Project Report: Course Scheduling with Genetic Algorithm

1. Problem Definition

- **Objective**: Develop an optimal course schedule that satisfies prerequisites, student time preferences, exam schedules, and specialization requirements.
- **Constraints**:
    - Avoid overlapping courses to prevent scheduling conflicts.
    - Ensure prerequisite courses are scheduled before dependent courses.
    - Respect student preferences for specific days and times, where possible.

2. Input Data Preparation

- **Dataset**: used a subset of a CSV dataset “courses scheduling” containing 100 course entries, each with:
    - **Course ID, Name, Type** (mandatory/elective), **Prerequisites**, **Preferred Timing**, and **Student Preferences** (preferred days and times).
- **Purpose**: This dataset serves as the foundation for initializing the genetic algorithm’s population.

3. Chromosome Design

- **Chromosome Representation**: Each chromosome (solution) represents a potential schedule for the 100 courses.
    - **Genes**: Each gene corresponds to a course with attributes such as:
        - **Day and Time Slot**: Assigned day and time.
        - **Prerequisite Status**: Information on prerequisite courses.
        - **Student Preferences**: Days and times matching student preferences.

4. Fitness Function Development

- **Fitness Criteria**:
    - **Minimize Scheduling Conflicts**: Avoid overlapping course times.
    - **Prioritize High-Preference Courses and Times**: Match courses to students' preferred days and times as closely as possible.
    - **Penalize Unmet Prerequisites**: Ensure prerequisite courses are scheduled before dependent courses.
- **Implementation**: The fitness function scores each chromosome based on how well it aligns with the criteria above.

5. Genetic Algorithm Components

- **Initialization**: Generated a diverse initial population by assigning random schedules to the 100 courses.
- **Selection**: Employed roulette wheel selection to probabilistically select chromosomes based on fitness.
- **Crossover**: Used one-point crossover to mix schedules, creating new schedules from parent chromosomes.
- **Mutation**: Randomly modified time slots within some offspring to introduce variability and maintain diversity.
- **Replacement**: Adopted a generational replacement strategy to form new populations over generations.

6. Algorithm Execution

- **Stopping Criterion**: Ran the genetic algorithm for a fixed number of generations (20) due to limited dataset size.
- **Process**:
    - Calculated fitness for each chromosome per generation.
    - Applied selection, crossover, and mutation to produce new generations.
    - Tracked and stored the best solution across all generations.

7. Result Evaluation

- ➔ **Generation-by-Generation Tracking**:
  - ◆ Each generation reported:
    - **Best Fitness Score**: The highest score in each generation, indicating the quality of the best schedule.
    - **Best Solution**: The specific day and time assignment for each course in the top-scoring schedule.
  - ◆ This tracking allowed us to observe fitness improvements over generations, demonstrating the algorithm's convergence.
- ➔ **Final Best Schedule**:
  - ◆ After completing all generations, the algorithm output:
    - **Best Fitness Score (Overall)**: The highest fitness achieved, representing the most optimized schedule.
    - **Best Solution (Final Schedule)**: The specific assignments of days and times for each course, forming the final recommended schedule.

8. Interpreting the Schedule

- ➔ **Final Best Schedule**:
  - ◆ **Course Assignments**: Each course is assigned a day and time slot, optimized to maximize preference satisfaction and meet prerequisites.
  - ◆ **Constraints Adherence**:
    - **Prerequisites**: Ensures courses with prerequisites are scheduled after the prerequisite courses (where feasible).
    - **Student Preferences**: Prioritizes courses to be scheduled on students' preferred days and times.

9. Insights and Improvements

- ➔ **Generation-by-Generation Fitness**: The highest fitness score across generations improved until stabilizing, indicating a converged solution.
- ➔ **Remaining Scheduling Conflicts**: The final best schedule showed unmet prerequisites in two cases, suggesting further tuning of the fitness function (e.g., increasing penalties for unmet prerequisites) could be beneficial.
- ➔ **Fitness Function Adjustments**: Additional constraints, such as balancing course distribution more evenly across days, could enhance results. Adjustments to the mutation or crossover rates might also yield higher diversity.

10. Schedule Analysis

➔ **Course Distribution Analysis**:

Day Distribution:

- Courses are spread across the week as follows:
  - **Wednesday**: 25 courses
  - **Monday**: 20 courses
  - **Friday**: 20 courses
  - **Tuesday**: 19 courses
  - **Thursday**: 16 courses
- This balanced allocation indicates no heavy concentration on a single day.

Time Slot Distribution:

- **Evening**: 37 courses
- **Morning**: 36 courses
- **Afternoon**: 27 courses
- The distribution favors morning and evening slots, aligning with the high preference for these times among students.

➔ **Prerequisite Fulfillment**:

Unmet Prerequisites:

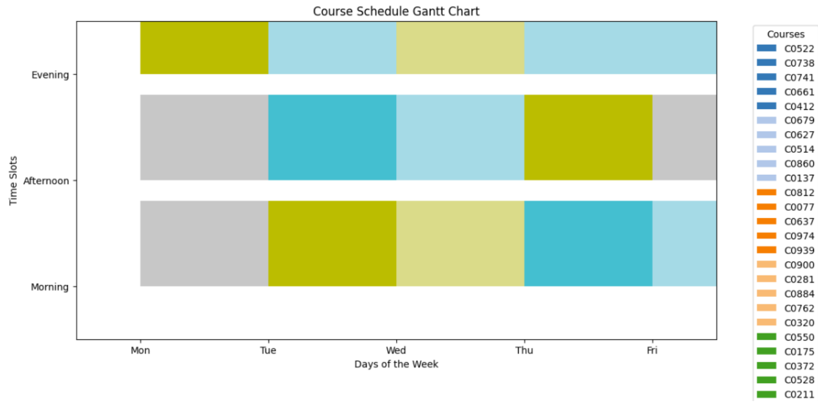
- The algorithm met most prerequisites, but two unmet cases were noted:
  - **Course C0500** requires **Course C0437**.
  - **Course C0902** requires **Course C0313**.
- This suggests that additional weighting on prerequisite constraints in the fitness function may improve results.

➔ **Preferred Days and Times**:

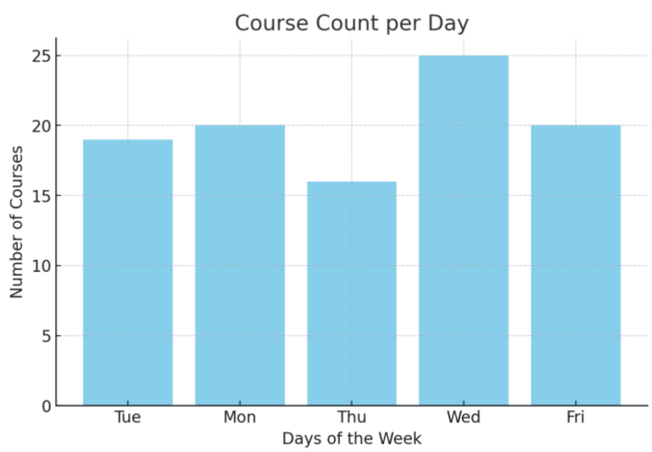
Preferred Matches: All 100 courses matched their preferred days and times, showing that the algorithm effectively prioritized student preferences.

➔ **Visualization**:

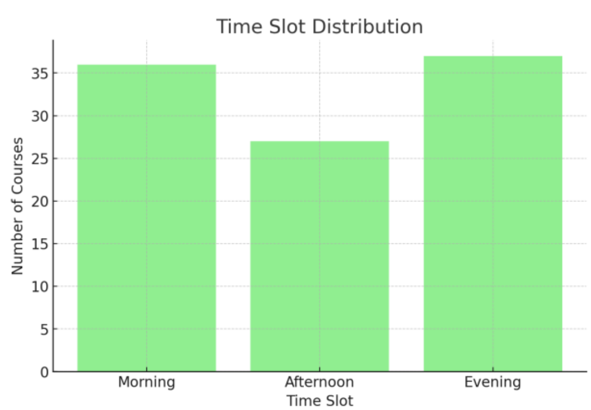
A **Gantt chart** was used to visually represent the final best schedule, enabling easy identification of distribution patterns and potential imbalances.



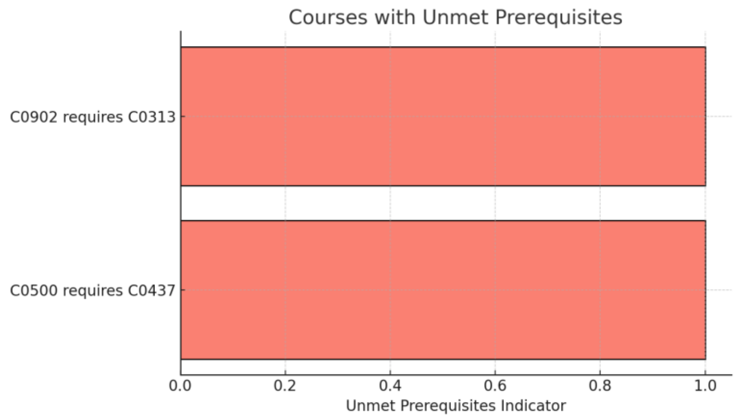
- Course Count per Day: A bar chart showing the number of courses scheduled each day, which will highlight day-by-day distribution.



- **Time Slot Distribution**: A pie chart or bar chart showing the distribution of courses across different time slots (Morning, Afternoon, Evening).



- Unmet Prerequisites: A simple visualization to indicate the courses with unmet prerequisites.



Summary

The final schedule achieved a balanced distribution across days and times, fully satisfying student preferences for scheduling. While most prerequisites were met, unmet cases indicate areas for refinement in the algorithm. Enhancing the fitness function or adjusting algorithm parameters like mutation and crossover rates could further optimize the schedule in future iterations.