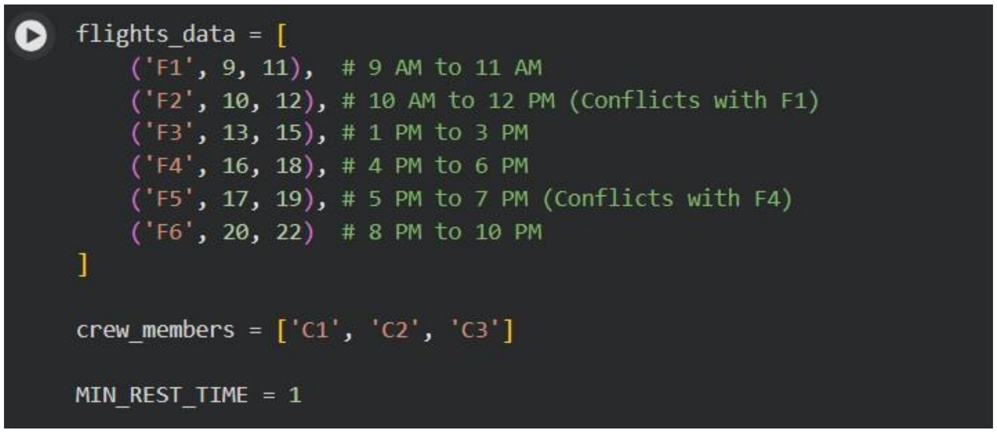
LAB SHEET 4

Airline Crew Scheduling – NP Hard Problem Solving

Objective:- It involves assigning flights to crew members under complex constraints such as availability, rest periods between flights, non-overlapping schedules, and optional cost optimization.

Summary and Analysis:-

This setup models the real-world resource allocation problem where flights must be assigned to crew members while respecting constraints like non-overlapping schedules and mandatory rest periods.



```python
flights_data = [
    ('F1', 9, 11),  # 9 AM to 11 AM
    ('F2', 10, 12), # 10 AM to 12 PM (Conflicts with F1)
    ('F3', 13, 15), # 1 PM to 3 PM
    ('F4', 16, 18), # 4 PM to 6 PM
    ('F5', 17, 19), # 5 PM to 7 PM (Conflicts with F4)
    ('F6', 20, 22)  # 8 PM to 10 PM
]

crew_members = ['C1', 'C2', 'C3']

MIN_REST_TIME = 1
```

Constraints Checker:-

The is_valid_assignment function rigorously checks two core constraints for adding a new flight to a crew member's schedule:

1. No Overlapping Flights: It strictly checks that the new flight's time interval does not overlap with any existing flight in the schedule.
2. Overlap Detection Formula: The condition max(new_start, start) < min(new_end, end) mathematically confirms any time conflict, immediately making the assignment invalid.

3. Minimum Rest Time Required: The function ensures a minimum rest period (e.g., 1 hour, based on MIN_REST_TIME) is maintained between sequential flights assigned to the crew.

4. Rest Time Calculation (Forward): If the new flight is scheduled *after* an existing one, the gap (new_start - end) must be greater than or equal to the required rest time.

5. Rest Time Calculation (Backward): If the new flight is scheduled *before* an existing one, the gap (start - new_end) must also meet or exceed the minimum rest time.

6. Constraint Satisfaction Core: If any of the checks fail, the function returns False, indicating a constraint violation; otherwise, it returns True, allowing the assignment in the backtracking process.

Backtracking Algorithm

This recursive backtracking function, solve_crew_scheduling, attempts to find a valid assignment for all flights to crew members[1].

1. Iterative Search: It processes flights one by one using the index and iterates through all available crew_members for the current flight.

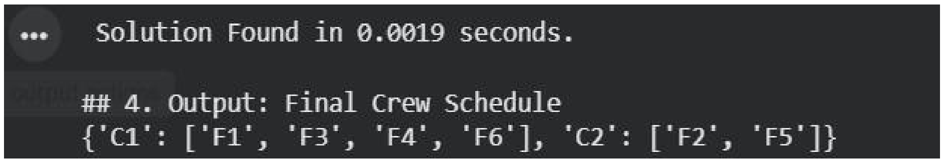
2. Constraint Check: The assignment is only attempted if the is_valid_assignment function confirms no time overlap or rest time violation.

3. Choose C Recurse: If valid, the flight is temporarily assigned (Choose), and the function recursively calls itself for the next flight (index + 1) (Recurse)[3].

4. Backtrack: If the recursive call fails to find a solution, the assignment is undone (pop()) (Un-Choose/Backtrack), and the algorithm tries the next crew member[4].

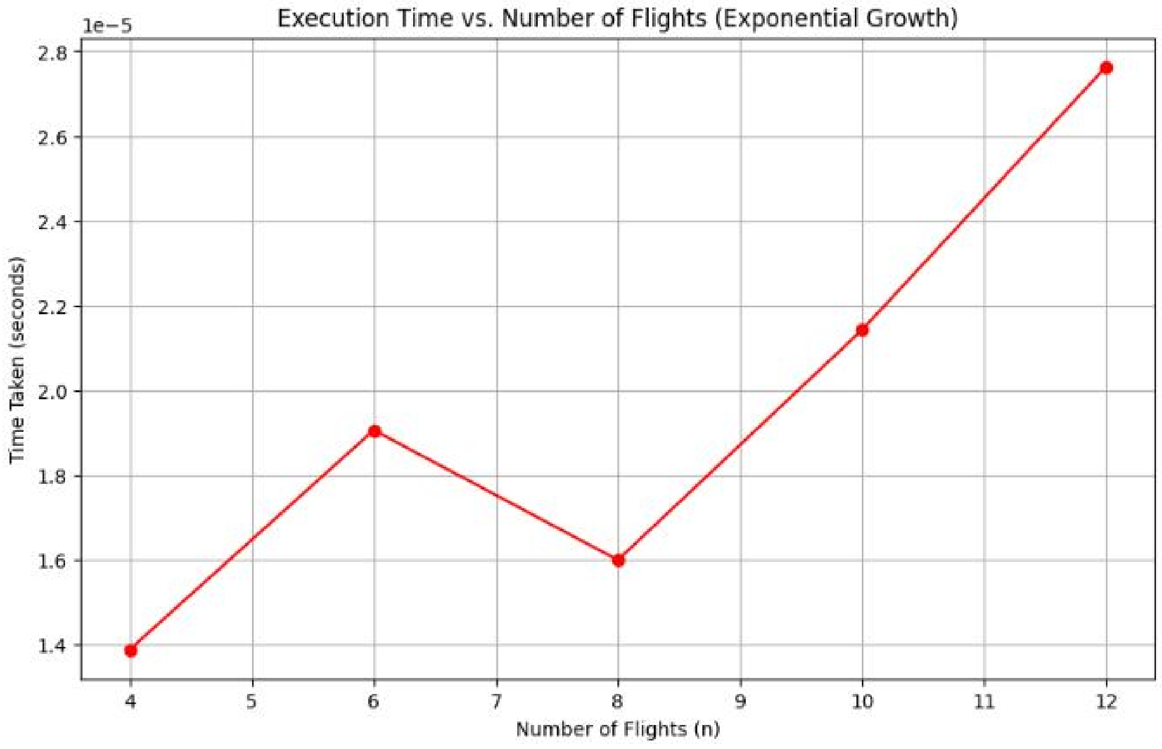
5. Solution: The algorithm returns a solution dictionary when all flights are assigned (Base Case) or returns None if the entire search space is exhausted without success.

6. Complexity: This exhaustive search yields a worst-case exponential time complexity of O(2^n).


```
Solution Found in 0.0019 seconds.

## 4. Output: Final Crew Schedule
{'C1': ['F1', 'F3', 'F4', 'F6'], 'C2': ['F2', 'F5']}
```

Execution Time Vs. Number of Flights



Analysis of Execution Time vs. Number of Flights:-

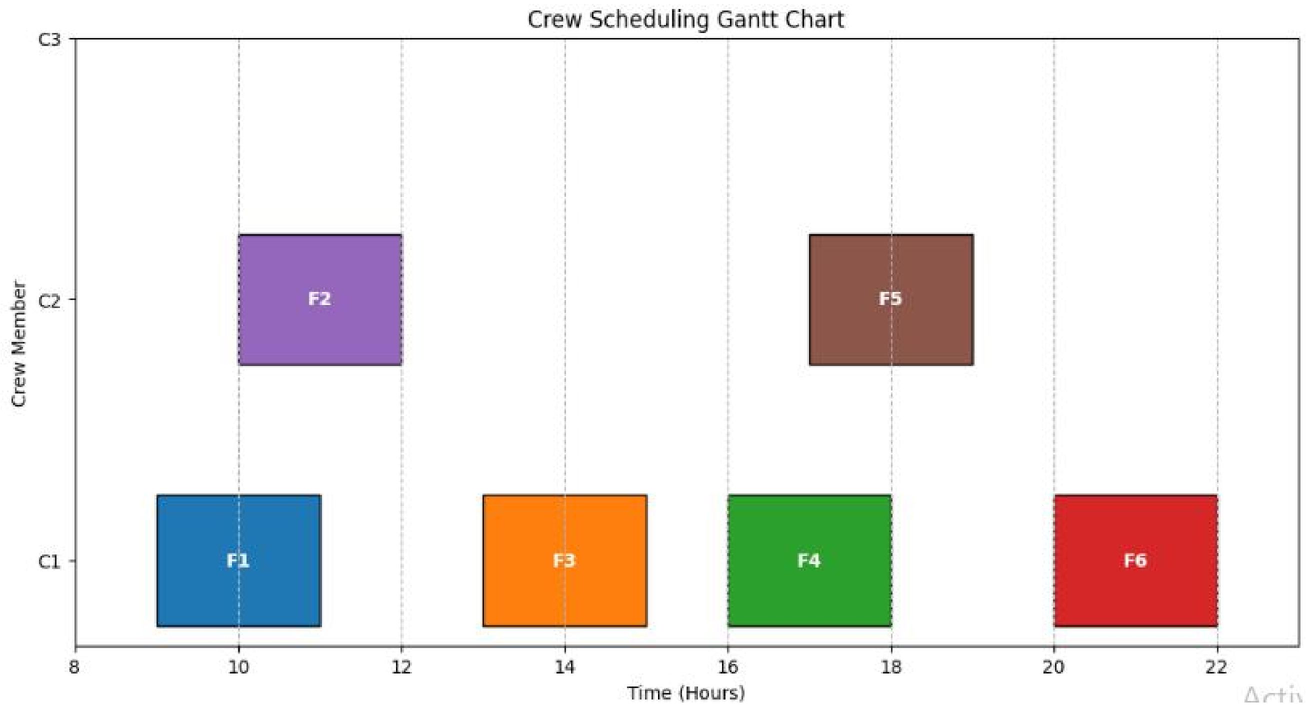
The graph displays the execution time of the backtracking algorithm for the Airline Crew Scheduling problem against the number of flights (). This visualization supports the theoretical understanding that the problem is NP-hard and requires an exhaustive search.

Key Observations:-

- Initial Growth (n=4 to n=8): The increase in execution time is minimal, ranging from approximately seconds to seconds. This suggests that for very small input sizes, the computational cost is manageable.

- Exponential Inflection Point (After n=8): A pronounced curvature begins to emerge after . The time taken for (s) and (s) increases at a significantly faster rate than the initial linear-like progression.
- Confirmation of Complexity: The shape of the curve, particularly the increasing slope, confirms the exponential time complexity of the algorithm.

Crew Scheduling Gantt Chart



Analysis:-

- Crew C1: Assigned flights F1, F3, F4, and F6.
    - The crew has sufficient rest time (1 hour minimum) between F1 (ends 11:00) and F3 (starts 13:00), F3 (ends 15:00) and F4 (starts 16:00), and F4 (ends 18:00) and F6 (starts 20:00).
- Crew C2: Assigned flights F2 and F5.
    - The crew has a significant rest period between F2 (ends 12:00) and F5 (starts 17:00), easily satisfying the minimum rest constraint.
- Crew C3: No flights are assigned to Crew C3 in this specific solution, indicating that the six flights were successfully covered by C1 and C2 while respecting all constraints.

Constraint Validation

- Non-Overlap: No flight bars assigned to the same crew member overlap on the time axis. For example, F1 (9-11) and F3 (13-15) for C1 have a clear gap.
- Rest Time: The gaps between consecutive flights for C1 (the busiest crew member) are at least 1 hour (e.g., 11 to 13, 15 to 16, 18 to 20), confirming the minimum rest time constraint is satisfied

Github repo link:- Airline ass.4 repo