Artificial Intelligence

AI2002

Rushda Muneer

Spring 2026

State Space Strategies

Problem Solving Agents

- Problem-solving agents use **atomic representations**, that is, states of the world are considered as wholes.
- Problem solving begins with **precise definitions** of problems and their solutions
- These function if the agent can adopt a goal and aim at satisfying it
- **Goal formulation,** based on the current situation and the agent's performance measure, is the first step in problem solving.
- **Problem formulation** is the process of deciding what actions and states to consider, given a goal.

State Space Search

- The process of looking for a sequence of actions that reaches the goal is called **search**.
- A search algorithm takes a problem as input and returns a **solution** in the form of an action sequence.
- Once a solution is found, the actions it recommends can be carried out. This is called the **execution** phase.

Well Defined Problem

- A problem can be defined formally by following components:
  - The **initial state** that the agent starts in.
  - A description of the possible **actions** available to the agent. Given a particular state s, ACTIONS(s) returns the set of actions that can be executed in s.
  - A description of what each action does; the formal name for this is the **transition model,** specified by a function RESULT(s, a) that returns the state that **results** from doing action a in state s.
- Together, the initial state, actions, and transition model implicitly define the **state space** of the problem—the set of all states reachable from the initial state by any sequence of actions.
- The **goal test** determines whether a given state is a goal state.

State Space Representation - Graph

- The state space forms a directed network or **graph** in which the **nodes** are **states** and the **links** between nodes are **actions**.
- A **path** in the state space is a sequence of states connected by a sequence of actions.
- A **path cost** function that assigns a numeric cost to each path. The problem-solving agent chooses a cost function that reflects its own performance measure.
- A **solution** to a problem is an action sequence that leads from the initial state to a goal state.
- An **optimal** solution has the lowest path cost among all solutions.

Example 1 – Vacuum Cleaner

- **States:** The state is determined by both the **agent location** and the **dirt locations**. The agent is in **one of two locations**, each of which **might or might not contain dirt**.
- **Initial state:** Any state can be designated as the initial state.
- **Actions:** In this simple environment, each state has just three actions: Left, Right, and Suck.
- **Transition model:** The actions have their expected effects, except that moving Left in the leftmost square, moving Right in the rightmost square, and Suck in a clean square have no effect.
- **Goal test:** This checks whether all the squares are clean.
- **Path cost:** Each step costs 1, so the path cost is the number of steps in the path.

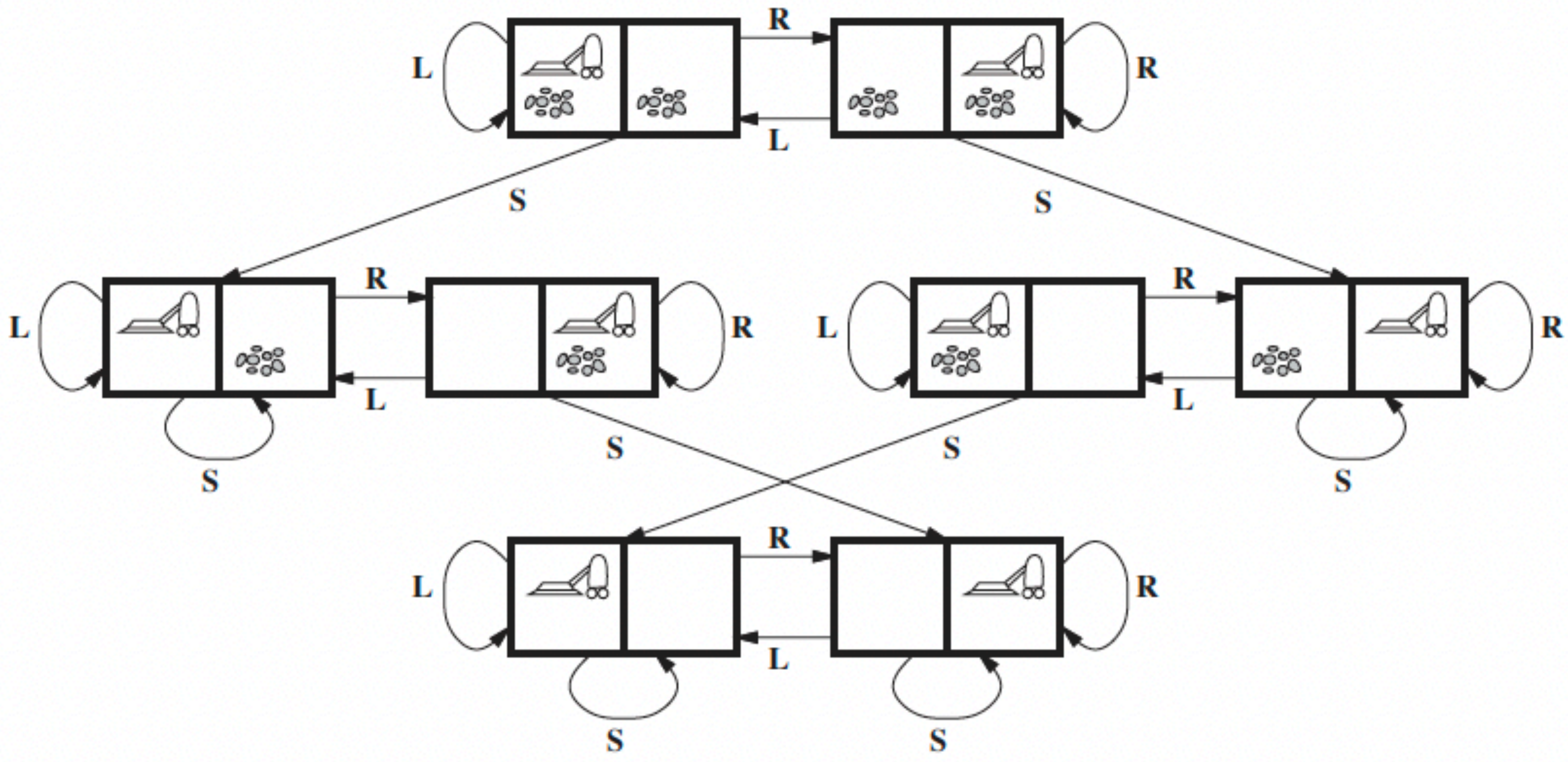
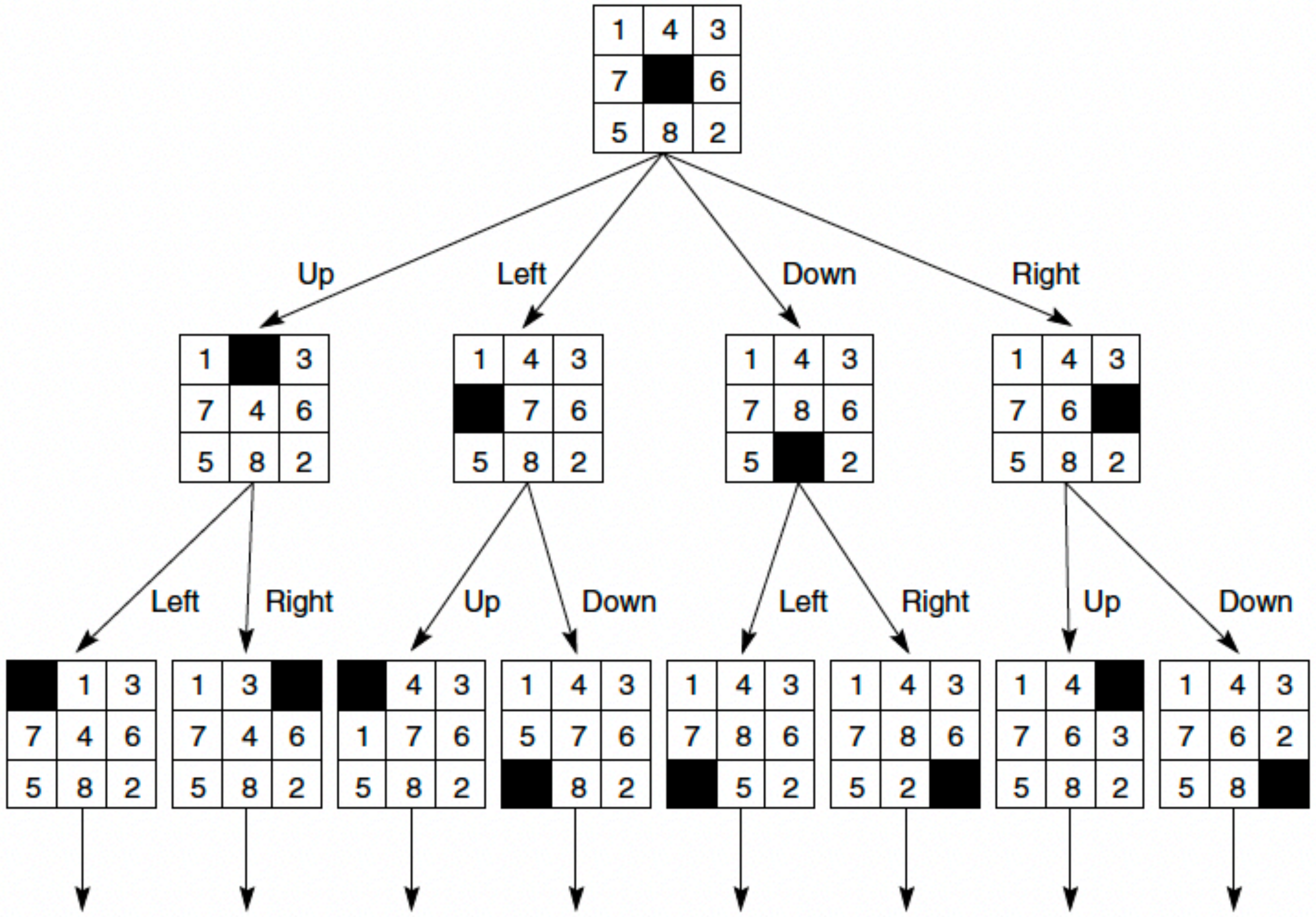


Figure 3.3 The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

Example 2 – 8-Puzzle

- **States:** A state description specifies the location of each of the eight tiles and the blank in one of the nine squares.
- **Initial state:** Any state can be designated as the initial state.
- **Actions:** The simplest formulation defines the actions as movements of the blank space Left, Right, Up, or Down.
- **Transition model:** Given a state and action, this returns the resulting state;
- **Goal test:** This 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.


Up
Left
Down
Right
Left
Right
Up
Down
Left
Right
Up
Down

Activity

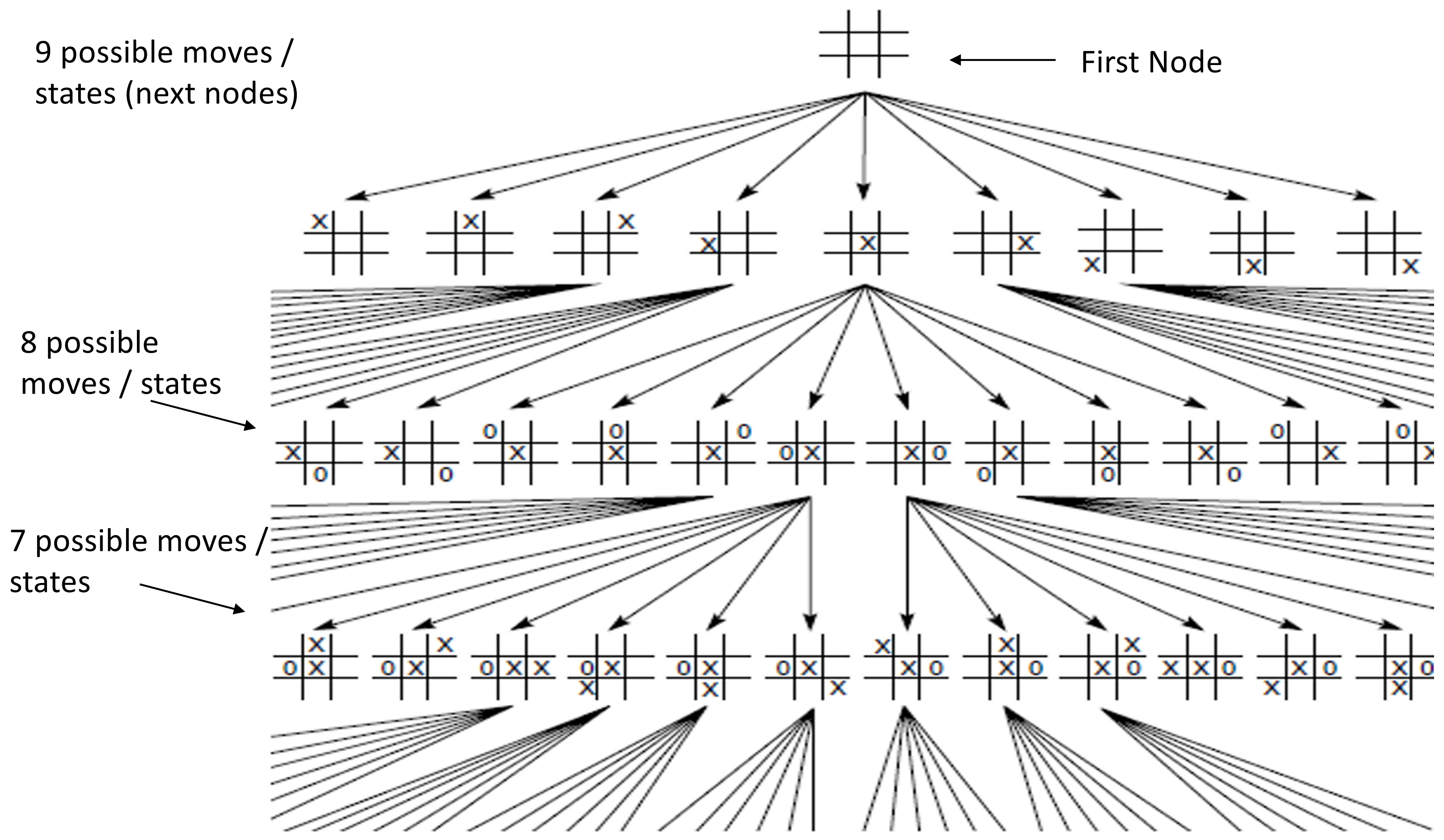
- Write down state space description (states, initial state, actions, transition model, goal test and path cost) for a game of TIC TAC TOE
- Give one example of state space search and write down the descriptors for it.

Example 3 – Tic Tac Toe

- **Board Setup:**
  - Nine spaces that can be filled with:
    - A space (empty)
    - A nought (O)
    - A cross (X)
- **Initial State:**
  - Empty board.
  - The game begins with your opponent placing a nought in the center square.
- **Goal State:**
  - Your goal is to get **three crosses (X)** in a line while **preventing three noughts (O)** in a line.
- **Search:**
  - From the **Initial State**, the task is to find the **next state** that brings you closer to achieving the **Goal State**.

Example 3 – Tic Tac Toe

- The game has **many possible states**, which can be categorized as: **Win, Lose, or Mid-game (intermediate).**
- From **any state**, only a **small subset of states** can be reached in **one move**, based on the rules and current position.
- A graph can represent this process:
- Start with the empty board (**first node**).
- Branch out into the nine possible moves (**new nodes**).
- Continue expanding the graph with additional moves until the **board is full or the game is won**.


9 possible moves / states (next nodes)
First Node
8 possible moves / states
7 possible moves / states

Example 4 – Travelling Salesman Problem

- **Problem Definition:** Given a map of cities and distances between them, find the minimum distance needed to visit each city once and return to the starting point.
- **Graph Representation:**
- Represent the cities as a tree:
  - Start at any city, then branch out to other cities.
  - At each node, keep track of cities visited and the total distance traveled.
- **Finite Tree:** The tree is finite since each city can only be visited once.
- The tree depth is limited to the number of cities.

Travelling Salesman Problem

- **Variations and Challenges:** If cities are not always directly reachable (e.g., needing to pass through other cities), some maps may have no solution.
- **Solution Approach:** The problem is solved using **search algorithms** to find an optimal or feasible solution.
- **Graph Theory:** Graph theory helps analyze the problem structure and guides the design of appropriate search techniques.

Graph Theory (Definitions)

- A graph consists of:
  - A set of "**nodes**" N_1, N_2, …, N_n, … which need not be finite.
  - A set of "**arcs**" that connect pairs of nodes. (Arcs are often described as an ordered pair of nodes; e.g. the arc (N_3,N_4) connects nodes N_3 with N_4).
  - A "**labeled graph**" has one or more descriptors (labels) attached to each node that distinguish that node from any other node in the graph.
- In the **state space model** of problem solving
  - The **nodes** of a graph are taken to represent **discrete "states"** in a problem solving process (such as the result of logical inferences or configurations of a game board).
  - The **arcs** of the graph represent **transitions between states**.
  - (These transitions correspond to logical inferences or legal moves of a game).

Graph Theory (Definitions)

- If a directed arc connects N_j to N_k, then N_j is called the "**parent**" of N_k and N_k is called the **child** of N_j.
- If the graph also contains an **arc** (N_j,N_l), then N_k and N_l are called "**siblings**".
- An **ordered sequence** of nodes $[N_1,N_2, \ldots, N_n]$, where each Ni, Ni+1 in the sequence represents an arc (N_i,N_{i+1}), is called a "**path**" of length n-1 in the graph.
- A "**rooted** graph" has a unique node N_s from which all paths in the graph originate. That is, the root has no parent in the graph.
- (The state space graphs of the most games are usually rooted graphs, with the start of the game at the root).
- A "**tip**" or "**leaf**" node is a node that has **no children**.
- On rooted graph, a node is said to be the "**ancestor**" of all nodes positioned after it (or to its right) and a "**descendant**" of all nodes before it (or to its left).

Graph Theory (Definitions)

- A path that contains any node more than once, is said to contain a “**cycle**” or “**loop**”.
- A “**tree**” is a graph in which there is a unique path between every pair of nodes. (The paths in a tree contain no cycles).
- The edges in a rooted tree are directed away from the root.
- Each node in a rooted tree has a unique parent.
- Two nodes in a graph are said to be “**connected**” if a path exists that includes them both.

State Space (Formal Definition)

- A “**State Space**” is represented by a **four-tuple** [N,A,S,GD] where:
- N is the **set of nodes or states** of the graph. These correspond to the states in a problem-solving process.
- A is the **set of arcs (links)** between nodes. These correspond to the steps in a problem-solving process.
- S is a **nonempty subset** of N, that contains the **start state(s)** of the problem.
- GD is a **nonempty subset** of N, that contains the **goal state(s)** of the problem.
- The states in GD are described using either:
- A **measurable property** of the states encountered in the search.
- A **property of the path** developed in the search.
- A “**solution path**” is a path through this graph from a node in S to a node in GD.

Homework Readings

- Artificial Intelligence a Modern approach – Chapter 3 (3.1 and 3.2)
- AI: Structures and Strategies – Chapter 3 (3.0 and 3.1)