UNIVERSITY TIMETABLE SCHEDULING

USING HILL CLIMBING ALGORITHM

TABLE OF CONTENT

01

INTRODUCTION

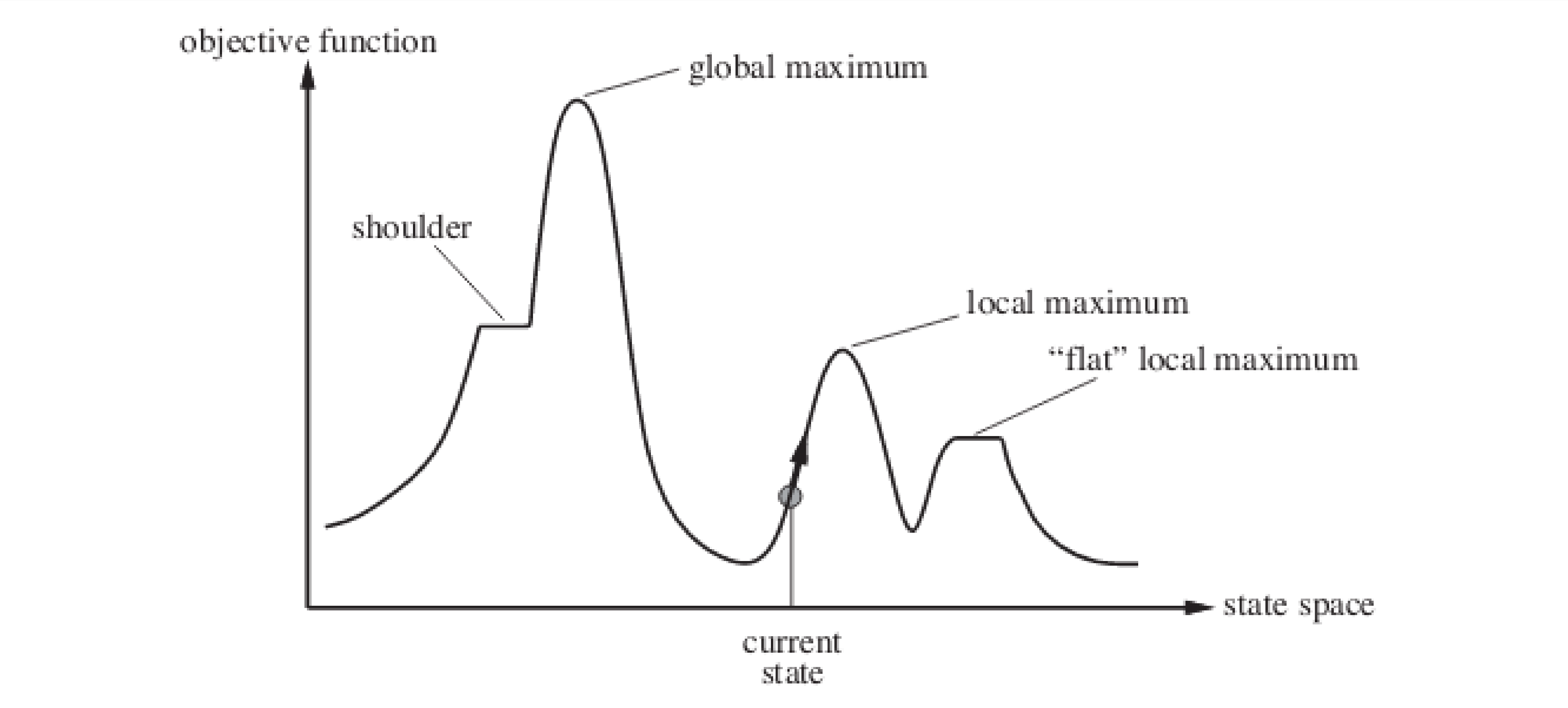
Hill Climbing Algorithm

Problem Statement :

Timetable scheduling is the process of assigning courses, professors, rooms, and time slots in an organized manner to avoid conflicts and ensure efficient resource utilization.

Common Challenges in Timetable Scheduling:

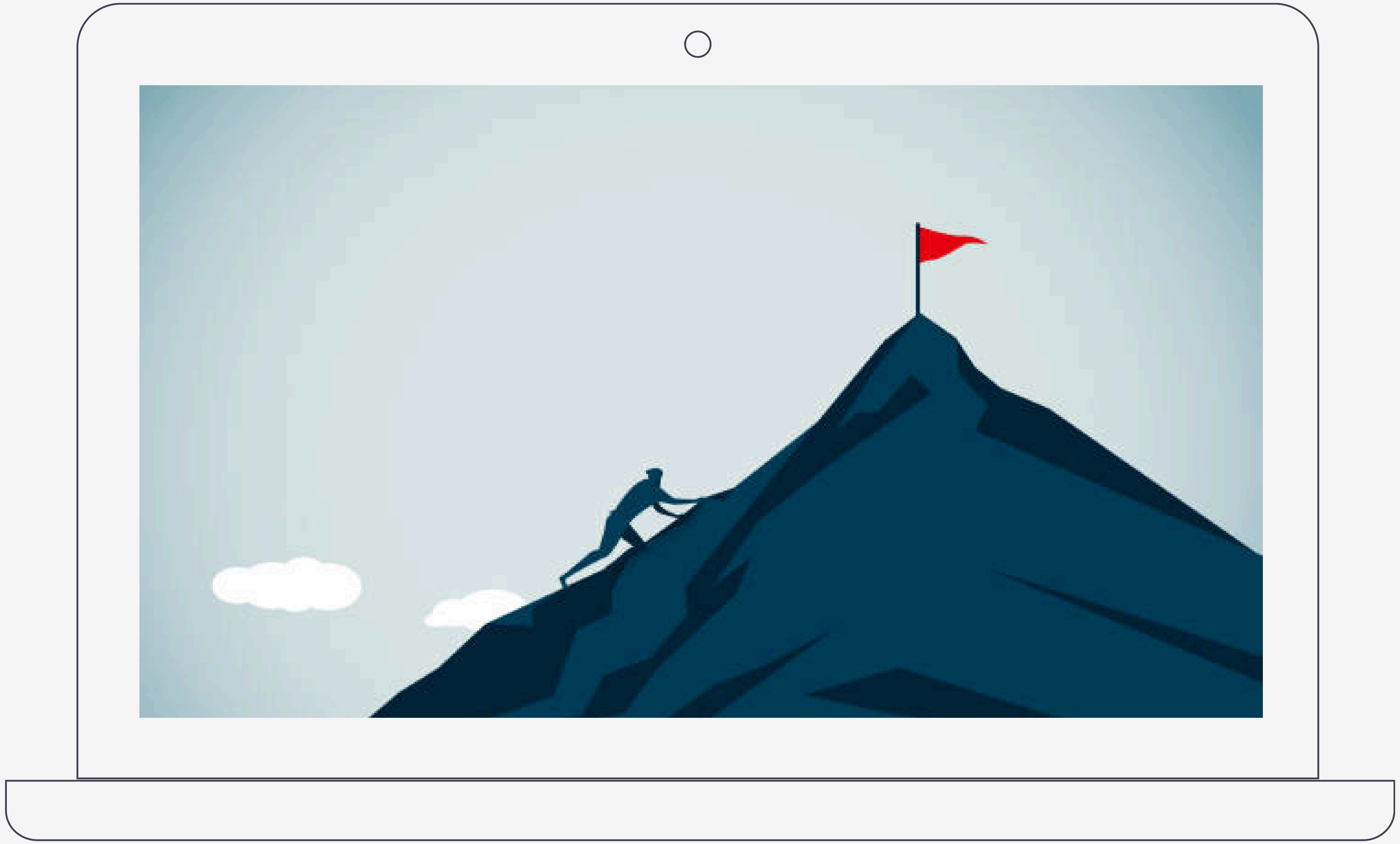
- **Room Overlaps** – Two classes assigned to the same room.
- **Professor Conflicts** – A professor is scheduled to teach two courses at the same time.
- **Course Clashes** – Two compulsory courses scheduled at the same time.
- **Scalability Issues** – Handling large datasets for universities with thousands of courses.



HILL CLIMBING ALGORITHM

How Hill Climbing Works (Algorithm):

- Start with an initial solution (can be random or a predefined heuristic-based solution).
- Evaluate the solution using a fitness function.
- Generate neighboring solutions by making small changes (e.g., swapping values, modifying elements).
- Select the best neighbor that improves the solution.
- Repeat the process until:
    - No better solution is found (local optimum reached).
    - A stopping condition (e.g., a time limit or a number of iterations) is met.



ALGORITHM

Initial Schedule Generation

FUNCTION generateInitialSchedule():

1. INITIALIZE an empty list schedule. → O(1)
2. CREATE a random number generator rand. → O(1)
3. FOR each course in the list of courses: → O(n) (no of courses)
    - 4. ASSIGN a random professor. → O(1)
    - 5. ASSIGN a random time slot. → O(1)
    - 6. ASSIGN a random room. → O(1)
    - 7. CREATE a new ScheduleEntry object with the selected values. → O(1)
    - 8 ADD the entry to schedule. → O(1)

9 RETURN the schedule list. → O(1)

Total Complexity :O(n) (Linear Complexity)

Conflict Detection Algorithm

FUNCTION countConflicts(schedule):

1. INITIALIZE conflicts = 0. → O(1)
    2. FOR each entry i in the schedule: → O(N)
        3. FOR each entry j after i in the schedule: → O(N)
            4. GET entryA (current entry). → O(1)
            5. GET entryB (next entry). → O(1)
            6. CHECK if entryA and entryB have the same room and time. → O(1)
                7. INCREMENT conflicts if true. → O(1)
            8. CHECK if entryA and entryB have the same professor and time. → O(1)
                9. INCREMENT conflicts if true. → O(1)

10. RETURN conflicts. → O(1)

Total complexity: O(N) × O(N) × O(1) = $O(N^2)$.

Neighbour Detection Algorithm

FUNCTION hillClimbTimetable()

1. INITIALIZE currentSchedule using generateInitialSchedule(). → $O(C)$
2. COMPUTE currentConflicts using countConflicts(). → $O(N^2)$
3. WHILE true: → $O(K)$
4.   GENERATE neighbors using generateNeighbors(). → $O(N)$
5.   INITIALIZE bestSchedule as the first neighbor. → $O(1)$
6.   COMPUTE bestConflicts using countConflicts(). → $O(N^2)$
7.   FOR each neighbor in neighbors: → $O(10)$
8.   COMPUTE conflicts using countConflicts(). → $O(N^2)$
9.   IF conflicts < bestConflicts, UPDATE bestSchedule, bestConflicts. → $O(1)$
10.   IF bestConflicts >= currentConflicts, RETURN currentSchedule. → $O(1)$
11.   UPDATE currentSchedule, currentConflicts to best values. → $O(1)$
12. END FUNCTION. → $O(1)$

Total complexity: $O(1) \times O(N) \times O(1) = O(N)$.

Hill Climbing Algorithm

hillClimbTimetable()

1. INIT currentSchedule → O(C)
2. COMPUTE currentConflicts → $O(N^2)$
3. WHILE true → O(K)
   4. GENERATE neighbors → O(N)
   5. SET bestSchedule = neighbors[0] → O(1)
   6. COMPUTE bestConflicts → $O(N^2)$
   7. FOR each neighbor → O(10)
   8. COMPUTE conflicts → $O(N^2)$
   9. IF conflicts < bestConflicts, UPDATE bestSchedule, bestConflicts → O(1)
   10. IF bestConflicts >= currentConflicts, RETURN currentSchedule → O(1)
   11. UPDATE currentSchedule, currentConflicts → O(1)
12. END → O(1)

Time Complexity: $O(K \times N^2)$

Test Case 1

```
Final Optimized Timetable:
DSA - Dr. Zen - Room 101 - 10 AM
Algebra - Dr. Ben - Room 101 - 9 AM
C++ - Dr. Charles - Room 101 - 2 PM
DBMS - Dr. Ross - Room 101 - 9 AM
UID - Dr. Charles - Room 102 - 9 AM
OOPS - Dr. Zen - Room 101 - 11 AM
Conflicts remaining: 1

Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5):
```

```
Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5): 1

Fully Overlapping Schedule Test

Before Optimization:
DSA - Dr. Zen - Room 101 - 9 AM
Algebra - Dr. Ben - Room 101 - 9 AM
C++ - Dr. Charles - Room 101 - 9 AM
DBMS - Dr. Ross - Room 101 - 9 AM
UID - Dr. Charles - Room 101 - 9 AM
OOPS - Dr. Zen - Room 101 - 9 AM
Conflicts remaining: 17

Conflict detected:
- Room Room 101 is used for DSA and Algebra at 9 AM.
- Room Room 101 is used for DSA and C++ at 9 AM.
- Room Room 101 is used for DSA and DBMS at 9 AM.
- Room Room 101 is used for DSA and UID at 9 AM.
- Room Room 101 is used for DSA and OOPS at 9 AM.
- Dr. Zen is assigned to DSA and OOPS at the same time.
- Room Room 101 is used for Algebra and C++ at 9 AM.
- Room Room 101 is used for Algebra and DBMS at 9 AM.
- Room Room 101 is used for Algebra and UID at 9 AM.
- Room Room 101 is used for Algebra and OOPS at 9 AM.
- Room Room 101 is used for C++ and DBMS at 9 AM.
- Room Room 101 is used for C++ and UID at 9 AM.
- Dr. Charles is assigned to C++ and UID at the same time.
- Room Room 101 is used for C++ and OOPS at 9 AM.
- Room Room 101 is used for DBMS and UID at 9 AM.
- Room Room 101 is used for DBMS and OOPS at 9 AM.
- Room Room 101 is used for UID and OOPS at 9 AM.
Total conflicts: 17
```

Test Case 2

```
Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5): 2

All Professors Busy at the Same Time Test

Before Optimization:
DSA - Dr. Zen - Room 101 - 10 AM
Algebra - Dr. Ben - Room 102 - 10 AM
C++ - Dr. Charles - Room 103 - 10 AM
DBMS - Dr. Ross - Room 101 - 10 AM
UID - Dr. Charles - Room 102 - 10 AM
OOPS - Dr. Zen - Room 103 - 10 AM
Conflicts remaining: 5

Conflict detected:
- Room Room 101 is used for DSA and DBMS at 10 AM.
- Dr. Zen is assigned to DSA and OOPS at the same time.
- Room Room 102 is used for Algebra and UID at 10 AM.
- Dr. Charles is assigned to C++ and UID at the same time.
- Room Room 103 is used for C++ and OOPS at 10 AM.
Total conflicts: 5

Final Optimized Timetable:
DSA - Dr. Zen - Room 103 - 10 AM
Algebra - Dr. Ben - Room 102 - 9 AM
C++ - Dr. Charles - Room 101 - 11 AM
DBMS - Dr. Ross - Room 101 - 10 AM
UID - Dr. Charles - Room 102 - 10 AM
OOPS - Dr. Zen - Room 101 - 2 PM
Conflicts remaining: 0
```

Test Case 3

```
Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5): 3

Minimal Schedule (Single Course) Test

Before Optimization:
DSA - Dr. Zen - Room 101 - 9 AM
Conflicts remaining: 0

Conflict detected:
Total conflicts: 0

Final Optimized Timetable:
DSA - Dr. Zen - Room 101 - 9 AM
Conflicts remaining: 0
```

Test Case 4

```
Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5): 4

Moderate Conflict Schedule Test

Before Optimization:
DSA - Dr. Zen - Room 101 - 9 AM
Algebra - Dr. Ben - Room 102 - 10 AM
C++ - Dr. Charles - Room 101 - 10 AM
DBMS - Dr. Ross - Room 103 - 1 PM
UID - Dr. Charles - Room 102 - 2 PM
OOPS - Dr. Zen - Room 103 - 3 PM
Conflicts remaining: 0

Conflict detected:
Total conflicts: 0

Final Optimized Timetable:
DSA - Dr. Zen - Room 101 - 9 AM
Algebra - Dr. Ben - Room 102 - 10 AM
C++ - Dr. Charles - Room 101 - 10 AM
DBMS - Dr. Ross - Room 103 - 1 PM
UID - Dr. Charles - Room 102 - 2 PM
OOPS - Dr. Zen - Room 103 - 3 PM
Conflicts remaining: 0
```

Test Case 5

```
Choose a test case:
1. Fully Overlapping Schedule
2. All Professors Busy at the Same Time
3. Minimal Schedule (Single Course)
4. Moderate Conflict Schedule
5. Exit
Enter choice (1-5): 5
Exiting...
```

Time Complexity

- generateInitialSchedule() : O(n)
- countConflicts() : O(n^2)
- generateNeighbors() : O(10n) → O(n)
- fillClimbingTimetable() : O(k*n^2)
- printSchedule() : O(n)

- Best Case[Zero Conflicts] : O(n) + O(n^2) = O(n^2)
- Worst Case[Many Iterations Needed]: O(k*n^2), k → number of iterations for convergence.

Overall Time Complexity : O(n^2) [dominant factor : conflict checking]

We tackled the University Timetable Scheduling problem using the Hill Climbing Algorithm to reduce conflicts.

- Fast and Efficient: Quickly finds an optimized schedule by making small, incremental improvements.
- Better than Brute Force: Instead of evaluating all possibilities, it focuses on improvements, making it practical for real-world use.
- Simple and Easy to Implement: Works well for small-to-medium scheduling problems without complex computations.

Presentation by

D VIRITHA - CB.SC.U4CSE23613

GEETHIKA G NAIR - CB.SC.U4CSE23616

MEERA J VAISHNAV - CB.SC.U4CSE23633

RAMA ROSHINEE S V - CB.SC.U4CSE23645