Artificial Intelligence Project

Timetable Scheduling

Syed Ata ul Muhaimen Ahmad

21I-0888

BS (CS) – G

- **Introduction:**

  The report consists of the implementation steps applied for timetable scheduling by using **genetic algorithm**. First of all, a population of chromosomes is generated in which each chromosome represents a module for a particular section's particular course. In addition to the standard steps of genetic algorithm i.e. mutation and crossover, there were several other constraints to be applied on our chromosome. By following all these steps and the provided guidelines a timetable is generated with a minimum number of clashes.

- **Implementation Steps:**

  ❖ **Population Initialization:**

  The first step involves the declaration of all the maximum possible resources i.e. courses, teachers, sections etc. The values are converted to binary form for further processing. Also, the maximum number of bits required to store that value is calculated along with initializing strings containing that many zeroes. After doing all these the initialized population method is used to create the required number of chromosomes i.e. courses*sections.

  ❖ **Fitness Calculation:**

  This step involves calculating the initial fitness of all the chromosomes created in the first step. For this, the hard constraints and some the soft constraints are used to calculate the clashes and in the end the method returns the inverse of the total number of clashes which shows the fitness of the chromosome.

- **Tournament Selection:**

  The tournament selection function is used to select worst two chromosomes for crossover. The tournament size is set, the smaller it is the better it will be as it will consider more unique groups and so the results are much better. So, this function selects two worst chromosomes from the tournament size and returns it for proceeding to crossover.

- **Crossover:**

  Once the worst two chromosomes are returned by the tournament selection function they are used for crossover. A two – point crossover is implemented. The first point is any random bit of the first lecture and the second point is any random bit of the second lecture. Thus, two children are generated and then proceeded for further processing.

- **Mutation:**

  After the children are generated because of crossover, their fitness is calculated and if their fitness is lower than that of their parents then it is sent for mutation. In mutation, for the specific section's specific course, random bit for first and second lecture is flipped. Thus, creating a mutated chromosome.

- **Comparing Results:**

  After the mutated chromosome is created, its fitness is calculated. If its fitness is better than the original chromosome then it replaces that chromosome otherwise the parent chromosomes are retained.

- **Generate Timetable:**

  All these steps are repeated for the whole population, until all of them have fitness of '0' which means no clashes. However, after a certain number of iterations if no change occurs the process is terminated.

- **Display Results:**

  After the process is terminated, the results are displayed. The classes held in rooms at a particular timeslot are displayed.

- **Results:**

  The results obtained show that initially the population contains chromosomes with low fitness i.e. the timetable have a lot of clashes, but after the genetic algorithm is applied the clashes are resolved and population with best fitness is obtained. Thus, our generated timetable contains very low or no clashes.