Exam Scheduler

Exam Scheduler Using Genetic Algorithm

Introduction

The exam scheduling system objective is to efficiently allocate exam slots for a group of students across different courses while looking out to set constraints and preferences. This report gives a detailed overview of the system's design, implementation, and evaluation, including the "genetic algorithm" used for scheduling and the comprehensive constraint testing suite.

System Overview

The system is built using Python language and leverages the pandas library for data handling. It uses a Genetic Algorithm (GA) to generate optimal schedules by iteratively evolving a population of potential solutions. The GA considers both hard and soft constraints to ensure the feasibility and fairness of the generated schedules.

Implementation Details

- Class Structures:
 - Course: Holds course code and name.
 - Teacher: Stores teacher names.
 - Student: Contains student names and their enrolled courses.
- Data Handling:
 - Data regarding courses, teachers, and students are loaded from CSV files. Each
 course, teacher, and student are represented as an object, hence easy
 manipulation, and access to relevant information during scheduling.
- Algorithm Parameters:

Population Size: 60Crossover Rate: 70%Mutation Rate: 40%

Generations: 50

Genetic Algorithm

def genetic_algorithm()

Main loop that evolves the population over multiple generations.

• Population Initialization:

def initialize_population(size)

Initial schedules are randomly generated, ensuring that each course is assigned a valid exam slot while avoiding conflicts.

Fitness Calculation:

def calculate fitness(individual)

The fitness function evaluates the quality of each schedule by deducting points for violations of constraints, both hard and soft. Higher fitness scores indicate better schedules meaning all constraints are being satisfied.

• Selection:

def roulette wheel selection(population, fitness)

Roulette wheel selection is used to choose parents for crossover, favoring the schedules with higher fitness.

Crossover and Mutation:

def crossover(parent1, parent2) def mutate(individual, mutation_rate=0.1)

Crossover and mutation operations introduce diversity and explore the solution space, aiming to improve the quality of schedules over the generations.

Termination:

The algorithm runs for a predefined number of generations, continually improving schedules towards better solutions.

Constraint Testing Suite:

A suite of functions is provided to comprehensively test the adherence of generated schedules to specified constraints. These tests cover both the hard and soft constraints, ensuring that the schedules are feasible and are optimized.

def test_hard_constraints(schedule, students, teachers, courses)
def test_soft_constraints(schedule, students, teachers, courses)

```
Testing Hard Constraints:

Pass: Test for scheduling an exam for each course.

Pass: Test for no student taking more than one exam at a time.

Pass: Test for no exams scheduled on weekends.

Pass: Test for all exams scheduled between 9 AM and 5 PM.

Pass: Test for each teacher not invigilating two exams at the same time.

Pass: Test for teachers not invigilating two consecutive exams.

Pass: Test for no double booking of rooms.

Pass: Test that all students are enrolled in at least two courses.
```

```
Testing Soft Constraints:

Pass: Test for Friday break from 1-2 PM.

Fail: Test for no more than one consecutive exam for any student.

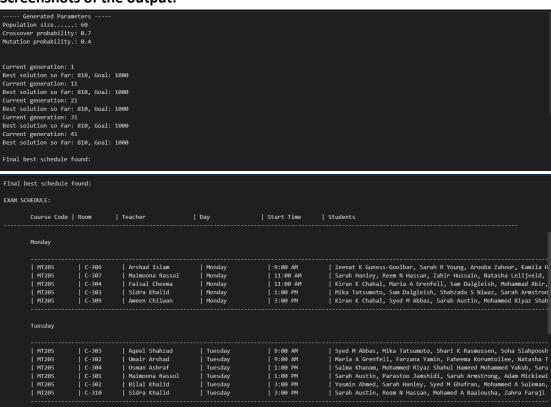
Pass: Test for MG course before CS course for any student enrolled in both.

Pass: Test for two hours of faculty break in the week.
```

Conclusion:

The exam scheduling system effectively uses the Genetic Algorithm to generate optimal schedules while also considering the various constraints and preferences. By including a detailed constraint testing suite, the system checks the integrity and fairness of generated schedules, facilitating efficient examination management for educational institutions. More enhancements can include additional optimization techniques and user interface improvements for enhanced usability.

Screenshots of the output:



Wednesday					
MT205	C-306	Mehwish Hassan	Wednesday	9:00 AM	Zahir Hussain, Maria A Grenfell, Shari K Rasmussen, Tina
MT205	C-302	Sumera Abbas	Wednesday	9:00 AM	Sarah Hanley, Nabila Altaf, Shahzada S Niwaz, Zaki Choudh
MT205	C-304	Hasan Mujtaba	Wednesday	1:00 PM	Farzana Yamin, Zahra Faraji Rad, Usman Rafiq, Nadine Meye
MT205	C-301	Usman Ashraf	Wednesday	3:00 PM	Mohammed Azam, Nabila Altaf, Arsheen Rajpar, Natasha Lees
MT205	C-305	Ejaz Ahmed	Wednesday	3:00 PM	Syed M Abbas, Nausheen Saleem, Usman Rafiq, Mohammad Abir
Thursday					
Thursday					
 MT205	C-3 0 2	Mehreen Alam	Thursday	9:00 AM	Yasmin Ahmed, Damian A Cummings, Mohammed A Suleman, Tina
MT205 MT205	C-309	Farwa Batool	Thursday	9:00 AM	Sam D Edwards, Sarah Nolasco, Ana Vukojevic, Leila C Payr
 MT205					Sam D Edwards, Sarah Nolasco, Ana Vukojevic, Leila C Payr
MT205 MT205	C-309	Farwa Batool	Thursday	9:00 AM	
MT205 MT205 MT205	C-309	Farwa Batool	Thursday Thursday 	9:00 AM	Sam D Edwards, Sarah Nolasco, Ana Vukojevic, Leila C Payn Yasmin Ahmed, Sarah J Roberts, Damian A Cummings, Ana Vul
MT205 MT205 MT205 MT205	C-309 C-309	Farwa Batool Zainab Abaid	Thursday Thursday	9:00 AM 11:00 AM	Sam D Edwards, Sarah Nolasco, Ana Vukojevic, Leila C Payr
MT205 MT205 MT205 MT205	C-389 C-389 C-389	Farwa Batool Zainab Abaid Farinab Abaid Sajid Khan	Thursday Thursday 	9:00 AM 11:00 AM 11:00 AM	Sam D Edwards, Sarah Nolasco, Ana Vukojevic, Leila C Payr Yasmin Ahmed, Sarah J Roberts, Damian A Cummings, Ana Vuk Salma Khanam, Sarah L Barber-Jones, Syed M Ghufran, Paras