1. genetic algorithm: A genetic algorithm is a powerful technique and a metaheuristic method used to solve complex problems that require huge search areas for possible solutions.GA is one of the evolutionary computations in AI so it's a category of evolutionary algorithms, it was developed by Professor John Holland in 1960.GA was inspired by the Darwinian theory of natural evolution which says that organisms multiply in geometric proportions leading to a struggle for survival because of the limited space and food and the stronger or the best will survive because they can make a species evolve due to those favorable variations so there is a small chance for the survival with injurious variations so the evolution is a process of natural selection.
GA often depend on adaptive systems to do good in different environments, in timetabling problem to increase the generality level we need a self-adaptive method & another example is the robot's complex behavior that needed to move in around the environment.

2. Advantages of the genetic algorithm: GA is able to look for solutions in many directions at the same time and Leave the paths which lead to suboptimal solutions because it has multiple offspring so there is a very good chance to find the favorable solution at each run.
On the other hand, the fitness schemata which are above average grow while the ones below average decay as per Holland's schema theorem. GA evaluates a very large group of individuals in a good period of time. GA can handle multiple parameters at the same time. Using one parameter and one individual optimizing GA can generate many solutions .so any person will be able to select the best solution.

3. Disadvantages of the genetic algorithm: in GA we must define the problem as clearly as possible. For no happens of errors we must write the solutions in a language that is capable of adapting the random changes.to define the individuals we use an (integer, binary, real list of numbers) it's difficult to determine the fitness function and must be written

carefully to get the desired solutions, if we determine a wrong fitness function, we may not find any solutions or solve a wrong problem. We must be attentive in the case of selecting the factors like (population, crossover rate, and mutation rates) because if we select a small population it leads to less solution space and premature convergence and therefore inaccurate results .in GA we can not find the optimal solution but we find the best solution.

4. Timetabling: genetic algorithm approach:

Scheduling faculty's timetable may encounter constraints (hard and soft) because of the difference as compared to a school timetable as the requirements are very limited. We had to find a functional solution for any problems that have hard constraints and we have to optimize the performance of the solution by considering the soft constraints and finding a solution for it.
In GA generate only one child solution with (selection, crossover, and mutation) for each generation and then the solution will be improved by local search. Finally, the worst population will be replaced with the new child individual.

5. Problem Definition:

- **The meeting time**
  Every week has a specific number of days and every day is divided into a fixed number of timeslots like 5 and each timeslot is 2 hours as(MT1, MT2,…..).
- **Rooms**
  Each room has a specific number of seats called the room capacity of students that can be accommodated.
- **Department**
  Each department contain a number of courses and It's an advantage that of courses in the same department is that they should not overlap.
- **Professors**
  Each professor has its own timetable for his/her In order to attend the university and specializes in offering certain subjects under his/her experience.

- **Courses**
  Each course included in a specific department and related to a particular subject.
- **The constraints**
  After specifying the (department, timeslot, professor, room to set of courses) we need to apply the hard constraints and the soft constraints as possible.
- **Timetabling (Hard constraints)**

1. Each room must not have more than one course in the same meeting time.
2. Each professor must not be assigned to more than one room in the same meeting time.
3. Each professor must only teach on days he/she is available at the university.
4. A room that is assigned to a course must have the needed capacity.
5. There is no student or teacher in two rooms at the same meeting time.
6. There are no two professors who teach two different courses to the same students at any time.
7. There is no allocation for two or more rooms for the same course for any students.
8. For each meeting, time must have an availability(room, professor, students, courses).
9. According to the type of the meetings, there is different & varying room capacity for each meeting.
10. The timetable must consist of all the available courses with flexibility and multilabel meeting time for all the registered students that cannot participate simultaneously.

- **Timetabling (Soft constraints)**

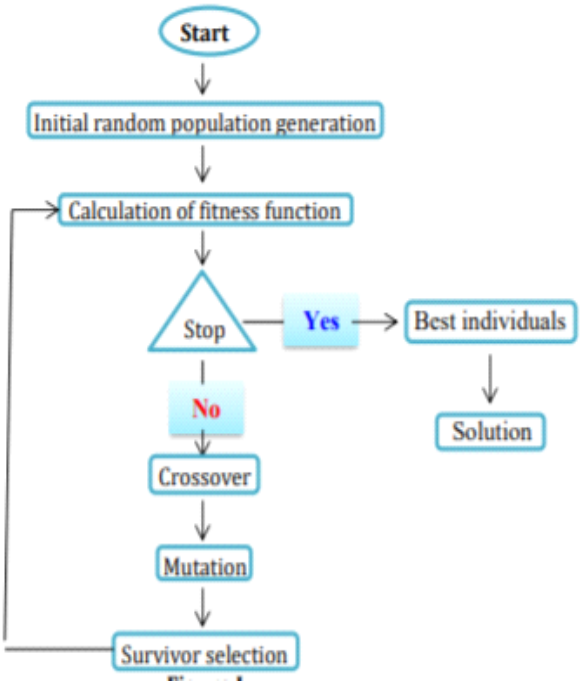
1. Each professor teaches the course that he/she has the experience on it.
2. Each professor should apply his/her timetable.
3. We must apply that each student or professor should have a maximum and minimum number of hours of attendance in a day.
4. All lectures that in the same day must be adjacent to each other.
5. Courses must not be at a late hour on a day.
6. Student presence in consecutive hours per day.

7. Assign the difficult & hard courses in the morning sessions to gain the student's full attention for these courses.
8. Additional choices for any professor must be available such as (favorite time, and favorite room).
9. These constraints should be applied that make time table high-quality.

6. How does Genetic Algorithm Work?

- **General pseudo-code of Genetic algorithms**

```
Initialize population
Calculate fitness of all solutions
Sort population by fitness
While termination condition not reached do
   Select two parents from population by tournament selection with size 2
   Create child solution using crossover with a probability P_c
   Apply mutation with a probability P_m to child solution
   Apply Local Search to child solution
   Replace child solution with the worst member of the population
   Sort population by fitness
End while
The best solution achieved as output
```



- **Chromosome Representation**
  The important factor that is lead to GA success is encoding the chromosome because it affects the efficiency and performance of GA and the speed and quality of the final result.

  We can represent the chromosome in the 4XNc matrix :

  - 4 represent the matrix row (department, professor ID, meeting time ID, room ID)
  - Nc represents the matrix columns as (number of courses).

  Chromosome matrix:

	C1	C2	C3	…….	CNc
department	MATH	EE	MATH		PHY
professor ID	I2	I2	I1		I1
meting time ID	MT1	MT2	MT2		MT1
room ID	R1	R3	R1		R2

- **Initial Population**
  Each chromosome represents a complete solution to the problem in the initial population generated by GA.

  Genes are the component of the chromosome that are initialized to random values. We can evaluate each chromosome by a specific function "fitness function " that can indicate the quality of the solution.

  The initial population is created randomly by taking the inputs from the following matrixes :

  (professor-course matrix, conflict-course matrix, room-course matrix, professor meeting time, room meeting time).

  - **professor-course matrix is NpXNc of values (0,1,2):**

    "0" P cannot teach the course.

"1" P can teach but he is not an expert in it.

"2" P is an expert in the course.

(Np -> the number of professors) (Nc-> the number of courses)

- **conflict-course matrix is Nc X Nc of values (0,1):**

  "0" c has no conflict.

  "1"c has no conflict.

- **room-course matrix Nr X Nc of values (0,1):**

  "0" R is not good for this c.

  "1" R is good for this c.

  (Nr -> the number of rooms)

- **professor meeting time matrix NpX60 of values (0,1,2):**

  "0" P is not in the university.

  "1" P in the university but want not to teach now.

  "2" P in the university and want to teach.

- **room meeting time matrix Nr X 60 of values (0,1):**

  "0" R is available at this time.

  "1" R is not available.

After this, we can make an ascending sort for the courses according to the number of professors. And each course will have a random (professor, meeting time, and room).

1. select a professor from the "professor-course matrix".

2. select meeting time according to the selected professor from the" professor meeting time matrix".

3. select a room from the "room-course matrix, room meeting time matrix".

- **Fitness Function**

  We say that this solution is valid if all the hard and soft constraints are satisfied. We have already satisfied all the hard constraints on the initial population, local search algorithms, and genetic operators because of this the fitness function is only dependent on applying the soft constraints but soft constraints are difficult to measure accurately.

  The fitness value depends on the number of broken constraints. We can calculate the fitness function by getting the inverse of the conflicts number according to this equation:

  (1/ (number of conflicts*1.0+1))

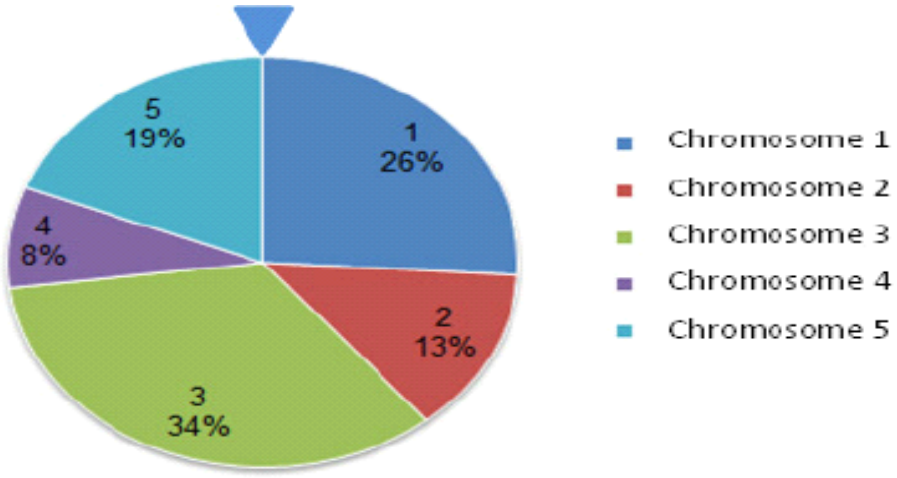
  the fitness value will be 1 If there is no conflict and will be the best solution.

- **Selection:**

  Here GA selects 2 random solutions from the population by "roulette wheel" and then selects the best solution between them. here we apply the selection process twice for each population to select the best two parents for the next generation. We can also use the tournament selection to find the best solution.

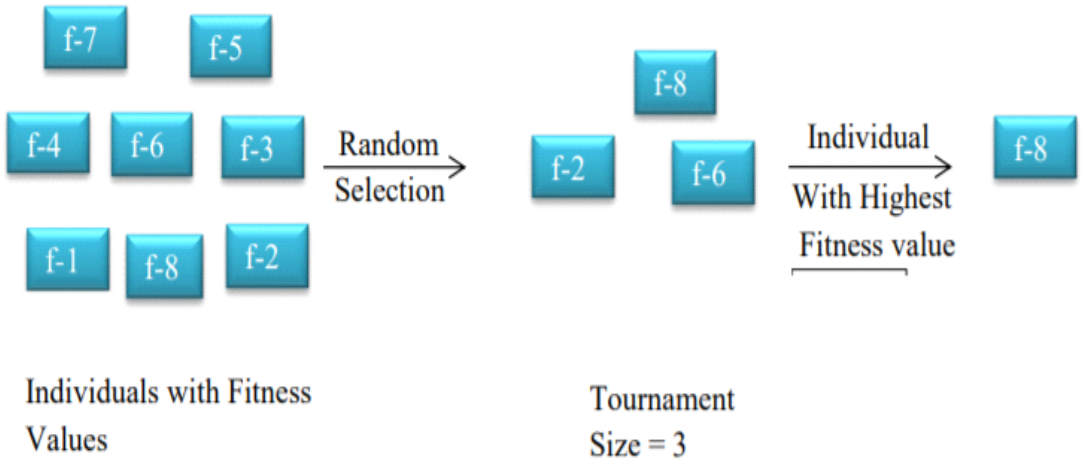
  ### 1.4.1 The roulette wheel:

  it's based on the probability of selecting the best solutions for the next generation.



1.4.2 Tournament selection:

This method depends on the fitness function to select individuals for the next generation, because of its simplicity and efficiency this method is considered as the most common use in GA. It's also dependent on the random selection, the selected individuals will compete to take a place in the next generation.



1.4.3 Stochastic Tournament selection:

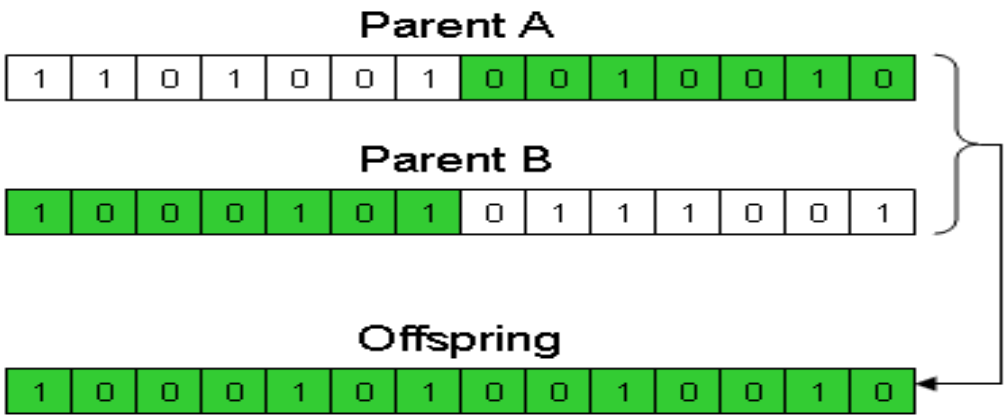
In this method, the individuals selected by the roulette wheel are put in the tournament.

- **Crossover:**

It's a very effective method, especially for numerical optimization problems. The probabilities of the selected individuals must be changed to improve their fitness function for the next generation by crossover and mutation.

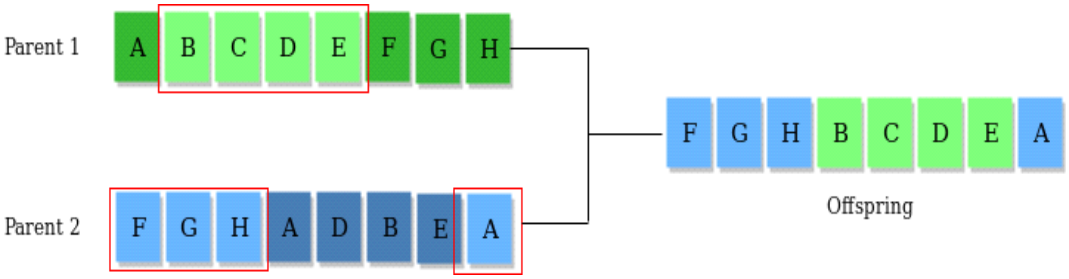
1.5.1 one-point crossover:

This is a fifty-fifty (50/50) manner by taking half of each parent and producing a new offspring.



1.5.2 random point crossover:

It's preferred to select randomly instead of (50/50) because it makes the next generation more variety than the old.



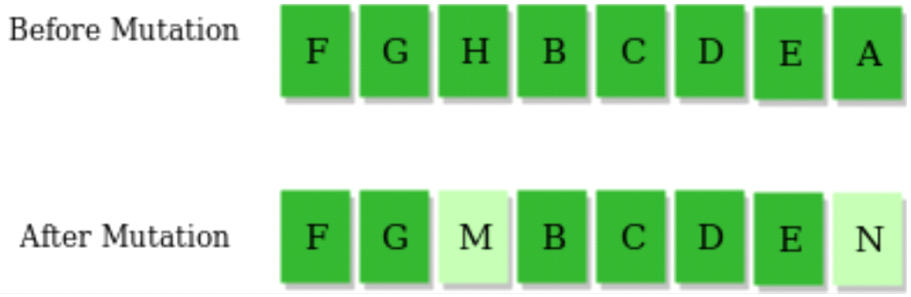
- **Mutation:**

  We can make the next generation more and more varied.

  This variety can be shown by using the mutation operation for the offspring produced by the crossover operation.

  The mutation, it's Random switching for any gen in the individuals by another gen.

  The individuals that are produced from mutation may be a desirable or undesirable individual. The mutated individuals will be determined by natural selection.

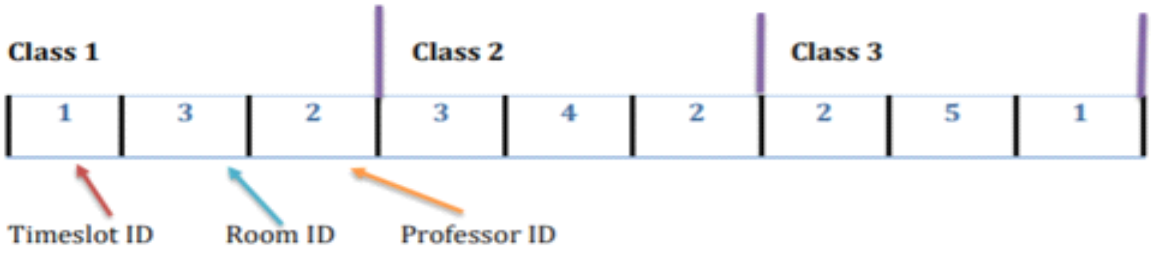


Here we select a random meeting time for the course based on the course time matrix. If we didn't find a solution, we can repeat the crossover operation.

We repeat and repeat the above operations until finding the desired solution for the problem.

7. Encoding

We must use a good encoding like numerical code to be able to encode all the class properties like(course timeslot, professor, classroom) so we can allocate a number for each feature. We can use a chromosome to encode an array of integers to represent each class.



Array is split into three to retrieve information for each class

- **The results of the experiments**

  We need to ensure that all the solutions obtained were correct and valid by apply the hard constraints. *****

  We found the best solution at this factor's values:

Factor	Values

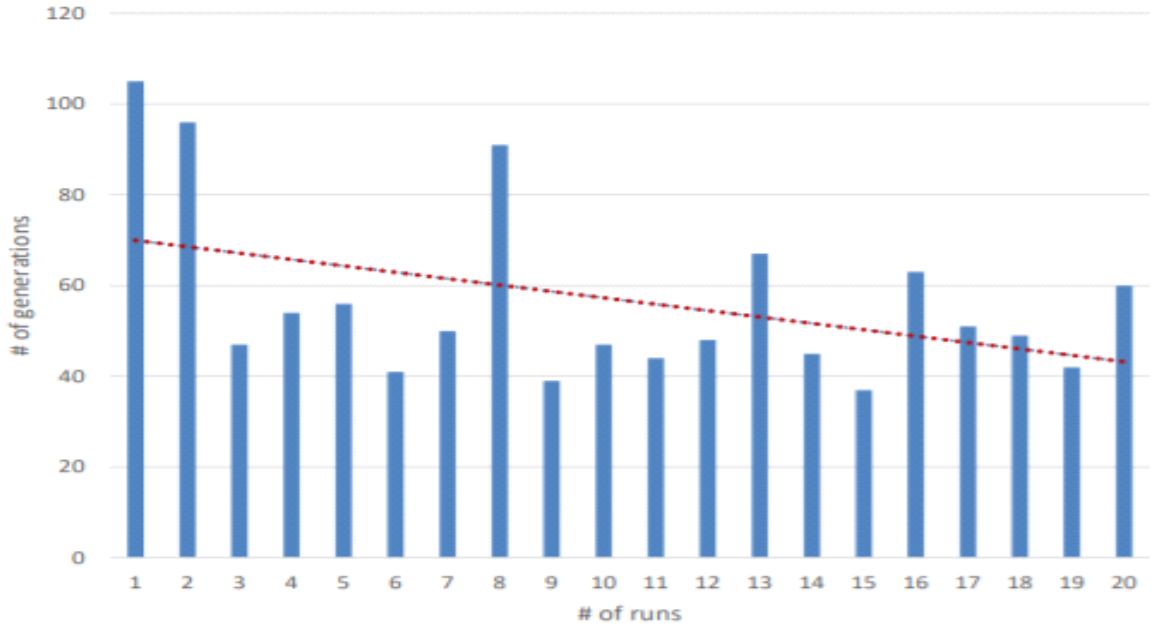
population size	100
mutation rate	0.01%
crossover rate	0.9%
number of best individuals	2
tournament size	5

We can get this factor's values through multiple runs of the system.

With these values we get the best solution for timetable problem with zero conflict at fitness function =1.

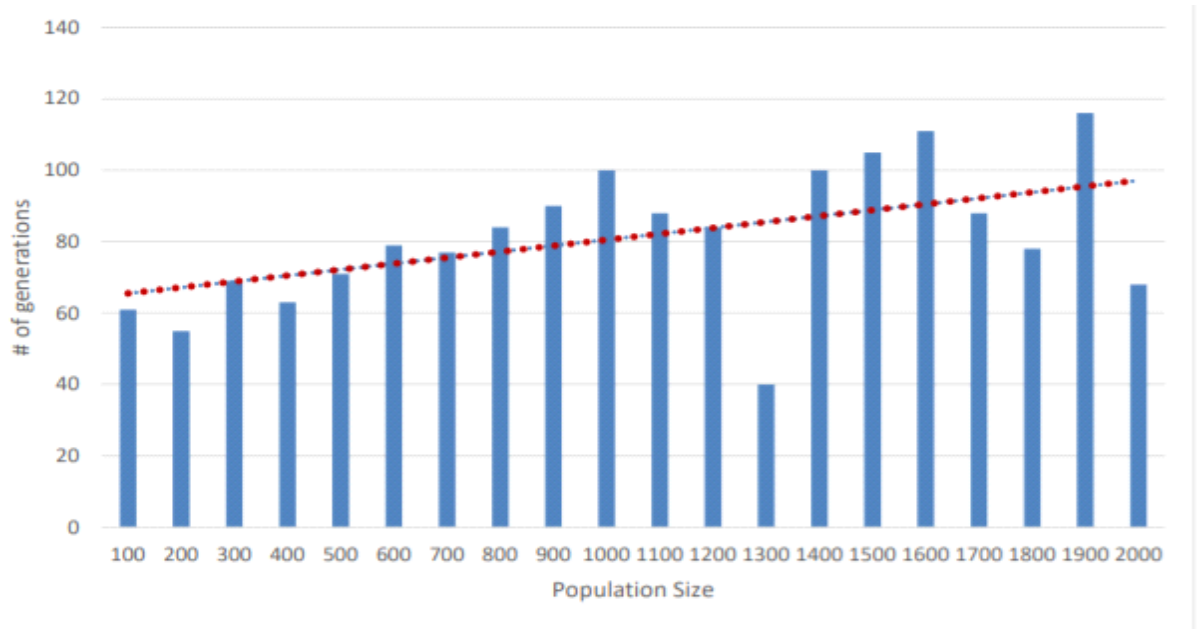
- **Performance Analysis:**

i. Course Scheduler using Genetic Algorithms



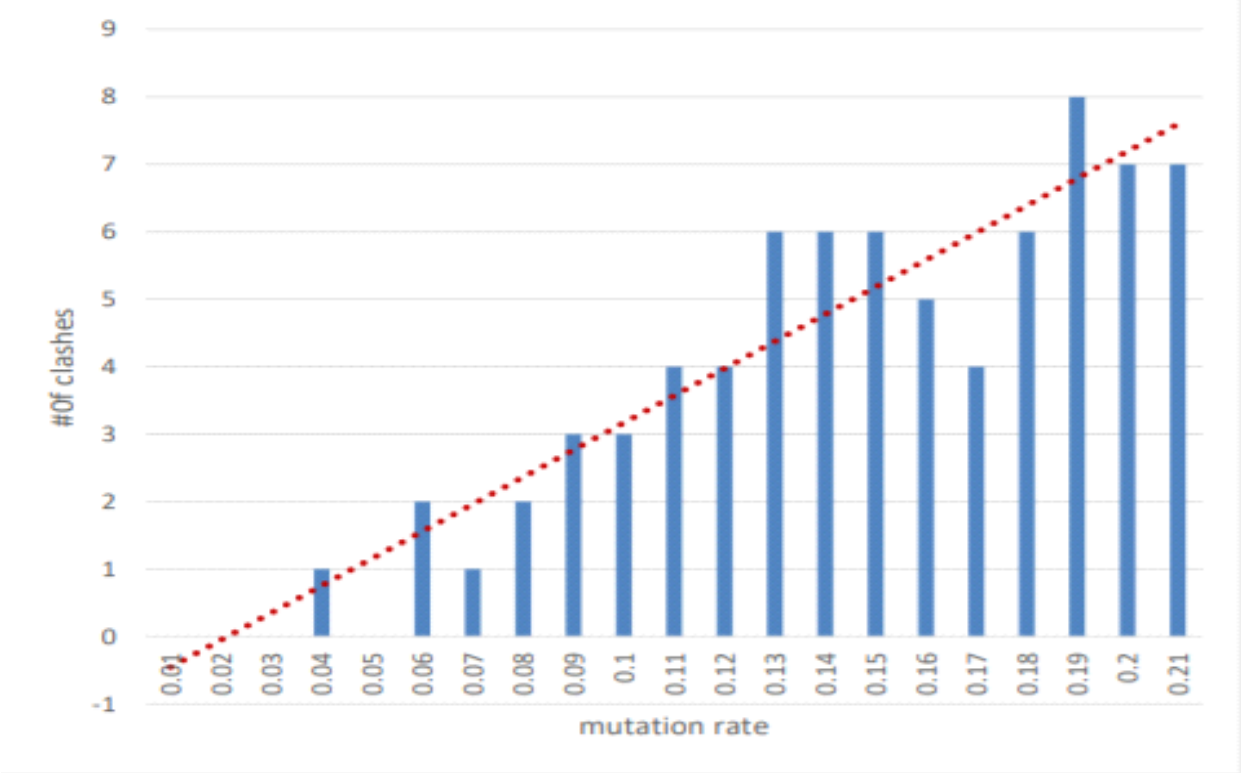
- This analysis represents the number of generations that the Course Scheduler takes to find an optimal solution.
- We will find a steady increase in the number of generations due to the increase in population size to get the best solution.

ii.Population Size vs Generations



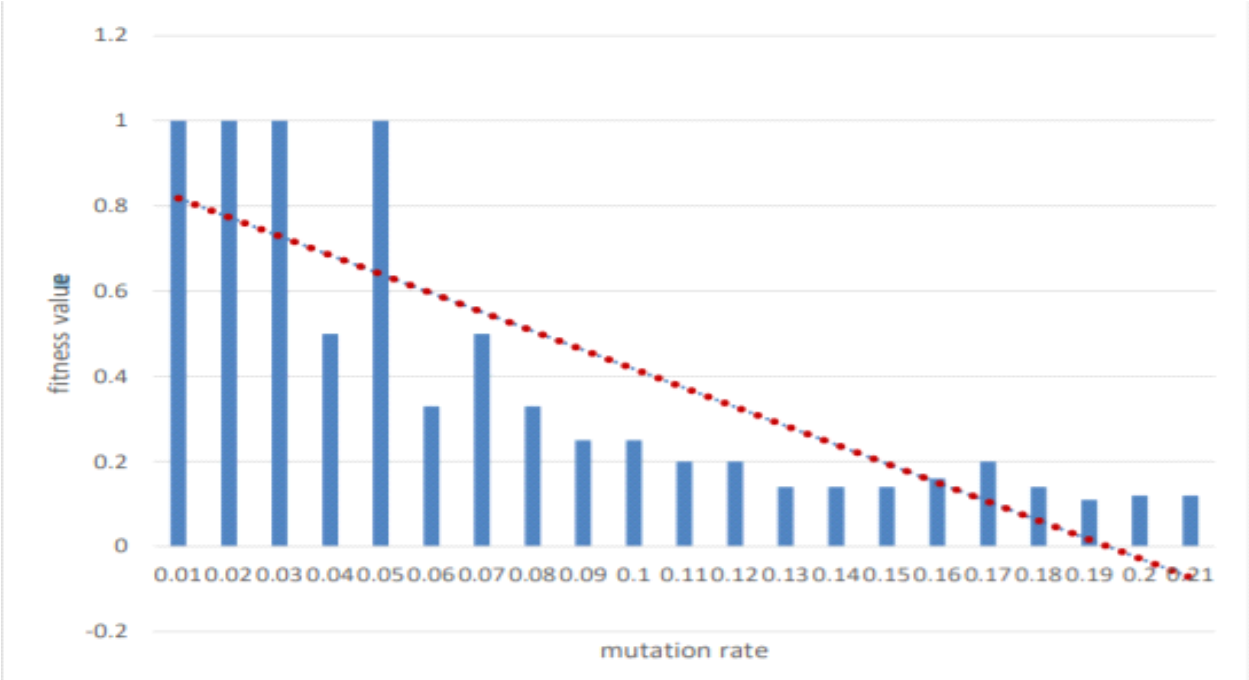
- This analysis represents Population size increase vs the number of generations to find the solution

iii. Mutation rate vs number of conflicts



- This analysis represents the mutation rate vs number of conflicts.
- Here the number of generations is 1000 and the mutation rate have range(0.01 :0.21).
- The number of conflicts increases because of the increase in the mutation rate.
- Here we find that the we can get zero conflict at mutation rate 0.01

iiii. Mutation rate vs fitness



- This analysis represents the mutation rate vs fitness function value.
- The fitness function values decrease due to the increase in mutation rate.