National University of Computer and Emerging Sciences

Applied Artificial Intelligence

"Assignment 2"

STUDENTS NAME: Sania Arshad

REGISTRATION NUMBER: i202425

DEGREE PROGRAM: BS-SE

SECTION: R

SUBJECT NAME: Applied Artificial Intelligence

DATE OF SUBMISSION: 25/03/2023

SUBMITTED TO: Ma'am Shahela Saif

STUDENT SIGNATURE:

_____

MARKS:

_____

REMARKS:

_____

TEACHERS SIGNATURE:

_____

Genetic Algorithm Code

```
import random

class Course:
    def __init__(self, name: str, timeslot: str):
        self.name = name
        self.timeslot = timeslot
class ExamHall:
    def __init__(self, name, maxTime):
        self.name = name
        self.maxTime = maxTime

c1 = Course("Mathematics", "9-10")
c2 = Course("English", "12-1")
c3 = Course("History", "1-2")
c4 = Course("German", "9-10")
c5 = Course("Chinese", "12-1")

h1 = ExamHall("H1", 6)
h2 = ExamHall("H2", 6)

courses = [c1, c2, c3, c4, c5]
timeslots=  ["9-10", "12-1", "1-2"]
halls= [h1, h2]

conflicts = [(c1, c2, 10), (c1, c4, 5), (c2, c5, 7), (c3, c4, 12), (c4, c5, 8)]

def initializePopulation(courses, timeslots, halls, conflicts, sizeOfSol):
    population = []
    for i in range(sizeOfSol):
        solutions = []
        for course in courses:
            timeslot = random.choice(timeslots)
            hall = random.choice(halls)
            solutions.append((course.name, timeslot, hall.name))
        population.append(solutions)
    return population

def checkFitness(solution, conflicts, halls, maxTime):
    fitness = 0
    for conflict in conflicts:
      course1, course2, students = conflict
      course1_assigned = False
      course2_assigned = False
      for c in solution:
```

```
        if c[0] == course1.name and c[1] == course1.timeslot:
            course1_assigned = True
        if c[0] == course2.name and c[1] == course2.timeslot:
            course2_assigned = True
      if course1_assigned and course2_assigned:
        fitness -=  100
    hall_time = {"H1": 0, "H2": 0}
    for course, timeslot, hall in solution:
        hall_time[hall] += 1
        if hall_time[hall] > maxTime:
            fitness -= 10
    return fitness

def rouletteWheelSelection(population, conflicts, halls, maxTime):
    fitness_values = []
    for solution in population:
        fitness_values.append(checkFitness(solution, conflicts, halls, maxTime))
    total_fitness = sum(fitness_values)

    probabilityOfFitness = []
    for fitness in fitness_values:
        probability=fitness/total_fitness
        probabilityOfFitness.append(probability)

    parents = []
    for i in range(2):
        maxFitnessIndex = probabilityOfFitness.index(max(probabilityOfFitness))
        parents.append(population[maxFitnessIndex])
        probabilityOfFitness[maxFitnessIndex] = 0 #so that it isnt chosen again
when called
    return parents

def singlePointCrossover(parent1, parent2):
   k = random.randint(0, 4)
   j= parent1[k]
   parent1[k]=parent2[k]
   parent2[k]= j
   return parent1

def mutation(solution, timeslots, halls):
    k = random.randint(0, 4)
    course, timeslot, hall = solution[k]
    new_timeslot = random.choice(timeslots)
    solution[k] = (course, new_timeslot, hall)
    return solution
```

```
def geneticAlgorithm(courses, timeslots, halls, conflicts):

  population = initializePopulation(courses, timeslots, halls, conflicts, 100)
  parents =rouletteWheelSelection(population, conflicts, halls, 6)
  parent1=0
  probabilityOfCrossover= 0.8
  probabilityOfMutation= 0.1
  if random.random() < probabilityOfCrossover:
    parent1= singlePointCrossover(parents[0], parents[1])
    if random.random() < probabilityOfMutation:
        parent1 = mutation(parent1, timeslots, halls)
  return parents[0]


maxSolutions = 1000
Sol = 1
solution = geneticAlgorithm(courses, timeslots, halls, conflicts)
fitnessCheck = checkFitness(solution, conflicts, halls, 6)

while fitnessCheck != 0 and Sol <= maxSolutions:
    print(f"Solution {Sol}: Fitness Value = {fitnessCheck}")
    Sol += 1
    solution = geneticAlgorithm(courses, timeslots, halls, conflicts)
    fitnessCheck = checkFitness(solution, conflicts, halls, 6)

if fitnessCheck == 0:
    print("Best Solution Obtained:")
    print(solution)
else:
    print("The limit for maximum solutions has been reached. Solution is not
found")
```

Explanation of the code

The above code uses Genetic Algorithm in order to find an optimal and feasible solution for scheduling of exams. The algorithm then uses the fitness function to evaluate the quality of each solution.

The algorithm generally starts with initializing the population. It takes the size of the population to be generated in a parameter. The population is randomly generated with the use of random function. It generates potential solution in forms of tuples in (course, timeslot, hall) format.

In order to select the two best potential parents, we used the roulette wheel selection method. This function uses the checkFitness method which evaluates the fitness of a solution by penalizing conflicts between courses and overuse of exam halls. If there are conflicting students the fitness value will be

subtracted 100 times for each student and if the time exceeds the maximum allotted time the penalty value is subtracted by 10. Probability of each potential solution is calculated after the evaluating their fitness. Two solutions with the best probability value are then selected as parents.

In order to generate a child solution, we used single point crossover method where we swapped an attribute of a tuple based on random selected value. To diversify the solution, we then applied mutation.

Testing the implementation

To test the code, multiple sets of input data were used. The data we used represented different scheduling problems with varying difficulty levels to check how well the algorithm performs.

Test Data 1

5 courses (Maths, English, History, German, Chinese) and 2 exam halls (H1, H2). Each course has 3 time slots available (9-10, 12-1, 1-2)

- Maths and English have 10 common students.
- Maths and German have 5 common students.
- English and Chinese have 7 common students.
- History and German have 12 common students.
- German and Chinese have 8 common students.

```
Solution 52: Fitness Value = -300
Solution 53: Fitness Value = -500
Solution 54: Fitness Value = -400
Solution 55: Fitness Value = -300
Solution 56: Fitness Value = -300
Solution 57: Fitness Value = -300
Best Solution Obtained:
[('Mathematics', '9-10', 'H1'), ('English', '9-10', 'H1'), ('History', '1-2', 'H1'), ('German', '1-2', 'H2'), ('Chinese', '12-1', 'H2')]
time taken to run: 0.22361048099999437
```

Solution with fitness value of 0 has been obtained after iterating the function 57 times. 0 value indicates that there are no conflicts therefore the solution obtained is accurate. The negative values above are generated considering the penalties of -100 for clashes and -10 for exceeding maximum time of each hall.

The total time taken to run the algorithm is 0.2 seconds approx.

Test Data 2

10 courses (Mathematics, English, History, German, Chinese, Biology, Chemistry, Physics, Computer Science, Art) and 3 time slots (9-10, 12-1, 1-2) available for each course. There are also 3 exam halls (H1, H2, H3) available for scheduling exams. The courses are:

- Mathematics and English have 10 common students.
- Mathematics and German have 5 common students.
- English and Chinese have 7 common students.
- History and German have 12 common students.
- German and Chinese have 8 common students.
- English and History have 6 common students.

- Biology and Chemistry have 9 common students.
- Physics and Computer Science have 4 common students.
- Chemistry and Art have 3 common students.
- History and Computer Science have 7 common students.

```
Solution 201: Fitness Value = -900
Solution 202: Fitness Value = -500
Solution 203: Fitness Value = -500
Best Solution Obtained:
[('Mathematics', '9-10', 'H1'), ('English', '9-10', 'H3'), ('History', '12-1', 'H3'), ('German', '1-2', 'H1'), ('Chinese', '12-1', 'H2'),
('Biology', '12-1', 'H3'), ('Chemistry', '1-2', 'H2'), ('Physics', '1-2', 'H2'), ('Computer Science', '1-2', 'H1'), ('Art', '1-2', 'H2')]
time taken to run: 0.8130842790000088
```

As the number of courses were increased the algorithm took 0.8130842790000088 seconds in order to find the most accurate solution.

Test Data 3

10 courses (Mathematics, English, History, German, Chinese, Biology, Chemistry, Physics, Geography, Economics)

8 time slots (9-10, 10-11, 11-12, 12-1, 1-2, 2-3, 3-4, 4-5)

5 exam halls (H1, H2, H3, H4, H5)

- Mathematics and English have 10 common students.
- Mathematics and German have 5 common students.
- English and Chinese have 7 common students.
- History and German have 12 common students.
- German and Chinese have 8 common students.
- English and History have 6 common students.
- Biology and Chemistry have 10 common students.
- Chemistry and Physics have 15 common students.
- Physics and Geography have 5 common students.
- Geography and Economics have 12 common students.
- Economics and Biology have 8 common students.
- Mathematics and Biology have 6 common students.
- English and Chemistry have 4 common students.
- History and Physics have 7 common students.
- German and Geography have 9 common students.
- Chinese and Economics have 11 common students.

```
Solution 197: Fitness Value = -600
Solution 198: Fitness Value = -500
Solution 199: Fitness Value = -300
Best Solution Obtained:
[('Mathematics', '1-2', 'H3'), ('English', '1-2', 'H2'), ('History', '9-10', 'H1'), ('German', '9-10', 'H2'), ('Chinese', '1-2', 'H3'),
('Biology', '9-10', 'H1'), ('Chemistry', '9-10', 'H1'), ('Physics', '1-2', 'H3'), ('Geography', '1-2', 'H1'), ('Economics', '12-1', 'H2')]
time taken to run: 1.0016030169999794
```

The total time taken to run the algorithm increases as the data dataset size increases this algorithm took 1.0016030169999794 seconds to execute.

Advantages/Disadvantages of using Genetic Algorithm

Following are the advantages and disadvantages of genetic algorithm based on my observations according to the dataset used.

Advantages

- The algorithm can generate optimal solutions in a short amount of time.
- It is customized according to suit various constraints such as conflicting students and exceeding the maximum hall time.

Disadvantages

- As the dataset is increased, the running time also increases.
- In order to find the best possible solution, the algorithm must be executed several times. There is a low chance that the best solution will be given in first try.
- Crossover and mutation might cause good solutions to be lost. This can cause the final solution to not be optimal at all.