.

كلمات مرشدة [.الخوارزمية الجينية ، مشاكل التحسين ، الطرق الاستكشافية ، إدارة الجداول الزمنية ، الجداول الزمنية**]**

الجداول الزمنية ، الجداول الزمنية