

Lecture 2

Problem Solving using State Space Representations

Overview

- Characteristics of agents and environments
- Problem-solving agents where search consists of
 - state space
 - operators
 - start state
 - goal states
- Abstraction and problem formulation
- Search trees: an effective way to represent the search process

Agents

- An agent is anything that can be viewed as perceiving its environment through sensors and acting upon that environment through actuators

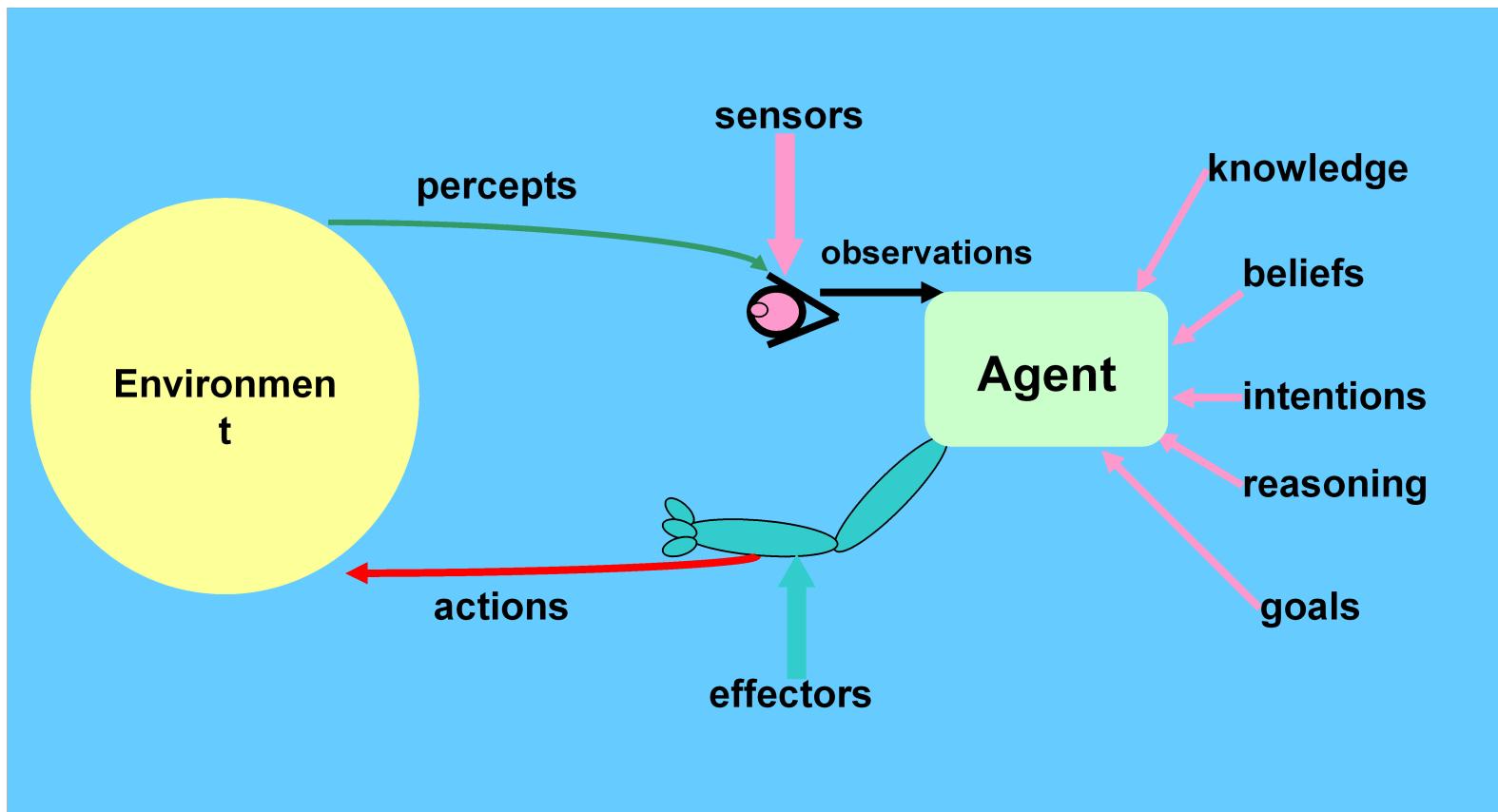
Human agent:

eyes, ears, and other organs for sensors;
hands, legs, mouth, and other body parts for
actuators

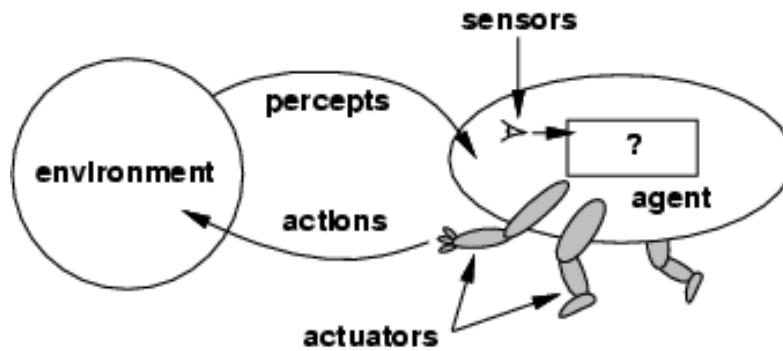
Robotic agent:

cameras and infrared range finders for sensors; various motors
for actuators

Schematic Diagram of an Agent



Agents and environments

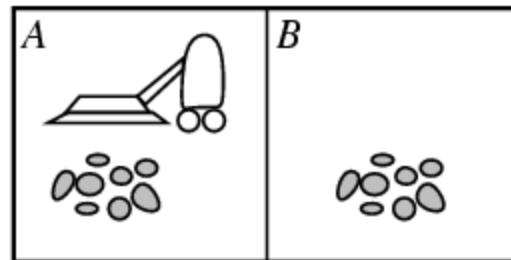


- The agent function maps from percept histories to actions:
 $[f: \mathcal{P}^* \rightarrow \mathcal{A}]$
- The agent program runs on the physical architecture to produce f
- agent = architecture + program

Rational agents

- Rationality is distinct from omniscience (all-knowing with infinite knowledge)
- Agents can perform actions in order to modify future percepts so as to obtain useful information (information gathering, exploration)
- An agent is autonomous if its behavior is determined by its own percepts & experience (with ability to learn and adapt) without depending solely on built-in knowledge

Vacuum-cleaner world



- Percepts: location and state of the environment, e.g., [A,Dirty], [A,Clean], [B,Dirty]
- Actions: *Left*, *Right*, *Suck*, *NoOp*

DARPA robot (Dec. 2013)

WHY THE DARPA ROBOTICS CHALLENGE TASKS?

The story of the DARPA Robotics Challenge (DRC) begins on March 12, 2011, the day after the Tohoku, Japan earthquake and tsunami struck the Fukushima-Daiichi nuclear power plant. On that day, a team of plant workers set out to enter the darkened reactor buildings and manually vent accumulated hydrogen to the atmosphere. Unfortunately, the vent team soon encountered the maximum level of radiation allowed for humans and had to turn back. In the days that followed, with the vents still closed, hydrogen built up in each of three reactor buildings, fueling explosions that extensively damaged the facility, contaminated the environment and drastically complicated stabilization and remediation of the site.

At Fukushima, having a robot with the ability to open valves to vent the reactor buildings might have made all the difference. But to open a valve, a robot first has to be able to get to it. The DRC tasks test some of the mobility, dexterity, manipulation and perception skills a robot needs to be effective in disaster response.



ROBOTICS
CHALLENGE
2013
TRIALS #DARPADRC



DARPA robot (Dec. 2013)

Aim to speed development of robots that could aid in response efforts after future natural and man-made disasters.

Prototype robots are requested to accomplish eight tasks, each designed to test the robots' ability to perform a number of critical real-world disaster-response skills.

They will test the robots' autonomous perception, autonomous decision-making, mounted and dismounted mobility, dexterity and strength.

DARPA robot (Dec. 2013)

Each task presents distinct challenges for the robots and their operators:

Drive and Exit Utility Vehicle: The hardest single task and the one that requires the most robot-human interaction. The operators must direct the robots to drive the vehicle safely despite occasional communications disruptions. Getting out of the driver's seat poses significant strength and dexterity challenges for the robots.

Walk Across Rough Terrain: The robots must maintain their balance and identify safe routes for placement of limbs.

Remove Debris from Doorway: Robots must demonstrate a wide range of motion, in addition to balance and strength, to clear a path forward.

Open Series of Doors: Moving the doors in an arc challenges the robots' perception and dexterity. The robots must figure out how to align and move themselves as they open each door.

DARPA robot (Dec. 2013)

Climb Industrial Ladder: To avoid falls, the robots must safely navigate the ladder and maintain their balance as they climb. Strength is required to stop a fall.

Cut Through Wall: Using power tools tests the robots' strength, dexterity and ability to perceive their environment. The robots must also simultaneously apply rigid force to hold a tool, yet demonstrate the flexibility to smoothly manipulate it.

Carry and Connect Fire Hose: The robots must identify the standpipe and then transport a bulky, non-rigid item (the fire hose) to it. The robots must then have sufficient dexterity and strength to attach the hose to a standpipe and open the spigot.

Locate and Close Leaking Valves: The robots must identify the valves, determine which ones are open and have sufficient range of motion to turn the valve wheels in an arc to close them.

DARPA robot (Dec. 2013) Winner: Google



Measuring the Agent's Performance

It includes the criteria for success of agents behavior

: Criteria

Is agent capable of achieving the task

What resources needed to achieve the task

What time needed to accomplish the task

Problems of measuring performance

It is not an easy task because it is based on measuring the average of each of the above criteria which is sometimes vague and not clear

Measuring the Agent's Performance: Examples

Rubbish collection agent

Measuring its performance by the following

Amount of rubbish collected

Quantity of electrical power used

Quality of cleaning the place

Mobile robot agent

Measuring its performance by the following

Going to a specific destination

Minimize the time to reach the destination

Minimize the power consumption

Minimize the number of collisions

Task Environment

- Before we design an intelligent agent, we must specify its “task environment”:

PEAS:

Performance measure

Environment

Actuators

Sensors

PEAS

- Example: Agent = robot driver in DARPA Challenge
 - Performance measure:
 - Time to complete course
 - Environment:
 - Roads, other traffic, obstacles
 - Actuators:
 - Steering wheel, accelerator, brake, signal, etc.
 - Sensors:
 - Optical cameras, lasers, sonar, accelerometer, speedometer, GPS, odometer, engine sensors, etc.

PEAS

- Example: Agent = Medical diagnosis system

Performance measure:

Healthy patient, minimize costs, lawsuits

Environment:

Patient, hospital, staff

Actuators:

Screen display (questions, tests, diagnoses, treatments, referrals)

Sensors:

Keyboard (entry of symptoms, findings, patient's answers)

Environment types

- Fully observable (vs. partially observable):
 - An agent's sensors give it access to the complete state of the environment at each point in time.
- Deterministic (vs. stochastic):
 - The next state of the environment is completely determined by the current state and the action executed by the agent.
 - If the environment is deterministic except for the actions of other agents, then the environment is **strategic**
 - Deterministic environments can appear stochastic to an agent (e.g., when only partially observable)
- Episodic (vs. sequential):
 - An agent's action is divided into atomic episodes. Decisions do not depend on previous decisions/actions.

Environment types

- **Static** (vs. dynamic):
 - The environment is unchanged while an agent is deliberating.
 - The environment is **semidynamic** if the environment itself does not change with the passage of time but the agent's performance score does
- **Discrete** (vs. continuous):
 - A discrete set of distinct, clearly defined percepts and actions.
 - How we **represent** or **abstract** or **model** the world
- **Single agent** (vs. multi-agent):
 - An agent operating by itself in an environment. Does the other agent interfere with my performance measure?

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle						
chess with clock						
taxi driving						
image analysis						
partpicking robot						

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully					
chess with clock	fully					
taxi driving	partial					
image analysis	fully					
partpicking robot	partial					

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.				
chess with clock	fully	strategic				
taxi driving	partial	stochastic				
image analysis	fully	determ.				
partpicking robot	partial	stochastic				

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential			
chess with clock	fully	strategic	sequential			
taxi driving	partial	stochastic	sequential			
image analysis	fully	determ.	episodic			
partpicking robot	partial	stochastic	episodic			

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static		
chess with clock	fully	strategic	sequential	semi		
taxi driving	partial	stochastic	sequential	dynamic		
image analysis	fully	determ.	episodic	semi		
partpicking robot	partial	stochastic	episodic	dynamic		

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	
chess with clock	fully	strategic	sequential	semi	discrete	
taxi driving	partial	stochastic	sequential	dynamic	continuous	
image analysis	fully	determ.	episodic	semi	discrete	
partpicking robot	partial	stochastic	episodic	dynamic	discrete	

task environm.	observable	deterministic/ stochastic	episodic/ sequential	static/ dynamic	discrete/ continuous	agents
crossword puzzle	fully	determ.	sequential	static	discrete	single
chess with clock	fully	strategic	sequential	semi	discrete	multi
taxi driving	partial	stochastic	sequential	dynamic	continuous	multi
image analysis	fully	determ.	episodic	semi	discrete	single
partpicking robot	partial	stochastic	episodic	dynamic	discrete	single

What is the environment for the DARPA Challenge?

- Agent = robotic vehicle
- Environment = 130-mile route through desert
 - Observable?
 - Deterministic?
 - Episodic?
 - Static?
 - Discrete?
 - Agents?

What is the environment for the DARPA Challenge?

- Agent = robotic vehicle
- Environment = 130-mile route through desert
 - Observable? Partially
 - Deterministic? Stochastic
 - Episodic? Sequential
 - Static? Dynamic
 - Discrete? Continues
 - Agents? Single

Agent types

- Five basic types in order of increasing generality:
 - Table Driven agent
 - Simple reflex agents
 - Model-based reflex agents
 - Goal-based agents
 - Problem-solving agents
 - Utility-based agents
 - Can distinguish between different goals

Table-driven agent: the function consists in a lookup table of actions to be taken for
. .every possible state of the environment

A simple reflex agent is a type of intelligent agent that performs actions based solely on the current situation, with an intelligent agent generally being one that perceives its environment and then acts. It does this through pre-determined rules for these conditions

A model-based reflex agent is one that uses its percept history and its internal memory to make decisions about an internal "model" of the world around it. Internal memory allows these agents to store some of their navigation history, and then use that semi-subjective history to help understand things about their current environment--even when everything . they need to know cannot be directly observed

Goal-based agents or Problem-solving agents expand on the concept of model-based agents, which understand the world around them by implementing the desired outcome or goal to be reached. A goal-based agent takes it a step further by using a goal in the future to help make decisions about how best to reach that outcome

A utility-based agent is an agent that acts based not only on what the goal is, but the best way to reach that goal. In short, it's the usefulness (or utility) of the agent that makes itself distinct from its counterparts

Problem-Solving Agents

- Intelligent agents can solve problems by searching a state-space
- State-space Model
 - the agent's model of the world
 - usually a set of discrete states
 - e.g., in driving, the states in the model could be towns/cities
- Goal State(s)
 - a goal is defined as a desirable state for an agent
 - there may be many states which satisfy the goal test
 - e.g., drive to a town with a ski-resort
 - or just one state which satisfies the goal
 - e.g., drive to Mammoth
- Operators (actions, successor function)
 - operators are legal actions which the agent can take to move from one state to another

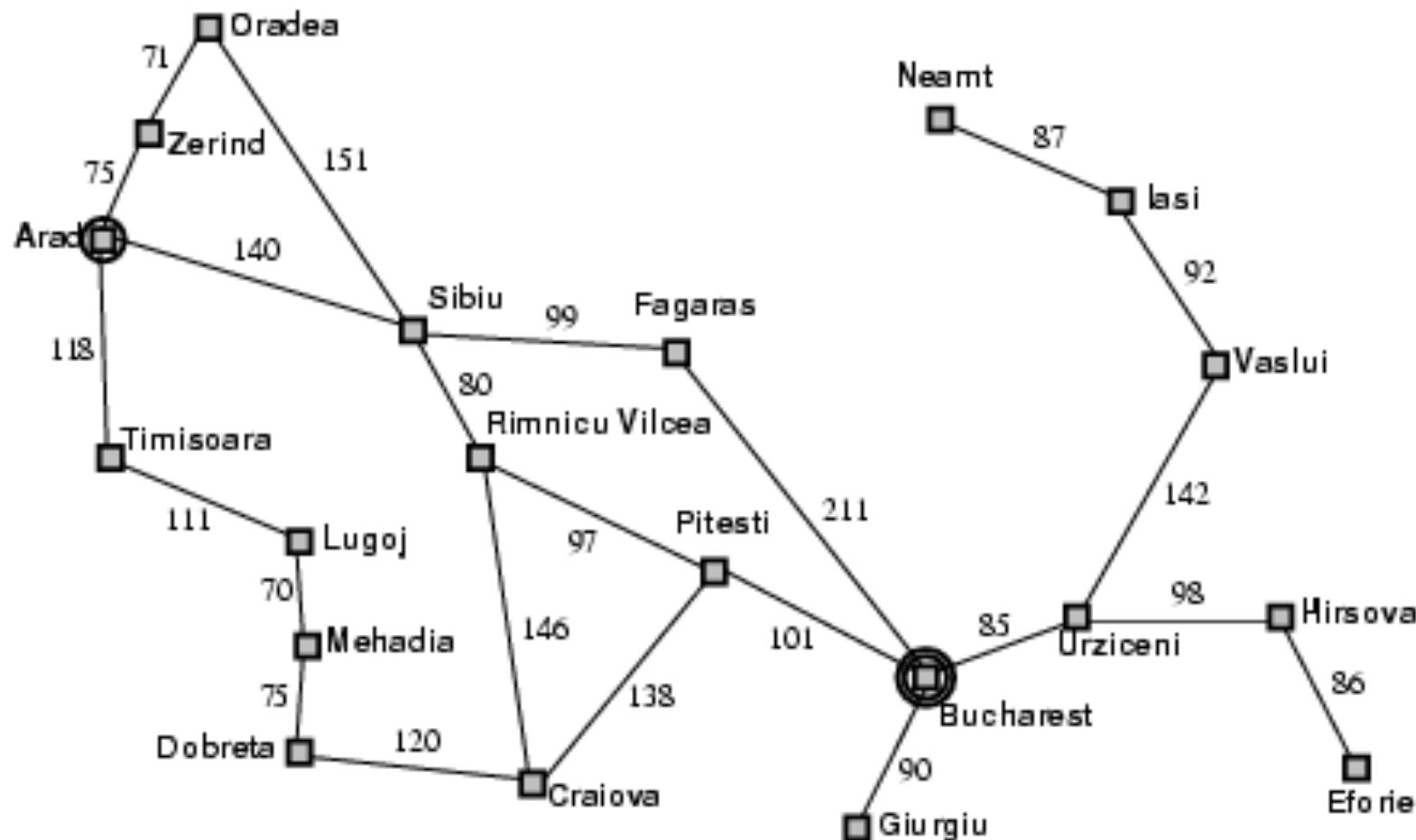
Initial Simplifying Assumptions

- Environment is static
 - no changes in environment while problem is being solved
- Environment is observable
- Environment and actions are discrete
 - (typically assumed, but we will see some exceptions)
- Environment is deterministic

Example: Traveling in Romania

- On holiday in Romania; currently in Arad
- Flight leaves tomorrow from Bucharest
- Formulate goal:
 - be in Bucharest
- Formulate problem:
 - **states**: various cities
 - **actions/operators**: drive between cities
- Find solution
 - By searching through states to find a goal
 - sequence of cities, e.g., Arad, Sibiu, Fagaras, Bucharest
- Execute states that lead to a solution

Example: Traveling in Romania



State-Space Problem Formulation

A **problem** is defined by four items:

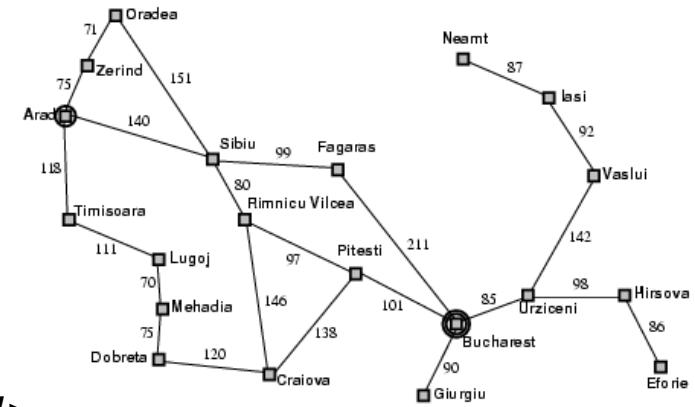
1. **initial state** e.g., "at Arad"
2. **actions** or **successor function**

$S(x)$ = set of action-state pairs
e.g., $S(Arad) = \{<Arad \rightarrow Zerind, Zerind>, \dots\}$

3. **goal test** (or set of goal states)
e.g., $x = \text{"at Bucharest"}$

4. **path cost** (additive)
e.g., sum of distances, number of actions executed, etc.
 $c(x,a,y)$ is the step cost, assumed to be ≥ 0

A **solution** is a sequence of actions leading from the initial state to a goal state



Example: Formulating the Navigation Problem

- Set of States
 - individual cities
 - e.g., Irvine, SF, Las Vegas, Reno, Boise, Phoenix, Denver
- Operators
 - freeway routes from one city to another
 - e.g., Irvine to SF via 5, SF to Seattle, etc
- Start State
 - current city where we are, Irvine
- Goal States
 - set of cities we would like to be in
 - e.g., cities which are closer than Irvine
- Solution
 - a specific goal city, e.g., Boise
 - a sequence of operators which get us there,
 - e.g., Irvine to SF via 5, SF to Reno via 80, etc

Abstraction

Definition of Abstraction:

Process of removing irrelevant detail
to create an abstract representation:
``high-level'', ignores irrelevant details



President



General



Sergeant



Foot Soldier

**High-level Abstract Description
to Low-level Implementation Details**

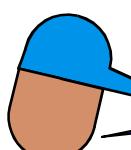
My poll ratings are low,
lets invade a small nation



Cross the river and take
defensive positions



Forward march, turn left
Stop!, Shoot



Abstraction

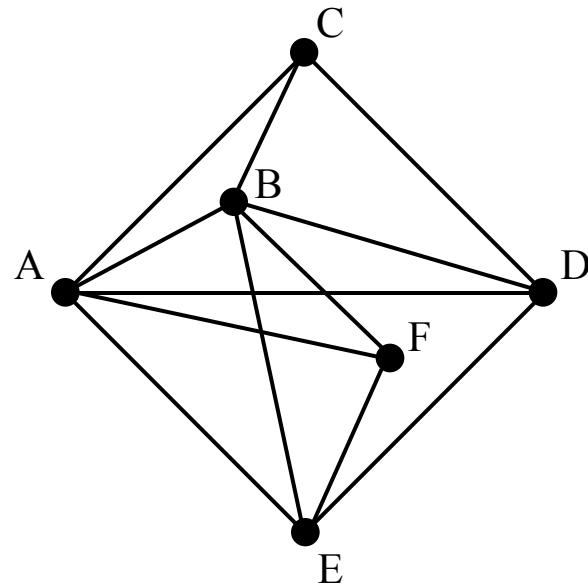
- Navigation Example: how do we define states and operators?
 - First step is to abstract “the big picture”
 - i.e., solve a map problem
 - nodes = cities, links = freeways/roads (a high-level description)
 - this description is an abstraction of the real problem
 - Can later worry about details like freeway onramps, refueling, etc
- Abstraction is critical for automated problem solving
 - must create an approximate, simplified, model of the world for the computer to deal with: real-world is too detailed to model exactly
 - good abstractions retain all important details

The State-Space Graph

- Graphs:
 - nodes, arcs, directed arcs, paths
- Search graphs:
 - States are nodes
 - operators are directed arcs
 - solution is a path from start S to goal G
- Problem formulation:
 - Give an abstract description of states, operators, initial state and goal state.
- Problem solving:
 - Generate a part of the search space that contains a solution

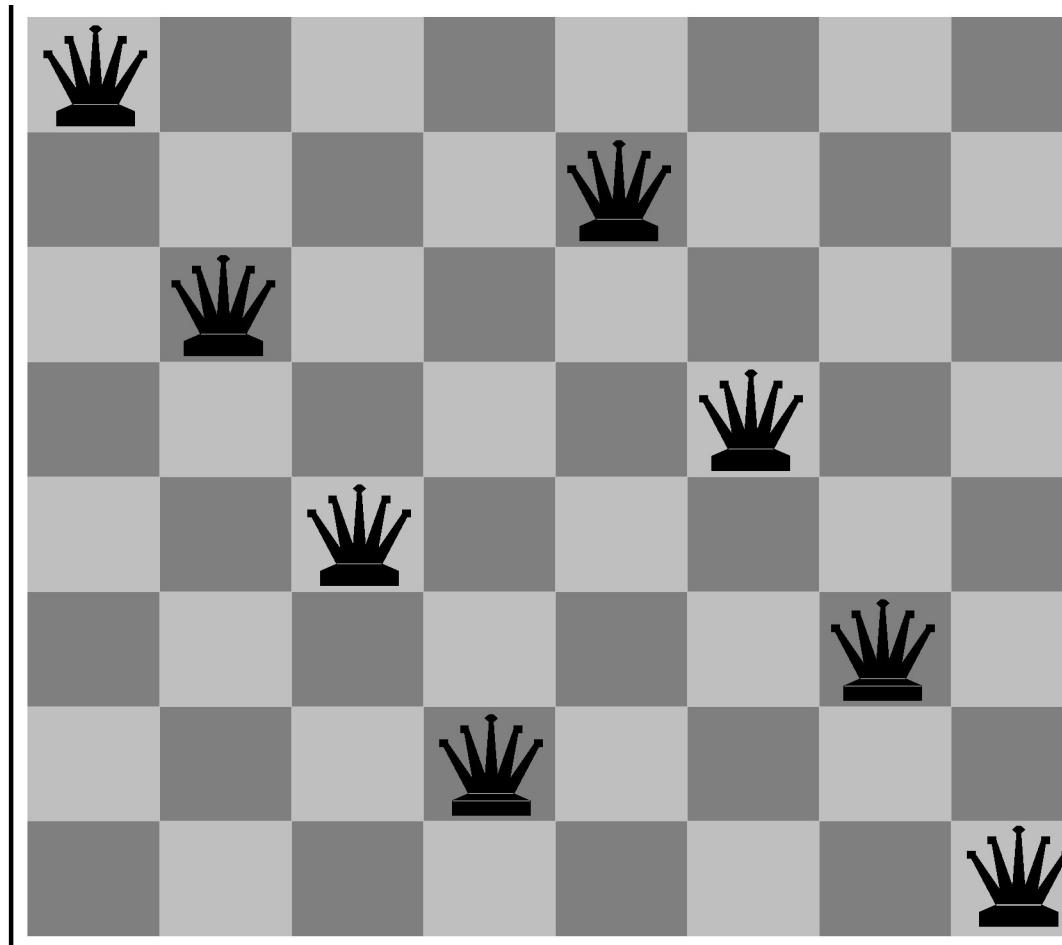
The Traveling Salesperson Problem

- Find the shortest tour that visits all cities without visiting any city twice and return to starting point.
- State: sequence of cities visited
- $S_0 = A$



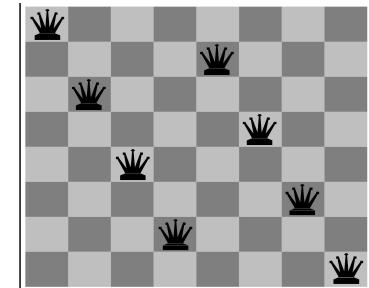
- $G = \text{a complete tour}$

Example: 8-queens problem

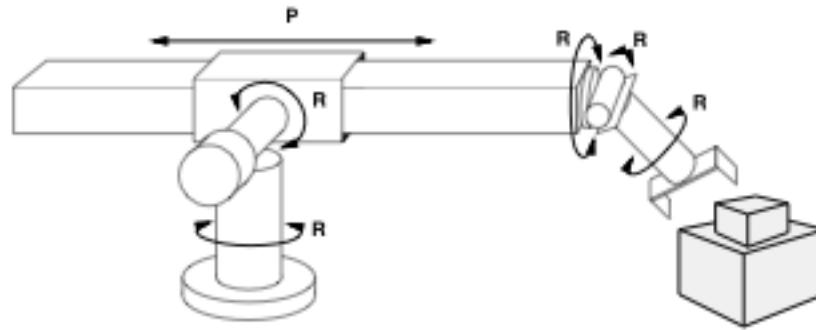


State-Space problem formulation

- states? -any arrangement of $n \leq 8$ queens
 - or arrangements of $n \leq 8$ queens in leftmost n columns, 1 per column, such that no queen attacks any other.
- initial state? no queens on the board
- actions? -add queen to any empty square
 - or add queen to leftmost empty square such that it is not attacked by other queens.
- goal test? 8 queens on the board, none attacked.
- path cost? 1 per move

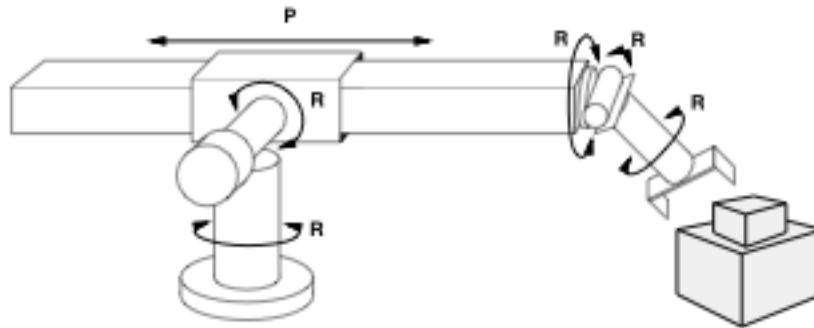


Example: Robot Assembly



- States
- Initial state
- Actions
- Goal test
- Path Cost

Example: Robot Assembly



- States: configuration of robot (angles, positions) and object parts
- Initial state: any configuration of robot and object parts
- Actions: continuous motion of robot joints
- Goal test: object assembled?
- Path Cost: time-taken or number of actions

Learning a spam email classifier

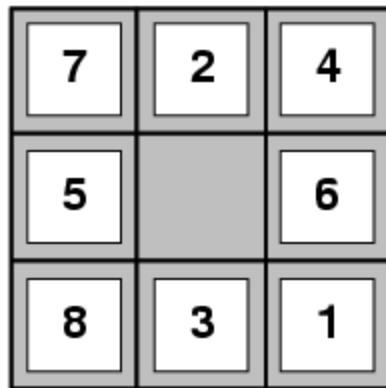
- States
- Initial state
- Actions
- Goal test
- Path Cost

Learning a spam email classifier

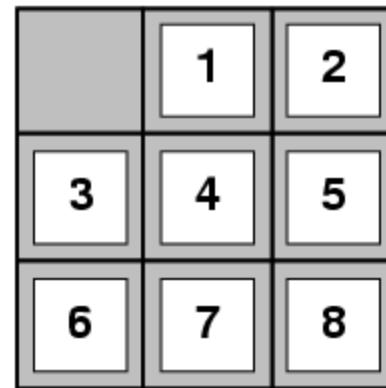
- States: settings of the parameters in our model
- Initial state: random parameter settings
- Actions: moving in parameter space
- Goal test: optimal accuracy on the training data
- Path Cost: time taken to find optimal parameters

(Note: this is an optimization problem – many machine learning problems can be cast as optimization)

Example: 8-puzzle



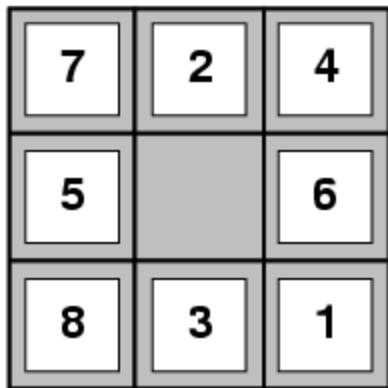
Start State



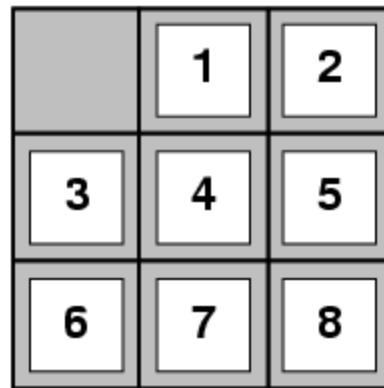
Goal State

- states?
- initial state?
- actions?
- goal test?
- path cost?

Example: 8-puzzle



Start State

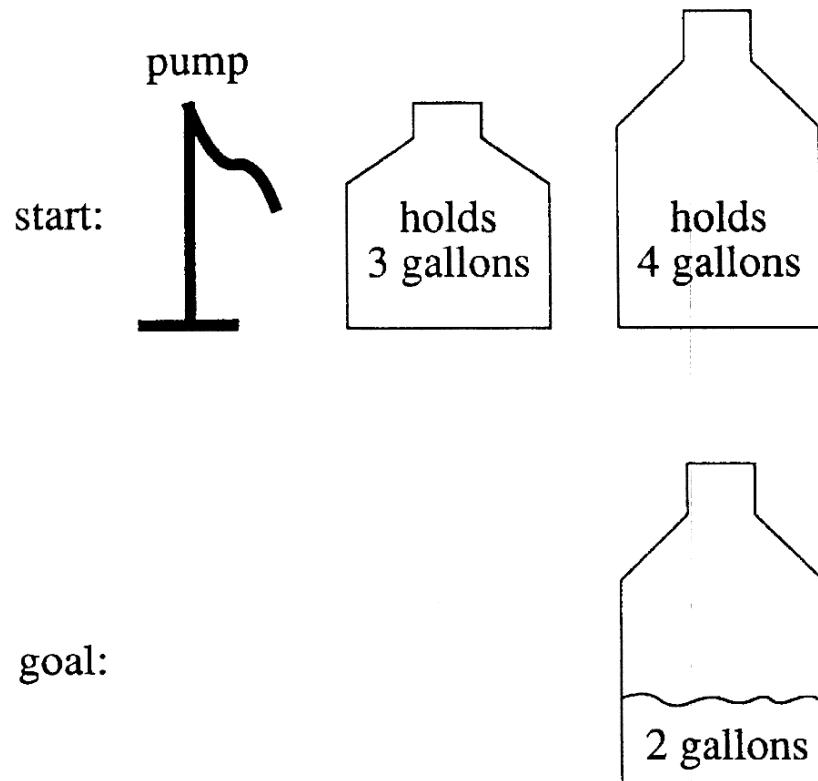


Goal State

- states? locations of tiles
- initial state? given
- actions? move blank left, right, up, down
- goal test? goal state (given)
- path cost? 1 per move

A Water Jug Problem

- You have a 4-gallon and a 3-gallon water jug
- You have a faucet with an unlimited amount of water
- You need to get exactly 2 gallons in 4-gallon jug



Puzzle-solving as Search

- State representation: **(x, y)**
 - x: Contents of four gallon
 - y: Contents of three gallon
- Start state: **(0, 0)**
- Goal state **(2, n)**
- Operators
 - Fill 3-gallon from faucet, fill 4-gallon from faucet
 - Fill 3-gallon from 4-gallon , fill 4-gallon from 3-gallon
 - Empty 3-gallon into 4-gallon, empty 4-gallon into 3-gallon
 - Dump 3-gallon down drain, dump 4-gallon down drain

Production Rules for the Water Jug Problem

- | | |
|--|---|
| 1 $(x,y) \rightarrow (4,y)$
if $x < 4$ | Fill the 4-gallon jug |
| 2 $(x,y) \rightarrow (x,3)$
if $y < 3$ | Fill the 3-gallon jug |
| 3 $(x,y) \rightarrow (x - d,y)$
if $x > 0$ | Pour some water out of the 4-gallon jug |
| 4 $(x,y) \rightarrow (x,y - d)$
if $x > 0$ | Pour some water out of the 3-gallon jug |
| 5 $(x,y) \rightarrow (0,y)$
if $x > 0$ | Empty the 4-gallon jug on the ground |
| 6 $(x,y) \rightarrow (x,0)$
if $y > 0$ | Empty the 3-gallon jug on the ground |
| 7 $(x,y) \rightarrow (4,y - (4 - x))$
if $x + y \geq 4$ and $y > 0$ | Pour water from the 3-gallon jug into the 4-gallon jug until the 4-gallon jug is full |

The Water Jug Problem (cont'd)

8 $(x,y) \rightarrow (x - (3 - y), 3)$
if $x + y \geq 3$ and $x > 0$

Pour water from the 4-gallon jug into the 3-gallon jug until the 3-gallon jug is full

9 $(x,y) \rightarrow (x + y, 0)$
if $x + y \leq 4$ and $y > 0$

Pour all the water from the 3-gallon jug into the 4-gallon jug

10 $(x,y) \rightarrow (0, x + y)$
if $x + y \leq 3$ and $x > 0$

Pour all the water from the 4-gallon jug into the 3-gallon jug

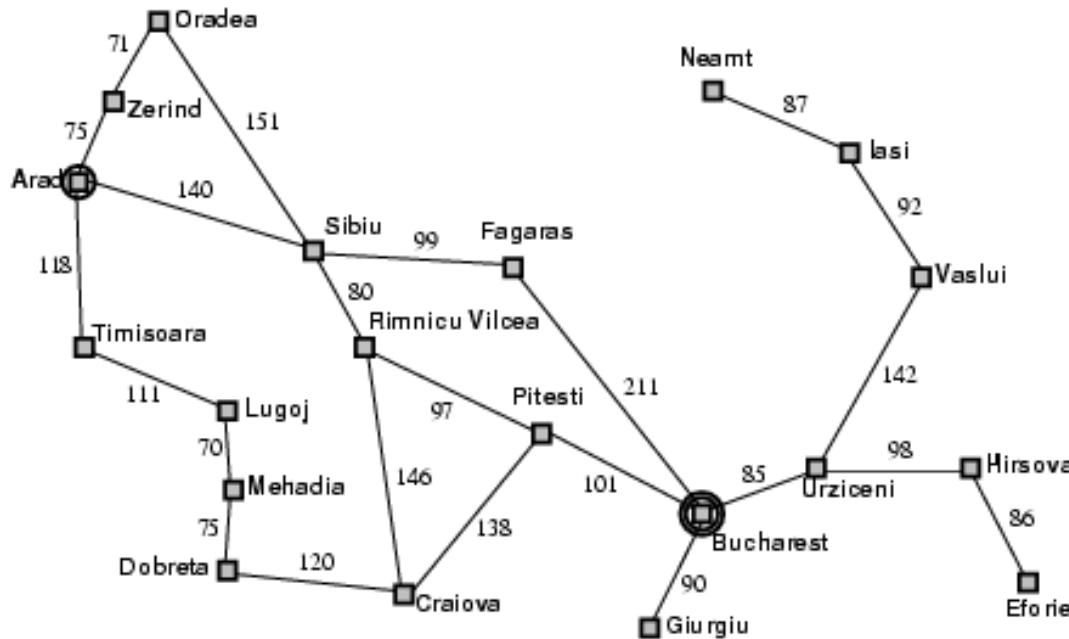
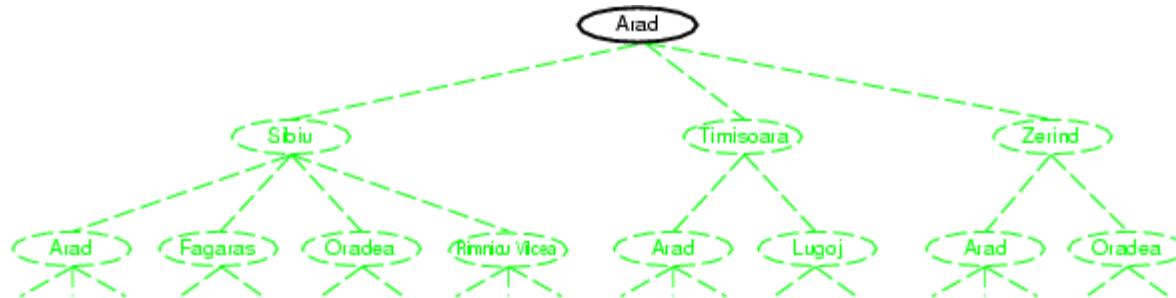
One Solution to the Water Jug Problem

Gallons in the 4-Gallon Jug	Gallons in the 3-Gallon Jug	Rule Applied
0	0	2
0	3	9
3	0	2
3	3	7
4	2	5
0	2	9
2	0	

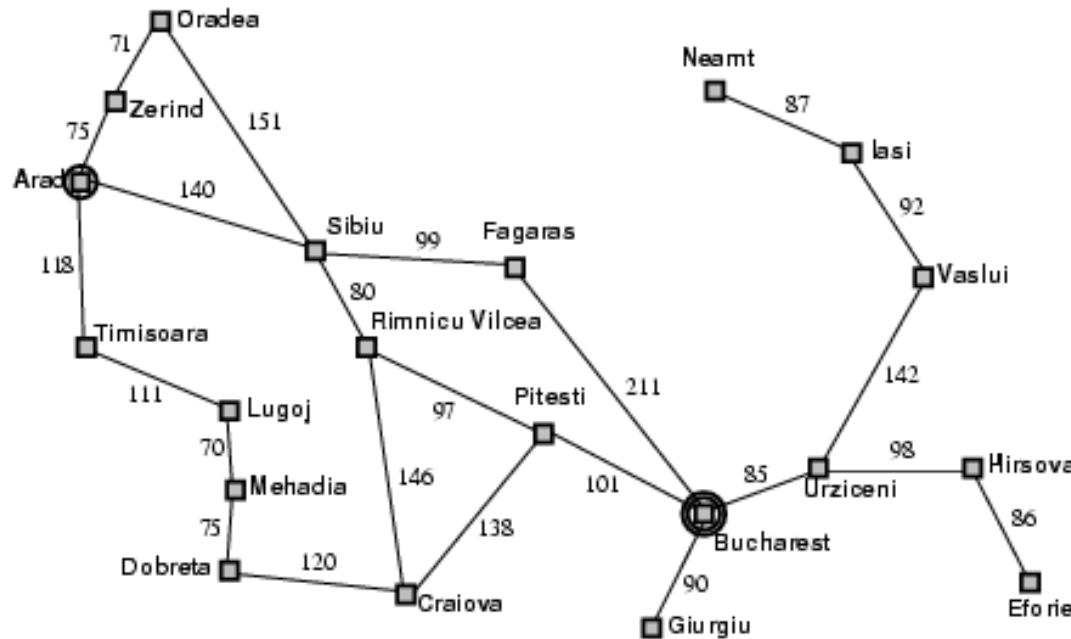
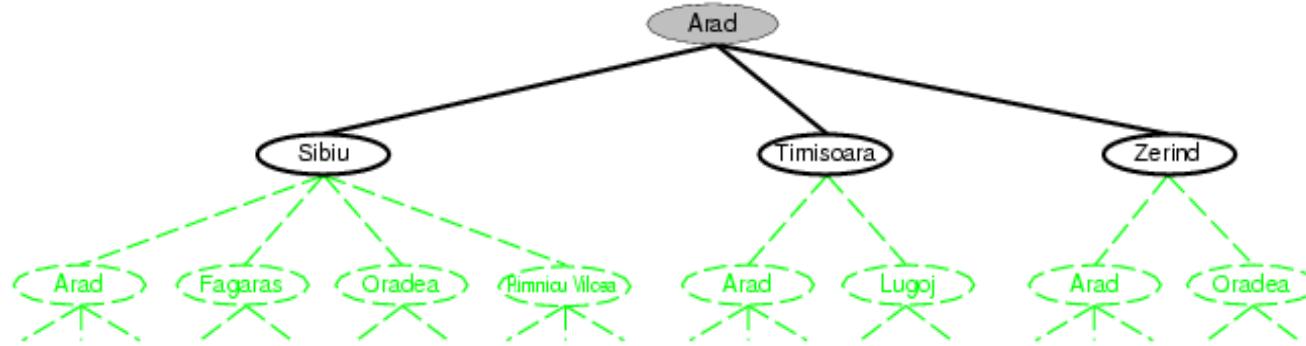
Tree-based Search

- Basic idea:
 - Exploration of state space by generating successors of already-explored states (a.k.a. expanding states).
 - Every state is evaluated: *is it a goal state?*
- In practice, the solution space can be a graph, not a tree
 - E.g., 8-puzzle
 - More general approach is graph search
 - Tree search can end up repeatedly visiting the same nodes
 - Unless it keeps track of all nodes visited
 - ...but this could take vast amounts of memory

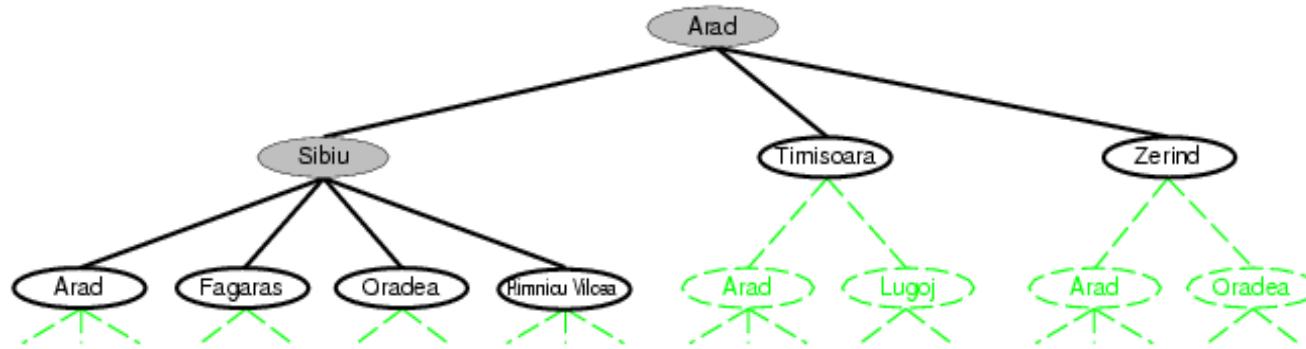
Tree search example



Tree search example

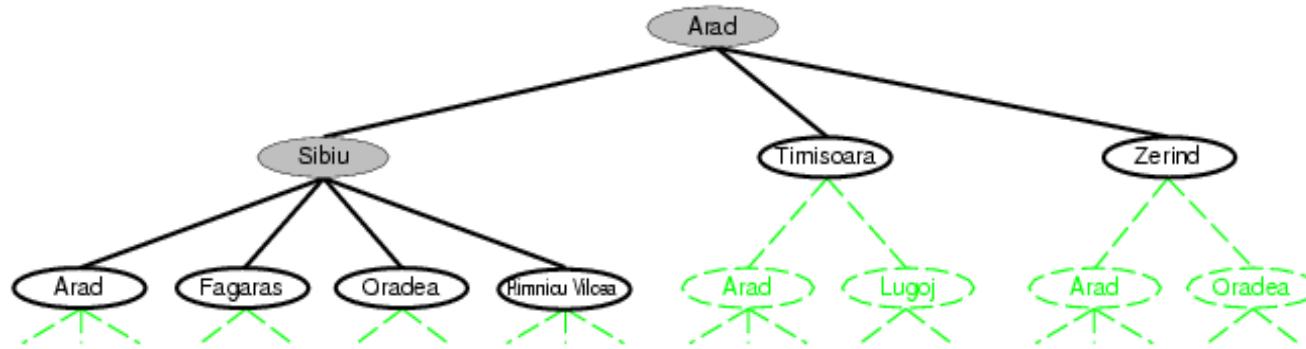


Tree search example



```
function TREE-SEARCH( problem, strategy ) returns a solution, or failure
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
```

Tree search example

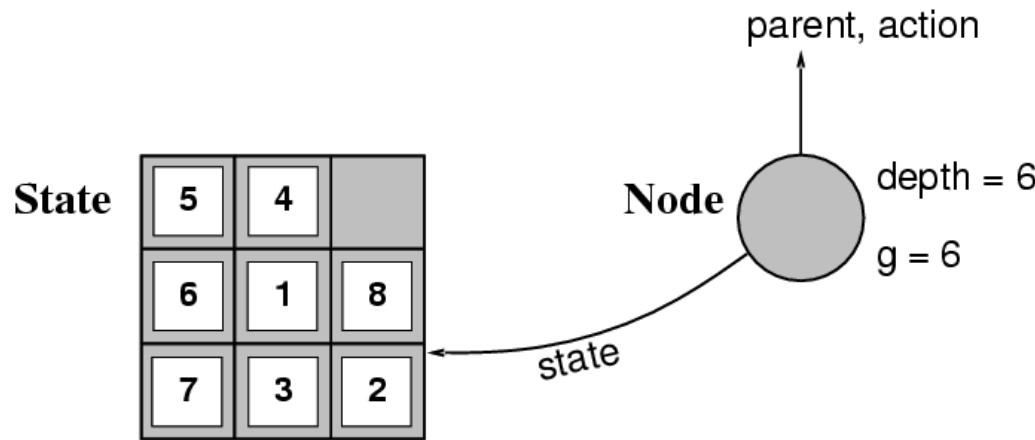


```
function TREE-SEARCH( problem, strategy ) returns a solution
    initialize the search tree using the initial state of problem
    loop do
        if there are no candidates for expansion then return failure
        choose a leaf node for expansion according to strategy
        if the node contains a goal state then return the corresponding solution
        else expand the node and add the resulting nodes to the search tree
```

This “strategy” is what differentiates different search algorithms

States versus Nodes

- A **state** is a (representation of) a physical configuration
- A **node** is a data structure constituting part of a search tree contains info such as: **state**, **parent node**, **action**, **path cost** $g(x)$, **depth**



- The `Expand` function creates new nodes, filling in the various fields and using the `SuccessorFn` of the problem to create the corresponding states.

State Spaces versus Search Trees

- State Space
 - Set of valid states for a problem
 - Linked by operators
 - e.g., 20 valid states (cities) in the Romanian travel problem
- Search Tree
 - Root node = initial state
 - Child nodes = states that can be visited from parent
 - Note that the depth of the tree can be infinite
 - E.g., via repeated states
 - Partial search tree
 - Portion of tree that has been expanded so far
 - Fringe
 - Leaves of partial search tree, candidates for expansion

Search trees = data structure to search state-space

Search Tree for the 8 puzzle problem

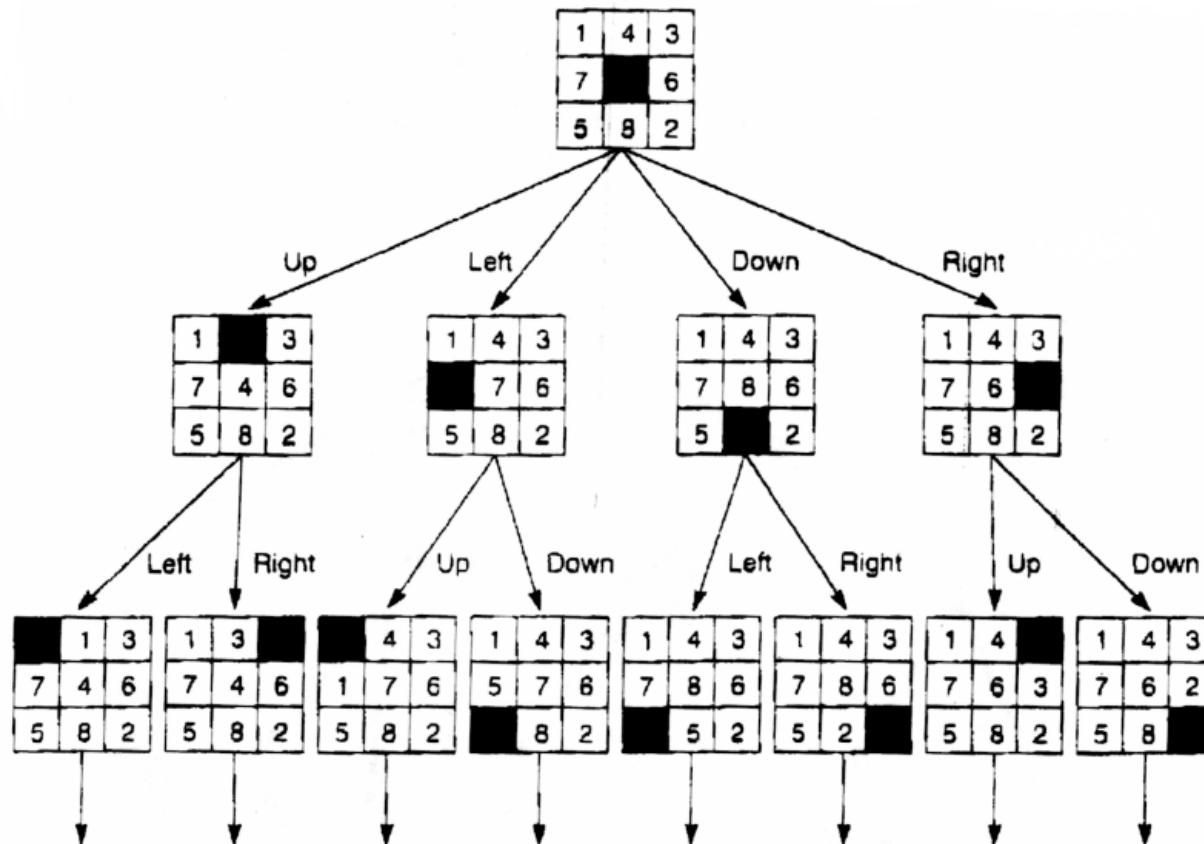


Figure 3.6 State space of the 8-puzzle generated by "move blank" operations.

Search Strategies

- A **search strategy** is defined by picking the order of node expansion
- Strategies are evaluated along the following dimensions:
 - completeness: does it always find a solution if one exists?
 - time complexity: number of nodes generated
 - space complexity: maximum number of nodes in memory
 - optimality: does it always find a least-cost solution?
- Time and space complexity are measured in terms of
 - b : maximum branching factor of the search tree
 - d : depth of the least-cost solution
 - m : maximum depth of the state space (may be ∞)

Why Search can be hard

Assuming $b=10$, time to expand a node = 1000 nodes/sec, memory = 100 bytes/node

Depth of Solution	Nodes to Expand	Time	Memory
0	1	1 millisecond	100 bytes
2	111	0.1 seconds	11 kbytes
4	11,111	11 seconds	1 megabyte
8	10^8	31 hours	11 giabytes
12	10^{12}	35 years	111 terabytes

Summary

- Characteristics of agents and environments
- Problem-solving agents where search consists of
 - state space
 - operators
 - start state
 - goal states
- Abstraction and problem formulation
- Search trees: an effective way to represent the search process