# JAYPEE INSTITUTE OF INFORMATION TECHNOLOGY, NOIDA

# CONFLICT RESOLUTION SCHEDULER USING GENETIC ALGORITHM


**GROUP MEMBERS**

| NAME | ENROLLMENT NUMBER |
|---|---|
| YASHIKA AGARWAL | 20803005 |
| AISHWARYA SHANKER | 20803009 |
| DHRUV GARG | 20803021 |

**SUBMITTED TO:-**

Dr. Shikha Jain

Dr. Ankita Verma

# Introduction:

A time table scheduler is a tool that helps to allocate and manage time slots for various tasks or events within a given time frame. The problem of creating an optimal time table scheduler is a complex task that requires the consideration of various factors, such as availability of resources, constraints, and preferences.

Our timetable scheduler effectively reduces conflicts by:

1. **Handling student enrollment:** Allocating courses to rooms with sufficient capacity to accommodate the enrolled students, minimizing conflicts arising from overcrowded classrooms.

2. **Considering instructor availability:** Scheduling classes at meeting times when instructors are available.

3. **Avoiding room conflicts:** Ensuring that no two classes are scheduled in the same room simultaneously.

4. **Ensuring instructor non-duplication:** Avoiding scenarios where the same instructor is assigned to teach multiple classes simultaneously.

# Problem Solving Strategy:

One approach to creating an optimal time table scheduler is to use a **Genetic algorithm**. A genetic algorithm is a search heuristic inspired by the process of natural selection. It works by mimicking the process of evolution, using a set of rules to generate a population of potential solutions, evaluating their fitness based on some criteria, and then selecting the most promising individuals for reproduction and mutation.

The genetic algorithm used in the code has the following main steps:

**1. Initialize the population:** The first step is to randomly generate a population of schedules that satisfy all the constraints and preferences. Each schedule is represented as a list of courses and their respective meeting times, rooms, and instructors.

**2. Evaluate the fitness:** The next step is to evaluate the fitness of each schedule in the population. The fitness function calculates the total cost or objective value of the schedule based on the constraints and preferences. The lower the cost or objective value, the fitter the schedule.

**3. Select the parents:** The next step is to select the parents for the next generation of schedules. This is done using a tournament selection strategy, where a subset of individuals is randomly selected from the population, and the fittest individual is chosen as the parent.

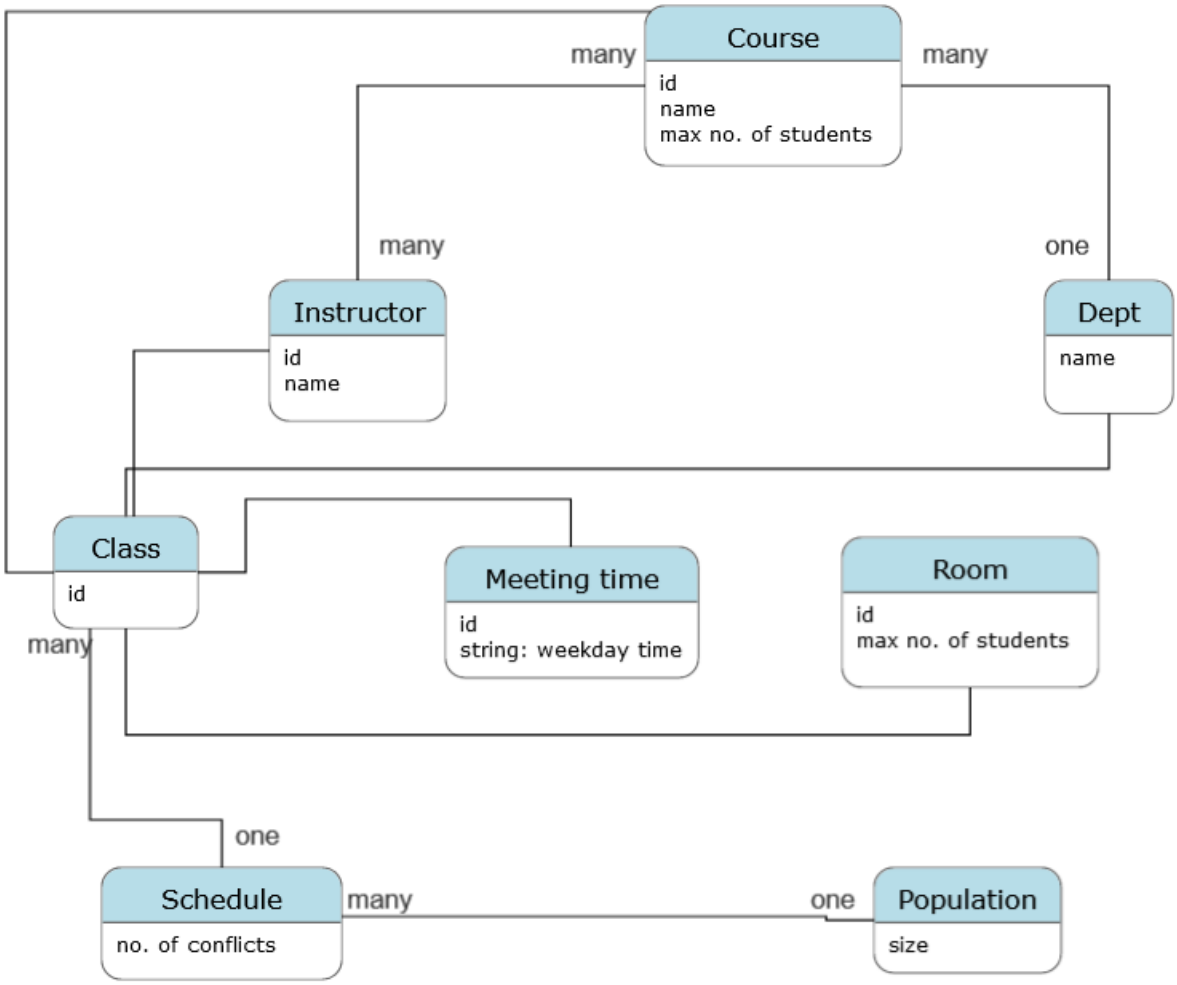
**4. Crossover:** The selected parents are then crossed over to produce new offspring schedules. The crossover operator combines the genetic information of the parents to generate new and potentially better schedules.

**5. Mutation:** The offspring schedules are then mutated with a certain probability to introduce diversity and prevent premature convergence. The mutation operator randomly modifies a small portion of the schedule to create new and potentially better schedules.

**6. Elitism:** The best schedules from the current population are carried over to the next generation unchanged to preserve the best solutions found so far.

**7. Repeat:** Steps 2-6 are repeated for a certain number of generations or until a satisfactory solution is found. The best schedule found so far is returned as the optimal solution.

## UML Diagram:




## Implementation:

**The code has been implemented in python programming language on pycharm IDE.**

The **sqlite3** library is used to connect to a SQLite database that stores data about available courses and teachers.

The **prettytable** library is used to format the output of the generated schedules in a readable table format.

The code creates a SQLite database and populates it with tables containing information about the rooms, meeting times, instructors, instructor availability, courses, and departments.

This database is used as input for implementing a time table scheduler using a genetic algorithm, as it contains all the necessary information for generating a feasible and optimal schedule.

In summary, we used the prettytable library in conjunction with SQLite to create a visually appealing table for displaying the results of a time table scheduler implemented using a genetic algorithm, making it easier to analyze and compare potential solutions.

## Result:

The result of a time table scheduler would be a feasible solution that satisfies all the constraints and requirements of the problem. Our solution using a genetic algorithm represents a schedule for classes to be held at specific times and in specific rooms, with specific instructors teaching specific courses.

It is important to note that the quality of the result would depend on the parameters and design choices of the genetic algorithm, such as the selection method, crossover operator, mutation rate, population size, and termination criteria. Therefore, we fine-tuned these parameters and compared multiple runs of the algorithm to ensure a good quality solution.

## References:

https://www.geeksforgeeks.org/genetic-algorithms/
https://www.sqlite.org/docs.html
https://pypi.org/project/prettytable/