AI PROJECT PROPOSAL

Solving Problem Using AI Search Algorithms

• The Problem Is :

the Traveling Salesman Problem (TSP)

• Introduction:

The goal of this project is to explore and apply fundamental Artificial Intelligence search strategies to solve one of the most well-known optimization problems: the **Traveling Salesman Problem (TSP)**.
This project allows hands-on implementation, experimentation, and evaluation of multiple AI search techniques, giving a deeper understanding of algorithmic behavior, performance tradeoffs, and heuristic design.

The project focuses on building a complete solution framework, analyzing algorithm efficiency, and comparing how different search strategies perform when attempting to minimize the total path cost between multiple cities.

• Problem Description:

The **TSP** asks the following question:

"Given a set of cities and the distances between them, what is the shortest possible route that visits each city exactly once and returns to the starting point?"

TSP is a classic **NP-hard optimization problem**, widely used in logistics, robotics path planning, transportation networks, manufacturing, and clustering analysis.

Input

- A graph of cities (nodes)
- Distances between each pair of cities (edges)

Output

- A complete tour visiting all cities once
- Minimum total travel cost

• Why TSP ?

- It is a perfect benchmark for search algorithms
- It demonstrates the difference between optimal search and heuristic search
- It clearly shows strengths/weaknesses in time, space, and optimalit

Algorithms Implemented

Uniform-Cost Search(UCS) **A* Search** **Hill Climbing Search** **Nearest Neighbor + 2-opt** **Genetic Algorithm (GA)**

1) Uniform-Cost Search (UCS)

UCS is an uninformed search that expands nodes based on the lowest cumulative path cost.

Why UCS fits TSP:

- Guarantees optimal solutions
- Works well for weighted graphs
- Serves as a baseline to compare other search techniques

Expected behavior:

- Very slow for large number of cities
- Extremely accurate (optimal)

The Part For → Eng : Shahd Mohamed Hamed Abdelrahman

---

2)A* Search

A* is an informed search algorithm combining path cost + heuristic estimate:

f(n) = g(n) + h(n)

Chosen heuristic for TSP:

- **Straight-line (Euclidean) distance** between the current city and nearest unvisited city
- Admissible and intuitive

Why A* fits TSP:

- Faster than UCS
- Reduces search space dramatically
- Still capable of reaching optimal or near-optimal solutions

The Part For → Eng : Omar Mohamed Ibrahim Badawy

---

3)Hill Climbing Search

A local optimization algorithm that iteratively improves the current solution by making small modifications.

Why Hill Climbing fits TSP:

- Very fast for large numbers of cities
- Produces good (not perfect) solutions
- Demonstrates the idea of local minima, plateaus, and heuristic landscapes

Enhancements used:

- Random-restart Hill Climbing
- Swap-based neighbor generation

The Part For → Eng : Noor Hussain Mwafi

---

4) Nearest Neighbor + 2-opt

- **Nearest Neighbor:** A greedy constructive algorithm that starts at an arbitrary city and always moves to the nearest unvisited city until all cities are included in the tour.
- **2-opt:** A local search operator that improves the initial tour by repeatedly selecting two edges in the tour and swapping their endpoints. This removes the "crossover" edges, often resulting in a shorter path.

Why Nearest Neighbor + 2-opt fits TSP:

- **Speed and Efficiency:** Nearest Neighbor is{O}(N^2), providing an excellent starting point quickly.
- **Practicality:** Demonstrates a common real-world approach where a decent, fast solution .

Expected Behavior:

- **Very fast execution**.
- **Near-optimal solutions** (better than simple Hill Climbing alone, due to the structured starting point and effective neighborhood search)

The Part For → Eng : Nour Yasser Hashem El-Sheikh

---

5) Genetic Algorithm (GA)

- **Core Principle:** GA maintains a "population" of potential solutions (tours/chromosomes) and iteratively applies **Selection**, **Crossover** (recombination), and **Mutation** operators to evolve better solutions over generations.

Why Genetic Algorithm (GA) fits TSP:

- **Global Search Capability:** Unlike local search (Hill Climbing), GA explores the entire solution space, making it less susceptible to getting stuck in local optima.
- **High Scalability:** Well-suited for very large TSP instances (N > 100) where other methods are too slow or memory intensive[6].
- **Demonstrates Metaheuristics:** Showcases an alternative class of AI optimization techniques based on nature-inspired concepts[7].

Expected Behavior:

- **High quality (near-optimal) solutions** for large N.
- **Convergence demonstrated** over generations, illustrating the evolutionary process

The Part For → Eng : Hend Mohamed Mohamed Fiala

• Evaluation & Expected Results

During experimentation we will evaluate each algorithm based on:

Performance Metrics

- **Execution time**
- **Memory usage**
- **Path cost (tour length)**
- **Optimality vs. speed**
- **Scalability with number of cities**

What we expect

- **UCS:** Best optimal result, but slowest
- **A*:** Fast + near-optimal, depending on heuristic strength
- **Hill Climbing:** Very fast but not always optimal

This comparison highlights the difference between uninformed search, heuristic search, and local optimization.

Algorithm	Expected Solution Quality	Expected Speed	Primary Role/Insight
Uniform-Cost Search (UCS)	**Optimal** (Guaranteed)	**Slowest** (Very slow for large N)	Baseline for optimal comparison
A* Search	**Optimal / Near-Optimal**	**Faster than UCS**	Demonstrates the power of heuristics
Hill Climbing	**Good (Local Optimum)**	**Very Fast**	Highlights local search and local minima problems
Nearest Neighbor + 2-opt	**Very Good (Strong Local Optimum)**	**Extremely Fast**	Practicality; provides a high-quality initial tour quickly.
Genetic Algorithm (GA)	**Near-Optimal (Global Search)**	**Moderate to Slow** (Due to large N focus)	Demonstrates metaheuristics and evolutionary search for large problems.

Deliverables

The final submission will be through a single Public GitHub Repository link, which will contain the following components:

1. **Source Code** for all five algorithms (UCS, A*, Hill Climbing, Nearest Neighbor + 2-opt, GA).
2. A `README.md` file providing clear instructions on how to set up, run, and test the project.
3. A detailed **PDF Report** including implementation specifics, comprehensive results, comparison tables, and visual analysis of the performance metrics across all algorithms.