Title: Solving Problems by Searching

AIMA: Chapter 3 (Sections 3.1, 3.2 and 3.3)

Introduction to Artificial Intelligence

CSCE 476-876, Spring 2002

URL: `www.cse.unl.edu/~choueiry/S02-476-876`

Berthe Y. Choueiry (Shu-we-ri)

`choueiry@cse.unl.edu`, `(402)472-5444`

List of terms for glossary:

Given: Friday, Feb 1, 2001

Due date: Monday, Feb 4, 2001

Goal test	(1 point)
Initial state	(1 point)
Missionaries & cannibals problem	(1 point)
n-queens problem	(1 point)
Operator	(1 point)
Path	(1 point)
Path cost	(1 point)
Problem: single-state, multiple-state, contingency, exploration	(4 points)
Search	(1 point)
Search cost (vs. total cost)	(1 point)
State set space	(1 point)
State space	(1 point)
Successor function (in search)	(1 point)
Toy problems	(1 point)
Traveling salesperson problem (TSP)	(1 point)

Summary

Intelligent Agents

- Definition and components (PAGE)
- Types of Intelligent Agents
  1. Self Reflex
  2. ?
  3. ?
  4. ?
- Types of environments:
  accessible or not, effectively accessible, deterministic or not, episodic or not, static vs. dynamic, discrete vs. continuous.

Outline

- Problem-solving agents
- Formulating problems
  - Problem types
  - Problem components
  - Importance of modeling

Simple reflex agent unable to plan ahead

- actions limited by current percepts
- no knowledge of what actions do
- no knowledge of what they are trying to achieve

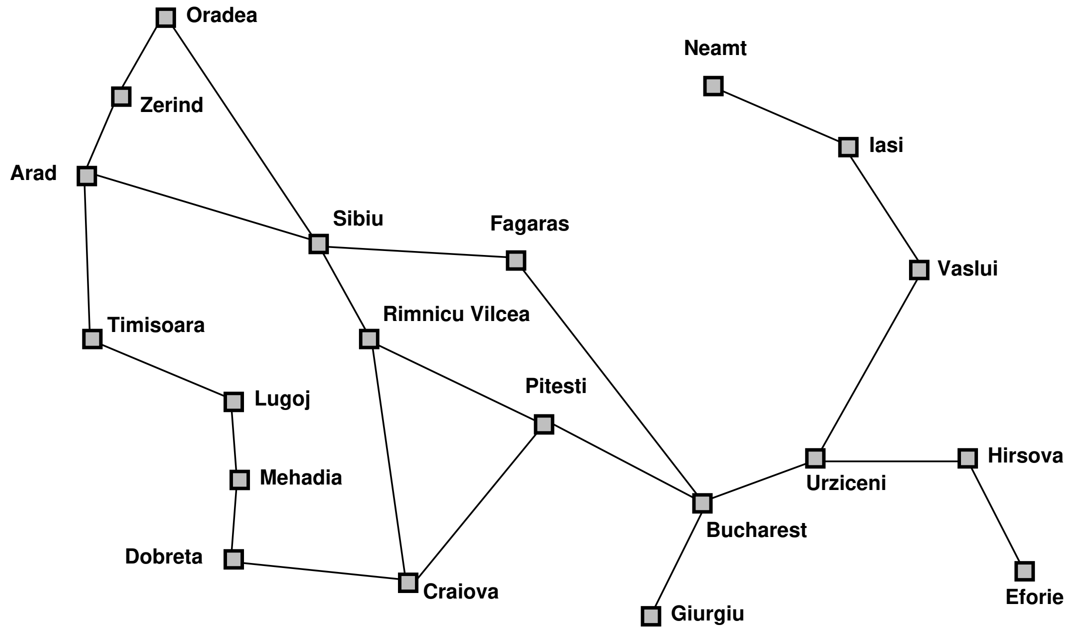
Problem-solving agent: goal-based agent

Given:

- a problem formulation: a set of states and a set of actions
- a goal to reach/accomplish

Find:

- a sequence of actions leading to goal



Example: Holiday in Romania

On holiday in Romania, currently in Arad, want to go to Bucharest

Formulate goal:

be in Bucharest

Formulate problem:

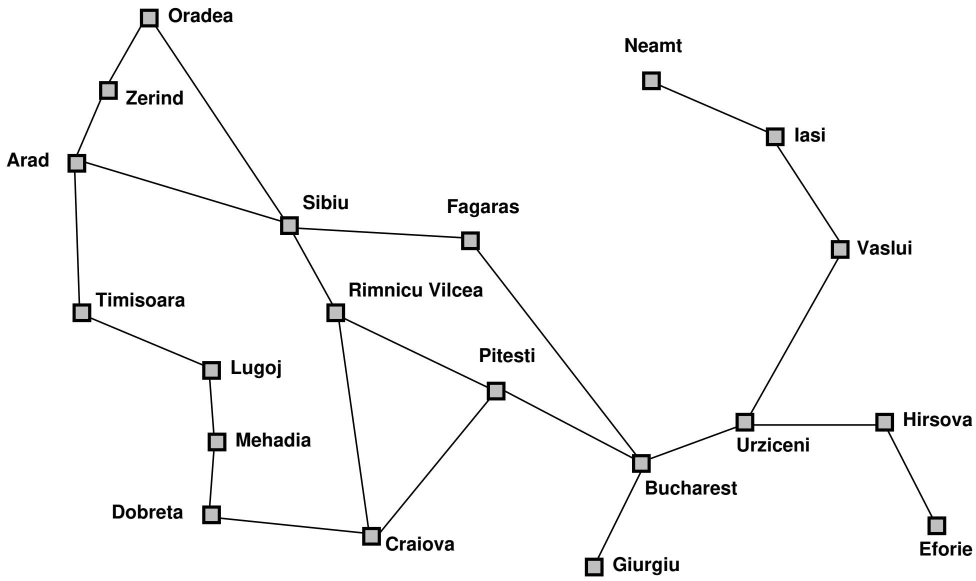
states: various cities

operators: drive between cities

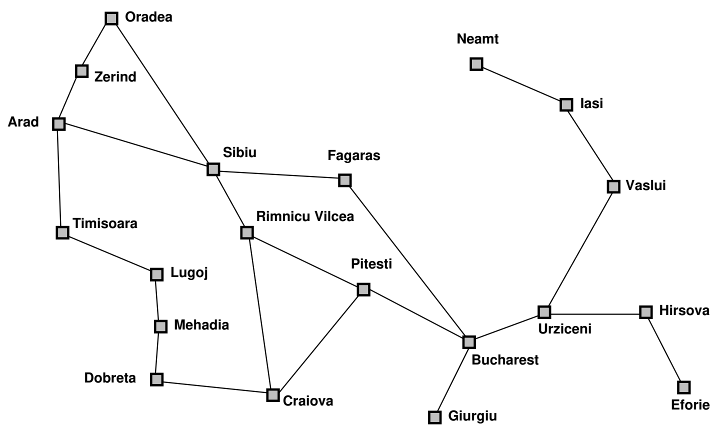
Find solution:

sequence of cities, *e.g.* Arad, Sibiu, Fagaras, Bucharest

Drive to Bucharest... how many roads out of Arad?



Use map to consider hypothetical journeys through each road until reaching Bucharest



Looking for a sequence of actions ⟶ search

Sequence of actions to goal ⟶ solution

Carrying out actions ⟶ execution phase

Formulate, search, execute

Formulate, search, execute

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

  state ← UPDATE-STATE(state, p)
  if s is empty then
     g ← FORMULATE-GOAL(state)
     problem ← FORMULATE-PROBLEM(state, g)
     s ← SEARCH(problem)
  action ← RECOMMENDATION(s, state)
  s ← REMAINDER(s, state)
  return action
```

×	Update-State	×	Formulate-goal
√	Formulate-Problem	√	Search
	Recommendation: first		Remainder: rest

Four types of problems

depending of the amounts of knowledge agent has of its $\begin{cases}\text{state}\\ \text{actions}\end{cases}$

1. Single-state problems
2. Multiple-state problems
3. Contingency problems
4. Exploration problems

Example: vacuum cleaner

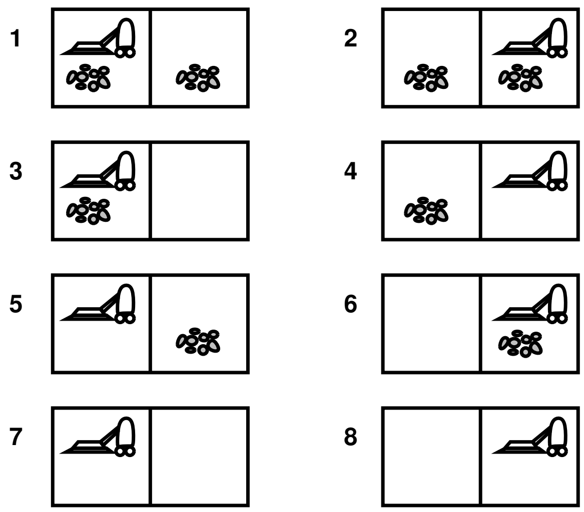
World:

- two locations
- dirt
- vacuum cleaner
- 8 states

Actions:

- *Left*
- *Right*
- *Suck* (100% effective)

Goal: clean up. States?



Single-state problem

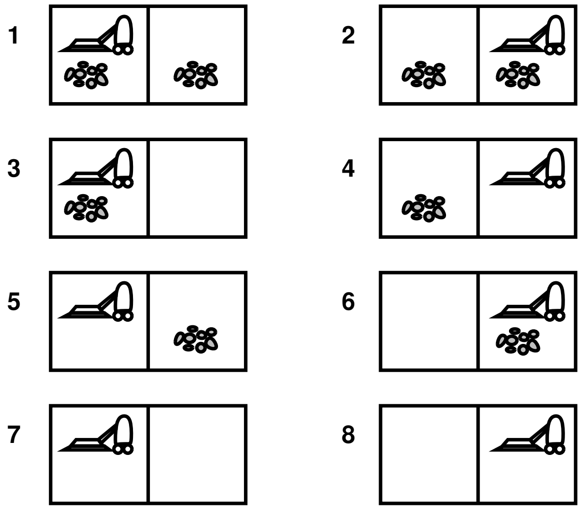
Agent knows exactly:

- where it is
- effects of actions

→ Environment properties?

Agent can exactly compute new state after any action

Action: 1 state ⟶ 1 state



Multiple-state problem

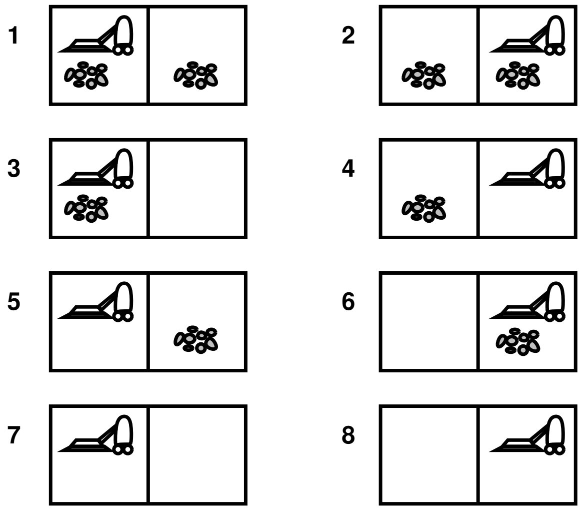
Agent $\left\{\begin{array}{l}\text{has limited access to world} \\ \textit{e.g.}\text{, extreme case: no sensors at all. Hopless?} \\ \text{knows exactly effects of actions}\end{array}\right.$

→ Environment properties?

→ Effect of *Right*?
→ [*Right*, *Suck*, *Left*, *Suck*]?

Agent must reason about *sets of states*

Action: state → state set

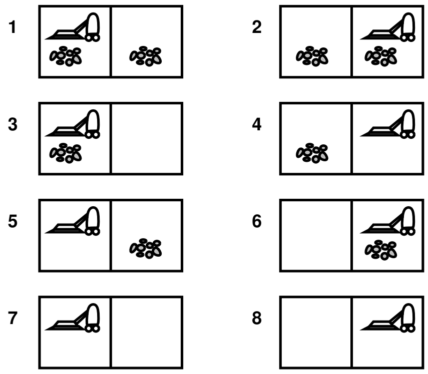


Contingency problem

Agent
- has limited access to world
- *e.g.*, position sensor, local dirt-sensor
- does not always know effects of actions
- *e.g.*, *sometimes* deposits dirt only $\nexists$ dirt there already

$\longrightarrow$ Environment properties?

Sensor: state = $\{1, 3\}$
Action: [Suck, Right, Suck]
New state set?
$\rightarrow$ No fixed sequence for success



Suck if sensor senses dirt
Monitor during execution

Open your eyes while walking..

Exploration problem

Agent
- has limited access to world
- <u>no information</u> about effects of actions
- *e.g.*, left in a new city with no map

$\longrightarrow$ State space completely unknown

No model to support search, must experiment

Solution techniques

Single-state problems
Multiple-state problems } **Search**

Contingency problems: **Planning**

must act before a full plan can be developed,
must use sensors during execution
start execution and consider contingencies as they arise
often interleave search and execution

Exploration problems: **Reinforcement learning**

to survive, must learn a model of the environment,
which can be used to solve subsequent problems

Single-state problem formulation

A *problem* is defined by four items:

1. *initial state*
   e.g., "at Arad"
2. *operators* (or *successor function* $S(x)$)
   e.g., Arad $\rightarrow$ Zerind, Arad $\rightarrow$ Sibiu, etc.
3. *goal test*, can be
   explicit, e.g., $x =$ "at Bucharest"
   implicit, e.g., $NoDirt(x)$
4. *path cost* (additive)
   e.g., sum of distances, number of operators executed, etc.

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

Multiple-state problem formulation

state space $\longrightarrow$ state <u>set</u> space

1. initial state *set*
2. operators specify actions: state $\rightarrow$ set of states
3. goal test: same
4. path cost: same

When operator: set (1) of states $\rightarrow$ set (2) of states
(2) is the union of all possible sets of states reachable from each of the states in (1)

A solution is a path to a set of states <u>all of which are goal states</u>

Problem solving performance

Measures for <u>effectiveness</u> of search:

1. Does it find a solution? — complete
2. Is it a good solution? — path cost low
3. Search cost? — time & space

Total cost = Search cost + Path cost
$\longrightarrow$ problem?

Example: Arad to Bucharest
Path cost: total mileage, fuel, tire wear f(route), etc.
Search cost: time, computer at hand, etc.

Importance of modeling

Real art of problem solving is modeling,

deciding what's in $\begin{cases} \text{state description} \\ \text{action description} \end{cases}$

choosing the right level of abstraction

State abstraction: road maps, weather forecast, traveling companions, scenery, radio programs, ...

Action abstraction: generate pollution, slowing down/speeding up, time duration, turning on the radio, ..

Combinatorial explosion. Abstraction by removing irrelevant detail make the task easier to handle

Example problems

Toy Problems:

$\rightarrow$ intended to illustrate or exercise $\begin{cases} \text{concepts} \\ \text{problem-solving methods} \end{cases}$

$\surd$ can be give concise, exact description

$\surd$ researchers can compare performance of algorithms

$\times$ yield methods that rarely scale-up

$\times$ may reflect reality inaccurately (or not at all)

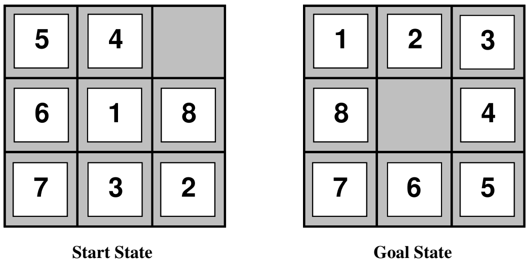
Real-world Problems:

$\rightarrow$ more difficult but whose solutions people actually care about

$\surd$ more credible, useful for practical settings

$\times$ difficult to model, rarely agreed-upon descriptions

Toy problem: 8-puzzle



States:

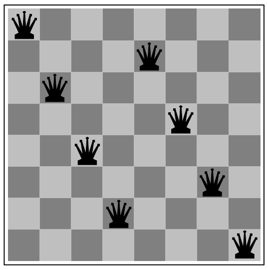
Operators:

Goal test:

Path cost:

→ instance of sliding-block puzzles, known to be **NP**-complete
→ Optimal solution of n-puzzle **NP**-hard
→ so far, nothing better than search
→ 8-puzzle, 15-puzzle traditionally used to test search algorithms

Toy problem: n-Queens



→ Formulation: incremental vs. complete-state

States: Any arrangement of $x \leq 8$ queens on board

Operators: add a queen (alt., move a queen)

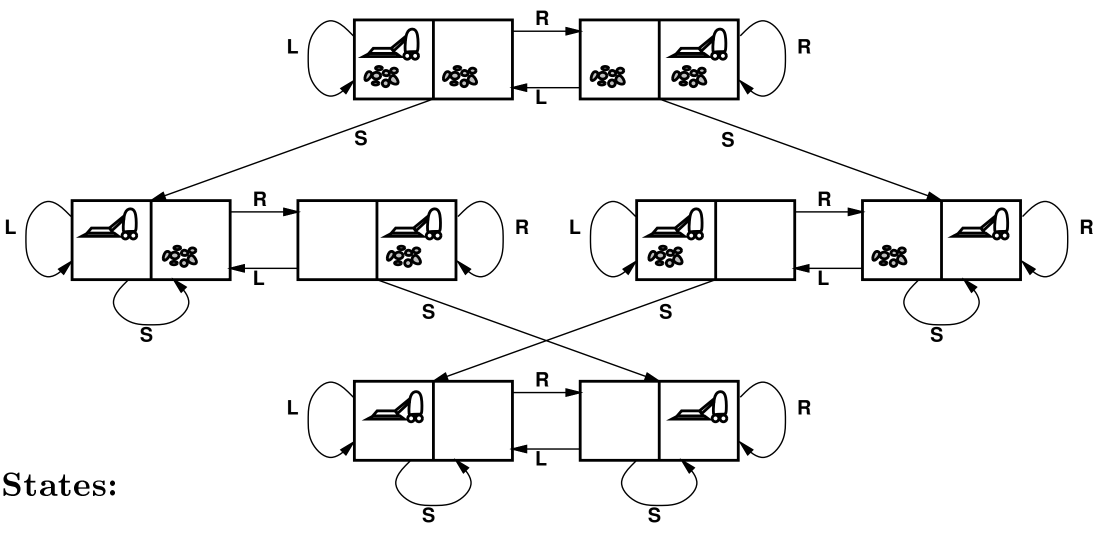
Goal test: 8 queens not attacking one another

Path cost: irrelevant (only final state matters)

→ 64^8 possible states
→ ∃ other more effective formulations (2057 states), see AIMA

Toy problem: vacuum

Single state case



States:

Operators:

Goal test:

Path cost:

Toy problems: requiring search

√	8 puzzles
√	n-queens
√	vacuum, single-state case
AIMA:	vacuum, multiple-state case
AIMA:	Missionaries & cannibals

Real-world problems: requiring search

- Route finding: state = locations, actions = transitions
  routing computer networks, travel advisory, etc.
- Touring: start in Bucharest, visit every city at least once
  Traveling salesperson problem (TSP) (exactly once, shortest tour)
- VLSI layout: cell layout, channel layout
  minimize area and connection lengths to maximize speed
- Robot navigation
- Assembly by Robot-arm
  States: robot joint angles, robot and parts coordinates
  Operators: continuous motions of the robot joins
  goal test: complete assembly
  path cost: time to execute