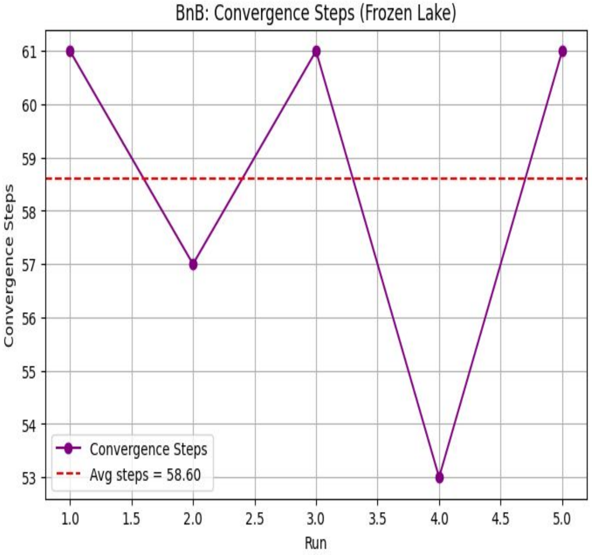
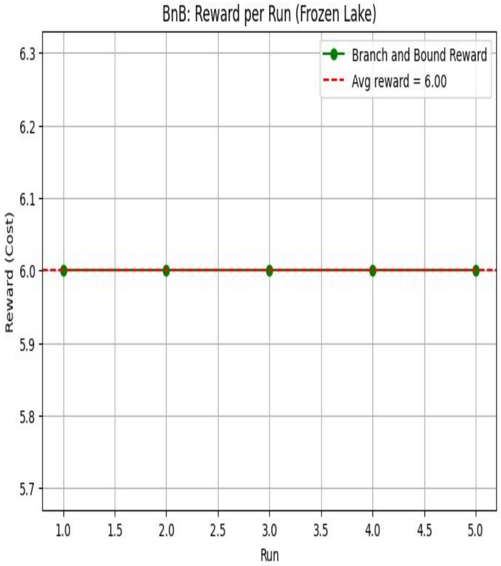
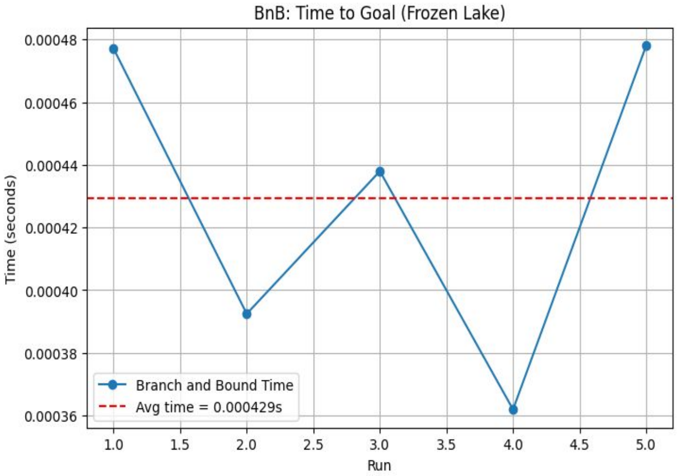
Assignment 2
Search and Optimization

CS24M109 - Pashaula Eswar Sai
CS24M123 - Gooty Bharadwaj

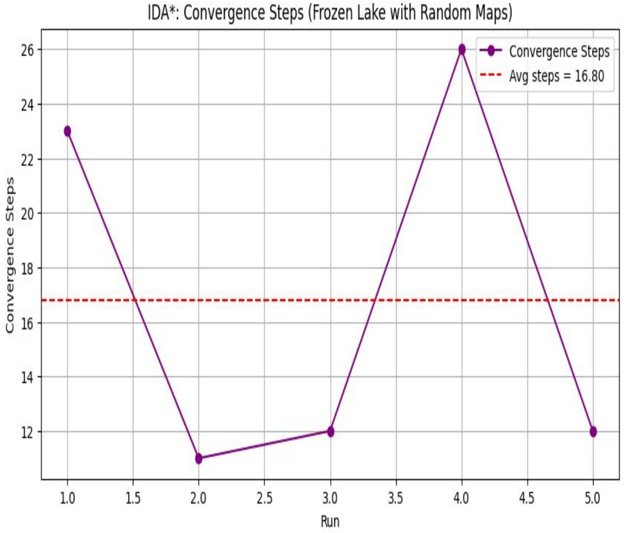
Branch and Bound

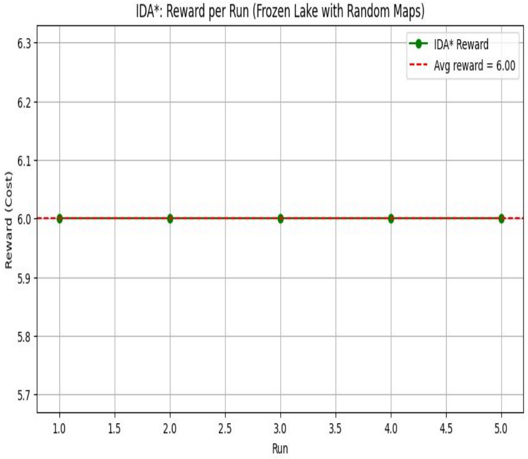


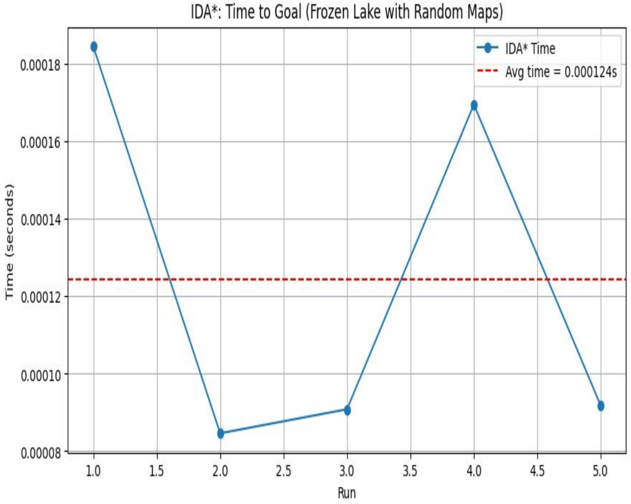




IDA*


IDA*: Convergence Steps (Frozen Lake with Random Maps)
Convergence Steps
Avg steps = 16.80
Convergence Steps
Run
26
24
22
20
18
16
14
12
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


IDA*: Reward per Run (Frozen Lake with Random Maps)
IDA* Reward
Avg reward = 6.00
Reward (Cost)
Run
6.3
6.2
6.1
6.0
5.9
5.8
5.7
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0


IDA*: Time to Goal (Frozen Lake with Random Maps)
IDA* Time
Avg time = 0.000124s
Time (seconds)
Run
0.00018
0.00016
0.00014
0.00012
0.00010
0.00008
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5
5.0

Heuristic Function

For BnB and IDA* the Manhattan distance:

```python
# Heuristic function: Manhattan Distance
def heuristic(state, goal, size=4):
    x1, y1 = state // size, state % size
    x2, y2 = goal // size, goal % size
    return abs(x1 - x2) + abs(y1 - y2)
```

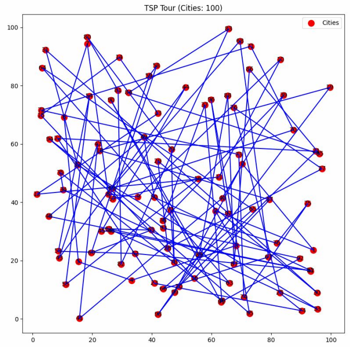
For Hill Climbing and Simulated Annealing

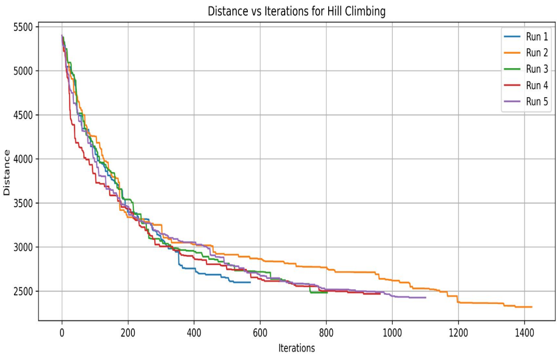
```python
def total_distance(tour, cities):
    return sum(np.linalg.norm(cities[tour[i]] - cities[tour[i - 1]]) for i in range(len(tour)))
```

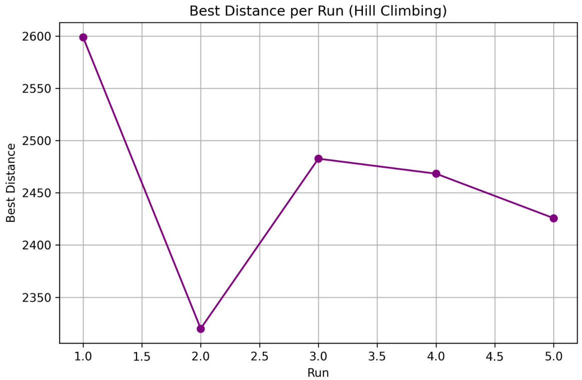
Observations for BnB and IDA*

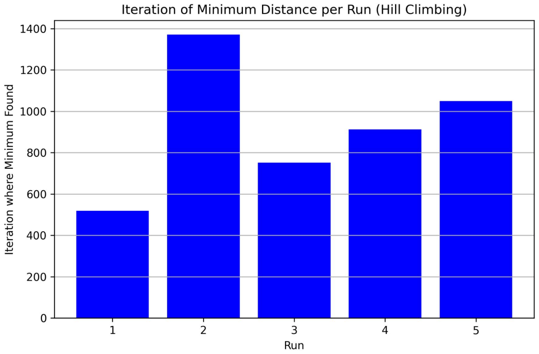
- Both IDA* and BnB were run on Frozen Lake for a fixed start and goal point while changing the positions of obstacles in the path.
- IDA* performed ~3.5x faster than BnB on average. This happens because IDA* leverages heuristic-guided along with depth limits pruning large parts of the search space early.
- It avoids exploring full subtrees that exceed the thresholds.
- BnB, while using cost-based pruning, still explores more suboptimal branches before updating the best path.
- Both algorithms are complete and optimal for the frozen lake problem.
- IDA* is better suited for Frozen Lake compared to BnB. BnB has overhead in terms of recursion and storage because of keeping track of visited states and costs
- Both algorithms found optimal solutions in all runs (Cost=6)

Hill Climbing for Travelling Salesman









1. **Consistent downward trend** – Hill climbing always chooses better solutions, leading to steady improvement.

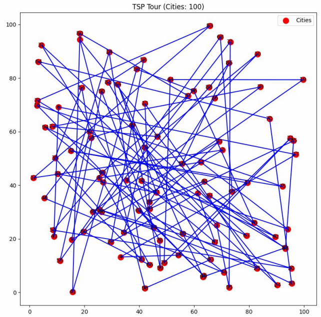
2. **No sudden jumps** – Only better neighbors are accepted, so there are no sharp distance drops.

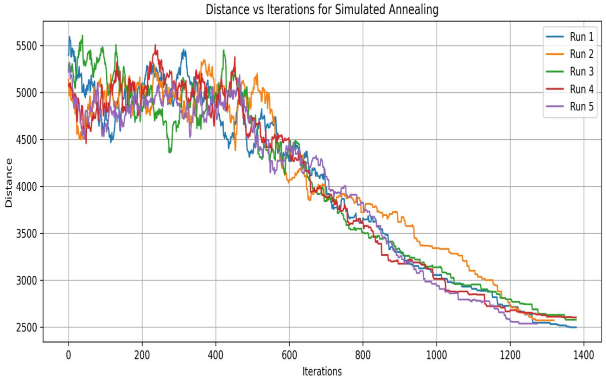
3. **Early improvements** – Most of the improvement happens in the initial iterations when better neighbors are abundant.

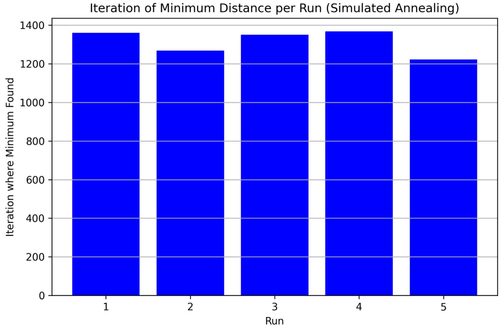
4. **Plateaus/stagnation** – Some runs get stuck in local minima, leading to flat lines (no improvement).

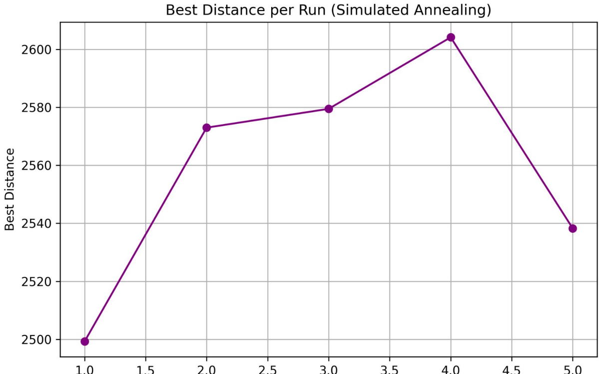
5. **Different end values** – The final distance varies due to different starting points and local optima.

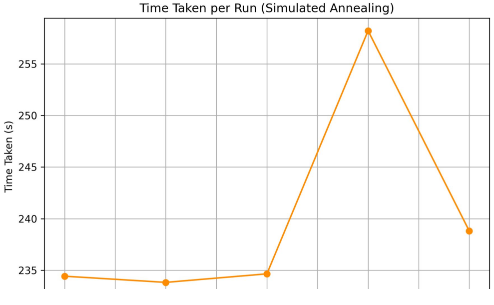
Simulated Annealing











1. **High fluctuations early** – Simulated annealing accepts worse solutions at high temperatures to escape local minima.

2. **Smoother drop later** – As the temperature decreases, it becomes more selective, accepting only better or similar solutions.

3. **Sudden drops** – Occasionally, a significantly better solution is found, leading to sharp improvements.

4. **Different early paths** – Random initial solutions cause each run to explore a different trajectory at the beginning.

5. **Similar final values** – Despite randomness, the algorithm converges towards a near-optimal solution in all runs.