I212478 Muhammad Tayyab Sohail

Artificial Intelligence Project Report

BC CS Sec B

Introduction

Timetable scheduling is a critical task in educational institutions, ensuring efficient allocation of resources while minimizing conflicts among various factors like sections, professors, and rooms. This project addresses the complex nature of timetable creation for university semesters, adhering to both hard and soft constraints. By employing a genetic algorithm approach, it aims to optimize timetable configurations, representing courses, instructors, and venues in a binary-encoded format. The algorithm iteratively evolves timetable solutions, prioritizing constraints fulfillment and minimizing conflicts, ultimately offering an efficient scheduling solution for academic semesters.

Code Explanation

- **generate_timeslot(day):** This function generates a random timeslot for a given day.

- **create_chromosome(subjects, teachers, sections, rooms, timings_class, timings_lab, days):** This function creates a chromosome representing a timetable configuration. It randomly assigns courses, instructors, rooms, and timeslots while adhering to specified constraints.

- **display_timetable_with_english_names(df):** This function converts the binary-encoded timetable dataframe into a human-readable format by mapping binary values to their corresponding English names (e.g., subjects, professors, timeslots, rooms).

- **display_timetable_with_names_and_binary(df):** This function prints the timetable dataframe with both English names and their corresponding binary values.

- **fitness_function(df):** This function calculates the fitness of a timetable configuration based on various constraints such as clashes between classes, room capacity, instructor availability, and scheduling preferences.

- **generate_and_evaluate_chromosomes(num_chromosomes):** This function generates a specified number of timetable chromosomes, evaluates their fitness, and returns a list of tuples containing the chromosome data and their corresponding fitness scores.

- **select_parents(population):** This function selects the top two individuals (chromosomes) from the population based on their fitness scores to serve as parents for the crossover operation.

- **crossover(parent1, parent2):** This function performs crossover between two parent chromosomes to produce offspring. It swaps timetable information between parents to generate new timetable configurations.

- **mutation(child):** This function introduces random changes to a child chromosome to explore new solution spaces. It shuffles timetable entries within each day to generate diverse offspring.

- **evolve(selected_parents):** This function evolves the population by performing crossover and mutation operations on selected parent chromosomes. It returns two offspring chromosomes.

- **replace_worst(population, child1, child2):** This function replaces the two least fit individuals in the population with the two offspring chromosomes.

- **find_best_chromosome(population):** This function identifies the best chromosome (timetable configuration) from the population based on its fitness score and returns it along with the corresponding fitness value.