FAST National University of Computer and Emerging Sciences

Artificial Intelligence

Project Report

Student Name:
Farzeen Qaiser (21i-0782)

Instructor Name:
Labiba Fatima

Submission date: 12/05/2024
Section: BSCS-G

Project Problem:

The timetable scheduling problem is a classic challenge encountered in university settings, involving the creation of schedules for each semester while minimizing conflicts between sections, professors, and room allocations. This problem entails assigning time slots to individual sections within specific rooms, with designated professors teaching particular courses.

Initializing Variables:

The code begins by defining key parameters such as the number of courses, sections, professors, days, timeslots per day, rooms, maximum room size, and maximum section strength. These parameters are crucial for designing an effective timetable.

1. Binary Representation:
    a. To facilitate the genetic algorithm's operations, each component (courses, sections, professors, etc.) is represented in binary form. The number of bits required for each component is computed using the logarithm function, ensuring sufficient representation space.
2. Generation of Unique Binary Representations:
    a. Unique binary representations are generated for sections, courses, professors, days, timeslots, and rooms. These representations enable efficient manipulation and storage of data during the scheduling process.
3. Assignment of Courses to Professors:
    a. Professors are randomly assigned a set number of courses, considering constraints such as the maximum number of courses per professor. This assignment is crucial for ensuring fair distribution of teaching responsibilities among faculty members.
4. Assignment of Courses to Sections:
    a. Courses are allocated to sections based on random selection, considering the maximum number of courses per section. This allocation ensures a diverse course offering for each section, catering to students' academic needs.
5. Creation of Global Rooms List:

    a. The code generates a list of rooms along with their respective sizes, represented in binary form. This step lays the foundation for room allocation during the timetable generation process.
6. Creation of Days and Timeslots List:
    a. Days and timeslots are combined to form a comprehensive list, facilitating the scheduling of classes across different time periods throughout the week.

Fitness Function Evaluation Criteria:

1. Lecture Clashes:
    a. The fitness function penalizes chromosomes with lecture clashes, where multiple lectures are scheduled in the same day, timeslot, and room. Such clashes are detrimental to effective timetable scheduling and are assigned a negative fitness score.
2. Section and Room Size Constraints:
    a. Chromosomes violating constraints related to section and room sizes are penalized. If the section strength is below 60 and the room size is 1, or if the section strength exceeds 60 and the room size is 0, the fitness score is decremented.
3. Professor's Teaching Schedule:
    a. The function ensures that a professor cannot teach two courses simultaneously. If two lectures by the same professor are scheduled in the same day and timeslot, the fitness score is reduced.
4. Adjacent Days Allocation:
    a. Chromosomes allocating courses on adjacent days are penalized. Courses scheduled on consecutive days are undesirable as they may lead to student fatigue and scheduling difficulties. Such allocations receive negative fitness scores.

Population Generation:

- The population is initialized with a specified size (pop_size), representing the number of individuals (chromosomes) in the population.

- Each chromosome represents a potential timetable solution and consists of various components such as courses, sections, professors, rooms, and lecture schedules.
- Chromosomes are generated iteratively using a loop, where each iteration creates a new chromosome.
- For each chromosome, the code randomly selects course-related information, including course ID, theory/lab designation, section, section strength, and professor, from the available section_courses list.
- Additionally, room-related information, including room ID and size, is randomly selected from the global_rooms list.
- Lecture schedules for both the first and second lectures are randomly assigned by selecting days and timeslots from the days_and_timeslots list.
- The selected components are appended to the chromosome in a predefined order, forming a complete timetable representation.

Mutation:

1. The mutation process begins by randomly selecting a chromosome from the population for mutation. This selection is made from the sorted_random_rows DataFrame, which contains potential parents for mutation.
2. Next, a random decision is made regarding which component of the chromosome will undergo mutation. The choice is between replacing either the first lecture's day and timeslot (columns 5 and 6) or the second lecture's day and timeslot (columns 9 and 10).
3. After determining the mutation point, a random index is selected from the days_and_timeslots list to obtain a new day and timeslot value.
4. Alternatively, a random index from the global_rooms list is chosen to obtain a new room and room size value.
5. The selected component of the chromosome is then replaced with the new value, introducing variability into the population.
6. The code utilizes conditional statements to determine which component of the chromosome to replace based on the randomly chosen option ('first' or 'second').

7. The replacement process involves updating the corresponding columns in the selected chromosome with the new values obtained from the random selection.

Crossover:

1. Parent Selection:
    a. The code selects pairs of parent chromosomes from the sorted_random_rows DataFrame, which contains potential parents for crossover. These parents are sorted based on their fitness scores, ensuring that fitter individuals have a higher chance of being selected as parents.
2. Crossover Point:
    a. The crossover operator defines specific columns (day, timeslot, room, and room size) as crossover points. These columns contain information crucial for timetable scheduling and are exchanged between parent chromosomes to produce offspring.
3. Exchange of Information:
    a. For each pair of parent chromosomes, the crossover process involves swapping the values of the specified columns between the two parents. This exchange of information creates offspring with characteristics inherited from both parents.
4. Implementation Details:
    a. The code iterates over pairs of parent chromosomes, swapping the values of the specified columns to create new offspring.

Results:

Sample 1:

	Course	Theory/Lab	Section	Section Strength	Professor	Day 1	Timeslot 1	Room 1	Room Size 1	Day 2	Timeslot 2	Room 2	Room Size 2	Fitness Score
0	0010	1	0000	0101110	0111	100	001	11010	0	001	000	10000	0	0
1	0001	0	0110	1000001	0101	001	010	01010	1	011	010	10001	1	0
2	0011	0	0010	0111100	0101	100	011	00000	1	011	011	11000	1	-4
3	1101	0	0001	1100010	0110	100	100	11011	0	001	001	01111	1	-1
4	0110	0	1000	0011111	1000	000	011	10110	0	001	001	10010	1	-3
5	1011	1	0111	1111000	1001	010	000	11001	0	000	001	00111	0	-2
6	0000	0	0101	1101110	0100	100	010	10110	0	000	100	01110	0	-2
7	1101	0	0111	1101001	0110	001	101	01010	1	010	010	10001	1	-1
8	0001	0	0111	0100011	0101	000	000	00010	0	001	011	00101	1	-6
9	0011	0	0010	0111100	0101	100	011	10110	0	001	001	10010	1	-6
10	0110	0	1000	0011111	1000	100	011	00000	1	011	011	11000	1	-5
11	0001	0	0111	0100011	0101	001	011	11000	1	100	100	11001	0	-5
12	0001	0	1000	0101111	0101	000	000	00010	0	001	011	00101	1	-4
13	0111	1	0011	1101011	0011	000	001	10000	0	001	011	10111	1	-1
14	1101	0	0111	1101001	0110	001	011	11000	1	100	100	11001	0	-4

Sample 2:

	Course	Theory/Lab	Section	Section Strength	Professor	Day 1	Timeslot 1	Room 1	Room Size 1	Day 2	Timeslot 2	Room 2	Room Size 2	Fitness Score
0	1110	0	0000	0011111	0000	000	001	00011	0	010	001	01110	0	-3
1	0001	0	1000	0101111	0101	001	101	11011	0	010	010	01101	1	-1
2	0101	1	1001	1011100	0111	000	101	01101	1	000	000	11010	0	-1
3	0011	0	0010	0111100	0101	010	100	00111	0	010	001	11000	1	-3
4	1101	0	0101	1011111	0110	011	010	00011	0	001	001	10111	1	-1
5	0011	0	0010	0111100	0101	000	010	01010	1	100	010	01100	1	-4
6	0111	1	0011	1101011	0011	011	010	01000	1	010	100	10111	1	-3
7	0001	0	0110	0000010	1001	100	100	01010	1	100	101	11011	0	-1
8	1110	0	0111	0110001	0011	000	001	10111	1	001	010	11100	1	-7
9	1110	0	0000	0011111	0000	011	010	01000	1	010	100	10111	1	-5
10	0111	1	0011	1101011	0011	000	001	00011	0	010	001	01110	0	-6
11	1110	0	0111	0110001	0011	000	010	01010	1	100	010	01100	1	-6
12	0011	0	0010	0111100	0101	000	001	10111	1	001	010	11100	1	-6
13	0111	1	0011	1101011	0011	000	000	11000	1	000	010	01100	1	0
14	0001	0	1000	1010001	1001	001	000	01010	1	000	000	11011	0	-1

Conclusion:

In conclusion, the genetic algorithm for timetable scheduling efficiently explores solutions through mutation and crossover, ensuring diversity and quality in generated schedules. By iteratively refining schedules based on fitness evaluation, the algorithm addresses complex scheduling requirements, offering promising solutions for optimizing university timetables.