ARTIFICIAL INTELLIGENCE

Assignment # 2

Instructions: **Deadline: 22 march, 2023**

1. Assignments are to be done individually. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class.
2. You have to use python programming language.
3. Plagiarism of any kind (copying from others and copying from the internet, etc.,) is not allowed and can result in zero marks in whole assignment category.
4. Your code must be properly commented.
5. Any assignment marked late by google will be considered late.
6. No marks will be assigned if any of the following deliverables are missing. The source code of the program. A pdf or word report containing a brief explanation of the steps involved in the program (each question) and the results obtained.
7. Put both source code and report in one folder, ZIP it and submit it. Your folder must be named as ROLLNO_NAME.ZIP

Problem Statement:

You are tasked with optimizing the scheduling of exams for a university. There are N courses to be scheduled and K exam halls available. Each course has a specific set of time slots available for scheduling its exam, and each exam hall can be used for a certain number of hours per day. In addition, some pairs of courses have conflicting students, meaning that they cannot be scheduled at the same time. Your task is to use a genetic algorithm to find a feasible and optimal schedule.

Requirements:

1. Define a representation for a solution to the scheduling problem. This could be a list of tuples, where each tuple represents an exam and contains information about its course, time slot, and exam hall.
2. Write a function to calculate the fitness of a solution. The fitness function should take a solution as input and output a score indicating how good the solution is. A good solution is one that satisfies all the constraints of the problem and minimizes the total number of conflicts between exams.
3. Implement a genetic algorithm to search for a good solution. Your implementation should include functions for initializing a population of solutions, selecting parents for crossover, performing crossover and mutation, and evaluating the fitness of the resulting offspring. You should experiment with different parameter settings to find the best combination.

4. Test your implementation on a set of sample problems with varying difficulty levels. You should report the results in terms of the fitness score achieved and the running time of the algorithm. You should also discuss the advantages and disadvantages of using a genetic algorithm for this type of problem compared to other optimization techniques.

Here is an example problem instance:

Suppose we have 5 courses (C1, C2, C3, C4, C5) and 2 exam halls (H1, H2). Each course has 3 time slots available (T1, T2, T3), and each exam hall can be used for a maximum of 6 hours per day. In addition, there are some pairs of courses with conflicting students:

- C1 and C2 have 10 common students.
- C1 and C4 have 5 common students.
- C2 and C5 have 7 common students.
- C3 and C4 have 12 common students.
- C4 and C5 have 8 common students.

Our task is to use a genetic algorithm to find a feasible and optimal schedule for these exams.

A possible solution could be:

[(C1, T1, H1), (C2, T2, H1), (C3, T3, H2), (C4, T1, H2), (C5, T2, H2)]

This solution schedules the exams for C1 and C2 in different time slots but the same exam hall, and schedules the exams for C3, C4, and C5 in different time slots and exam halls. It satisfies all the constraints of the problem and has a fitness score of 0.

We can use a genetic algorithm to search for other good solutions, and compare their fitness scores with the optimal solution found. For example, we can start with an initial population of 100 solutions, use tournament selection with a tournament size of 5, apply single-point crossover with a probability of 0.8, and apply mutation with a probability of 0.1, and run the algorithm for 100 generations. We can use the fitness function defined above with penalty values of 10 for each hour exceeding the maximum for an exam hall, and 100 for each conflicting student pair scheduled at the same time.

After running the genetic algorithm, we find the best solution with a fitness score of 20:

[(C1, T1, H1), (C2, T3, H1), (C3, T2, H2), (C4, T1, H2), (C5, T3, H2)]

This solution schedules the exams for C1 and C2 in different time slots but the same exam hall, schedules the exams for C3 and C5 in different time slots but the same exam hall, and schedules the exams for C4 in a different time slot and exam hall. It satisfies all the constraints of the problem but incurs penalties for exceeding the maximum usage time for exam hall H2.

Bonus: Extend your implementation to include a local search algorithm to improve the quality of the best solution found by the genetic algorithm. You should report the results of this extended implementation on the same set of sample problems and compare the results with those obtained by the basic genetic algorithm.

Good Luck! 😊