A hand is shown interacting with a 3x3 sliding puzzle board. The board has a light beige background with a dark brown border. It contains nine squares. Some squares have small circles in them, and others have small U-shaped pieces. The hand is positioned over the bottom-left square.

Solving the 8-Puzzle Problem Using AI Algorithms

Team members

Abdelrhman Aiman Abdelgamil Mostafa Elbarawy.

Eslam Osama Ebrahim Elfadaly.

Khaled Fawzy Mahmoud Elgazzar.

Abdelrhman Elsaid Mahmoud Fahmy Mohamed Raslan.

Ahmed Mohamed Ahmed Grida.

Anas Ashraf Abdelhamid Elsaid Elnagar.

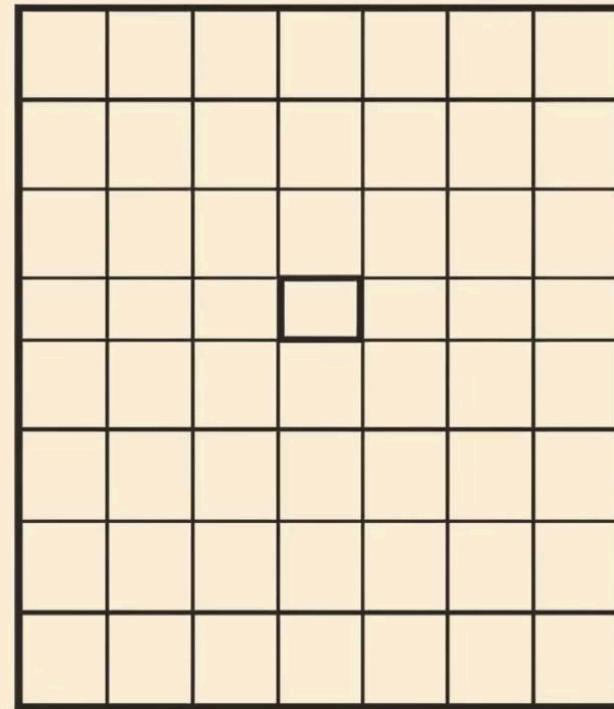
Omar Mohamed Mansour Abdelqader Wahdan.

Zyad Ehab Talaat Abdelrahim.

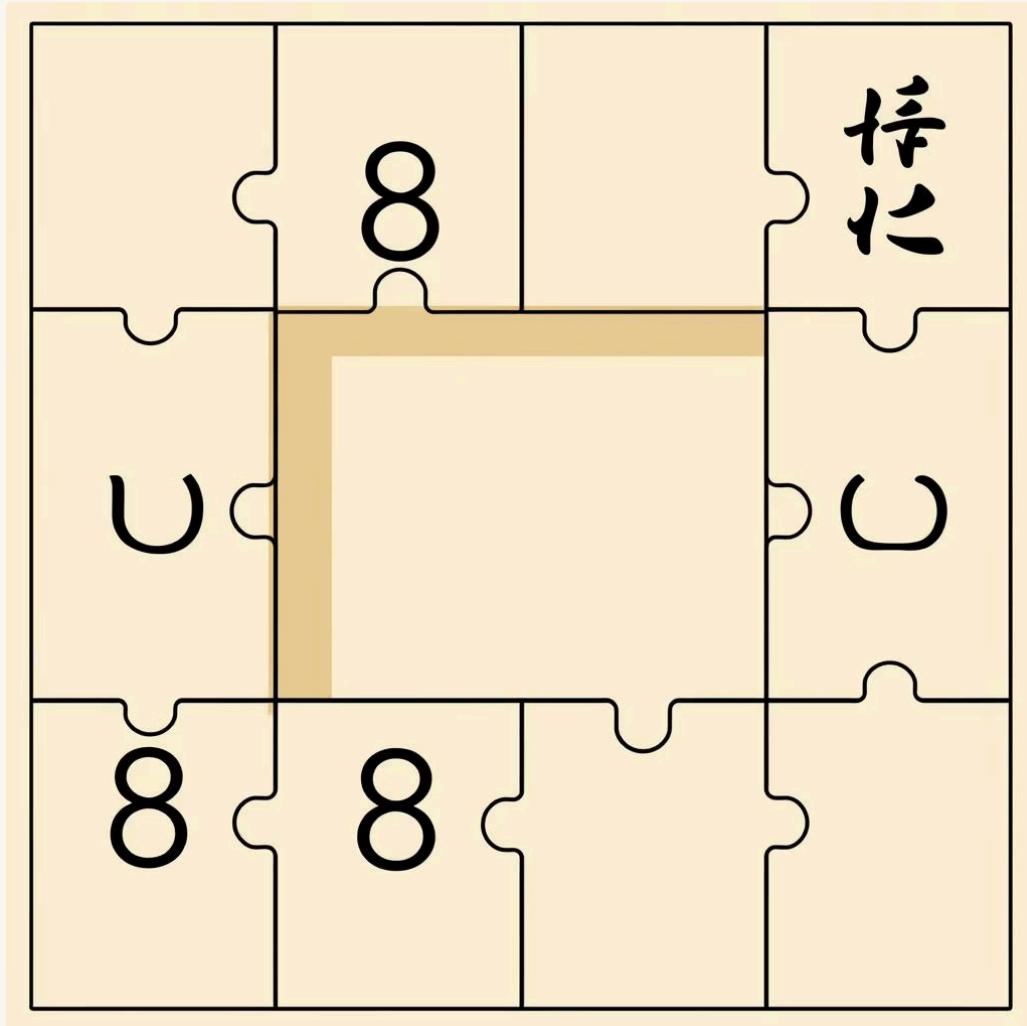
What is the 8-Puzzle Problem?

The 8-Puzzle is a sliding puzzle game invented by Noyes Palmer Chapman in the 1870s. It consists of a 3×3 grid with eight numbered tiles and one blank space. The objective is to slide the tiles until they reach a specific goal configuration.

This simple yet challenging problem serves as an excellent testbed for various Artificial Intelligence search algorithms, allowing us to compare their performance, optimality, and efficiency.



Rules of the 8-Puzzle



→ **Limited Movement**

Only tiles adjacent to the blank space can be moved into it.

→ **Directional Moves**

Allowed moves include Up, Down, Left, and Right.

→ **Defined Goal State**

The ultimate goal is to arrange the tiles in ascending order, with the blank space in the bottom-right corner.

Representing the Problem in AI

1

States

Each unique arrangement of tiles on the 3x3 grid represents a distinct state in the problem space.

2

Actions

The legal moves (sliding a tile into the blank space) are defined as the actions that transition between states.

3

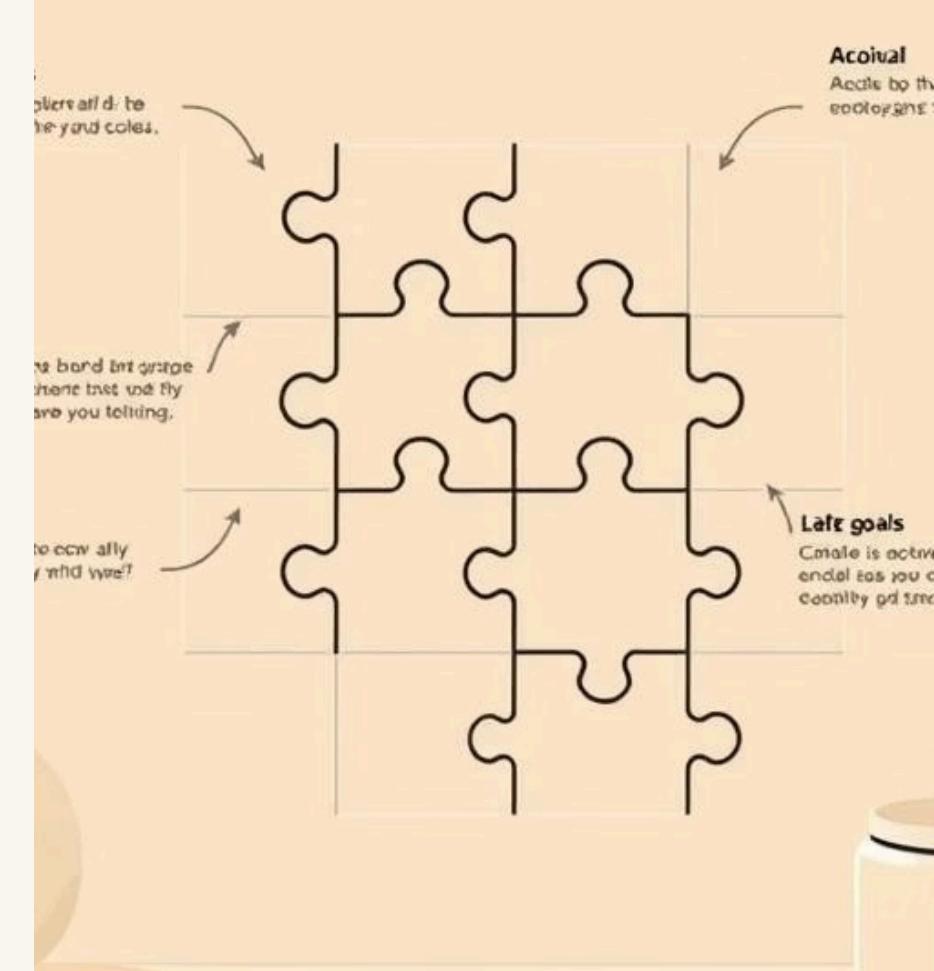
Goal Test

A goal test function continuously checks if the current tile configuration matches the desired target state.

4

Cost

Each move typically incurs a uniform cost, usually set to 1, simplifying path evaluation.



Search Algorithms to Solve the Puzzle

Breadth-First Search (BFS)

Explores the state space **level by level**.

- **Guarantees the shortest path** when all moves have equal cost.
- **Disadvantage:** Requires large memory for complex problems.

Depth-First Search (DFS)

Prioritizes exploring deep paths first. It's **memory-efficient** but might not find the shortest solution and can get stuck in infinite loops.

Uniform Cost Search (UCS)

Explores states based on the lowest cumulative path cost. It guarantees finding the optimal solution when all costs are positive, but it can be slower than BFS when many nodes have similar costs.

A* Search

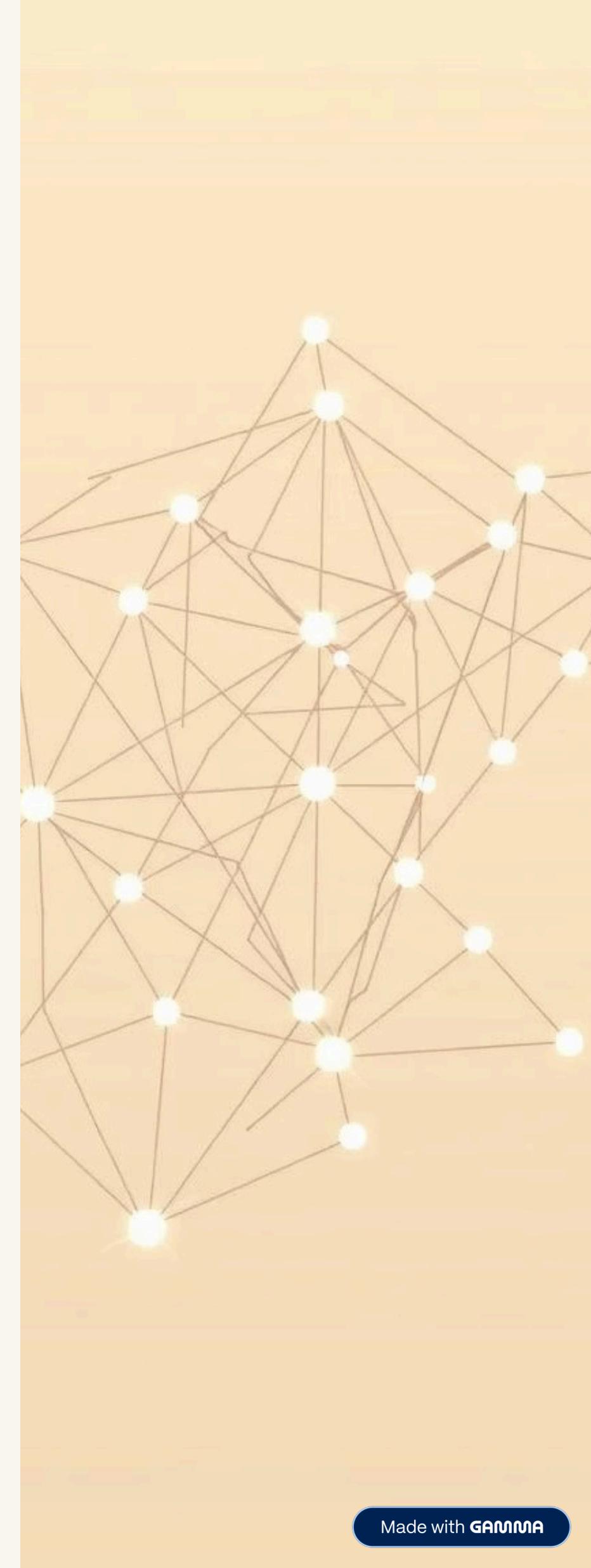
Combines path cost and heuristic evaluation to choose the most promising states. It guarantees an optimal solution when the heuristic is admissible, making it one of the most effective search algorithms.

Hill Climbing

A local search algorithm that continuously moves toward states with better heuristic values. It is simple and fast but can get stuck in local optima or plateaus, failing to reach the global solution.

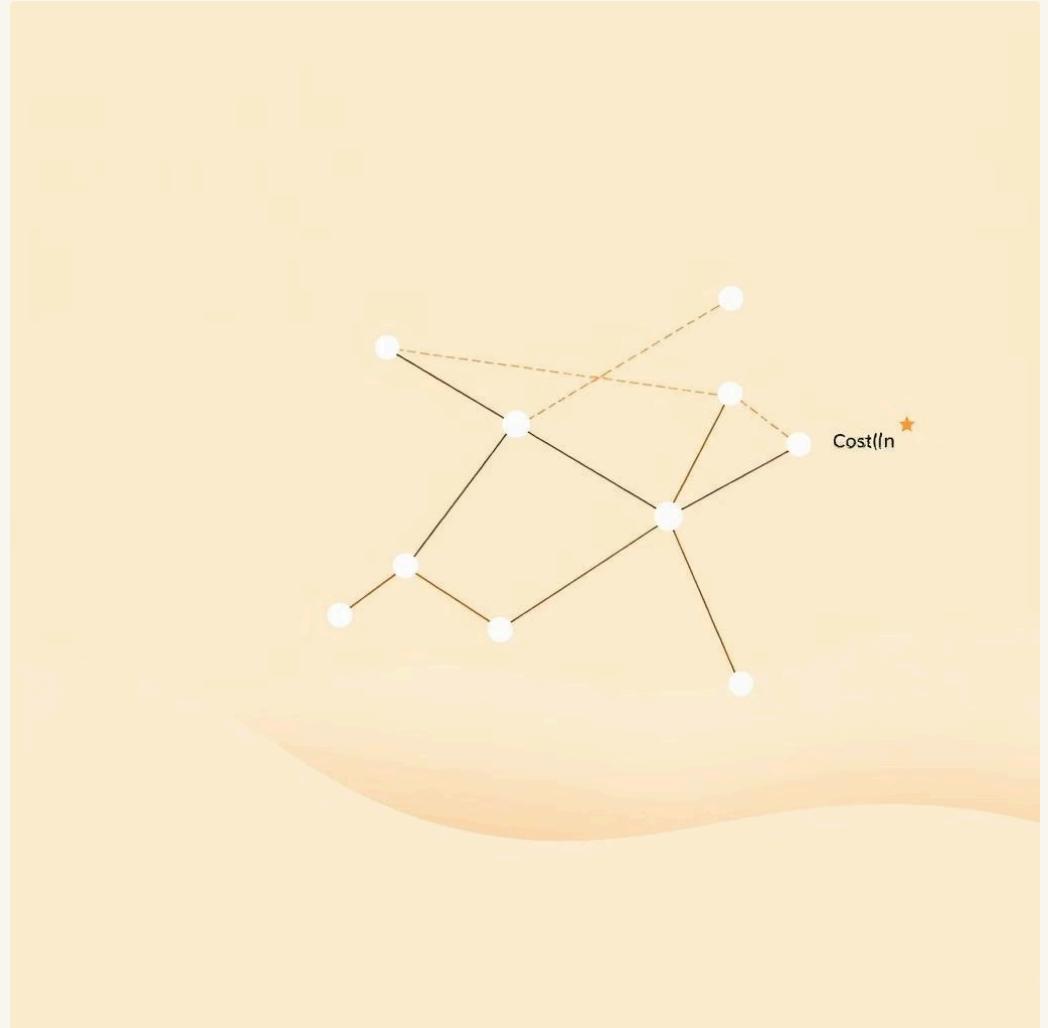
Greedy Best-First Search

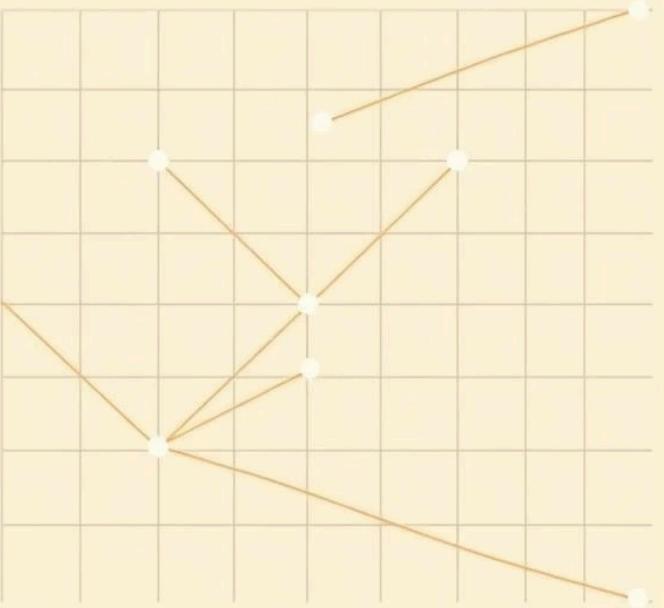
Selects states based solely on a heuristic function that estimates how close the current state is to the goal. It is fast and memory-efficient but does not guarantee finding the optimal solution.



Performance Comparison

Algorithm	Time	Space	Optimal	Suitable for 8-Puzzle
BFS	High	High	Yes	Not Suitable
DFS	High	Low	No	Not Suitable
UCS	High	High	Yes	Partially Suitable
IDS	Medium	Low	Yes	Partially Suitable
A*	Low	High	Yes	Highly Suitable
Hill Climbing	Fast	Very Low	No	Not Suitable
Genetic	Variable	Medium	No	Limited Suitable
Greedy BestFirst Search	Very Fast	Low	No	Suitable





Conclusion

- Informed search performs better
- A* is the best overall
- UCS guarantees optimality
- Different algorithms → different trade-offs