Exam Scheduling Systems

Team:

Mostafa Walid A20000908

Maryam Elgohary A20000882

Shahd Ahmed A20000476

Asser Mohamed 2221022487

Karim Wael 221100389

Sondos Ahmed 221101426

Outlines:

- Project Overview
- Detailed Problem
- Datasets
- Technical Implementation (Key Details)
- Model
- GUI
- Results

Project Overview:

This project aims to automate and optimize the creation of exam schedules for educational institutions. The core challenge lies in managing numerous courses, diverse student enrollment patterns, limited resources (such as classrooms and invigilators), and various constraints (both hard and soft). Hard constraints include ensuring no student has overlapping exams and that each exam has an allocated room and invigilator. Soft constraints might involve accommodating student or instructor preferences, such as avoiding early morning exams or minimizing consecutive exam days for students.

The project leverages a genetic algorithm (GA), a biologically inspired optimization technique, to explore a vast solution space efficiently. Genetic algorithms mimic the process of natural selection by using operations such as selection, crossover, and mutation to evolve a population of potential solutions over successive generations. By iteratively refining these solutions, the GA seeks to minimize conflicts and optimize the exam schedule according to the predefined constraints and preferences.

This approach allows for a high degree of flexibility and adaptability, making it well-suited to the complex and dynamic nature of scheduling problems. The genetic algorithm's ability to balance multiple objectives and constraints ensures that the final schedule is not only feasible but also optimized for the best possible outcomes. This includes minimizing the number of conflicts, efficiently utilizing available resources, and meeting the needs and preferences of students and staff.

Additionally, the use of a genetic algorithm can significantly reduce the time and effort required to generate exam schedules compared to traditional manual methods. This automation can lead to more efficient administrative processes within educational institutions, freeing up valuable time for staff to focus on other critical tasks. Overall, this project aims to enhance the scheduling process, improve resource management, and provide a more balanced and fair exam timetable for all stakeholders involved.

Detailed Problem:

The exam scheduling problem is a classic example of a combinatorial optimization problem, where the number of possible schedules grows exponentially with the number of courses and constraints, making it computationally intractable to find the absolute best solution using brute-force methods. This problem encompasses both hard and soft constraints.

Hard constraints, which must be satisfied, include ensuring no overlapping exams for students, allocating a unique exam slot for each course, scheduling exams within predetermined time windows, ensuring invigilators do not supervise multiple exams simultaneously, making sure the number of students in an exam does not exceed classroom capacity, accommodating exams of varying durations, respecting unavailable exam days (such as weekends or holidays), considering part-time instructors' specific availability, and preventing scheduling conflicting courses with significant student overlap in the same time slot.

Soft constraints, which are preferable to satisfy, involve minimizing instances where students have back-to-back exams, scheduling certain courses before others due to prerequisites or academic reasons, distributing invigilation duties fairly among instructors to avoid overloading anyone, considering room preferences for courses that require specific facilities (such as a projector for presentations), gathering and accommodating students' preferred exam times if feasible, refining the constraint to avoid consecutive exams by prioritizing avoiding consecutive exams in the same subject area or for the same student on the same day, and considering preferences for exams in the morning or afternoon for both students and faculty.

Methodology:

Data Input and Preprocessing:

CSV Files: The project reads data from CSV files containing course details, instructor information (including full-time/part-time status), student lists, course registrations, and classroom capacities.

Data Structures: The input data is organized into classes and collections to facilitate manipulation and access during the GA process.

Genetic Algorithm:

Chromosome Representation: Each chromosome is a potential solution (exam schedule). It's a list of genes, where each gene represents a single exam and contains information about its time slot, assigned rooms, and invigilators.

Initialization: The initial population is generated randomly. Each chromosome is created by assigning random values to the genes (time, room, invigilator) for each exam.

Fitness Function: The fitness function quantifies the quality of a chromosome by evaluating how well it satisfies the constraints. The calculation involves penalties for violating hard constraints and rewards for adhering to soft constraints.

Selection: The selection mechanism (e.g., roulette wheel selection) biases the selection of parent chromosomes based on their fitness. Fitter chromosomes have a higher probability of being chosen.

Crossover: Crossover creates new offspring chromosomes by combining parts of two parent chromosomes. This can be done by various techniques like single-point, two-point, or uniform crossover.

Mutation: Mutation randomly alters genes in offspring chromosomes. This introduces diversity into the population and helps prevent premature convergence to suboptimal solutions.

Evolutionary Process: The algorithm iteratively applies selection, crossover, and mutation over multiple generations. This process simulates natural selection, where fitter solutions are more likely to survive and reproduce.

Stopping Criterion:

The algorithm stops when a solution with an acceptable fitness level (ideally, fulfilling all constraints) is found or when a maximum number of generations is reached.

Datasets:

The datasets provided appear to contain information relevant to scheduling exams, including details about courses, teachers, students, registrations, and classrooms. To provide a comprehensive overview of each dataset without delving into specific numbers, we will examine the columns and their meanings.

Given that the courses.csv file appears to have been read in incorrectly, with course codes as column names and course names as values in the first row, we will use the version of this dataset for this analysis.

The datasets provide the following information, essential for exam scheduling:

Courses Data: Contains a list of courses, each identified by a unique code and a descriptive name.

Teachers Data: Lists the instructors (referred to as "Doctors") along with the courses they teach and their employment status (full-time or part-time).

Students Data: Provides a list of student names.

Registrations Data: Indicates which students are enrolled in which courses, establishing the relationship between students and the exams they need to take.

Classrooms Data: Details the available classrooms, including a unique identifier for each room and its maximum capacity.

The exam schedule is spread across two weeks and includes 26 exams in total. Exams are scheduled from Saturday to Thursday, with no exams on Fridays. Each day has one or two exam slots: 12:00 AM and/or 3:00 PM.

Each exam entry in the schedule specifies the course code, course name, start time, room number, and the two assigned invigilators (teachers). The schedule also indicates the week number for each set of exams.

The classrooms used for the exams are: I001, I003, Q121, J204, Q100, J203, J300, J201, N224, I002, N147, N247, NG24, J302, and Q123.

Some teachers, like Mohamed Abdelaziz, Manar Elshazly, Mohamed Ghetas, and Samy Ghoniem, are assigned to invigilate multiple exams.

The schedule seems to be well-organized, with no apparent conflicts in terms of room or teacher availability. However, without additional information on student enrollment and course prerequisites, it's difficult to assess whether the schedule fully optimizes for student preferences or potential conflicts.

Technical Implementation (Key Details):

Programming Language: Python is used for its versatility and rich ecosystem of libraries for data manipulation, optimization, and visualization.

Custom Classes: Classes are defined to represent courses, students, registrations, exams, and classrooms. These classes encapsulate relevant data and methods for the algorithm.

Data Structures: Lists, dictionaries, and tuples are used to store and organize information efficiently. Randomization: The random module is employed for generating initial populations and introducing randomness during mutation.

Genetic Operators: Functions implement selection, crossover, and mutation operations to drive the evolutionary process.

Model:

The model used to generate the exam schedule is a genetic algorithm (GA). A GA is a search heuristic that is inspired by Charles Darwin's theory of natural evolution. This algorithm reflects the process of natural selection where the fittest individuals are selected for reproduction in order to produce offspring of the next generation.

In this context, the GA is designed to find an optimal or near-optimal solution to the exam scheduling problem. The model takes into account various constraints, including the availability of teachers and classrooms, the number of students enrolled in each course, and the duration of each exam.

The GA starts by generating a population of potential solutions, where each solution represents a possible exam schedule. Each solution is evaluated based on how well it satisfies the given constraints. The solutions that best meet the constraints are then selected to "reproduce" by combining their characteristics to create new solutions. This process is repeated over many generations, with the hope that the solutions will improve over time.

The final output of the GA is an exam schedule that best satisfies the given constraints. This schedule is represented in a human-readable format, detailing the time, room, and invigilators for each exam.

The specific implementation of the GA in this project involves several key components:

Representation: Each solution (exam schedule) is represented as a chromosome, which is a string of numbers. Each number in the chromosome represents a particular aspect of the schedule, such as the time slot for an exam, the room where it will be held, or the teachers who will invigilate it.

Fitness Function: The fitness function evaluates how well a solution satisfies the constraints. It assigns a higher fitness score to solutions that better meet the constraints.

Selection: The selection process chooses solutions from the current population to be parents of the next generation. The probability of a solution being selected is proportional to its fitness score.

Crossover: Crossover combines the genetic material of two parent solutions to create new offspring solutions.

Mutation: Mutation introduces random changes into the offspring solutions. This helps to maintain diversity in the population and prevent the algorithm from getting stuck in a local optimum.

GUI:

Input Fields:

Courses File: Allows the user to select the CSV file containing course information.

Teachers File: Allows the user to select the CSV file containing teacher/instructor details.

Students File: Allows the user to select the CSV file containing student names.

Registrations File: Allows the user to select the CSV file containing student-course registrations.

Classrooms File: Allows the user to select the CSV file containing classroom information.

Each input field has a default value that suggests the expected filename, making it easier for users to locate the correct files.

Browse Buttons:

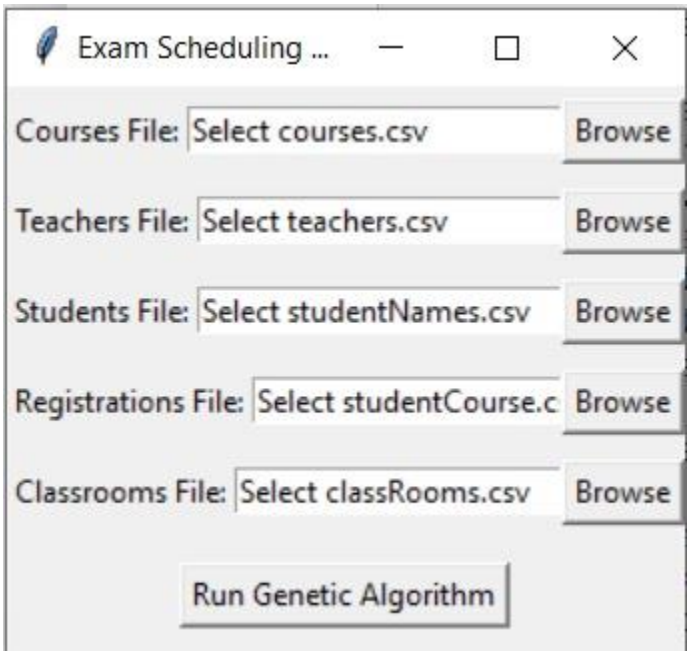
Each input field is accompanied by a "Browse" button.

Clicking this button opens a file dialog, allowing the user to navigate their file system and select the appropriate CSV file.

Run Genetic Algorithm Button:

This is the main action button of the GUI.

Clicking it triggers the application to read the data from the selected files, apply the genetic algorithm, and generate an optimized exam schedule



Results:

The exam schedule is spread across two weeks and includes 26 exams in total. Exams are scheduled from Saturday to Thursday, with no exams on Fridays. Each day has one or two exam slots: 12:00 AM and/or 3:00 PM.

Each exam entry in the schedule specifies the course code, course name, start time, room number, and the two assigned invigilators (teachers). The schedule also indicates the week number for each set of exams.

The classrooms used for the exams are: I001, I003, Q121, J204, Q100, J203, J300, J201, N224, I002, N147, N247, NG24, J302, and Q123.

Some teachers, like Mohamed Abdelaziz, Manar Elshazly, Mohamed Ghetas, and Samy Ghoneim, are assigned to invigilate multiple exams.

Exam Time table are also extracted to a file

Week 2

Saturday

CSE015	Object Oriented Programming	3:00 PM	Room 1: NG24	Mahmoud Safwat , Mohamed Amir

Sunday

CSE383	Computer Vision	12:00 AM	Room 1: J302	Samy Ghoniem , Radwa
MAT123	Mechanics	3:00 PM	Room 1: J204	Ahmed Heikal , Ibrahim Attia
CSE273	Parallel and Distributed Systems	3:00 PM	Room 1: Q100	Mohamed Ghetas , Manar Elshazly

Tuesday

AIE111	Artificial intelligence	12:00 AM	Room 1: I001	Mohamed Abdelaziz , Ahmed Nader
AIE121	Machine Learning	3:00 PM	Room 1: N147	Shaker Elsappagh , Manar Elshazly
CSE272	Embedded Systems	3:00 PM	Room 1: J201	Ashraf Elshrief , Kariem

Wednesday

CSE221	Database Systems	12:00 AM	Room 1: J204	Shaker Elsappagh , Samy Ghoniem
AIE343	Text Mining	3:00 PM	Room 1: J301	Manar Elshazly , Radwa
AIE323	Data Mining	3:00 PM	Room 1: N247	Mohamed Abdelaziz , Sameh Basha
PHY232	Modern physics	3:00 PM	Room 1: J201	Mohamed Emam Ismail , Mahmoud Hamed

Thursday

MAT121	Dynamics	12:00 AM	Room 1: N247	Ahmed Hazem , Ahmed Nader
CSE243	Secure Programming	12:00 AM	Room 1: J302	Mohamed Abdelaziz , Berhan