Teacher Preference-Driven Schedule Optimization

CS-371 Artificial Intelligence

Session: 2021-2025

Project Supervisor

Samyan Qayyum Wahla

Group Members

Muhammad Subhan 2021-CS-35

Department of Computer Science
University of Engineering and Technology, Lahore
Pakistan

‘

Contents

1 Introduction

1.1 Background

University class scheduling is a mindful task for educational institutions which involves the allocation of courses to specific times, instructors, and rooms. It plays an important role in ensuring efficient resource utilization and providing students with a smooth academic experience. This problem involves many constraints, preferences, and parameters catagorizing this problem to NP-Hard Problem. So, finding an optimal solution in a reasonable amount of time is difficult, making it a suitable application of advanced scheduling techniques and optimization algorithms.

1.2 Problem Statement

The problem is the manual creation of class schedules in educational institutions is a complex task due to the many constraints and preferences involved. This process is often time-consuming, error-prone, and can lead to suboptimal resource allocation and scheduling conflicts.

1.3 Objective

The primary objective of this project is to create an automated class scheduling system that generates optimal schedules without conflicts. This system aims to provide a solution that satisfies both the teacher preferences and the institution resource constraints.

1.4 Quality Education (SDG 4)

By optimizing educational schedules and ensuring that students have access to well-organized and efficient class timetables, we add a little contribution to the advancement of SDG 4, which aims to ensure inclusive and equitable quality education for all.

2 Literature Review

2.1 A Novel Optimization Approach for Educational Class Scheduling with considering the Students and Teachers' Preferences

Explanation

The research study have proposed a mathematical model for the solution of university class scheduling. The model uses linear programming approach which keeps in mind soft constrainsts, hard constraints, preference and various parameters. This model uses the ideology to minimizing deviation from ideal scheduling plan incooperating all the constraints and parameters.

The implementation involves the usage of genetic algorith as a metaheuristic approach. It finds the optimal solution through a process of evolution and selection. The chromosomes are representing the scheduling of courses over different time slots in a days. In each iteration, crossover and mutation evolves the solution from previous iteration and selects the one for next iteration that fits the best decided by selection operator.

The model divides the objective function into two parts, one for professors and one for students, and aims to maximize the number of classes within a specific time period. It defines various decision variables, parameters, and sets to formulate the scheduling problem.

In a nutshell, this mathematical model provides a method for optimizing university class scheduling. It considers the all constraints while striving to keep minimum deviations from the ideal plan. This approach uses a genetic algorithm to find the most efficient solution.

Limitations

- This approach does not consider the need for breaks in the schedule.
- This approach does not consider the availability of rooms.
- No preference of professors and teachers for a time slot is introduced as soft constraints.

Research Paper

Chen, Y., Bayanati, M., Ebrahimi, M., & Khalijian, S.(2022).*A Novel Optimization Approach for Educational Class Scheduling with considering the Students and Teachers' Preferences.* Discrete Dynamics in Nature and Society, 2022. [1]

2.2 A Heuristic Approach to Course Scheduling Problem

Explanation

The research study have proposed a sorting and searching based algorithm.. It begins by sorting course sections based on their start times and categorizes them by weekdays, creating an efficient sequence for teacher-course assignments. The algorithm also considers teacher preferences, ensuring that courses are assigned to teachers who have indicated a preference for them. Teachers are assigned courses based on their preferences, provided they have available time slots and their current class load does not exceed predefined limits. This approach aims to optimize teacher-course assignments, resulting in a well-structured timetable that minimizes conflicts and overlaps.

```
CourseScheduler()
1. Define maxLoad
2. For each teacher t:
3.   Set load[t] = 0
4. Sort sectionList by start time
5. Classify the sections by weekdays
6. For each day d:
7.   For each section s:
8.     For each teacher t:
9.       If preferred course and free time slot and load[t] < maxLoad:
10.         Assign s to t
11.         If s is in other days:
12.           Remove s
13.           Go to step 7
14.       If s is unassigned:
15.         For each teacher t:
16.           If free time slot and load[t] < maxLoad:
17.             Assign s to t
18.             If s is in other days:
19.               Remove s
20.               Go to step 7
21.           Mark s as unassigned
```

```
22. Return scheduled and unassigned sections
```

Limitations

- Method does not tackle the case when multiple teachers prefer the same course.
- Less Automated as unassigned courses may require manual assignment.
- The algorithm's scalability in larger educational institutions with numerous courses and teachers is a big concern.
- The algorithm assumes all classes have the same duration, which is not always the case.

Research Paper

Karmaker, Debjyoti, Mir Riyanul Islam, H. Rahman, A. Bhowmik, and M. N. Imteaj. *"A heuristic approach to course scheduling problem. "* American International University-Bangladesh (2015).[2]

2.3 Intelligent Timetable Scheduler: Comparison of Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms

Explanation

This study compares Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms

When solving exam timetable scheduling using graph coloring algorithm, subjects act as vertices. Edges are drawn depending on graph type [15]. One type is a conflicting graph. There should be common modules that cannot have the same time slots that type of conflicts represented in a conflicting graph. In this algorithm initially found the maximum degree vertex selected. Found out all its triples [16]. The first and third vertex cannot be adjacent to each other. Then first and third vertex can have the same color. Likewise coloring the whole graph Algorithm for generating best graph,

Iterated Local search algorithm starts with an initial solution and iterates several times that create a candidate solution based on the existing one. If the candidate solution found to be better than the existing one, the existing solution will be updated with the candidate. The search procedure continues until no solution improvement has occurred during several iterations.

The findings primarily suggest that the Iterated Local Search (ILS) Algorithm performed better than the other algorithms in terms of accuracy and efficiency in solving the exam timetable scheduling problem. Additionally, the study highlighted the Graph Coloring Algorithm suitable in identifying scheduling conflicts.

The ranking from fastest to slowest is: Graph Coloring Algorithm, Genetic Algorithm, Heuristic Algorithm, and Iterated Local Search Algorithm (ILS).

Limitations

- The study conducted experiments using SLIIT (Sri Lanka Institute of Information Technology) as a case study.
- The study conducted does not consider complex constraints and preferences.

Research Paper

Tiny Wijerathna Ekanayake, Pavani Subasinghe, Shawn Ragel, Anjalie Gamage, Suchini Attanayaka.*"Intelligent Timetable Scheduler: Comparison of Genetic, Graph Coloring, Heuristic and Iterated Local Search Algorithms ".*[3]

2.4 A Decision Support System for a Three-Stage University Course Scheduler

Explanation

This research paper proposes Decision Support System (DSS) for solving university course scheduling problem. The problem involves the assignment of courses to faculty members, scheduling courses into time slots, and allocating rooms to those courses. The DSS is a Three-Stage Scheduling Model using goal programming that efficiently addresses these challenges.

In the first stage, the DSS assigns courses to faculty members while considering various rules and departmental regulations.The objective is to optimize course assignments and ensure faculty satisfaction. The second stage of the model assigns time slots to the courses assigned in the first stage. It takes into account various factors such as morning vs. afternoon sessions, odd vs. even days, and faculty preferences for class times. The third and final stage allocates rooms to the courses and focuses on ensuring that each course is located in a room of the right size, close to the department offering the course.

The DSS was implemented using a combination of VB.Net for the user interface, MS SQL Server for data management, and Large Scale LP/Quadratic Engine APIs for solving the optimization model.

Limitations

- DSS can include the computational complexity of solving large scheduling problems.For this purpose large Scale LP/Quadratic Engine APIs were used.

Research Paper

Karmaker, Debjyoti, Mir Riyanul Islam, H. Rahman, A. Bhowmik, and M. N. Imteaj. *"A Decision Support System for a Three-Stage University Course Scheduler with an Application to College of Business Administration, Kuwait University "*[4]

Table 1: A summary of the Three-Stage Scheduling Model

Stage	Goals	Goal priority	Goal Equation	Stage objective function	Hard Constraints	Hard Constraints Equation
I	**1.** Each faculty member i should take an exact maximum course load, L_i	P_1	A2	*Equation A1:* Minimize all the negative and positive deviation of goals 1 and 2 plus all the negative deviations of goals 3,4, and 5	Each faculty member i is allowed to take a maximum of two sections for the same course j	A7
	2. The number of course sections N_j for course j should all be covered by faculty members.	P_1	A3			
	3. Each faculty member i should take at least one College Level Course (CLC) course section. This goal is a department level requirement	P_2	A4			
	4. Each faculty member i should take at least one Major Level Course (MLC) section. This goal is a department level requirement	P_3	A5			
	5. Maximize the total preference for each faculty member i that has a course load L_i .	P_4	A6			
II	**1.** Total number of courses assigned in a specific time slot *cannot* exceed the number of rooms available for that time slot.	P_1	A9	*Equation A8:* Minimize all the positive deviations of goals 1, 2 and 3 plus the all the negative deviations of goals 5,6,7, and 8.	**1.** Sum of sections taught for every faculty in every specific time slot must be at most equal to 1	A17
	2. Total number of similar CLC assigned during a specific time slot in morning-time cannot exceed 2 sections for the same course.	P_2	A10			
	3. Total number of similar CLC assigned during a specific time slot in afternoon-time cannot exceed 1 section for the same course **Note:** Goals 2 and 3 eliminate timing conflict of courses that can be taken at the same time for similar CLC.	P_3	A11			
	4. The MLC should be 4 times more condensed during the morning-time than they are during the afternoon-time, where 4 is just any number that the department picks. **Note:** This reduces the gaps between MLC.	P_4	A12		**2.** Sum of MLC offered during a specific time slot during same day must equal at most 1.	A18
	5. 60% of courses should preferably be offered during the odd days and 40% during the even days	P_5	A13			
	6. 70% of courses should preferably be offered during the morning-time and 30% during the afternoon-time **Note:** Goals 4, 5, 6 are like a guideline specific to the needs of the institution, i.e. at Kuwait University.	P_6	A14 and A15		**3.** Sum of time slots for each section for every faculty, every course, and every section must equal 1.	A19
	7. Maximizes the faculty preferences on their class times.	P_7	A16			
III	**1.** Assigned each previously scheduled course from Stage II to a room of the right size as close as possible to the department that is offering the course.	P_1	A21	*Equation 20*: Minimize the negative deviation of goal 1	**1.** Each section of a course that has been assigned a specific faculty and time should be located in one room only.	A22
					2. Each room is assigned to at most one faculty in a specific time period.	A23

Algorithm	Running Time (%)	Strengths	Weaknesses
Genetic Algorithm	25	Iteratively improves solutions.	Some subjects still have conflicts.
Graph Coloring Algorithm	11	Effectively identifies clashes.	Scheduling constraints may be complex.
Heuristic Algorithm	20	Practical, rule-based approach.	May not guarantee an optimal solution.
Iterated Local Search (ILS)	44	Provides more accurate solutions.	Time-consuming for large instances.

Table 1: Comparison of Exam Timetable Scheduling Algorithms

References

[1] Novel Scheduling Approach

[2] Heuristic Scheduling Approach

[3] Intelligent Timetable Scheduler

[4] Decision Support System for University Course Scheduler

3 Methodology

3.1 Classes

3.1.1 Group

The `Group` class represents a group of students. It has class variable `groups` to store all instances of groups. Each group has a name and a size. The class provides a `find` method to search for a group by name, returning its index. The `__repr__` method is implemented to provide a string representation of a group.

3.1.2 Professor

The `Professor` class represents a professor. It has class variable `professors` to store all instances of professors. Each professor has a name. The class provides a `find` method to search for a professor by name, returning its index. The `__repr__` method is implemented to provide a string representation of a professor.

3.1.3 CourseClass

The `CourseClass` class represents a class or course. It has class variable `classes` to store all instances of classes. Each class has a code, credit hours (`cr`), lab indicator, and an optional full name. The class provides a `find` method to search for a class by code, returning its index. The `__repr__` method is implemented to provide a string representation of a class.

3.1.4 CourseAssignment

The `CourseAssignment` class represents an assignment of a course class to a professor and a group. It has attributes `course_class`, `professor`, and `group`.

3.1.5 Room

The `Room` class represents a room. It has class variable `rooms` to store all instances of rooms. Each room has a name, size, and lab indicator. The class provides a `find` method to search for a room by name, returning its index. The `__repr__` method is implemented to provide a string representation of a room.

3.1.6 Slot

The `Slot` class represents a time slot for a class. Each instance has attributes `start`, `end`, `day`, and `is_lab_slot` representing the start time, end time, day of the week, and whether it is a lab slot. The `__repr__` method provides a string representation of the time slot.

3.2 The Structure of Genetic Algorithm

The following Setps are performed:

3.3 Bit Masking

A list of Course Asignment to professor with Group is converted into binary using `bin` with padding of 0's on right using `rjust` by finding maximum bits needed using `bits_needed` for each entity. Then a random Lecture theater converted into binary is assigned with a random time slot.

Class, Group, Professor , Room and Lecture Theater represent a Gene and A Chromosome is list of genes that represent a solution to problem under discussion.

3.4 Evalute

- `use_spare_classroom(chromosomes)`: This function seem to contribute to the overall score by evaluating how well spare classrooms are utilized in the schedule.
- `faculty_member_one_class(chromosomes)`: This function likely checks whether each faculty member is assigned to only one class, contributing to the overall score accordingly.
- `classroom_size(chromosomes)`: This function may assess the size of classrooms used in the schedule, contributing to the score.
- `group_member_one_class(chromosomes)`: This function probably checks if each group member is assigned to only one class, impacting the overall score.
- `appropriate_room(chromosomes)`: This function likely evaluates whether the assigned rooms are appropriate for the classes, contributing to the score.
- `appropriate_timeslot(chromosomes)`: This function may assess whether the assigned time slots are appropriate for the classes, contributing to the overall score.

The scores from each of these criteria are accumulated in the score variable, and then the total score is normalized by dividing it by max_score before being returned. Normalizing the score to a value between 0 and 1 allows for better comparison and interpretation, especially when the criteria have different scales.

3.5 Selection

The `selection` function performs selection within the population based on the evaluation scores of the chromosomes. It sorts the population in descending order of their scores, calculated using the `evaluate` function. Then, it removes excess chromosomes beyond the desired population size (`n`).

This operation ensures that only the fittest individuals survive and are used to create the next generation. It plays a crucial role in driving the evolution of the population towards better solutions.

```
// Python code for the selection function
def selection(population, n):
    population.sort(key=evaluate, reverse=True)
    while len(population) > n:
        population.pop()
```

3.6 Crossover

The `crossover` function implements a crossover operation, a fundamental genetic algorithm operation. It selects two parent chromosomes `a` and `b` randomly from the population and a random crossover point (`cut`). It then creates a new chromosome by combining the genes before the crossover point from `a` with the genes after the crossover point from `b`.

This process mimics the genetic recombination in natural evolution and helps propagate promising genetic material. The new chromosome is added to the population, contributing to the next generation.

```
// Python code for the crossover function
def crossover(population):
    a = random.randint(0, len(population) - 1)
    b = random.randint(0, len(population) - 1)
    cut = random.randint(0, len(population[0]))
    population.append(population[a][:cut] + population[b][cut:])
```

3.7 Mutate

The `mutate` function is responsible for introducing random changes or mutations to a given chromosome. It randomly selects a position in the chromosome, represented by the index `a`. Then, it replaces the gene at that position with a new gene, formed by combining bits from the original course, professor, and group, along with randomly chosen slot and lecture theater.

The process is designed to introduce variability in the population and explore different solutions. This mutation operation helps the genetic algorithm avoid getting stuck in local optima and increases the diversity of the population.

```
// Python code for the mutate function
def mutate(chromosome):
    rand_slot = random.choice(slots)
    rand_lt = random.choice(lts)
    a = random.randint(0, len(chromosome) - 1)
```

```
    chromosome[a] = course_bits(chromosome[a]) + professor_bits(chromosome[a])+
    group_bits(chromosome[a]) + rand_slot + rand_lt
```

3.8 Preference as Soft Constraint

Features:

1. **Professor Grade:** This feature represents the preference or suitability of a professor for a particular time . It could be encoded as a numerical value, with higher values indicating a stronger preference.
2. **Time Preference (Morning/Evening/Either):** This feature captures the time preferences of a professor to conduct classes. It could be a categorical variable with values like "Morning," "Evening," or "Either."
3. **Subject:** This feature indicates the preference of a professor for a specific subject or course for a specific time.

Score Calculation using Linear Regression: Linear regression can be used to model the relationship between these features and the score assigned to a particular schedule. The formula for the linear regression model could be:

$$\text{Score} = \beta_0 + \beta_1 \times \text{Professor Grade} + \beta_2 \times \text{Time Preference} + \beta_3 \times \text{Subject Preference} + \epsilon$$

Here,

- $\beta_0, \beta_1, \beta_2, \beta_3$ are the coefficients to be learned from the data.
- ϵ is the error term.

Data Preparation:

1. **Encoding Categorical Variables:** Convert categorical variables (Time Preference, Subject Preference) into numerical representations using one-hot encoding or label encoding.
2. **Training Data:** Prepare a dataset with examples of schedules and their corresponding scores. The scores could be assigned based on expert opinions or historical data.

Model Training: Use a linear regression algorithm to train the model on the prepared dataset. The goal is to learn the coefficients that best fit the relationship between the features and the score.

Prediction: Once the model is trained, you can use it to predict the score for new schedules based on the professor's grade, time preference, and subject preference.

Integration into Genetic Algorithm: Integrate the linear regression model into your genetic algorithm evaluation function. After evaluating hard constraints, calculate the soft constraint score using the linear regression model and combine it with the overall score.

```
def evaluate_soft_constraint(schedule):

    professor_grade, time_preference, subject_preference = extract_features(schedule)
    predicted_score = linear_regression_predict(professor_grade, time_preference, subje

    return predicted_score
```

This approach allows you to incorporate teacher preferences as soft constraints, providing a more nuanced evaluation of schedules in your optimization process.

4 Evaluation Metrics

When evaluating the performance of the scheduling model, several metrics can be considered. These metrics provide insights into the effectiveness and efficiency of the generated schedules. Some common evaluation metrics include:

- **Fitness Score:** A comprehensive metric that considers both hard and soft constraints. It provides an overall measure of how well a schedule satisfies the specified criteria.
- **Preference Score:** The Preference Score is a quantitative measure indicating the satisfaction level with a particular schedule. It is calculated based on individual teacher preferences and is expressed as a percentage. Each teacher provides a preference score on a scale of 1 to 5, and the overall score is computed by summing up the individual scores and normalizing it with respect to the total number of teachers. The formula for the Preference Score (PS) is given by:

$$PS = \frac{\text{Sum of Individual Scores}}{5 \times \text{Total Number of Teachers}} \times 100$$

  This metric offers insights into how well the generated schedule aligns with the preferences of the teaching staff, with a higher score indicating a more satisfactory schedule.
- **Execution Time:** The time taken by the scheduling algorithm to generate a feasible schedule. This metric is crucial for assessing the efficiency of the algorithm, especially in large-scale scheduling scenarios.

These evaluation metrics collectively provide a holistic view of the scheduling model's performance and can guide future enhancements.

5 Results

Time in Coloab: 19-25 min Time in Local Machine: 48-61 min

```
A21
+----------+-------------+-------------+-------------+-------------+
| Day\Slot | 08:00-09:00 | 09:00-10:00 | 10:00-11:00 | 11:00-12:00 |
+----------+-------------+-------------+-------------+-------------+
| Mon      | PPSD N1     | SE N3       | -           | AI N1       |
+----------+-------------+-------------+-------------+-------------+
| Tue      | -           | AI N1       | PPSD N1     | SE N1       |
+----------+-------------+-------------+-------------+-------------+
| Wed      | SE N1       | PPSD N1     | -           | AI N2       |
+----------+-------------+-------------+-------------+-------------+

B21
+----------+-------------+-------------+-------------+-------------+
| Day\Slot | 08:00-09:00 | 09:00-10:00 | 10:00-11:00 | 11:00-12:00 |
+----------+-------------+-------------+-------------+-------------+
| Mon      | -           | AI N2       | -           | -           |
```

```
+----------+-------------+-------------+-------------+-------------+
| Tue      | SE N2       | PPSD N2     | SE N3       | PPSD N2     |
+----------+-------------+-------------+-------------+-------------+
| Wed      | -           | SE N2       | PPSD N2     | -           |
+----------+-------------+-------------+-------------+-------------+

C21
+----------+-------------+-------------+-------------+-------------+
| Day\Slot | 08:00-09:00 | 09:00-10:00 | 10:00-11:00 | 11:00-12:00 |
+----------+-------------+-------------+-------------+-------------+
| Mon      | SE N2       | PPSD N3     | AI N2       | PPSD N3     |
+----------+-------------+-------------+-------------+-------------+
| Tue      | AI N3       | -           | SE N2       | -           |
+----------+-------------+-------------+-------------+-------------+
| Wed      | AI N2       | -           | SE N1       | PPSD N3     |
+----------+-------------+-------------+-------------+-------------+

D21
+----------+-------------+-------------+-------------+-------------+
| Day\Slot | 08:00-09:00 | 09:00-10:00 | 10:00-11:00 | 11:00-12:00 |
+----------+-------------+-------------+-------------+-------------+
| Mon      | AI N3       | SE N1       | PPSD N2     | -           |
+----------+-------------+-------------+-------------+-------------+
| Tue      | PPSD N1     | AI N3       | -           | -           |
+----------+-------------+-------------+-------------+-------------+
| Wed      | SE N3       | SE N3       | -           | PPSD N1     |
+----------+-------------+-------------+-------------+-------------+

Teacher Statisfiablility(%): 76.3%
```

6 Test Case

Google Colab = 19-25 mins
Local Machine = 48-61 mins

```
        cpg = [CourseClass.find("PF"),
            Professor.find("Dr. Awais Hassan"),
            Group.find("A23"),
           CourseClass.find("PF"), Professor.find(
               "Dr. Awais Hassan"), Group.find("B23"),
           CourseClass.find("PF"), Professor.find(
               "Mr. Laeeq uz Zaman"), Group.find("C23"),
           CourseClass.find("PF"), Professor.find(
               "Ms. Maida"), Group.find("D23"),
           CourseClass.find("PF (Lab)"), Professor.find(
               "Ms. Maida"), Group.find("A23"),
           CourseClass.find("PF (Lab)"), Professor.find(
               "Ms. Maida"), Group.find("B23"),
           CourseClass.find("PF (Lab)"), Professor.find(
               "Ms. Maida"), Group.find("C23"),
```

```
CourseClass.find("PF (Lab)"), Professor.find(
    "Ms. Maida"), Group.find("D23"),
CourseClass.find("AICT"), Professor.find(
    "Ms. Abqa Javed"), Group.find("A23"),
CourseClass.find("AICT"), Professor.find(
    "Ms. Abqa Javed"), Group.find("B23"),
CourseClass.find("AICT"), Professor.find(
    "Dr. Amna Zafar"), Group.find("C23"),
CourseClass.find("AICT"), Professor.find(
    "Dr. Amna Zafar"), Group.find("D23"),
CourseClass.find("AICT (Lab)"), Professor.find(
    "Ms. Abqa Javed"), Group.find("A23"),
CourseClass.find("AICT (Lab)"), Professor.find(
    "Ms. Abqa Javed"), Group.find("B23"),
CourseClass.find("AICT (Lab)"), Professor.find(
    "Dr. Amna Zafar"), Group.find("C23"),
CourseClass.find("AICT (Lab)"), Professor.find(
    "Dr. Amna Zafar"), Group.find("D23"),
CourseClass.find("DMI"), Professor.find(
    "Waqas Ali"), Group.find("A23"),
CourseClass.find("DMI"), Professor.find(
    "Waqas Ali"), Group.find("B23"),
CourseClass.find("DMI"), Professor.find(
    "Waqas Ali"), Group.find("C23"),
CourseClass.find("DMI"), Professor.find(
    "Waqas Ali"), Group.find("D23"),
CourseClass.find("FE"), Professor.find(
    "Humanities Department I"), Group.find("A23"),
CourseClass.find("FE"), Professor.find(
    "Humanities Department I"), Group.find("B23"),
CourseClass.find("FE"), Professor.find(
    "Humanities Department I"), Group.find("C23"),
CourseClass.find("FE"), Professor.find(
    "Humanities Department I"), Group.find("D23"),
CourseClass.find("C"), Professor.find(
    "Mathematics Department I"), Group.find("A23"),
CourseClass.find("C"), Professor.find(
    "Mathematics Department I"), Group.find("B23"),
CourseClass.find("C"), Professor.find(
    "Mathematics Department I"), Group.find("C23"),
CourseClass.find("C"), Professor.find(
    "Mathematics Department I"), Group.find("D23"),
CourseClass.find("WP"), Professor.find(
    "Mechanical Department I"), Group.find("A23"),
CourseClass.find("WP"), Professor.find(
    "Mechanical Department I"), Group.find("B23"),
CourseClass.find("WP"), Professor.find(
    "Mechanical Department I"), Group.find("C23"),
CourseClass.find("WP"), Professor.find(
    "Mechanical Department I"), Group.find("D23"),
```

```
        CourseClass.find("DSA"),
        Professor.find("Mr. Nazeef ul Haq"),
        Group.find("A22"),
        CourseClass.find("DSA"),
        Professor.find("Mr. Nazeef ul Haq"),
        Group.find("B22"),
        CourseClass.find("DSA"),
        Professor.find("Mr. Faraz"),
        Group.find("C22"),
        CourseClass.find("DSA"),
        Professor.find("Mr. Faraz"),
        Group.find("D22"),
        CourseClass.find("DSA (Lab)"),
        Professor.find("Mr. Nazeef ul Haq"),
        Group.find("A22"),
        CourseClass.find("DSA (Lab)"),
        Professor.find("Mr. Nazeef ul Haq"),
        Group.find("B22"),
        CourseClass.find("DSA (Lab)"),
        Professor.find("Mr. Faraz"),
        Group.find("C22"),
        CourseClass.find("DSA (Lab)"),
        Professor.find("Mr. Faraz"),
        Group.find("D22"),
        CourseClass.find("TW"),
        Professor.find("Humanities Department II"),
        Group.find("A22"),
        CourseClass.find("TW"),
        Professor.find("Humanities Department II"),
        Group.find("B22"),
        CourseClass.find("TW"),
        Professor.find("Humanities Department II"),
        Group.find("C22"),
        CourseClass.find("TW"),
        Professor.find("Humanities Department II"),
        Group.find("D22"),
        CourseClass.find("COAL"),
        Professor.find("Mr. Tauqir Ahmed"),
        Group.find("A22"),
        CourseClass.find("COAL"),
        Professor.find("Mr. Tauqir Ahmed"),
        Group.find("B22"),
        CourseClass.find("COAL"),
        Professor.find("Mr. Tauqir Ahmed"),
        Group.find("C22"),
        CourseClass.find("COAL"),
        Professor.find("Mr. Tehseen"),
        Group.find("D22"),
        CourseClass.find("COAL (Lab)"),
        Professor.find("Mr. Tehseen"),
```

```
Group.find("A22"),
CourseClass.find("COAL (Lab)"),
Professor.find("Mr. Tehseen"),
Group.find("B22"),
CourseClass.find("COAL (Lab)"),
Professor.find("Mr. Tehseen"),
Group.find("C22"),
CourseClass.find("COAL (Lab)"),
Professor.find("Mr. Tehseen"),
Group.find("D22"),
CourseClass.find("LA"),
Professor.find("Mathematics Department II"),
Group.find("A22"),
CourseClass.find("LA"),
Professor.find("Mathematics Department II"),
Group.find("B22"),
CourseClass.find("LA"),
Professor.find("Mathematics Department II"),
Group.find("C22"),
CourseClass.find("LA"),
Professor.find("Mathematics Department II"),
Group.find("D22"),
CourseClass.find("DMII"),
Professor.find("Dr. Ayesha Altaf"),
Group.find("A22"),
CourseClass.find("DMII"),
Professor.find("Dr. Ayesha Altaf"),
Group.find("B22"),
CourseClass.find("DMII"),
Professor.find("Dr. Ayesha Altaf"),
Group.find("C22"),
CourseClass.find("DMII"),
Professor.find("Dr. Ayesha Altaf"),
Group.find("D22"),
CourseClass.find("CN"),
Professor.find("Dr. Junaid Arshad"),
Group.find("A21"),
CourseClass.find("CN"),
Professor.find("Dr. Junaid Arshad"),
Group.find("B21"),
CourseClass.find("CN"),
Professor.find("Dr. Junaid Arshad"),
Group.find("C21"),
CourseClass.find("CN"),
Professor.find("Ms. Aroosh"),
Group.find("D21"),
CourseClass.find("CN (Lab)"),
Professor.find("Ms. Aroosh"),
Group.find("A21"),
CourseClass.find("CN (Lab)"),
```

```
Professor.find("Ms. Aroosh"),
Group.find("B21"),
CourseClass.find("CN (Lab)"),
Professor.find("Ms. Aroosh"),
Group.find("C21"),
CourseClass.find("CN (Lab)"),
Professor.find("Ms. Aroosh"),
Group.find("D21"),
CourseClass.find("AI"),
Professor.find("Prof. Dr. M. Aslam"),
Group.find("B21"),
CourseClass.find("AI"),
Professor.find("Mr. Samyan Qayyum"),
Group.find("A21"),
CourseClass.find("AI"),
Professor.find("Mr. Samyan Qayyum"),
Group.find("C21"),
CourseClass.find("AI (Lab)"),
Professor.find("Ms. Tazeem"),
Group.find("A21"),
CourseClass.find("AI (Lab)"),
Professor.find("Ms. Tazeem"),
Group.find("B21"),
CourseClass.find("AI (Lab)"),
Professor.find("Ms. Tazeem"),
Group.find("C21"),
CourseClass.find("AI (Lab)"),
Professor.find("Ms. Tazeem"),
Group.find("D21"),
CourseClass.find("PPSD"),
Professor.find("Dr. Faiza Iqbal"),
Group.find("A21"),
CourseClass.find("PPSD"),
Professor.find("Dr. Faiza Iqbal"),
Group.find("B21"),
CourseClass.find("PPSD"),
Professor.find("Dr. Faiza Iqbal"),
Group.find("C21"),
CourseClass.find("PPSD"),
Professor.find("Dr. Faiza Iqbal"),
Group.find("D21"),
CourseClass.find("SE"),
Professor.find("Mr. Aatif Hussain"),
Group.find("A21"),
CourseClass.find("SE"),
Professor.find("Mr. Aatif Hussain"),
Group.find("B21"),
CourseClass.find("SE"),
Professor.find("Prof. Dr. Shazia Arshad"),
Group.find("C21"),
```

```
CourseClass.find("SE"),
Professor.find("Prof. Dr. Shazia Arshad"),
Group.find("D21"),
CourseClass.find("SE (Lab)"),
Professor.find("Mr. Laeeq uz Zaman"),
Group.find("A21"),
CourseClass.find("SE (Lab)"),
Professor.find("Mr. Laeeq uz Zaman"),
Group.find("B21"),
CourseClass.find("SE (Lab)"),
Professor.find("Mr. Laeeq uz Zaman"),
Group.find("C21"),
CourseClass.find("SE (Lab)"),
Professor.find("Mr. Laeeq uz Zaman"),
Group.find("D21"),
CourseClass.find("MAD"),
Professor.find("Dr. Amjad Farooq"),
Group.find("A21"),
CourseClass.find("MAD"),
Professor.find("Dr. Amjad Farooq"),
Group.find("B21"),
CourseClass.find("MAD"),
Professor.find("Dr. Amjad Farooq"),
Group.find("C21"),
CourseClass.find("MAD"),
Professor.find("Mr. Waseem"),
Group.find("D21"),
CourseClass.find("CC"),
Professor.find("Dr. Talha Waheed"),
Group.find("A20"),
CourseClass.find("CC"),
Professor.find("Dr. Talha Waheed"),
Group.find("B20"),
CourseClass.find("CC"),
Professor.find("Dr. Talha Waheed"),
Group.find("C20"),
CourseClass.find("CC (Lab)"),
Professor.find("Ms. Aaliya"),
Group.find("A20"),
CourseClass.find("CC (Lab)"),
Professor.find("Ms. Aaliya"),
Group.find("B20"),
CourseClass.find("CC (Lab)"),
Professor.find("Ms. Aaliya"),
Group.find("C20"),
CourseClass.find("NLP"),
Professor.find("Dr. Usman Ghani"),
Group.find("A20"),
CourseClass.find("NLP"),
Professor.find("Dr. Usman Ghani"),
```

```
        Group.find("B20"),
        CourseClass.find("NLP"),
        Professor.find("Dr. Usman Ghani"),
        Group.find("C20"),
        CourseClass.find("IPS"),
        Professor.find("Department of Islamic Studies"),
        Group.find("A20"),
        CourseClass.find("IPS"),
        Professor.find("Department of Islamic Studies"),
        Group.find("B20"),
        CourseClass.find("IPS"),
        Professor.find("Department of Islamic Studies"),
        Group.find("C20"),
        CourseClass.find("IR"),
        Professor.find("Dr. Khaldoon"),
        Group.find("A20"),
        CourseClass.find("IR"),
        Professor.find("Dr. Khaldoon"),
        Group.find("B20"),
        CourseClass.find("IR"),
        Professor.find("Dr. Khaldoon"),
        Group.find("C20"),
        CourseClass.find("CCom"),
        Professor.find("Nauman Shafee"),
        Group.find("A20"),
        CourseClass.find("CCom"),
        Professor.find("Nauman Shafee"),
        Group.find("B20"),
        CourseClass.find("CCom"),
        Professor.find("Nauman Shafee"),
        Group.find("C20"),

        ]
```

7 Challenges

7.1 Lack of Previous Dataset

No previous dataset was available for reference, which posed a significant challenge in obtaining historical data to inform the model. The absence of such data makes it difficult to analyze past patterns and trends, potentially impacting the accuracy and robustness of the model.

7.2 Feature Selection for Teacher Preference

The decision-making process for selecting features related to Teacher Preference presented another notable challenge. Determining which features to include in the model, such as Professor Grade, Timing Preferences (Morning, Evening, Either), and Subject, required careful consideration. The challenge lies in identifying the most relevant and influential features that contribute to the accuracy of the model's predictions. (You may consider adding more details or challenges

here.)

8 Limitations

- **No Constraint on Credit Hours:** The current model does not account for credit hour constraints, which could impact the overall scheduling process.
- **Assumption of Teacher Already Assigned to Courses:** The model assumes that teachers are already assigned to specific courses, which might not always reflect the real-world scenario.
- **Course Division:** The model does not support the division of a course into multiple parts, limiting its flexibility in handling complex scheduling scenarios.

9 Future Improvements

- **Dynamic Constraint Adaptation:** Implement a mechanism for the scheduling model to dynamically adapt to changing constraints and preferences. This could involve real-time adjustments based on evolving academic requirements or teacher availability.
- **Intelligent Credit Hour Handling:** Enhance the model to intelligently consider credit hour constraints, optimizing schedules while adhering to academic credit requirements. This would provide a more realistic representation of course schedules.
- **Support for Course Division:** Extend the model's capabilities to handle courses that need to be divided into multiple parts. This enhancement would accommodate scenarios where a single course spans multiple sessions or has different components.
- **Integration with Student Preferences:** Explore the integration of student preferences into the scheduling model. This could lead to schedules that not only meet academic requirements but also consider student preferences for optimal learning experiences.
- **Scalability and Performance Optimization:** Optimize the scheduling algorithm to handle large-scale scenarios efficiently. This includes improving computational efficiency, reducing execution time, and enhancing the model's scalability for complex scheduling tasks.
- **Visualization Tools:** Develop visualization tools to represent generated schedules in an intuitive and user-friendly manner. This would facilitate easier interpretation of schedules by administrators, teachers, and students.
- **User Interface Enhancements:** Improve the user interface for better interaction with the scheduling model. A user-friendly interface can streamline the input of constraints, preferences, and the interpretation of generated schedules.
- **Feedback Mechanism:** Implement a feedback mechanism that collects input from users, including teachers and administrators, to continuously refine and enhance the scheduling model based on real-world experiences and requirements.

10 Conclusion

This project successfully implemented a Teacher Preference-Driven Schedule Optimization system, employing genetic algorithms and constraint satisfaction techniques for university class

scheduling. The system demonstrated its ability to generate optimal schedules considering various constraints and preferences, with reasonable execution times on both Google Colab and a local machine.

While achieving its objectives, opportunities for future work include incorporating additional constraints like room availability, breaks between classes, and specific professor preferences. Further refinement of genetic algorithm parameters and exploration of alternative optimization techniques could contribute to enhanced system performance. The Teacher Preference-Driven Schedule Optimization system provides a foundation for automated and efficient university class scheduling, with ongoing refinements holding the potential for even more robust and tailored scheduling solutions.