AI PROJECT REPORT

NAME: Salman Jan

ROLL NO: 21I-2574.

SECTION: F.

PRE-PROCESSING

In this phase, I created hard coded arrays for **Courses, No of Section/course, Professors, Days, Time Slots, Rooms.** Then I assigned each course section a random strength between 30 and 120. Then I assigned random professors to each section, while making sure that no professor is assigned more than 3 course sections. For each course I am keeping the record of whether the course is **Lab** or **Theory** in two ways, first I am explicitly storing the type against course, and secondly, I am placing Labs after certain index in Courses array. For rooms, I am storing rooms information as a tuple in array, first value of room is room name and 2nd value of tuple is room size.

Gene- Structure.

We are provided with the following gene-structure.

Course, Theory/Lab, Section, Section-Strength, Professor, First-lecture-day,

First- lecture-timeslot, First-lecture-room, First-lecture-room-size, Second-lecture-day, Second lecture-timeslot, Second lecture-room, Second lecture-room-size.

We can see that “Course, Theory/Lab, Section, Section-Strength, Professor” this part of the gene remains same throughout the code, so I made separate array **Course Details**, storing this information for each course section.

For other parts of the gene, I take the random index in range of array size and convert the index to its binary form and append it to the gene structure. For example, I select random index from Days array and convert it to binary and append it to the gene.

So, the final structure of the gene follows this pattern.

3-bits for First Lecture-day.

3-bits for First Lecture time slot.

6-bits for First Lecture room.

3-bits for Second Lecture-day.

3-bits for Second Lecture time slot.

6-bits for Second Lecture room.

3-3-6-3-3-6 = 24 bits length of 1 gene.

This gene is for one course section, and in similar way I calculate the gene for all course sections and append them and I get the **Chromosome.**

In my case, total course sections are 46(not hard coded, can be changed without effecting the output) so my **Chromosome length is 46 * 24(gene length) = 1104 bits**.

GENETIC ALGORITHM.

It’s time for the most awaited part, the Genetic Algorithm. Dividing into 5 parts.

1. **Generating Initial Population (1000 in my case, not hard coded).**
2. **Fitness Function**
3. **Selection (Tournament Selection)**

4. **Crossover (Two-Point)**
5. **Mutation (Single-Point)**

Fitness Function:

In Fitness function I check all the clashes for hard and soft constraints. For hard constraints I add a value of -1 to total clashes and for soft constraints I add a value of -0.5 to total clashes. All checks are properly commented in code, so not explaining them here.

Tournament Selection:

Now I have Initial population and fitness scores of the population, I send these values to the tournament selection function. Here I have to select new parents for mating. So, I select two random Individual from population and do a selection fight between them, the one with greater fitness score wins and is send to the mating pool. For new generations I need new parents equal to the size of initial population, so I do N (Initial population size) such rounds and get N new parents for mating.

Crossover:

I get two parents in this function. First, I select two random points for crossover and then perform the crossover to get two new children.

Mutation:

For each child produced after crossover, I apply mutation to them. For mutation I select one random point and flip the bit of that point.

Correct the Gene:

During crossover since it is randomly taking points and changing individuals, there is a chance of getting values out of range for specific part of gene, so I check each gene and if there is a part getting out of range, I re assign a random Index to that part, and finally get a **New Generation.**

After getting the new generation, the cycle continues. First, calculating the fitness of each individual, then performing tournament selection, then crossover, then mutation and finally correcting the gene. The cycle

continues until **Max Iterations** are allowed.

Generating the output:

When I get the individual with fitness value of 0, I pass this individual to function which decodes it and puts the value in **Excel File**. For this part I took the help from **ChatGPT.**