CS:4420 Artificial Intelligence

Spring 2018

Problem Solving by Search

Cesare Tinelli

The University of Iowa

Readings

- Chap. 3 of [Russell and Norvig, 2012]

Example: Romania

Problem: On holiday in Romania; currently in Arad. Flight leaves tomorrow from Bucharest. Find a short route to drive to Bucharest.

Formulate problem:

states: various cities

actions: drive between cities

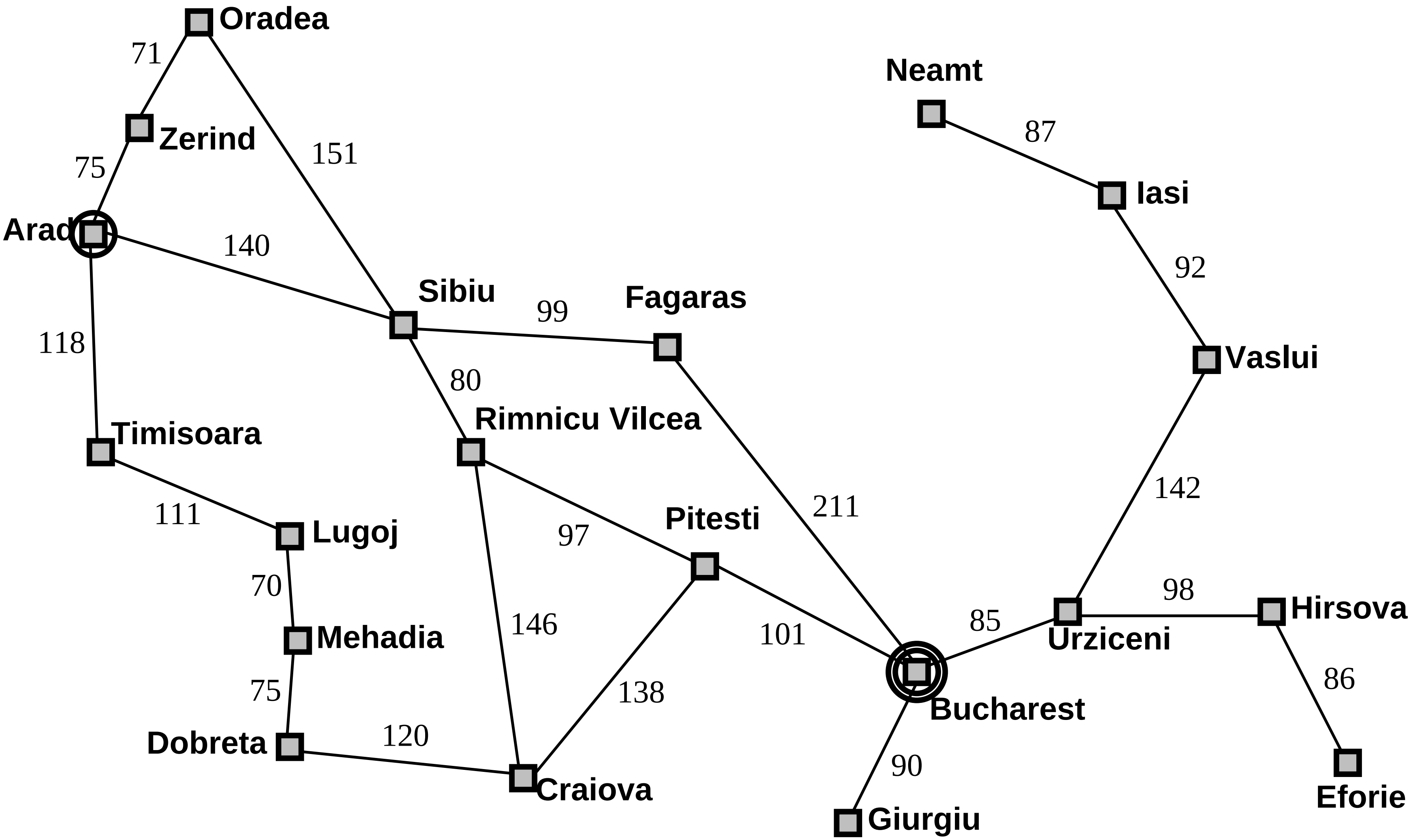
Formulate goal:

be in Bucarest

Formulate solution:

sequence of cities (eg, Arad, Sibiu, Fagaras, Bucharest)

Romania's map



Problem-solving agents

Restricted form of general agent:

```
function SIMPLE-PROBLEM-SOLVING-AGENT( percept) returns an action
    static: seq, an action sequence, initially empty
            state, some description of the current world state
            goal, a goal, initially null
            problem, a problem formulation

    state ← UPDATE-STATE(state, percept)
    if seq is empty then
        goal ← FORMULATE-GOAL(state)
        problem ← FORMULATE-PROBLEM(state, goal)
        seq ← SEARCH( problem)
    action ← RECOMMENDATION(seq, state)
    seq ← REMAINDER(seq, state)
    return action
```

Problem-solving agents

Restricted form of general agent:

function SIMPLE-PROBLEM-SOLVING-AGENT(*percept*) **returns** an action
  static: *seq*, an action sequence, initially empty
    state, some description of the current world state
    goal, a goal, initially null
    problem, a problem formulation

  $state \leftarrow$ UPDATE-STATE($state, percept$)
  if *seq* is empty **then**
    $goal \leftarrow$ FORMULATE-GOAL($state$)
    $problem \leftarrow$ FORMULATE-PROBLEM($state, goal$)
    $seq \leftarrow$ SEARCH($problem$)
  $action \leftarrow$ RECOMMENDATION($seq, state$)
  $seq \leftarrow$ REMAINDER($seq, state$)
  return *action*

Note: this is offline problem solving; solution executed "eyes closed."
Online problem solving involves acting without complete knowledge.

Problem Types

- Deterministic, fully observable environment $\Longrightarrow$ *single-state problem*
  - Agent knows exactly which state it will be in
  - Solution is a sequence of actions
- Non-observable environment $\Longrightarrow$ *conformant problem*
  - Agent know it may be in any of a number of states
  - Solution, if any, is a sequence of actions
- Nondeterministic and/or partially observable environment $\Longrightarrow$ *contingency problem*
  - Percepts provide new information about current state
  - Solution is a tree or policy
  - Often interleave search and execution

Problem Types (cont.)

- Unknown state space $\Longrightarrow$ *exploration problem* ("online")

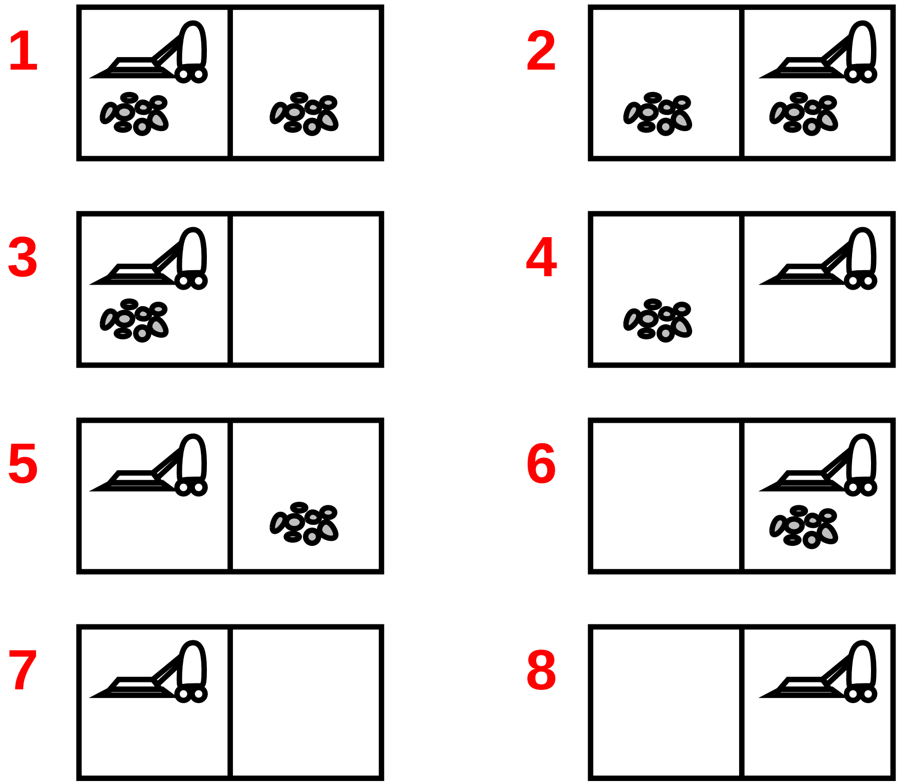
Example: Vacuum World

Single-state problem

initial state = 5

goal states = $\{7, 8\}$

Solution?



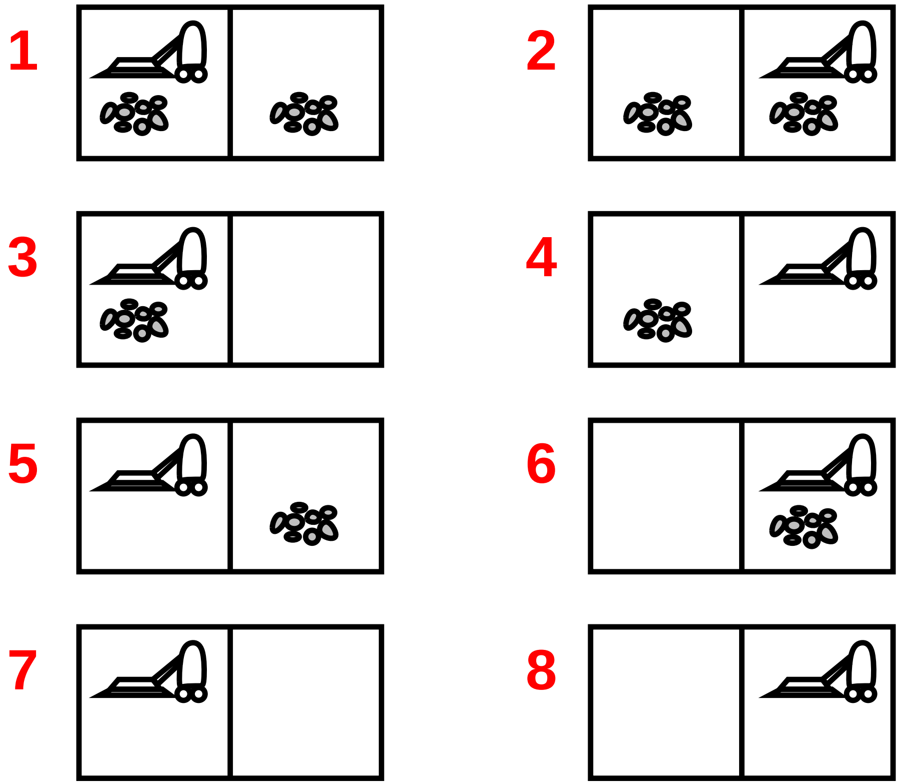
Example: Vacuum World

Single-state problem

initial state = 5

goal states = $\{7, 8\}$

Solution? $[Right, Suck]$

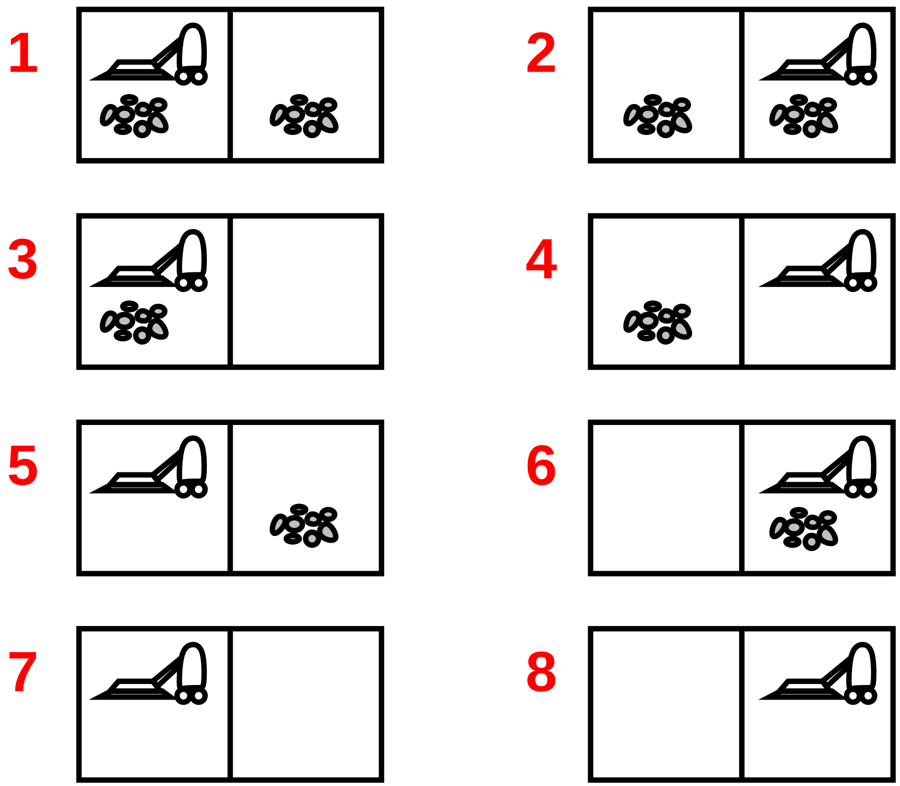


Example: Vacuum World

Conformant problem, initial state $= \{1, 2, 3, 4, 5, 6, 7, 8\}$

$Right \Longrightarrow \{2, 4, 6, 8\}$, $Left \Longrightarrow \{1, 3, 5, 7\}$, $Suck \Longrightarrow \{4, 5, 7, 8\}$

Solution?

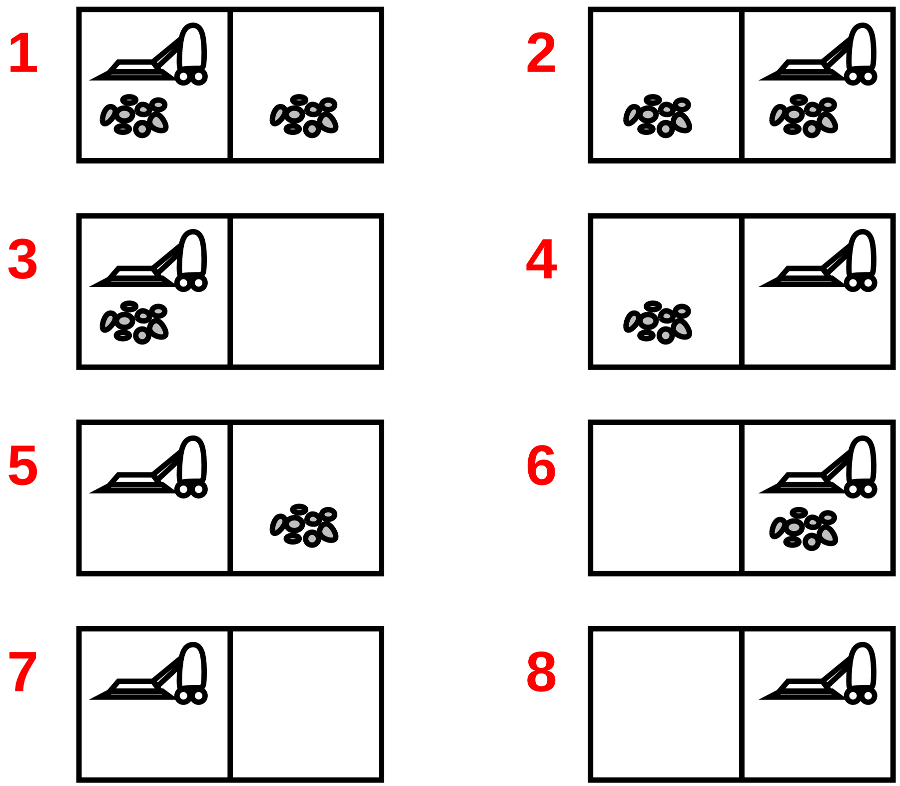


Example: Vacuum World

Conformant problem, initial state $= \{1, 2, 3, 4, 5, 6, 7, 8\}$

$Right \Longrightarrow \{2, 4, 6, 8\}$, $Left \Longrightarrow \{1, 3, 5, 7\}$, $Suck \Longrightarrow \{4, 5, 7, 8\}$

Solution? $[Right, Suck, Left, Suck]$

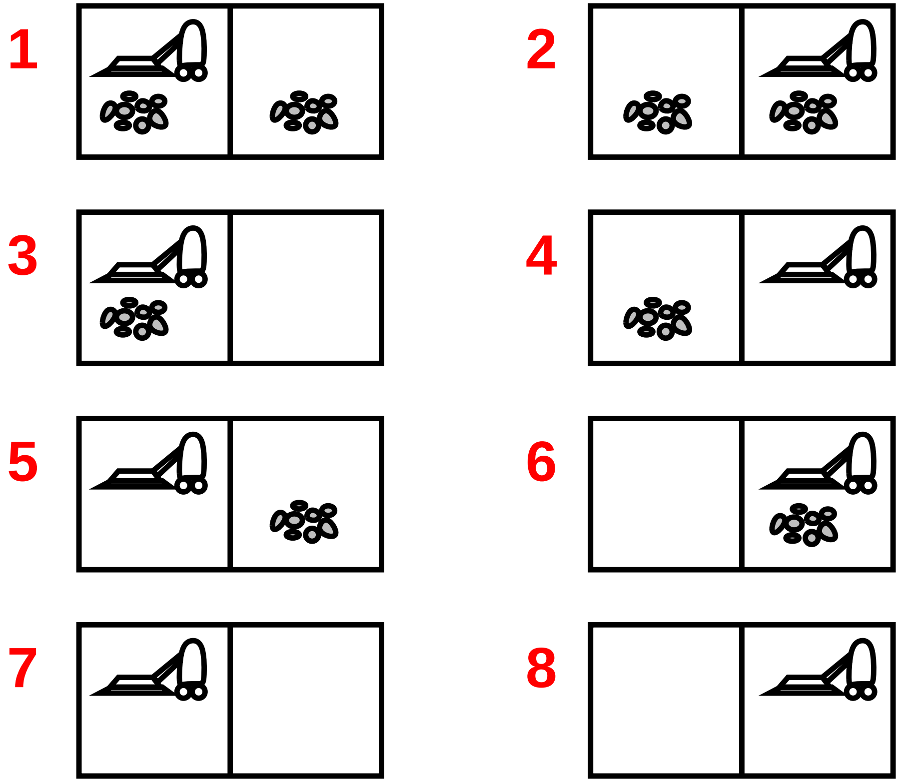


Example: Vacuum World

Contingency problem, initial state $= 5$

$Suck$ occasionally fails. Local sensing: dirt, location.

Solution?

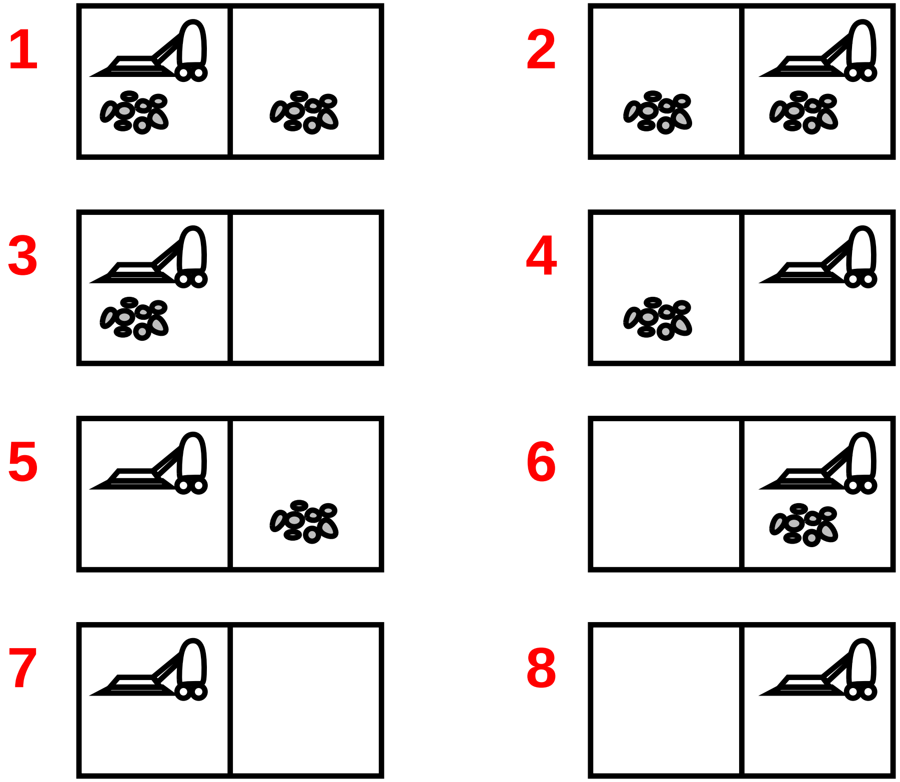


Example: Vacuum World

Contingency problem, initial state $= 5$

$Suck$ occasionally fails. Local sensing: dirt, location.

Solution? $[Right, \textbf{if } dirt \textbf{ then } Suck]$



Problem Solving

We start by considering the simpler cases in which the environment is fully observable, static and deterministic

In such environments the following holds for an agent A:

- A's world is representable by a discrete set of states
- A's actions are representable by a discrete set of operators
- the next world state is completely determined by the current state and A's actions
- the world's state transitions are caused exclusively by A's actions

Single-state Problem Formulation

Formally, a problem is defined by four components:

- An initial state (eg, $In(Arad)$)
- A successor function S returning sets of action–state pairs (eg, $S(Arad) = \{\langle GoTo(Zerind), In(Zerind)\rangle, \ldots\}$)
- A goal test, explicit (eg, $x = In(Bucharest)$) or
  implicit, (eg, $NoDirt(x)$)
- A path cost
  (eg, sum of distances, number of actions executed, ...) Usually additive and given as $c(x, a, y)$, the step cost from x to y by action a, assumed to be ≥ 0

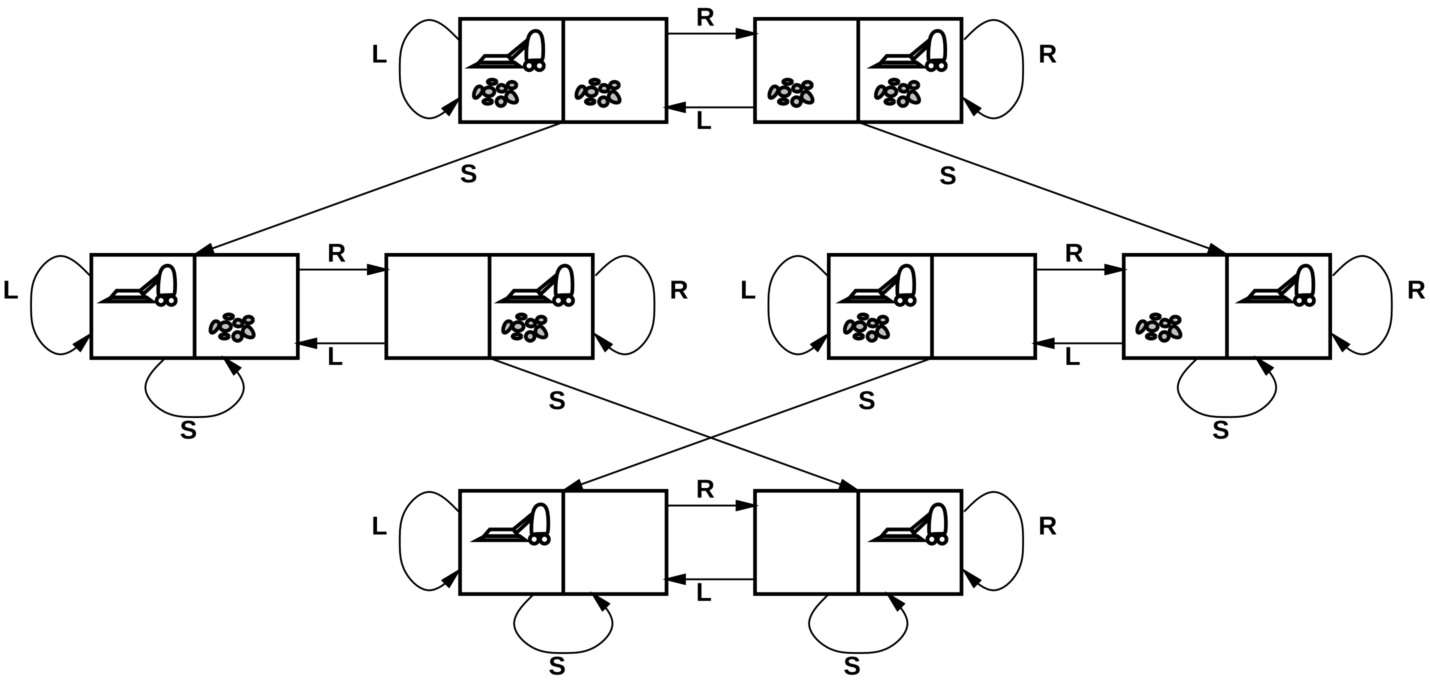
A solution is a sequence of actions leading from the initial state to a goal state

Selecting a State Space

Since the real world is absurdly complex the state space must be abstracted for problem solving.

- Abstract state = set of real states
- (Abstract) action = complex combination of real actions eg, $GoTo(Zerind)$ from $Arad$ represents a complex set of possible routes, detours, rest stops, etc.
  - For guaranteed realizability, any real state corresponding to $In(Arad)$ must get to some real state corresponding to $In(Zerind)$
  - Each abstract action should be "easier" than the original problem!
- (Abstract) solution = set of real paths that are solutions in the real world

Example: vacuum world state space graph



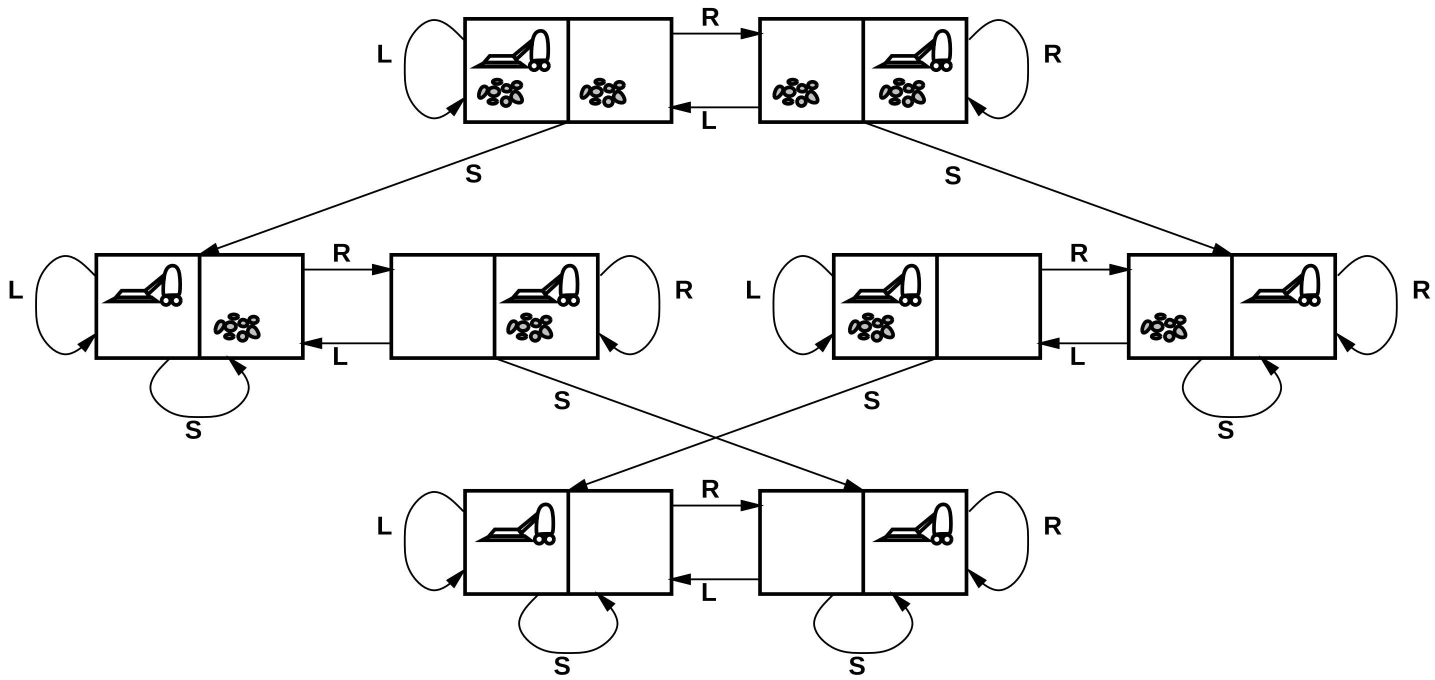
States?

Actions?

Goal test?

Path cost?

Example: vacuum world state space graph



States? ⟨dirt flag, robot location⟩ (ignore dirt amount)

Actions? *Left*, *Right*, *Suck*, *NoOp*

Goal test? $\neg dirty$

Path cost? 1 per action (0 for *NoOp*)

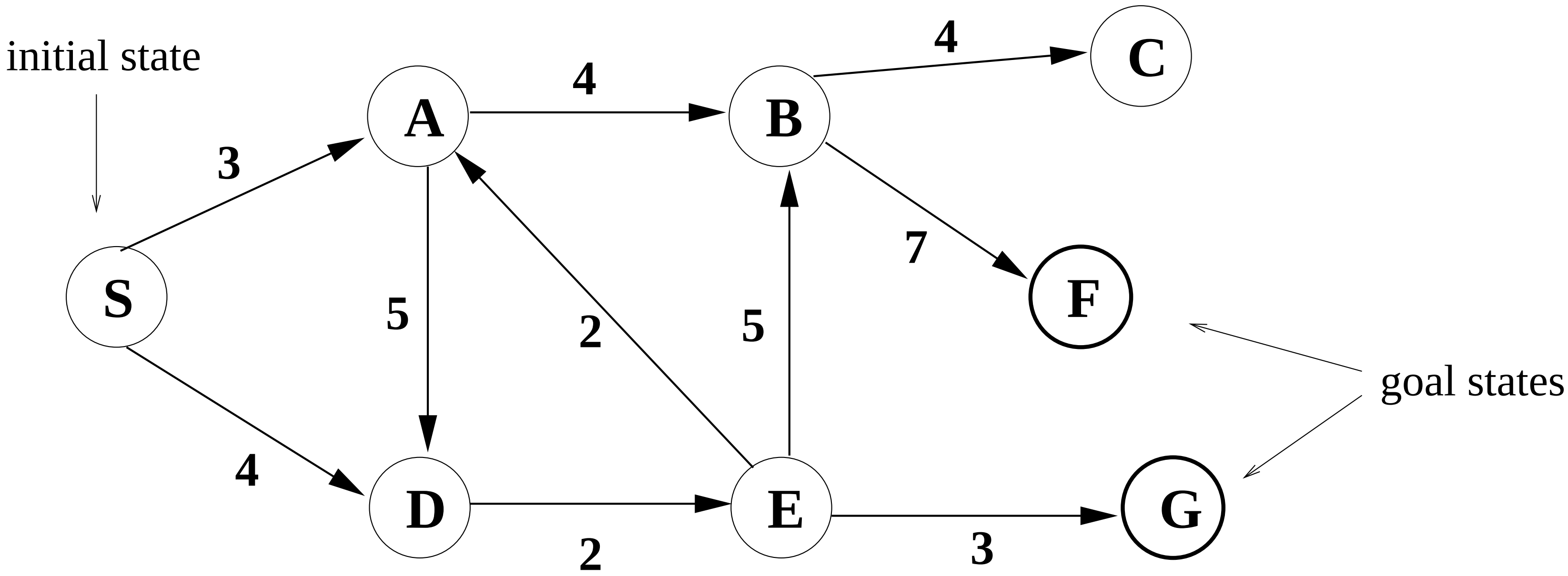
Formulating Problem as a Labeled Graph

In the graph

- each node represents a possible state
- a node is designated as the initial state
- one or more nodes represent goal states, states in which the agent's goal is considered accomplished
- each edge represents a state transition caused by a specific agent action
- associated to each edge is the cost of performing that transition

Search Graph

How do we reach a goal state?



There may be several possible ways. Or none!

Factors to consider:

- cost of finding a path
- cost of traversing a path

Problem Solving as Search

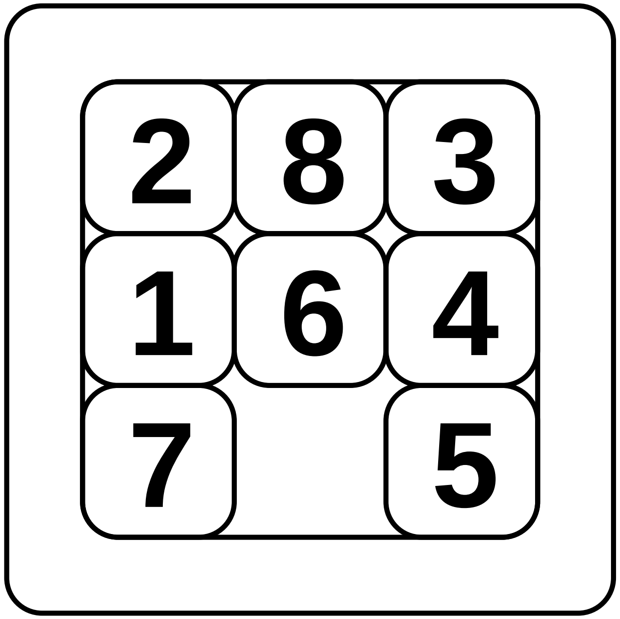
Search space: set of states reachable from an initial state S_0 via a (possibly empty/finite/infinite) sequence of state transitions

To achieve the problem's goal

1. search the space for a (ideally optimal) sequence of transitions starting from S_0 and leading to a goal state
2. execute (in order) the actions associated to each transition in the identified sequence

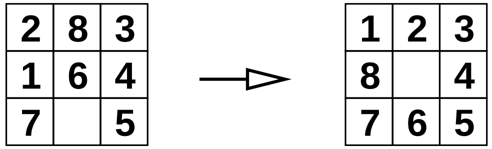
For contingency problems, two steps above need to be interleaved

Example: The 8-puzzle

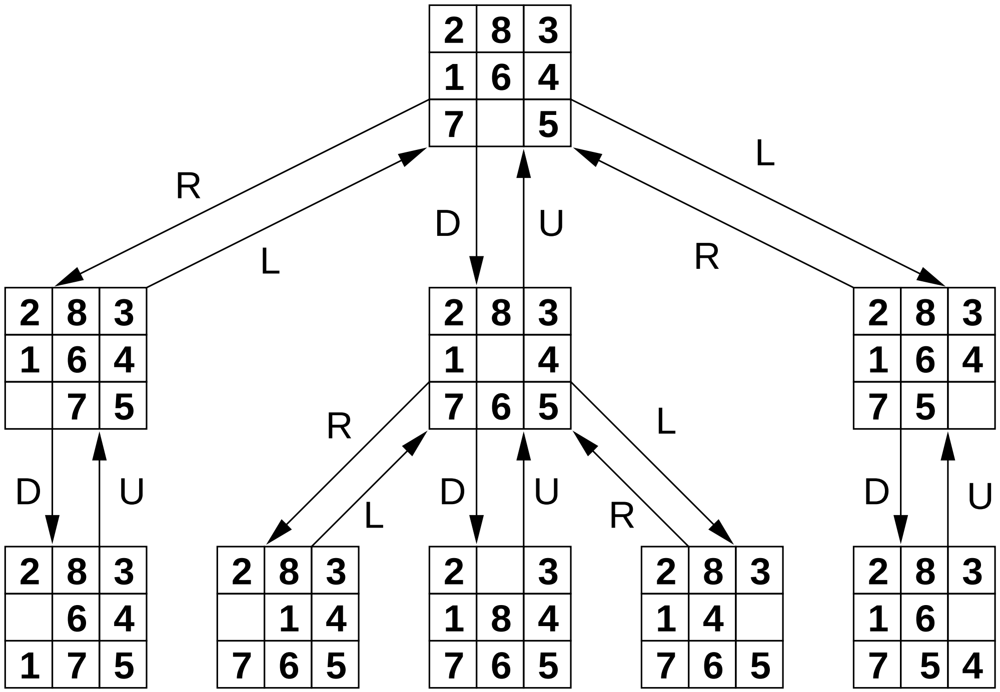


Example: The 8-puzzle

Problem: Go from state S to state G.



(S) (G)



Example: The 8-puzzle

States: configurations of tiles

Operators: move one tile Up/Down/Left/Right

Note:

- There are $9! = 362,880$ possible states: all permutations of $\{0,1,2,3,4,5,6,7,8\}$ where 0 is the empty space
- Not all states are directly reachable from a given state

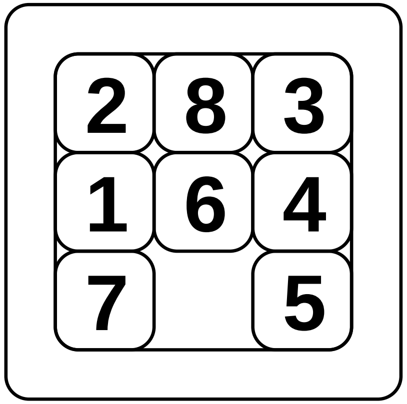
How can an artificial agent represent the states and the state space for this problem?

Problem Formulation

1. Choose an appropriate data structure to represent the world states
2. Define each operator as a precondition/effects pair where the
   - precondition holds exactly in the states the operator is applicable to
   - effects describe how a state changes into a successor state by the application of the operator
3. Specify an initial state
4. Provide a description of the goal—to check if a reached state is a goal state

Formulating the 8-puzzle Problem

States: each represented by a 3×3 array of numbers in $[0 \ldots 8]$, where value 0 is for the empty cell



becomes $\quad A = \begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix}$

Formulating the 8-puzzle Problem

- Operators: 24 operators of the form $OP_{r,c,d}$ where $r, c \in \{1, 2, 3\}$, $d \in \{L, R, U, D\}$
- If the empty space is at position (r, c), $OP_{r,c,d}$ moves it in direction d

Example:

$$\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix} \overset{OP_{3,2,L}}{\Longrightarrow} \begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 0 & 7 & 5 \end{vmatrix}$$

Preconditions and Effects

Example: $OP_{3,2,R}$

$$\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix} \overset{OP_{3,2,R}}{\Longrightarrow} \begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 5 & 0 \end{vmatrix}$$

Preconditions: $A[3,2] = 0$

Effects: $\begin{cases} A[3,2] & \leftarrow & A[3,3] \\ A[3,3] & \leftarrow & 0 \end{cases}$

Preconditions and Effects

Example: $OP_{3,2,R}$

$$\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix} \stackrel{OP_{3,2,R}}{\Longrightarrow} \begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 5 & 0 \end{vmatrix}$$

Preconditions: $A[3,2] = 0$

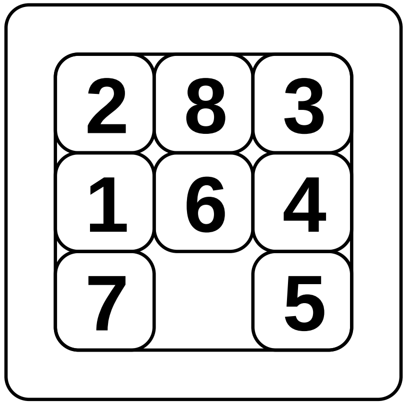
Effects: $\begin{cases} A[3,2] & \leftarrow & A[3,3] \\ A[3,3] & \leftarrow & 0 \end{cases}$

We have 24 operators in this problem formulation ... 20 too many!

A Better Formulation

States: each represented by a pair $(A, (i, j))$ where:

- A is a 3×3 array of numbers in $[0 \ldots 8]$
- (i, j) is the position of the empty space (0) in the array



becomes $\left(\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix}, (3, 2) \right)$

A Better Formulation

Operators: 4 operators of the form OP_d where $d \in \{L, R, U, D\}$

OP_d moves the empty space in the direction d

Example:

$$\left(\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix}, (3,2) \right) \quad \stackrel{OP_L}{\Longrightarrow} \quad \left(\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 0 & 7 & 5 \end{vmatrix}, (3,1) \right)$$

Preconditions and Effects

Example: OP_L

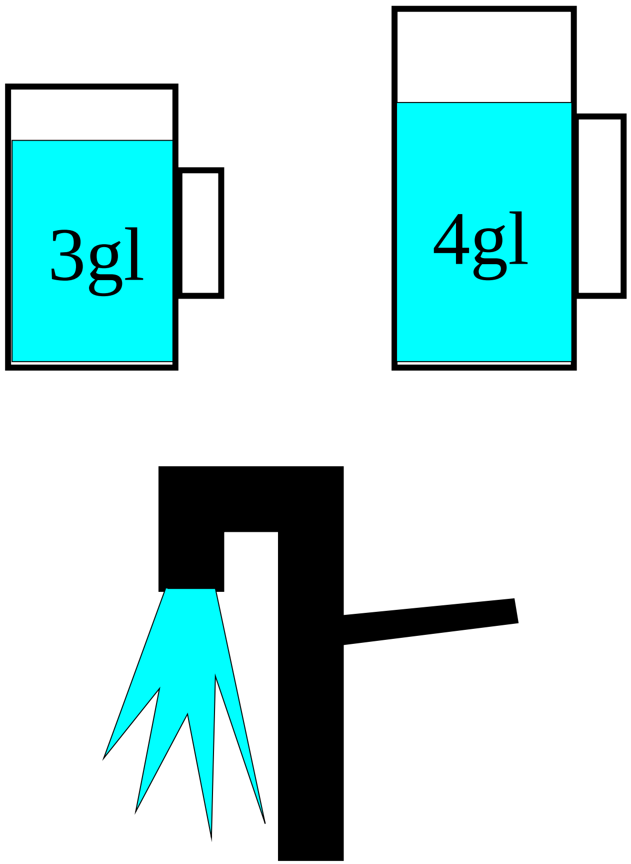
$$\left(\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 7 & 0 & 5 \end{vmatrix}, (3,2) \right) \xrightarrow{OP_L} \left(\begin{vmatrix} 2 & 8 & 3 \\ 1 & 6 & 4 \\ 0 & 7 & 5 \end{vmatrix}, (3,1) \right)$$

Let (r_0, c_0) be the position of 0 in A

Preconditions: $c_0 > 1$

Effects:
$$\begin{cases} A[r_0, c_0] & \leftarrow & A[r_0, c_0 - 1] \\ A[r_0, c_0 - 1] & \leftarrow & 0 \\ (r_0, c_0) & \leftarrow & (r_0, c_0 - 1) \end{cases}$$

The Water Jugs Problem



Get exactly 2 gallons of water into the 4gl jug

The Water Jugs Problem

States: Determined by the amount of water in each jug

State Representation: Two real-valued variables, J_3, J_4, indicating the amount of water in the two jugs, with the constraints:

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

Initial State Description

$$J_3 = 0, \qquad J_4 = 0$$

Goal State Description:

$$J_4 = 2 \qquad \text{(non exhaustive description)}$$

The Water Jugs Problem: Operators

E4: empty jug4 on the ground

precond: $J_4 > 0$ **effect:** $J'_4 = 0$

E4-3: pour water from jug4 into jug3 until jug3 is full

precond: $J_3 < 3,$ $J_4 \geq 3 - J_3$ **effect:** $J'_3 = 3,$ $J'_4 = J_4 - (3 - J_3)$

P3-4: pour water from jug3 into jug4 until jug4 is full

precond: $J_4 < 4,$ $J_3 \geq 4 - J_4$ **effect:** $J'_4 = 4,$ $J'_3 = J_3 - (4 - J_4)$

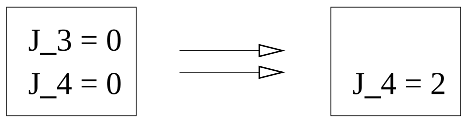
E3-4: pour water from jug3 into jug4 until jug3 is empty

precond: $J_3 + J_4 < 4,$ $J_3 > 0$ **effect:** $J'_4 = J_3 + J_4,$ $J'_3 = 0$

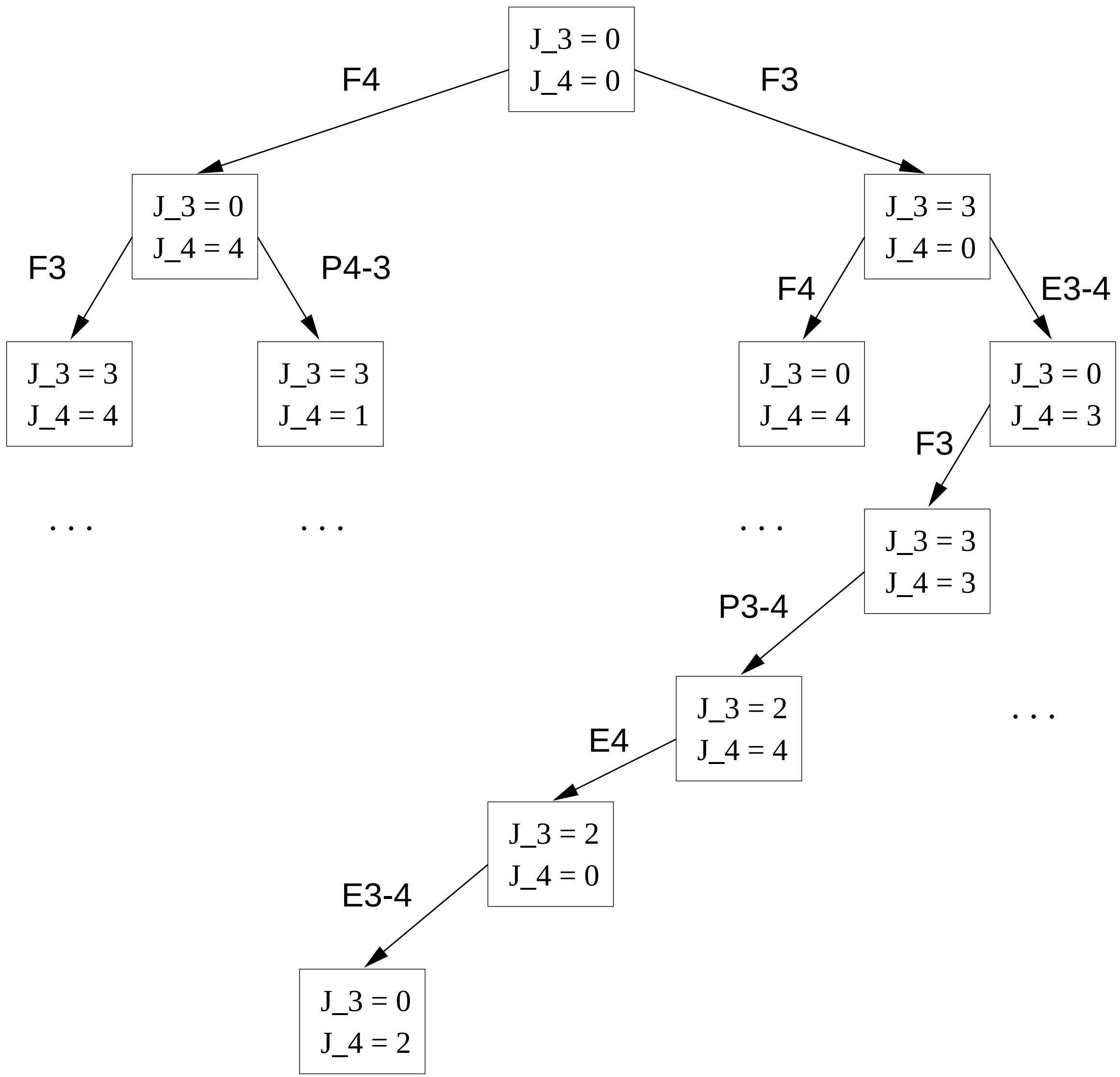
...

The Water Jugs Problem

Problem



Search Graph



Real-World Search Problems

- Route Finding
  (computer networks, airline travel planning system, ...)
- Travelling Salesman Optimization Problem
  (package delivery, automatic drills, ...)
- Layout Problems
  (VLSI layout, furniture layout, packaging, ...)
- Assembly Sequencing
  (assembly of electric motors, ...)
- Task Scheduling
  (manufacturing, timetables, ...)
- ...

Problem Solution

Typically, a problem's solution is a *description of how to reach a goal* state from the initial state:

Examples:

- n-puzzle
- route-finding problem
- assembly sequencing

Occasionally, a problem's solution is simply a *description of the goal* state itself:

Examples:

- 8-queen problem
- scheduling problems
- layout problems