National University of Computer and Emerging Sciences
Islamabad Campus

Assignment 03

Exam Schedule Generator Using Genetic Algorithm

ARTIFICIAL INTELLIGENCE

Submitted by: Muhammad Azan Afzal

Roll number: 22I-1741

Date: April 5, 2025

Table of Contents

- # Introduction

This report outlines the implementation of a genetic algorithm (GA) to generate an optimal exam schedule for a university while satisfying various constraints. The GA approach is particularly well-suited for this problem because it can effectively search through a large solution space to find near-optimal solutions for complex constraint-based problems.

- # Steps

Dataset Overview

The implementation uses four CSV files:

1. **courses.csv**: Contains course codes and names
2. **studentNames.csv**: Contains student names
3. **teachers.csv**: Contains teacher names
4. **studentCourse.csv**: Maps students to their enrolled courses

Genetic Algorithm Components

1. Chromosome Representation

Each chromosome represents a complete exam schedule. It is structured as a dictionary where:

- Keys are course codes
- Values are dictionaries containing the exam details (day, time slot, room, teacher, duration)

This representation allows for easy access to exam details and efficient evaluation of constraints.

2. Initial Population

The initial population is created by generating random schedules. Each schedule assigns random values for:

- Day (Monday to Friday)
- Time slot (9 AM to 4 PM)

- Room (C301 to C310)
- Invigilating teacher

The population size is set to 50 individuals to balance exploration and computational efficiency.

3. Fitness Function

The fitness function evaluates how well a schedule satisfies both hard and soft constraints:

Hard Constraints (Essential Requirements):

- Every course must have an exam scheduled
- No student should be assigned to overlapping exams
- Exams are scheduled only on weekdays (Monday-Friday)
- Exam timings must be between 9 AM and 5 PM
- Each exam must have an invigilating teacher
- No teacher is assigned to multiple exams simultaneously
- No teacher should have consecutive exam invigilation duties

Soft Constraints (Optimization Criteria):

- A common break on Friday from 1-2 PM for all students and teachers
- Students should not have back-to-back exams
- Management course exams should be scheduled before Computer Science course exams
- A two-hour break for faculty meetings

Violations of hard constraints result in large penalties, while satisfaction of soft constraints adds rewards. The overall fitness score is calculated as:

Fitness = 1000 – penalties + rewards

4. Selection Mechanism

As specified in the requirements, roulette wheel selection is used to select parents for reproduction. The selection probability for each individual is proportional to its fitness value. This ensures that fitter individuals have a higher chance of being selected while still maintaining diversity in the population.

Justification: Roulette wheel selection provides a good balance between exploration and exploitation. It favors better solutions but still gives weaker solutions a chance, which helps maintain genetic diversity and avoids premature convergence.

5. Crossover Operator

A single-point crossover is implemented with a crossover rate of 0.8 (80% chance of crossover). During crossover:

1. A random crossover point is selected
2. Genes before the crossover point come from the first parent
3. Genes after the crossover point come from the second parent

Justification: Single-point crossover is effective for this problem because it preserves blocks of related assignments while allowing for genetic material exchange. The high crossover rate (0.8) encourages exploration of new solutions.

6. Mutation Operator

Mutation introduces small random changes to individuals with a mutation rate of 0.1 (10% chance per gene). For selected genes, one attribute (day, time, room, or teacher) is randomly changed.

Justification: This mutation rate provides a good balance between maintaining good solutions and introducing diversity. A higher rate might disrupt promising solutions, while a lower rate might lead to premature convergence.

7. Elitism

The algorithm implements elitism by preserving the best individual from each generation. This ensures that the best solution is never lost during evolution.

Algorithm Flow

1. Generate an initial random population
2. Evaluate fitness of each individual
3. For each generation:
    - Select parents using roulette wheel selection
    - Create offspring through crossover
    - Apply mutation to offspring
    - Evaluate fitness of offspring

- Replace the old population with the new one (keeping the best individual)

4. Return the best solution found

Termination Criteria

The algorithm terminates after a fixed number of generations (100) or when a perfect solution is found (fitness score of 1000 or higher).

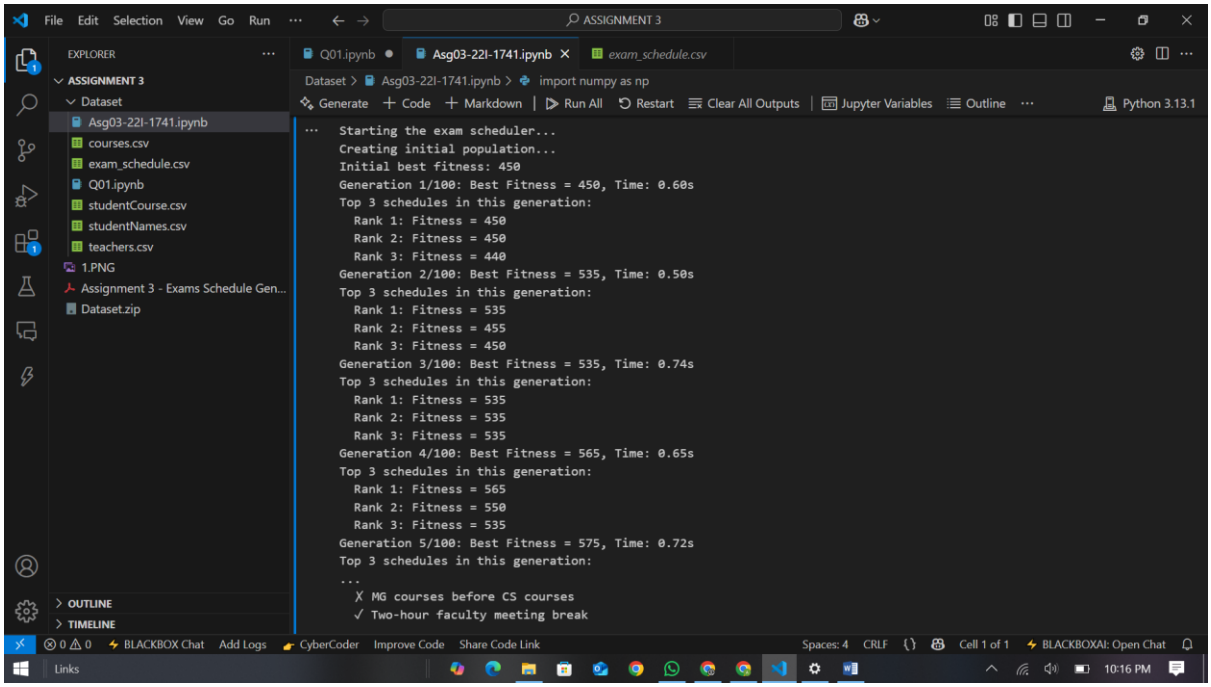
Output Format

The final schedule is formatted as a table with columns for:

- Course code
- Day of the week
- Time slot
- Room
- Invigilating teacher
- Duration

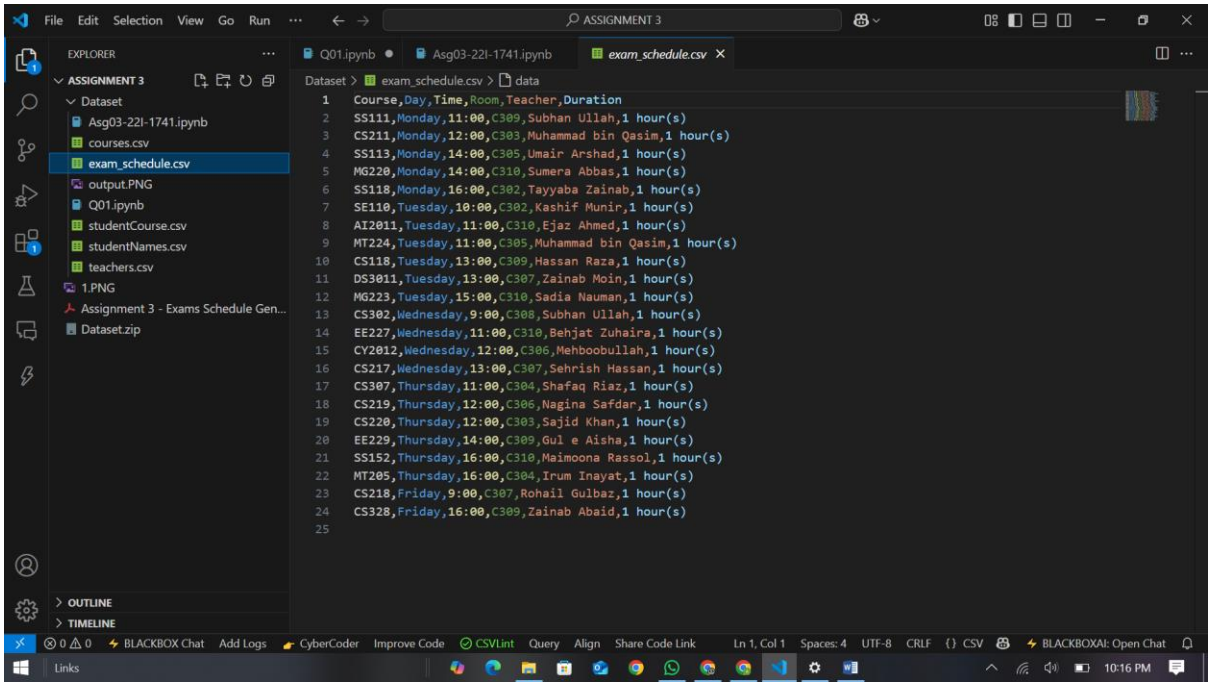
The output also includes a check of which hard and soft constraints are satisfied.

OUTPUT SCREENSHOTS



This shows the execution of what appears to be a genetic algorithm for exam scheduling optimization:

- Starting with an initial population
- Initial best fitness: 450
- The algorithm runs through multiple generations (showing generations 1-5 of 100)
- Fitness scores improve over generations (from 450 to 575)
- Some constraints are listed at the bottom:
  - ✕ MG courses before CS courses
  - ✓ Two-hour faculty meeting break



This shows an exam_schedule.csv file containing exam information with the following columns:

- Course (e.g., SS111, CS211)
- Day (Monday through Friday)
- Time (ranging from 9:00 to 16:00)
- Room (e.g., C309, C310)
- Teacher (teacher names)
- Duration (all exams are 1 hour)

The schedule contains various courses across different departments (CS, SS, MT, AI, EE, CY, DS) spread throughout the week.

Performance and Results

The algorithm maintains progress information, showing:

- Evolution of fitness scores across generations
- Top 3 schedules in each generation
- Time taken per generation
- Final schedule with constraint satisfaction details

Constraints Satisfaction

The implementation focuses on satisfying all hard constraints first. Once these are met, it attempts to optimize for soft constraints. The weight of penalties for hard constraint violations is much higher than rewards for soft constraint satisfaction, ensuring that valid solutions are prioritized.

- ## CONCLUSION

The genetic algorithm implementation successfully generates exam schedules that satisfy all hard constraints while optimizing for soft constraints. The algorithm's modular design allows for easy modification and extension to include additional constraints if needed.

The combination of appropriate chromosome representation, fitness function design, and genetic operators ensures efficient exploration of the solution space and convergence toward optimal or near-optimal solutions.