**PROGRAM STRUCTURES AND ALGORITHM**

**INFO 6205**

**Prof. Robin Hillyard**

**TIME TABLE GENERATION USING GENETIC ALGOTITHM**

****

By

Akilan Rajendaran

Niranjhani Vasudevan

**PROBLEM STATEMENT**:

Time Table creation is really a tedious and arduous job. It really requires patience to create a timetable given the set of data. By using genetic algorithm, the time taken to create time table can be easily be reduced without any errors. Given a set of data that has list of professors, list of courses associated with each professors list of class rooms available, list of time slots, and capacity of each classrooms. Using genetic algorithm the timetable can easily calculated without any conflicts. Timetabling has vast set of search space, and lot of solutions can be created in each search space though it is not feasible. The solution which are feasible will satisfy the soft-constraint and does not violate the soft constraints.

**LITERATURE SURVEY**:

Genetic algorithm is mainly used for optimization and general search algorithms. We can obtain one optimal solution at any time. Each population generated represents a solution set. The individuals are chosen from the parent population and they are mated (Crossoverpopulation) or mutated (Mutatepopulation) to obtain a new set of population. In this way new generation can be created. In GA it uses three process “Selection”, ”Crossover”, ”Mutation”. Genetic algorithm has these following functions in common.

1. Population (List of chromosomes)
2. Selection depending on the fitness.
3. Crossover or mutate to generate new offspring.

**GA PROCESS FOR TIME TABLE SCHEDULING:**

1. **Chromosomes generation:**

Chromosomes are proposed solution in the search space. Here the chromosomes are the Classes that are combination of departments, class, room, instructor, meeting time.

The fitness function is based on how well the chromosomes matches with problem solution ie., Time table generation.

1. **Individual 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.

1. **Selection:**

Here we are taking the fittest individual for mutation and crossover. We are selecting a random set of individuals using a TOURNAMENT\_SIZE where it is initialized as 3followed by selecting fittest among the three.

1. **CrossOver Population:**

Crossover varies the chromosome pattern from one generation to the next. It simply means selecting fittest parents (Individuals).The selection process is carried on as explained previously and if the probability is less than 0.5 the chromosomes from parent one is selected else the chromosomes from the parent two is selected.

Initialize the crossoverpopulation

Set the elite individuals in the crossover population.

//to set the other individuals in the population

Select one schedule1 which are the fittest among the three schedules selected

Select one schedule2 which are the fittest among the three schedules selected

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. **Mutate Population :**

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

Set the elite individuals in the mutate population.

//to set the other individuals in the population

If there is a possibility of mutation rate

Set the remaining individuals randomly from the initial 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. The lesser the number of clashes, the more fit the schedule is.

If (fitnessChanged)

fitness =Cal Fitness

fitnessChanged = false;

return fitness

**Calc Fitness**

forEach class x

If roomCapacity is greater than the student

Increment the conflict

forEach class y

If x.meetingtime is equal to y.meetingtime

If x.room is equal to y.room

Increment the conflict

If x.prof is equal to y.prof

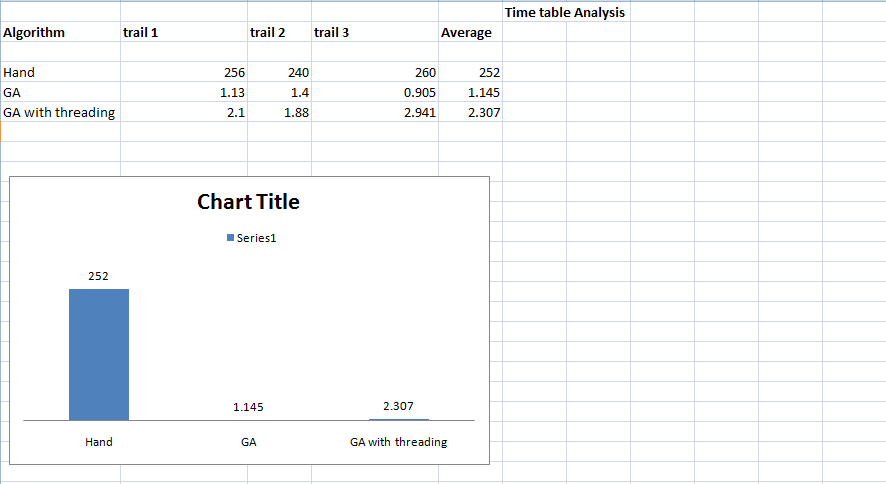
Increment the conflict

return 1/number of conflict +1

1. **Termination Condition:**

If there are no clashes with the time table generated, then the fitness function would be equal to 1. If it is so, terminate.

**ANALYSIS:**



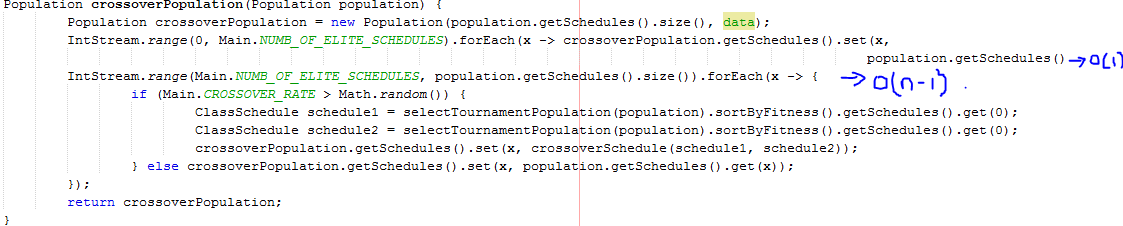
**Time Complexity:**

**Initialization:**

population = geneticAlgorithm.evolve(population).sortByFitness(); -- > O(n) where n is population size.

**Evolution:**

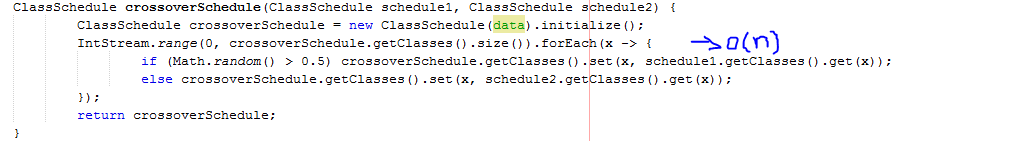
**CrossOverPopulation**

****

For sort by Fitness it takes O(n^2) but since we are sorting only 3 schedules (n=3)

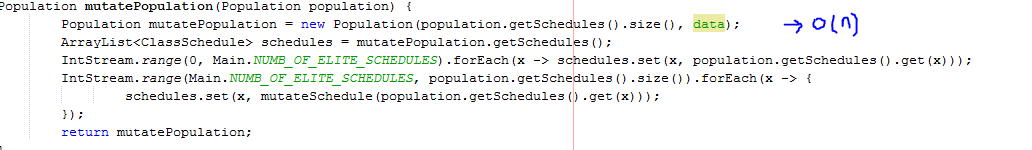
which is negligible.

**CrossOverSchedule**



For Mutation

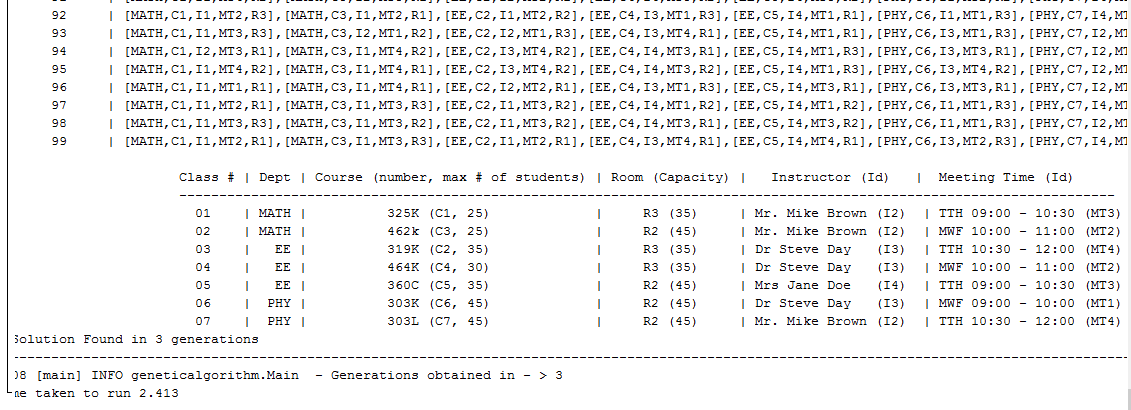
**MutatePopulation**



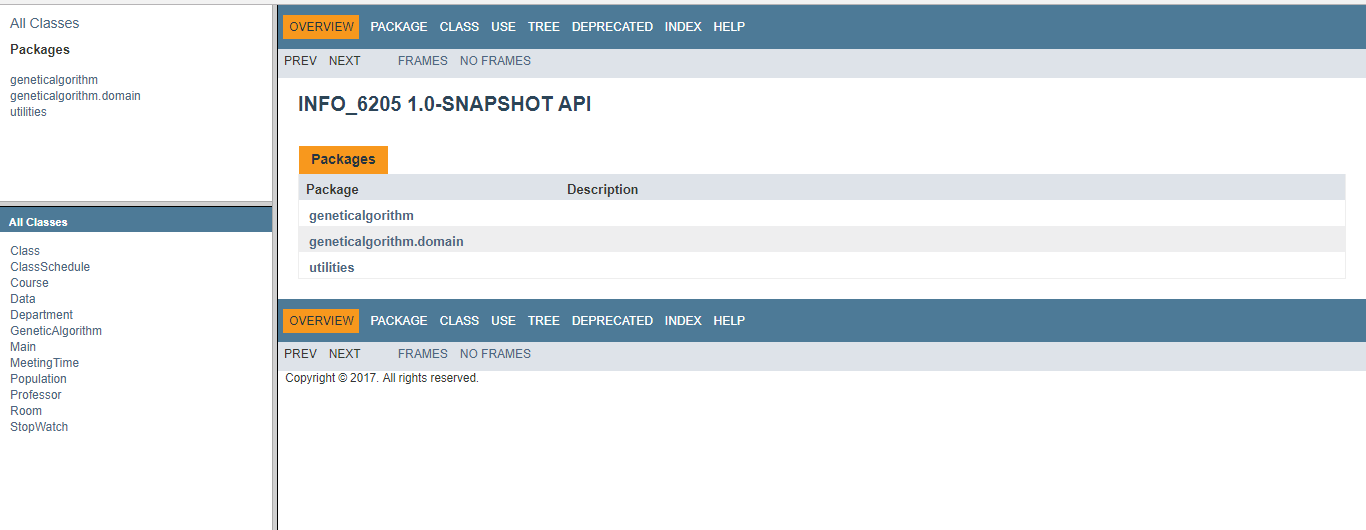
population = geneticAlgorithm.evolve(population).sortByFitness(); 🡪 O(n)

In best case scenario the fittest schedule is found in first occurrence. Since it is the randomized algorithm, the average case scenario can be taken.

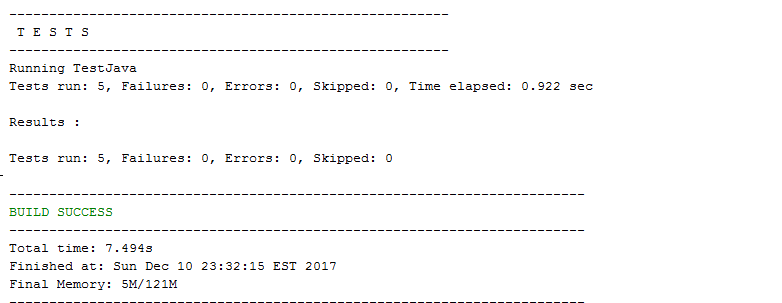
**Output:**



**APIDOCS:**



**JUnit Test Case:**

****

Reference :

1. <https://www.researchgate.net/publication/2253354_A_Genetic_Algorithm_To_Solve_The_Timetable_Problem>
2. <https://www.ijarcce.com/upload/2015/february-15/IJARCCE4I.pdf>
3. <http://ieeexplore.ieee.org/document/4682035/>
4. <http://cdn.intechopen.com/pdfs/30303.pdf>