Problem Solving and Search Algorithms

Er. Rudra Nepal

November 18, 2024

Introduction to Problem Solving in AI

Problem solving is central to artificial intelligence. It involves finding solutions to specific challenges or tasks by exploring various possible solutions.

- A **problem** consists of an initial state, a goal state, and actions that transform one state into another.
- AI agents use problem-solving techniques, such as search algorithms, to navigate through possible states to reach the goal state.
- Problem solving in AI is often seen as a search through a problem space, using algorithms to explore different possibilities.

Problem Solving Agents

A **problem-solving agent** is an intelligent agent designed to solve problems by searching through a set of possible states.

- **Agent's goal:** The agent starts from an initial state and attempts to reach a goal state.
- **Search process:** The agent uses a search algorithm to explore the state space—often represented as a tree or graph of possible actions and states.

Problem Solving Agents (Continued)

- The agent selects actions based on available choices and proceeds toward the goal state.
- **Examples of Problem-Solving Agents:**
  - Games: Chess, Checkers (agents search for optimal moves).
  - Pathfinding: Navigation systems (find the shortest path between locations).
  - Robotics: Robots solving tasks like object picking or navigation.
- Problem-solving agents are essential in many AI applications such as decision-making, game playing, and autonomous systems.

State Space Search

A state space consists of:

- A set of nodes representing each state of the problem.
- Arcs between nodes representing legal moves from one state to another.
- An initial state and a goal state.

The state space can take the form of:

- A tree, or A graph.

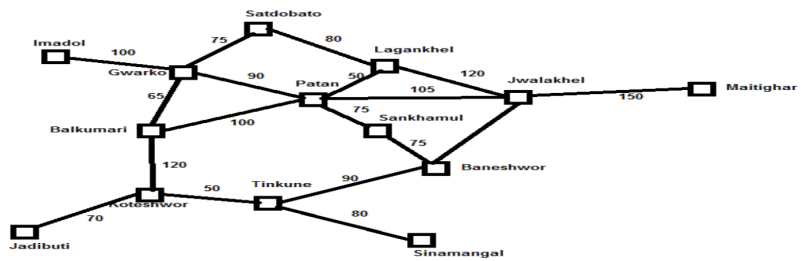


Figure: Example of a state space representation

State Space Search and AI

State space search forms the basis of most AI methods because:

- It allows for formal definition of a problem as the need to convert some given situation into a desired situation using a set of permissible operations.
- It provides a framework to define the process of solving a problem using known search techniques.

State Space Representation: Towers of Hanoi

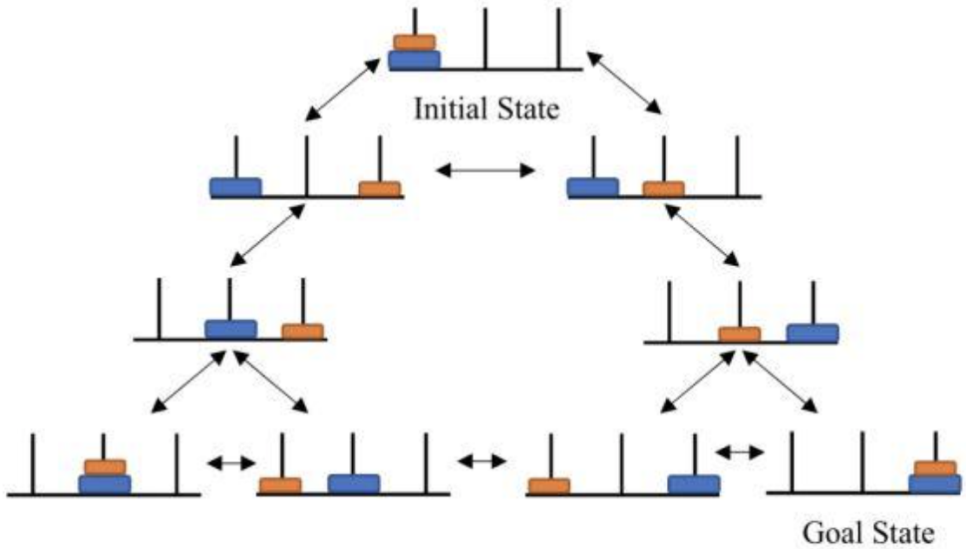


Figure: State space representation of the Towers of Hanoi problem using a graph

The Problem Solving Process

The problem-solving process in AI involves several systematic steps that guide the agent to the solution.

1. **Define the Problem:**
   - Identify the initial state, the goal state, and the allowable actions or operators.
   - Example: In a navigation problem, the initial state is the starting location, the goal is the destination, and actions are the possible moves.
2. **Generate Possible Actions:**
   - Create a set of valid actions that can move the agent from one state to another.
   - Actions are the building blocks for transitioning between states.

The Problem Solving Process (Continued)

3. **Search for a Solution:**
   - Use a search algorithm (e.g., depth-first search, breadth-first search) to explore the state space.
   - The goal is to find a sequence of actions that lead to the goal state.
4. **Evaluate and Choose the Solution:**
   - Once a solution is found, the agent evaluates it to ensure it's optimal (if necessary).
   - The final solution can be implemented to achieve the goal.

Production Systems

A **production system** is a framework used for solving problems, based on a set of production rules and a database (or working memory) of facts.

- **Production rules:** These are condition-action pairs, which specify what should be done under certain conditions.
- **Database (working memory):** The database stores information (facts) about the current state of the problem.
- **Execution:** The system applies a production rule whenever its condition is met, updating the database until the goal state is reached.
- Example: In an expert system for medical diagnosis, rules could be "if the patient has fever and cough, then check for flu."

Well-Defined vs. Ill-Defined Problems

Problems can be categorized into two types based on the clarity of their components: well-defined and ill-defined.

- **Well-Defined Problems:**
  - The problem has a clear initial state, goal state, and set of actions.
  - The solution space is clearly defined.
  - Examples: Solving a puzzle, playing chess, pathfinding in a grid.
  - These problems are easier to solve because they have clearly defined boundaries and rules.

Well-Defined Problem

A problem can be defined formally by four components:

- **Initial State**: The state from where an agent starts.
- **Successor Function**: Describes the possible actions available to an agent. Given a particular state x, a successor function returns a set of $\langle \text{action}, \text{successor} \rangle$ pairs, where each action is one of the legal actions in state x, and each successor is a state that can be reached from x by applying the action.
- **Goal Test**: Determines whether a given state is a goal state.
- **Path Cost**: Assigns a numeric cost to each path.

Together, the initial state and successor function implicitly define the state-space of the problem.

Well-Defined vs. Ill-Defined Problems (Continued)

- **Ill-Defined Problems:**
  - The problem lacks a clear initial state, goal state, or set of actions.
  - Ambiguous goals or constraints.
  - Examples: Writing a novel, improving education, defining a good life.
  - These problems are harder to solve because they involve ambiguity and undefined criteria.

Problem Formulation

Problem formulation is the process of defining a problem in a way that it can be tackled with problem-solving techniques.

- **State Space:** The set of all possible states that can be reached from the initial state by applying any number of actions.
- **Actions:** The set of operations that can be applied to a state to reach a new state.
- **Goal State:** The target or desired state the agent is trying to reach.

Problem Formulation (Continued)

- **Example of Problem Formulation:**
  - In a robot navigation problem, the state space could be a grid map, with actions representing movements (up, down, left, right).
  - The problem can be defined by a starting location (initial state) and a goal location (goal state).
- Proper problem formulation simplifies the complexity of the problem and makes it solvable using algorithms.

Example Problem: 8-Puzzle

The 8-puzzle consists of a 3x3 board with eight numbered tiles and a blank space. A tile adjacent to the blank space can slide into the space. The problem definition is as follows:

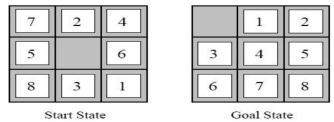
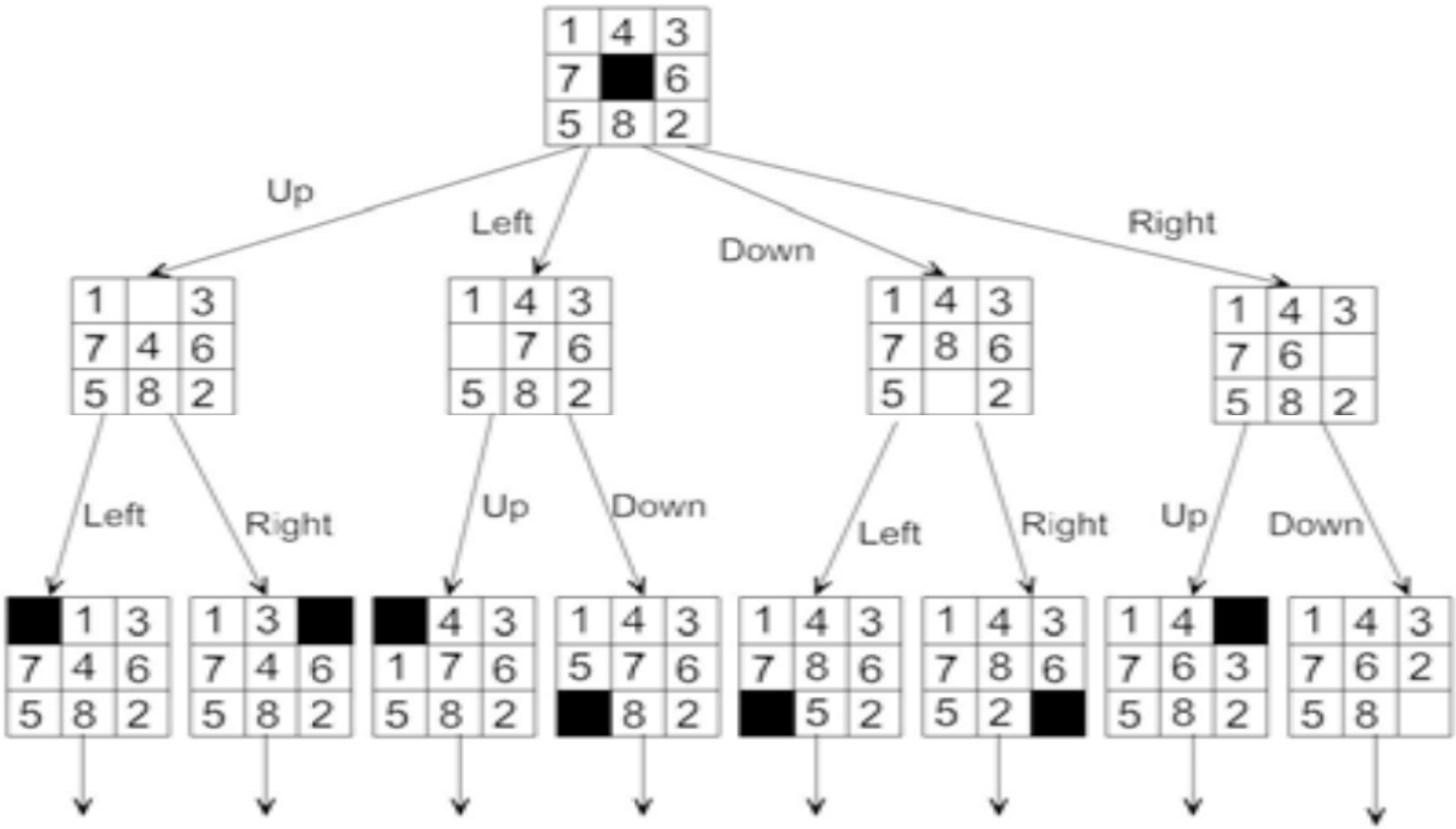


Figure: 8-Puzzle Example

- **States**: Specifies the location of each of the tiles and the blank in one of the nine squares.
- **Initial State**: Any state can be designated as the initial state.
- **Successor Function**: Generates legal states that result from trying the four actions:
  - Blank moves *Left*, *Right*, *Up*, or *Down*.
- **Goal Test**: Checks whether the state matches the goal configuration.
- **Path Cost**: Each step costs 1, so the path cost is the number of steps in the path.

Example Problem: 8-Puzzle

The state space for the 8-puzzle. Arcs denote actions: Blank space Left, Right, Up, Down



Water Jug Problem

You are given two jugs, a 4-gallon one and a 3- gallon one, a pump which has unlimited water which you can use to fill the jug, and the ground on which water may be poured. Neither jug has any measuring markings on it. How can you get exactly 2 gallons of water in the 4-gallon jug?

Water Jug Problem: Solution

The Water Jug Problem can be formulated as follows:

- **States**: Determined by the amount of water in each jug.
- **State Representation**: Two real-valued variables, J_3 and J_4, indicating the amount of water in the two jugs, with the constraints:

$$0 \leq J_3 \leq 3, \quad 0 \leq J_4 \leq 4$$

- **Initial State**: $J_3 = 0, J_4 = 0$
- **Goal State**: $J_4 = 2$

Water Jug Problem: Actions/Rules

The following actions (rules) are available in the Water Jug Problem:

Action/Rule	**Precondition**	**Effect**
F4	$J_4 < 4$	$J_4 = 4$ (Fill jug 4 to full)
F3	$J_3 < 3$	$J_3 = 3$ (Fill jug 3 to full)
E4	$J_4 > 0$	$J_4 = 0$ (Empty jug 4)
E3	$J_3 > 0$	$J_3 = 0$ (Empty jug 3)
P4-3	$J_3 < 3$	$J_3 = 3$ (Pour water from jug 4 into jug 3 until jug 3 is full)
P3-4	$J_4 < 4$	$J_4 = 4$ (Pour water from jug 3 into jug 4 until jug 4 is full)

Water Jug Problem: Solution

State Space Representation

- Done in Class

Farmer, Goat, Wolf, and Cabbage Problem

A farmer needs to get a goat, a wolf, and a cabbage across a river. He has a rowboat that can carry himself and one item at a time. The problem conditions are:

- If the wolf and goat are left together, the wolf will eat the goat.
- If the goat and cabbage are left together, the goat will eat the cabbage.

Formulate this problem as a state-space search problem and find the solution.

Farmer, Goat, Wolf, and Cabbage Problem: Solution

Problem Formulation

- **States**: Determined by the position of the farmer, goat, wolf, and cabbage on each side of the river.
- **State Representation**: Four variables, F, G, W, and C, representing the positions of the farmer, goat, wolf, and cabbage respectively.
- **Constraints**:
  - Goat and cabbage (GC) cannot be left together alone.
  - Wolf and goat (WG) cannot be left together alone.
- **Initial State**: [West: FGWC, East: -]
- **Goal State**: [West: -, East: FGWC]

Actions/Operators/Rules

The actions that can be performed in the Farmer, Goat, Wolf, and Cabbage problem are as follows:

Action/Rule	Description
F_{w-e}	Farmer alone move from west side to east
F_{e-w}	Farmer alone move from east side to west
FG_{w-e}	Farmer and goat move from west side to east
FG_{e-w}	Farmer and goat move from east side to west
FW_{w-e}	Farmer and wolf move from west side to east
FW_{e-w}	Farmer and wolf move from east side to west
FC_{w-e}	Farmer and cabbage move from west side to east
FC_{e-w}	Farmer and cabbage move from east side to west

Farmer, Goat, Wolf, and Cabbage problem

State Space Representation

- Done in Class

Missionaries and Cannibals Problem

Three missionaries and three cannibals must cross a river using a boat which can carry at most two people, under the constraint that, for both banks and the boat, if there are missionaries present on the bank (or the boat), they cannot be outnumbered by cannibals (if they were, the cannibals would eat the missionaries). The boat cannot cross the river by itself with no people on board.

- **Formulate the problem for state space representation.**
- **Construct a complete search tree.**

Search Algorithms

Search algorithms are used to explore the state space and find solutions to a problem.

- **Uninformed Search:**
  - The algorithm has no additional information about the goal state other than what is in the current state.
  - Examples: Depth-first search (DFS), breadth-first search (BFS).
- **Informed Search:**
  - The algorithm uses heuristic information to guide its search toward the goal state.
  - Example: A* search, which uses both path cost and heuristic to find the optimal solution.

Search Algorithms (Continued)

- Search algorithms can be classified based on whether they are **complete** (guarantee a solution) or **optimal** (find the best solution).
- **DFS vs. BFS:**
  - DFS explores deeply into the search tree before backtracking.
  - BFS explores level by level and guarantees the shortest path (if unweighted).

Summary

Problem solving in AI is a structured approach to finding solutions by searching through state spaces. Key concepts include:

- **Problem-solving agents** that explore state spaces to find solutions.
- A **production system** for solving problems using rules and facts.
- The distinction between **well-defined** and **ill-defined** problems.
- Effective **problem formulation** to simplify the problem and apply search algorithms.
- **Search algorithms** that guide the agent toward a solution, whether informed or uninformed.

These techniques are foundational for AI systems and are applied in many domains such as robotics, game playing, and decision-making.