**Faculty Timetable Schedule**

**Using Genetic Algorithm**

**Team member :**

**Name : ID :**

**Karim abdelfattah saad 20208180 MI**

**Yousef Shaban Ahmed 20208299 MI**

**Alaa Ahmed Hamdy 20208063 MI**

**Lara rabeh hamdy 20208183 MI**

**Dalia Ahamed lofty 20208095 MI**

**shared folder url :** https://drive.google.com/file/d/1FDaCDxD5crwSLEaehP3JXpjuOyV01goA/view?usp=sharing

**Introduction and overview**

This project covers methods used for solving the timetable scheduling problem specific for timetable formats at the Faculty using genetic algorithm.

The application similar to using genetic algorithm for scheduling the faculty timetable is the implementation of Genetic Algorithms (GA) for job scheduling on computational grids that optimizes the makespan and the total flowtime.

A Literature reviews of academic publications :

- Goldberg, D. E. (1989). Genetic Algorithms in Search, Optimization, and Machine Learning. Boston: Addison-Wesley.

- Davis, L. (1991). Handbook of Genetic Algorithm. Von Nostrand Reinhold, Newyork.

- Mitchell, Melanie (1996). An Introduction to Genetic Algorithms. MIT Press.

- Bryant, Kylie (2000). Genetic Algorithms and the Traveling Salesman Problem, in Proceedings of 1st GNT Regional Conference on Mathematics, Statistics and Applications.

- Coello, Carlos (2000). An updated survey of GA-based multiobjective optimization techniques, ACM

Computing Surveys, vol. 32, no. 2, pp. 109-143

- Hanne, Thomas (2000). Global multiobjective optimization using evolutionary algorithms. Journal of

Heuristics, vol. 6, no. 3, pp. 347-360.

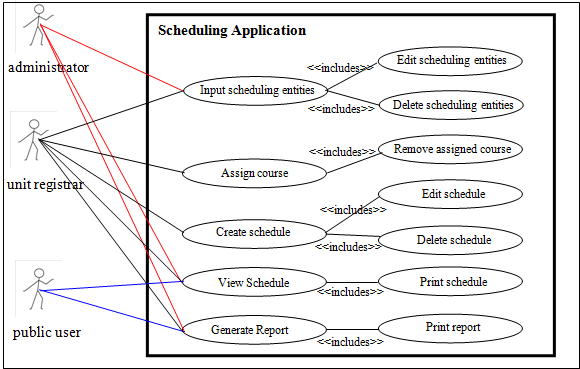
**Proposed solutions and Dataset**

Calculate the fitness of the classes and make a new generation to have the best population by genetic operators and the methods of initialize and calculate the fitness

**Main functionalities :**

**Input data** for each class are *Instructor’ name*, *course*, *meeting Time*, *department* and list of *allowed courses of each department and the get methods*.  
**Output data** are generations a table in terminal and show the conflicts and the best class with 1.0 conflict for each class.

**Use case diagram.**

****

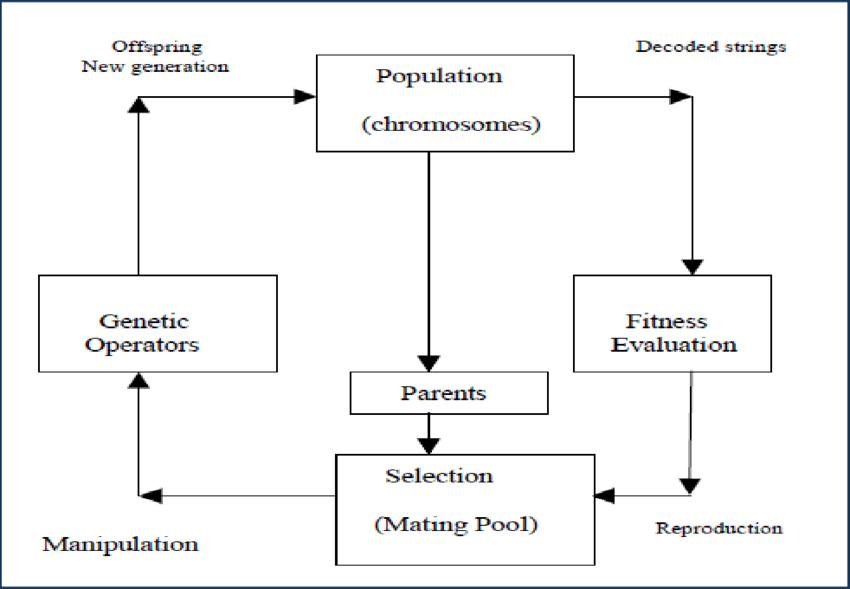
The dataset get from the instance data or database and the user can manually register the partly of data by using GUI

**Applied algorithm**

the algorithm calculate the fitness and the number of conflicts of the classes in the populations and select the individuals of classes that have the high fitness and make crossover between them and using mutation to improve the fitness

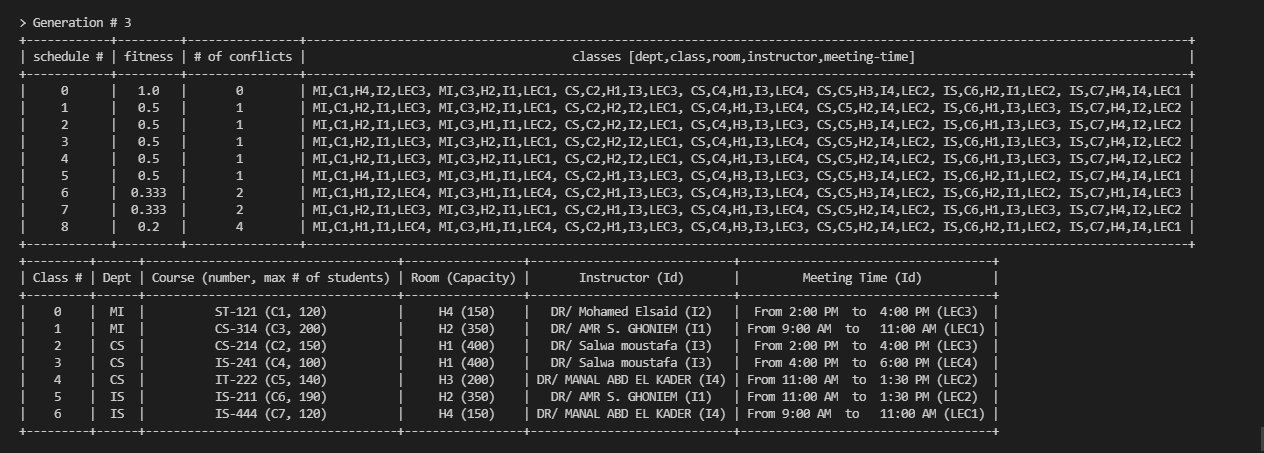
Crossover – Selection - Mutation

**Block diagram.**

****

**Experiments & results**

The result will show in terminal as a timetable and represent all the generations of the algorithm and show the number of conflicts and the fitness of each schedule.

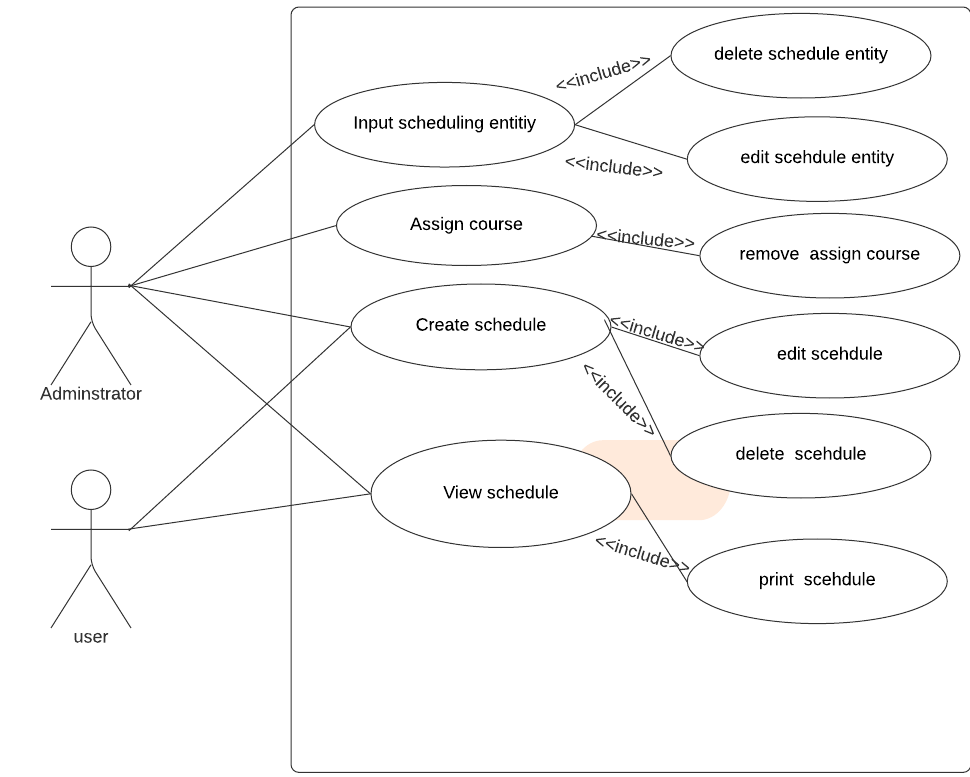
****

**Analysis, Discussion, and future work**

Faculty Timetable show the generations of the schedules and its fitness , conflicts , etc.. and will modify large and wide solution space search ability

* The advantages that It can find fit solutions in a very less time. (fit solutions are solutions which are good according to the defined heuristic)
* The disadvantages of using the genetic algorithm that Its hard to choose parameters like number of generations, population size etc. When we are working even though our heuristic was right we were not realizing it because we were running for a fewer generations.

**Use case Diagram :**



**Genetic Algorithm Diagram :**

