AI assignment - 2

SEARCH & OPTIMIZATION

Atreya M - CS22B009
Bhukya Lachiram Nayak - CS22B012

Frozen Lake - Environment

In Frozen Lake

- Each cell is a state.
- Actions = {Up, Down, Left, Right}.
- Transitions are deterministic (non-slippery).
- Goal is to find the **shortest path to the goal**.

1.BRANCH & BOUND

Branch and Bound is a best-first search algorithm that systematically explores paths by expanding the most promising nodes first—those with the lowest estimated total cost ($\mathrm{cost\_so\_far} + \mathrm{heuristic}$).

How it Works

1. **Start from initial state** (S).
2. Maintain a **priority queue** (min-heap) of paths, prioritized by their cost + heuristic.
3. At each step:
   - Pop the **path with the lowest priority**.
   - If it reaches the **goal (**G**)**, return it.
   - Otherwise, **expand** the path by moving in all possible directions (up/down/left/right).
   - Add each new path to the queue if the resulting state hasn't been visited.
4. **Repeat** until goal is found or queue is empty

$\mathrm{cost\_so\_far}$: Number of steps taken.

$\mathrm{heuristic}$: Estimated cost to goal (we use **Manhattan distance**).

Advantages

- Explores only the best candidate paths first.
- Finds **optimal** solution if heuristic is admissible .

Disadvantages

- May explore many unnecessary paths if heuristic is weak.
- Memory intensive (stores many partial paths).

BnB

Iterative Deepening A*

IDA* combines **depth-first search (DFS)** with the cost-based pruning of **A***. It performs a series of **depth-limited searches**, increasing the cost bound at each iteration.

HOW IT WORKS:

- **Set an initial cost bound** using: bound = heuristic(start)
- Run a **DFS**, only exploring paths where: cost_so_far + heuristic ≤ bound
- If the goal is found, return it
- If not found, **increase the bound** to the minimum value that exceeded the last one and **repeat**.

Process

- Each iteration is a DFS up to a given cost threshold.
- The threshold is incrementally increased (hence "iterative deepening").

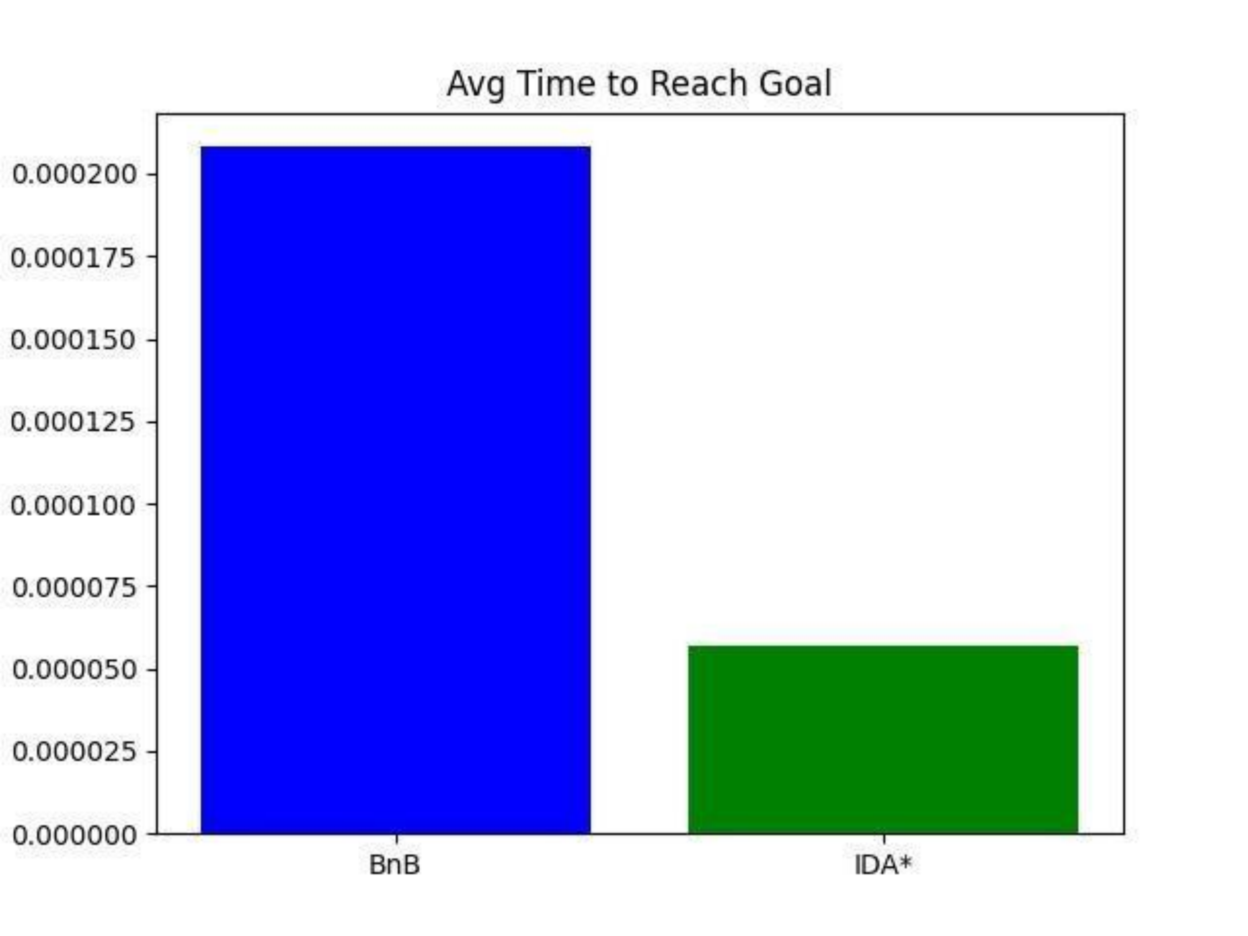
Advantages

- **Memory efficient** (like DFS).
- **Optimal solution** .
- Works well in large state spaces.

Disadvantages

- **Slower than A*** due to repeated work.
- Requires careful tuning to avoid recomputation overhead.

IDA*


Avg Time to Reach Goal
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
0.000000
BnB
IDA*

What’s the key difference with BnB and IDA*?

BnB tries to explore all possibilities but skips branches that are worse than the best solution found so far.

IDA* combines the benefits of A* (heuristics) with iterative deepening to explore efficiently.

Even with pruning, BnB often has to **explore a large part of the search space**, especially early on. If a better solution is found late, earlier pruning isn’t very effective.

Traveling Salesman Problem - Environment

Experiment Setup

- **Environment**: 20 random 2D points (mk_env).
- **Objective**: Minimize total path length (TSP).
- **Evaluation**:
    - Each algorithm runs **5 times**.
    - Average cost is calculated and visualized.
    - Final tour is saved as a plot (hc_tour.png, sa_tour.png).

Generating the cities using np.random.rand() , and we can compute distances between cities using numpy op

3.HILL CLIMBING

Hill Climbing is a greedy local search algorithm. It continuously moves towards a better solution by exploring its neighbors, hoping to reach the optimal solution.

How It Works

1. Start with a random tour (a random permutation of points).
2. Repeatedly:
    - Pick two indices $i < j$.
    - Reverse the segment between i and j (a 2-opt move).
    - If the new tour has a lower cost, accept it.
    - Else, revert the change.
3. Run this for a fixed time (t seconds in the code).

Cost Function: The total distance of the tour

Advantages of Hill Climbing

1. Simple to implement
2. Fast and efficient
3. Uses little memory

Disadvantages of Hill Climbing

1. Can get stuck in local optima
2. No backtracking
3. Not good for complex landscapes

Simulated Annealing

Simulated Annealing is an advanced local search algorithm inspired by the annealing process in metallurgy. Unlike hill climbing, SA accepts worse solutions with a certain probability to escape local minima.

How It Works

1. Start with a **random solution** .
2. Initialize **temperature** T
3. Repeat until time expires or T is very small:
    - Choose two points $i < j$ and reverse the segment (2-opt move).
    - Compute **cost difference** $d = new\_cost - old\_cost$.
    - Accept the move if:
        - $d < 0$ (better solution), or
        - $\exp(-d/T) > random()$ (probabilistic acceptance).
    - Gradually **decrease temperature**: T *= $cool\_rate$ (like 0.995).

Temperature Schedule

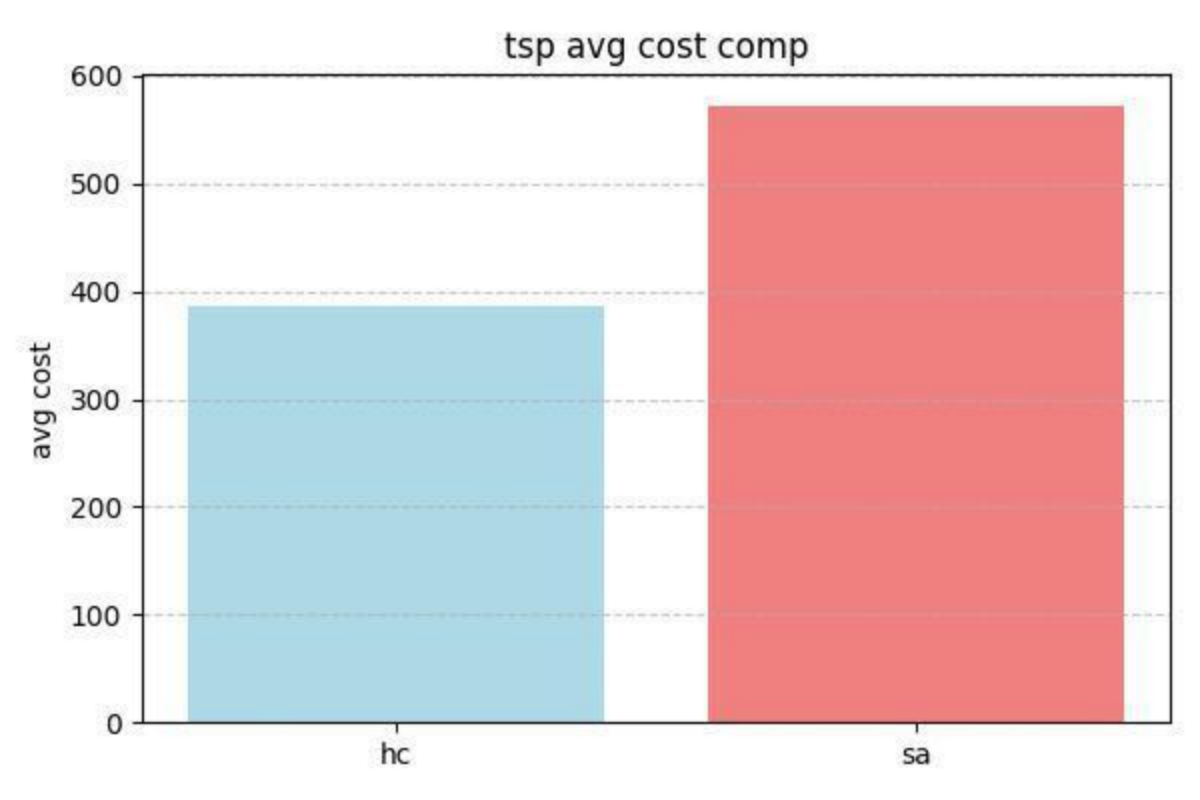
- Starts high to allow exploration (accepting worse paths).
- Gets stricter over time (low temperature = greedy).

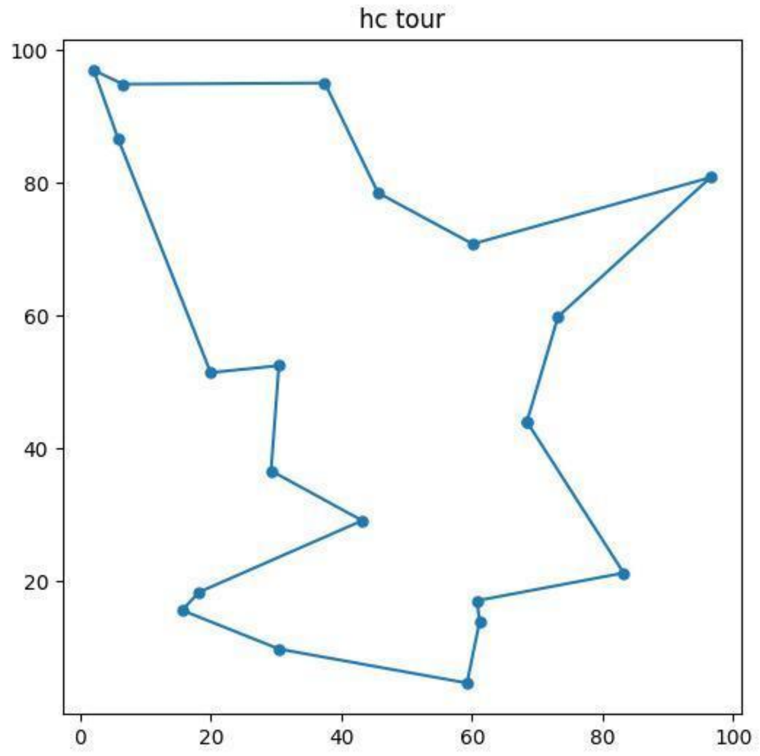
Advantages

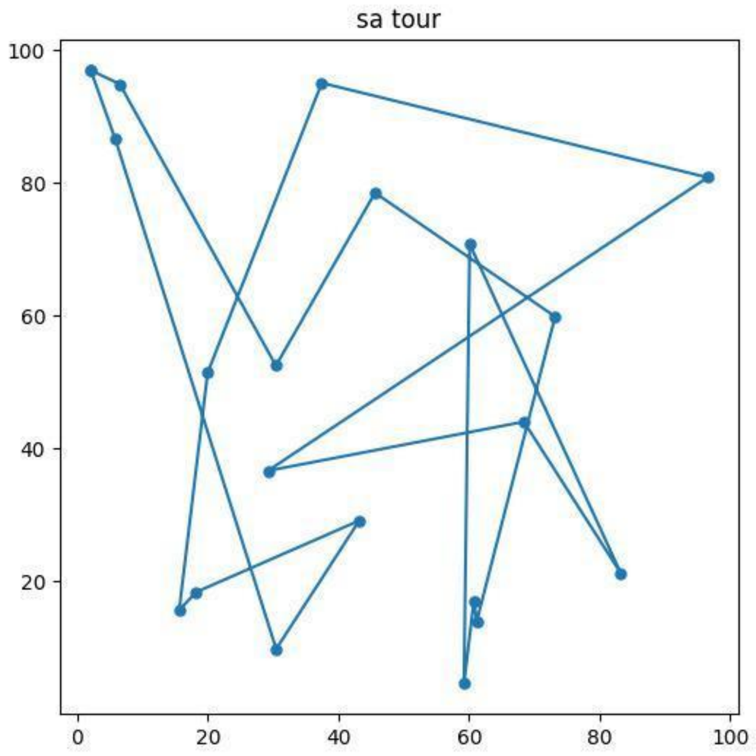
- Escapes local minima.
- Can find **global optima** over time.

Disadvantages

- Slower than hill climbing.
- Requires careful tuning (initial temperature, cooling rate).


tsp avg cost comp
avg cost
600
500
400
300
200
100
0
hc
sa


hc tour
0
20
40
60
80
100
0
20
40
60
80
100


sa tour
0
20
40
60
80
100
0
20
40
60
80
100