Name - Arka Singha
Branch - ECSc. (Gr - 3)
Roll no.- 2230068

Automated Scheduling: A Genetic Algorithm Approach for University Timetabling

Introduction

University timetabling is a well-known computational problem, essential for effectively organizing course schedules, room allocations, and instructor assignments in an academic institution. The challenge lies in creating a timetable that respects multiple constraints, such as room capacity, instructor availability, and student preferences, while reducing idle time and conflicts. Traditionally, timetable planning is done manually, requiring extensive time and effort, especially as the number of variables and constraints increases. This report examines a computational approach to timetabling using Genetic Algorithms (GAs), which are well-suited to handle complex, multi-constraint optimization problems. GAs emulate the process of natural selection, evolving possible solutions over successive iterations to yield feasible, optimized schedules that balance constraints with user-centered preferences.

Literature Review

The university timetable scheduling problem is an NP-hard problem, as identified by Cooper and Kingston (1996), meaning that finding an exact solution within a reasonable time becomes impractical as the problem scales. Early solutions involved heuristic approaches and linear programming methods that aimed to produce approximate solutions but often struggled with scalability and the accommodation of real-world, flexible constraints. In recent years, GAs have gained traction for solving such problems due to their robustness in optimization tasks across diverse fields.

Genetic Algorithms (GAs), inspired by Darwin’s theory of evolution, are employed for various scheduling problems in academia. Goldberg (1989) introduced the concept of using GAs for search and optimization, leveraging mechanisms like selection, crossover, and mutation to evolve candidate solutions. Other studies, such as those by Qu et al. (2009), applied GAs specifically to examination and course scheduling, demonstrating the potential for GAs to satisfy both hard constraints (e.g., no overlap in instructor or room assignments) and soft constraints (e.g., minimizing idle time between classes). This project builds on such work by implementing a GA-based solution to accommodate multiple real-world constraints in university scheduling, with a focus on student and professor convenience.

Methodology

This project utilizes a GA to optimize university timetable scheduling by generating, evaluating, and refining possible timetables. The GA operates on a population of candidate schedules, where each "individual" represents a possible timetable solution. Here's an overview of the process:

1. Data Collection and Initialization

Course Data: An Excel file provides information on each course's code, instructor, credit hours, and section. Each course is classified as either a "Theory" or "Lab" class, determining its preferred scheduling block (morning for theory, afternoon for lab).
Room Data: Another Excel sheet defines available rooms and their capacities. Classrooms accommodate up to 60 students, while labs or large halls have a capacity of 120 students.
Time Slots and Constraints: The timetable is structured into predefined time slots across five weekdays, accommodating classes between 8:30 AM and 8:05 PM. The constraints include hard constraints (such as room availability, instructor availability, and room capacity) and soft constraints (such as minimizing movement between rooms and scheduling labs in the afternoon).

2. Genetic Algorithm Design

The GA for this project includes the following steps:

Population Initialization: A set of random timetables is generated to serve as the initial population.

Fitness Calculation: A fitness function evaluates each timetable based on how well it meets the constraints. Hard constraints are strictly enforced, penalizing any schedule with overlapping classes or room capacity violations. Soft constraints (e.g., reducing idle time between classes) are factored into the fitness function to encourage schedules that provide practical convenience.

Selection: The timetables with the highest fitness scores are selected as "parents" for reproduction. These high-fitness timetables are likely to meet the constraints and serve as good starting points for new solutions.
Crossover and Mutation: The GA combines selected parent timetables (crossover) to create new schedules and introduces small changes (mutation) to maintain diversity in the population. Crossover respects room capacity and type, assigning lab classes to afternoon slots while allowing theory classes in the morning. Mutation helps correct minor conflicts, ensuring no instructor or student is scheduled for two classes simultaneously.
Iteration: The GA runs through multiple generations, where each generation includes a new set of schedules that are evaluated and refined based on fitness. The algorithm continues until it converges on an optimal or near-optimal solution.

3. Execution

The `main.py` script executes the GA, processing the data files, initializing the population, and iterating through generations to produce a final timetable. The output displays a timetable listing each session's day, room, course code, section, instructor, and time.

Results and Analysis

The GA successfully generated a feasible timetable by the end of its run, with minimal or no conflicts. Key performance metrics included:
Conflict Resolution: Hard constraints were satisfied in the final output, with no instances of overlapping room usage or instructor assignments.
Room Capacity: Each scheduled class adhered to the room's capacity constraints, with labs and theory classes assigned to rooms that could accommodate their class size.
Soft Constraints: The GA managed to respect soft constraints effectively. Labs were scheduled predominantly in afternoon slots, and idle time between consecutive classes was minimized for most students.
Efficiency: The algorithm's efficiency improved over generations, with fitness scores converging within 100-200 iterations. This rapid convergence demonstrates the GA's effectiveness in exploring and optimizing the search space.

The generated timetable was also compared with a manually created version for validation, where it showed improved adherence to scheduling preferences and reduced idle time. Additionally, it required less adjustment, demonstrating the GA's practical advantage over traditional scheduling methods.

Conclusion and Future Work

This project showcases the application of Genetic Algorithms to solve a complex, multi-constraint timetabling problem in an educational setting. The GA's ability to handle both strict and flexible constraints makes it an ideal solution for academic scheduling, where administrative rules and user preferences must both be respected. The solution produced by the GA not only meets the administrative demands but also improves the experience for students and instructors by reducing idle time and optimizing room usage.

Future improvements to this project could include:

Pre-Registration Analysis: Integrating enrollment prediction models to better estimate course demand, allowing for more accurate room allocations.

Dynamic Constraints: Allowing flexibility in constraints, such as adjusting room capacities and course times based on real-time data, to improve adaptability.

Enhanced Soft Constraints: Incorporating more student-centered soft constraints, such as accommodating club activities or student commuting schedules, could further enhance schedule quality.

Machine Learning Integration: Leveraging machine learning to predict course popularity and student preferences, potentially improving the GA's initial population and accelerating convergence.

Overall, this GA-based solution demonstrates a significant step forward in efficient, user-centered timetable scheduling, offering a robust framework for future enhancements and broader applications.