REPORT

Name: Marrium Jilani (20k-1748)

This report contains a brief explanation of the steps involved in the program and the results obtained.

- Firstly, I have initialized a population that will have a randomly generated schedule each. The exams will be scheduled according to hall, timeslots, duration of each timeslots and course names. The population size here was defined to be 100. For the sake of this question, I have used the descriptions provided in the example
- After I have population the schedule population with 100 schedules, I have calculated the fitness values of each schedules and stored it in a variable.
- The value of the fitness function has been decided based on the conflicting students, each time a course with conflicting students is scheduled in the same timeslots in different halls, there will be a conflict. The number of conflicting students that arise due to this will affect the fitness score. Conflict penalty on each pair of students is 100. However, if the exam is scheduled in the same hall in the same time slot, there will be no conflict. If the usage of the hall exceeds 6 hours per hall, a penalty of 10 will be multiplied with each exceeding hour.
- After this I have applied tournament selection on the population with tournament size being 5. This variable can be changed without affecting the functionality to make the code generic. The solutions with the least function values will be considered the best.
- Then, I have implemented single point crossover using the parents generated form this tournament selection and applied a mutation function.
- Finally, I have passed the parameters to the genetic algorithm. The algorithm starts with an initial population of solutions and iteratively generates new solutions through selection, crossover, and mutation. In each generation, the algorithm selects two parents from the population using tournament selection, applies single-point crossover to the parents with the probability given, and applies mutation to the resulting offspring with the given probability. The resulting offspring and parents are then combined into a single population and sorted based on their fitness scores. The best solutions from the combined population are selected as survivors to form the next generation's population. The algorithm terminates after a certain number of generations or when a satisfactory solution is found. Finally, the function returns the best solutions found by the algorithm.
- Then, I have printed the final best population whose each solution will have best possible fitness score.