**Class Scheduling Using Genetic Algorithm**

By: Karan Barai, Krunal Nanda

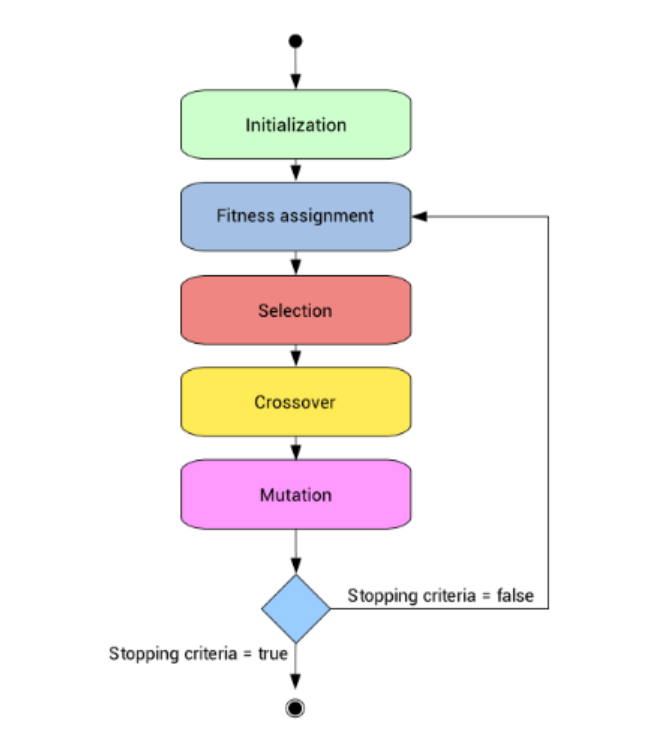
**Problem Statement:**

Genetic Algorithm is an optimization technique which tries to find out such values of input so that we get the best output values or results. By using genetic algorithm, we are trying to automate class scheduling problem. Given a set of data that has list of professors, list of courses associated with each professor, list of class rooms available, and list of time slots a time table can be prepared automatically by the algorithm. This information becomes a chromosome and we evaluate fitness of each chromosome and select best possible chromosomes and then proceed with crossover and mutation.

**Literature Survey:**

Genetic algorithm is a heuristic search method used in artificial intelligence and computing. It is used for finding optimized solutions to search problems based on the theory of natural selection and evolutionary biology. Genetic algorithms are excellent for searching through large and complex data sets. They are considered capable of finding reasonable solutions to complex issues as they are highly capable of solving unconstrained and constrained optimization issues. The individuals are chosen from the parent population and they are mated (Crossover population) or mutated (Mutate population) to obtain a new set of population. In this way new generation can be created. GA uses three processes “Selection”,” Crossover”,” Mutation”.

Working process of genetic algorithm:



**Genetic Algorithm Process for class scheduling:**

1. **Chromosome generation**: the chromosomes are the classes that are combination of course, room, professor and time. The fitness function is based on how well the chromosomes matches with problem solution i.e., Time table generation.
2. **Population:** To start with the genetic algorithm, we have to initialize the population. Then the fitness function for each individual that is schedule is calculated by determining the number of clashes for given schedule. In our approach we have created the population by selecting random data from the datasets given.
3. **Selection:** we are taking the fittest individual for mutation and crossover. We are selecting a random set of individuals using a ‘tournamentsize’ where it is initialized as 5 followed by selecting fittest among the five.
4. **Crossover Population**: The crossover operator is analogous to reproduction and biological crossover. In this more than one parent is selected and one or more off-springs are produced using the genetic material of the parents. Crossover is usually applied in a GA with a high probability – pc .

Initialize the crossoverpopulation

Set the elite individuals in the crossover population.

//to set the other individuals in the population

Select one individual1 which is the fittest among the five individual selected from the population

Check whether to apply crossover to selected individual

Select one individual2 which is the fittest among the five individual selected from the population

If crossover rate > randomValue

If probability >0.5

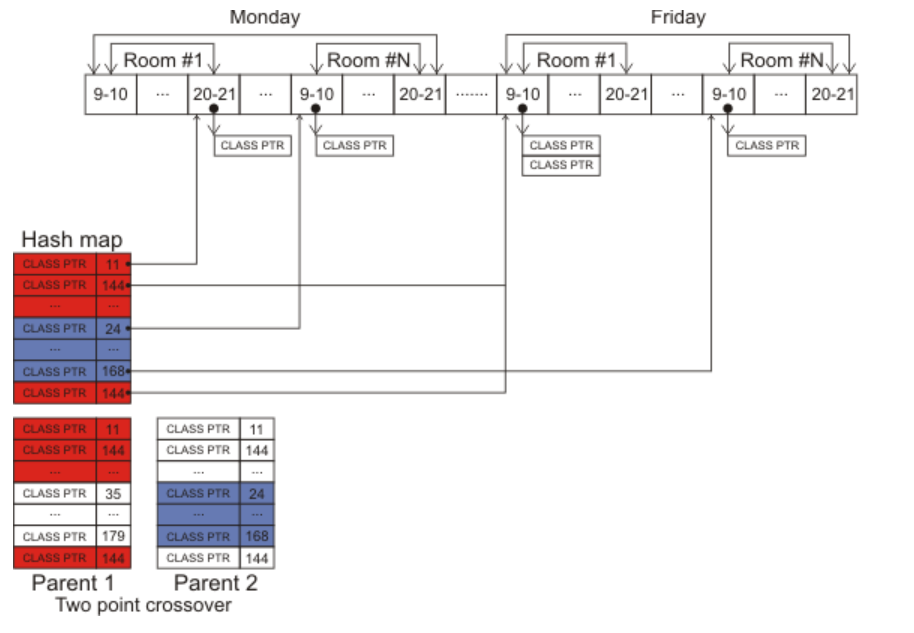
Select the chromosomes from schedule1

Else

Select the chromosomes from schedule2

Else

Select the schedules from the existing population.



1. **Mutation:** Mutation is similar to the biological mutation. Mutation changes one or more genes from the chromosomes. In our approach we have generated the population randomly if there is the possibility of mutation rate.

Initialize the mutate population

Loop over current population by fitness

Create a random individual to swap genes with

Loop over individual’s genes

Skip mutation if this is an elite individual

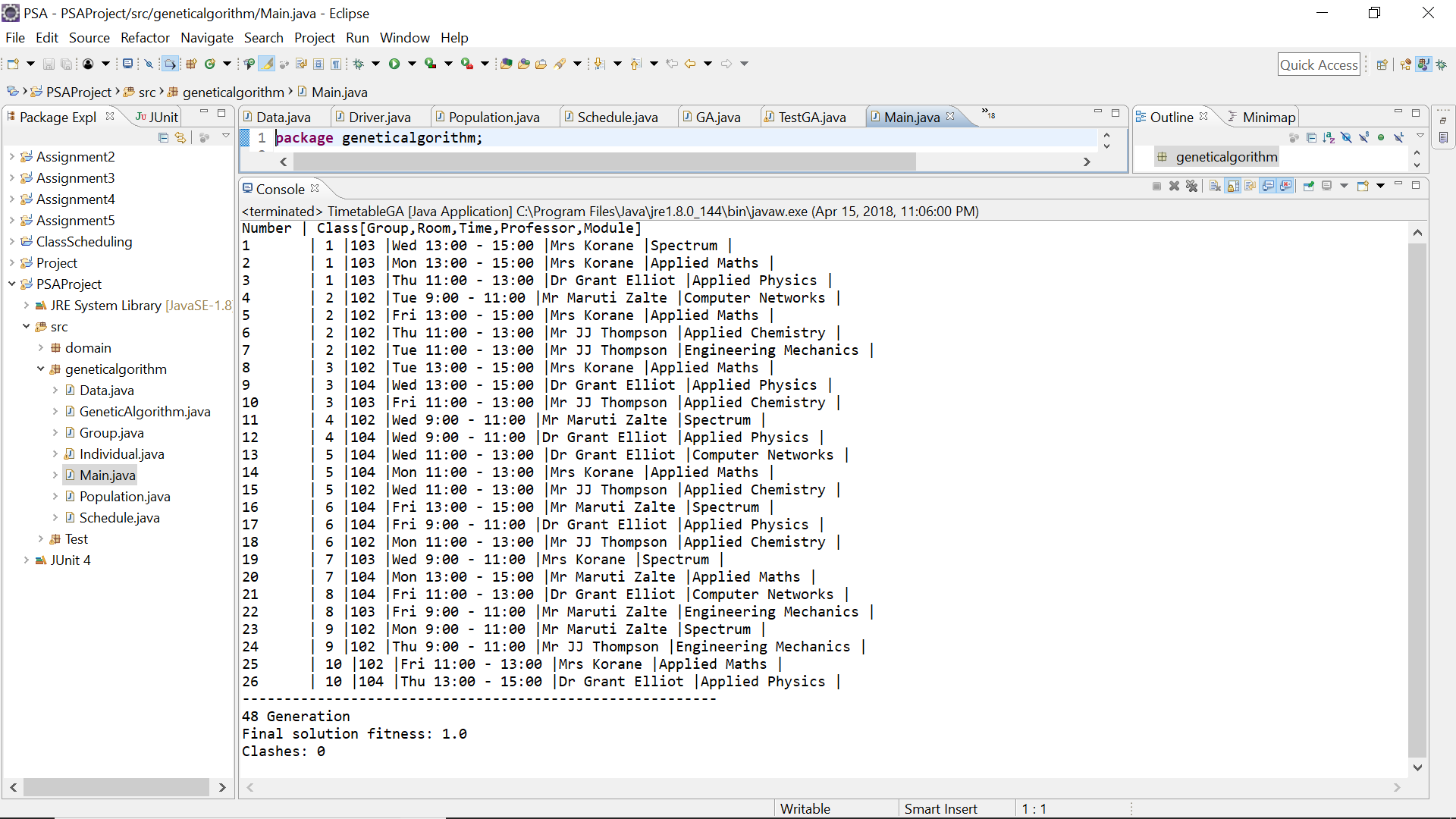
Check if gene needs mutation

Swap for new gene

Add individual to population and return new population

1. **Fitness Function:** The fitness function determines the quality of the given solution set. In this problem set the fitness function depends on the number of clashes. Clashes like room size and group size clash, professor availability clash etc. The lesser the number of clashes, the more fit the individual is.
2. **Terminating Condition:** If there are no clashes then fitness function will be equal to 1 and we will get our solution. Also, if number of generations passed is greater than max generations program will terminate.

**Output:**

****

**Conclusion:**

Our problem was to develop a class scheduling algorithm that will automatically generate a time table considering all the parameters. According to our study and tests, the genetic algorithm generates a result which is quite close to the optimal result. We observed that, when we take a small number of generations number of clashes increases and we don’t get optimal result. Whereas, when we take generations greater then 100 we achieved optimal results. Genetic algorithms have a drawback that it doesn’t guarantee finding global maxima. Also, we usually need a decent sized population and a lot of generations before we see good results.

**References:**

1. Introduction to Genetic Algorithm:

<https://www.analyticsvidhya.com/blog/2017/07/introduction-to-genetic-algorithm/>

1. Making a class schedule using genetic algorithm

<https://www.codeproject.com/Articles/23111/Making-a-Class-Schedule-Using-a-Genetic-Algorithm>