IART - Artificial Intelligence

Lecture 2a: Search Problems

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.

Associate Professor at DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Portugal

President of APPIA – Portuguese Association for Artificial Intelligence

Problem Solving using Search

Presentation Structure:

- Problem Solving Methods
- Problem Formulation
- State Space
- Blind/Uninformed Search:
  - Breadth First, Depth First, Uniform Cost, Iterative Deepening, Bidirectional Search
- Intelligent/Informed Search:
  - Greedy Search, A* Algorithm
- Practical Application Examples

Problem Solving using Search

- **How can an agent act, setting goals and considering possible sequences of actions to achieve those goals!**
- **Problem Solving:**
  – Formulating a problem as a search problem
  – Uninformed search (search strategies)
  – Informed Search (greedy search, A* Algorithm)
  – Iterative Improvement Algorithms
  – Search in Games (including a hostile agent!)

Agents for Problem Solving

- **"Problem Solving Agent": Tries to find the sequence of actions that leads to a desirable state!**
- **Problem Formulation:**
  - What are the possible actions? (what is its effect on the state of the world?)
  - What are the possible states? (how to represent them?)
  - How to evaluate states
- Search problem
  - Solution: sequence of actions Final stage is execution!
- **Formulate $\rightarrow$ Search $\rightarrow$ Execution**

Search Problems

- **Many of the problems in computer science can be defined as:**
  - A set S of STATES (possibly infinite)
  - An INITIAL state: s € S
  - A TRANSITION relationship T across this state space
  - A set of FINAL (objective) states: O € S
- **Problem can be represented by a GRAPH, where the nodes represent states and the arcs (connections) the pairs of the transition relation**
- **Problem can be solved through search finding a path between the initial state and an objective state**

Agents for Problem Solving

- **Problem Formulation:**
  - State Representation
  - Initial (Current) State
  - Objective Test (defines the desired states)
  - Operators (Name, Preconditions, Effects, cost)
  - Solution Cost

Simple Agents for Problem Solving

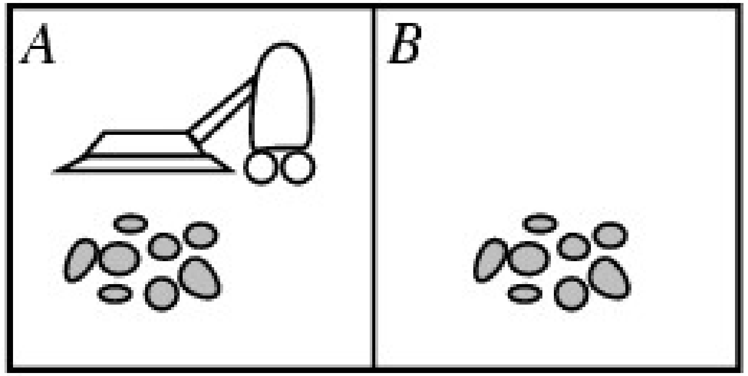
```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
   static: seq, an action sequence, initially empty
           state, some description of the current world state
           goal, a goal, initially null
           problem, a problem formulation

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

Problem Formulation

- **What is the agent's knowledge of the state of the world and his actions?**
- **Four different types of problems:**
  - Single state problems (deterministic and accessible environment)
  - Multiple state problems (deterministic but inaccessible environment)
  - Contingency problems (non-deterministic and inaccessible environment, sensors must be used during execution, solution is a tree or policy)
  - Exploration problems (unknown state space)

Example: Vacum Cleaner Problem

- **2 localizations, 3 Actions (left, right, suck), 8 possible States, Objective: clear the garbage!**
- **Problem:**



  - Single state problem if… (deterministic and accessible)
  - Multiple state problem if … (deterministic but inaccessible)
  - Contingency problem if … (non-deterministic and inaccessible)
  - Exploration problem if (unknown state space)

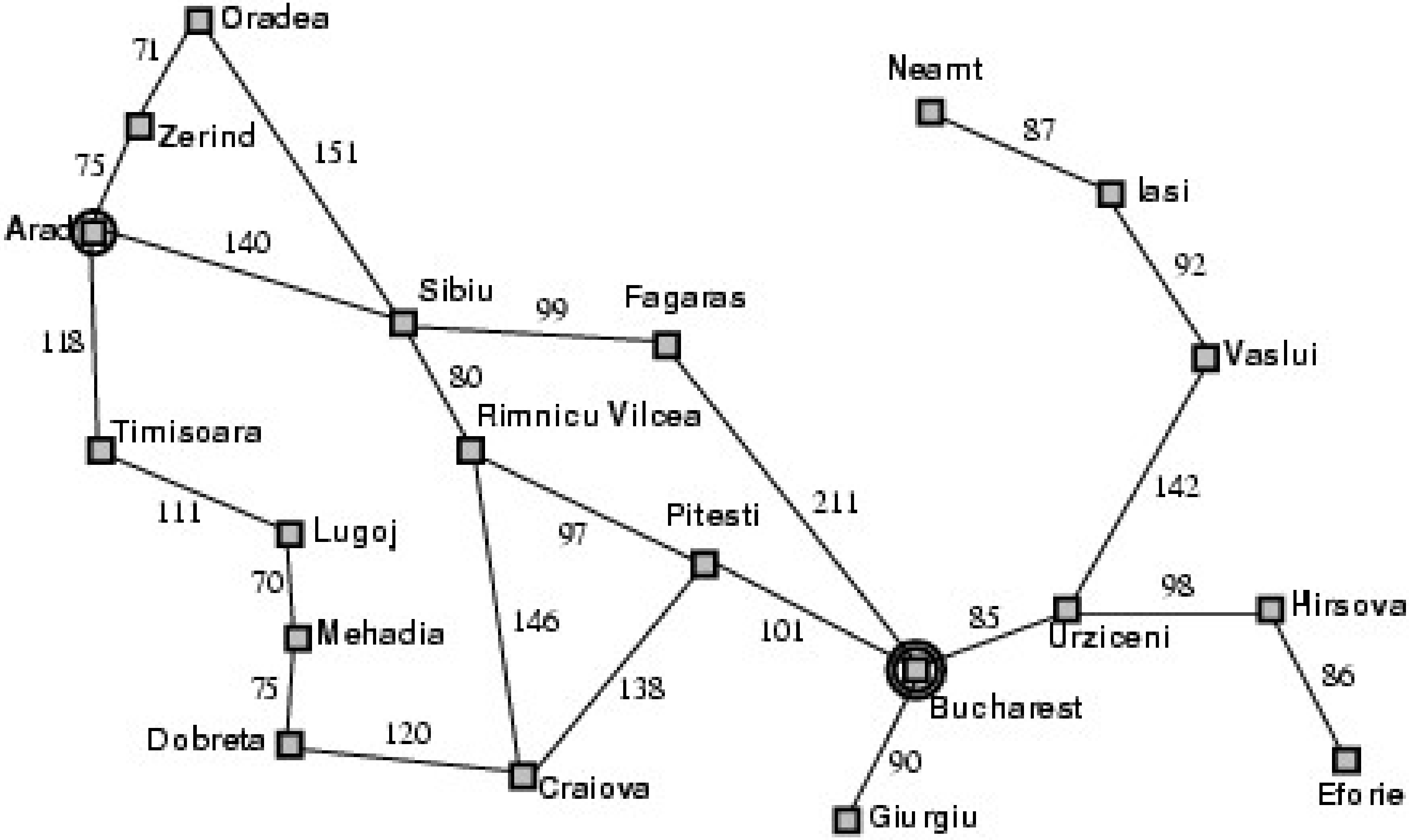
Well Defined Problems (single state)

- **Problem: Collection of information the agent will use to decide what to do!**
- **Problem Formulation:**
  - **State Space:**
    - **Initial State**
    - **Possible Actions Set (operators, successors role)**
  - **Goal Testing**
  - **Solution Cost Function**
- **datatype** PROBLEM

  components: INITIAL-STATE, OPERATORS, GOAL-TEST, PATH-COST-FUNCTION
- **Solution: Path from initial state to goal**
- **Total Cost = Solution Cost + Search Cost**

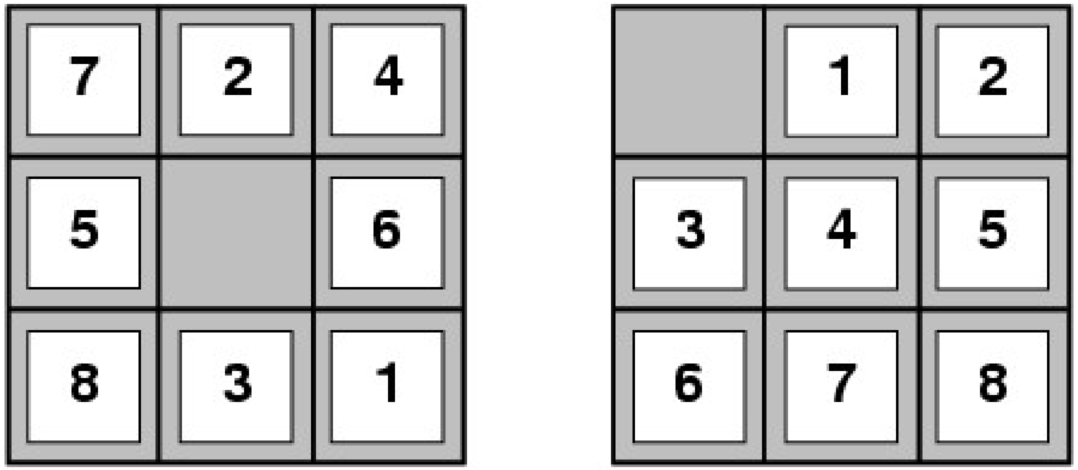
Example: Street Map or Romania

- Initial State: Arad; Objective State: Bucharest
- Operators: Arcs with respective costs



Example: N-Puzzle Problem

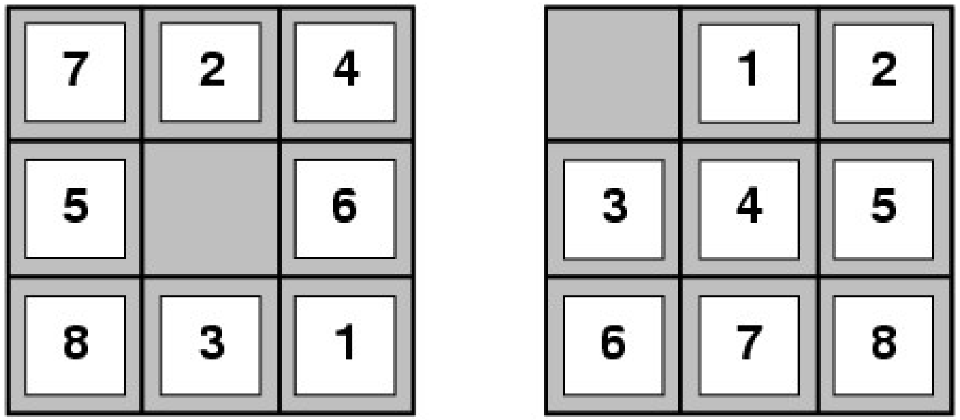
- **States, Operators, Objective Testing, Cost of Solution**



Start State   Goal State

Example: N-Puzzle Problem

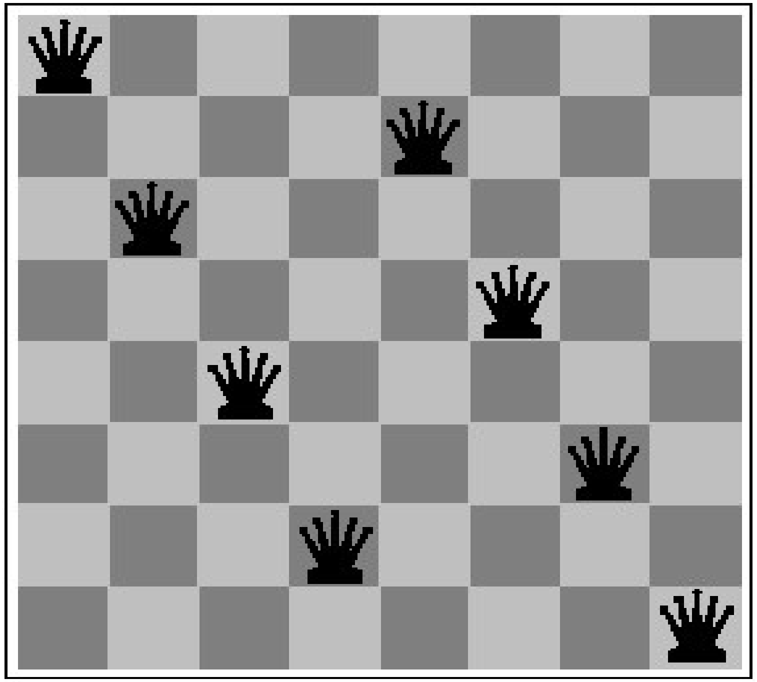
- **States, Operators, Objective Testing, Cost of Solution**
  - States: Specifies the position of each piece and the empty space (several representations are possible)
  - Initial state: Represented in the figure
  - Successor operators: generates valid states that result from execution. These are the four actions (move empty space left, right, up or down)
  - Objective test: Checks whether the state corresponds to the objective configuration (represented in the figure)
  - Solution cost: Each step costs 1, the cost of the solution being the number of steps to solve the problem



Start State Goal State

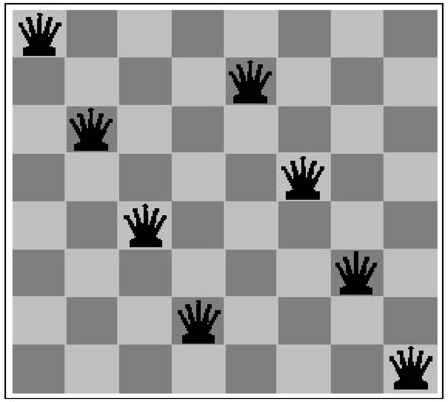
Example: N-Queens Puzzle

- **States, Operators, Objective Testing, Cost of Solution**



Example: N-Queens Puzzle

- **Objective Test: 8 Queens on the board without any attack**
- **Solution Cost: 0 (all solutions are the same)**
- **Formulation 1:**
  - Status: Any arrangement from 0 to 8 Queens on the board
  - Operator: Add a queen to any square
  - We have 64^8 possible sequences!



- **Formulation 2:**
  - Status: Arrangements from 0 to 8 Queens, one in each column, no attacks!
  - Operator: Add a queen to the leftmost column that is empty, without attacking any other
  - We have 2057 possible sequences!
- **Formulation 3:**
  - Status: Arrangements of 8 Queens on the board, one in each column!
  - Operator: Move attacked queen to home from the same column

Cryptograms

- **Find digits (all different), one for each letter so that the sum is correct!**
- **States: Puzzle with some letters replaced by numbers**
- **Operators: Replace all occurrences of a letter with a digit**
- **Objective Test: Puzzle only contains digits and the sum is correct!**
- **Solution Cost: 0 (all solutions are the same)**

```
   F O R T Y
       T E N
+      T E N
----------------
   S I X T Y
```

$C_1\ C_2\ C_3\ C_4$

```
   S E N D
+ M O R E
------------------
 M O N E Y
```

Cryptograms

- **Cryptogram Solution:**
  - Cryptogram 1: F=2, O=9, R=7, T=8, Y=6, E=5, N=0, S=3, I=1, X=4
    - Any more solutions?
  - Cryptogram 2: S=9, E=5, N=6, D=7, M=1, O=0, R=8, Y=2

```
  F O R T Y
      T E N
+     T E N
-----------------
  S I X T Y
```

```
  2 9 7 8 6
      8 5 0
+     8 5 0
-----------------
  3 1 4 8 6
```

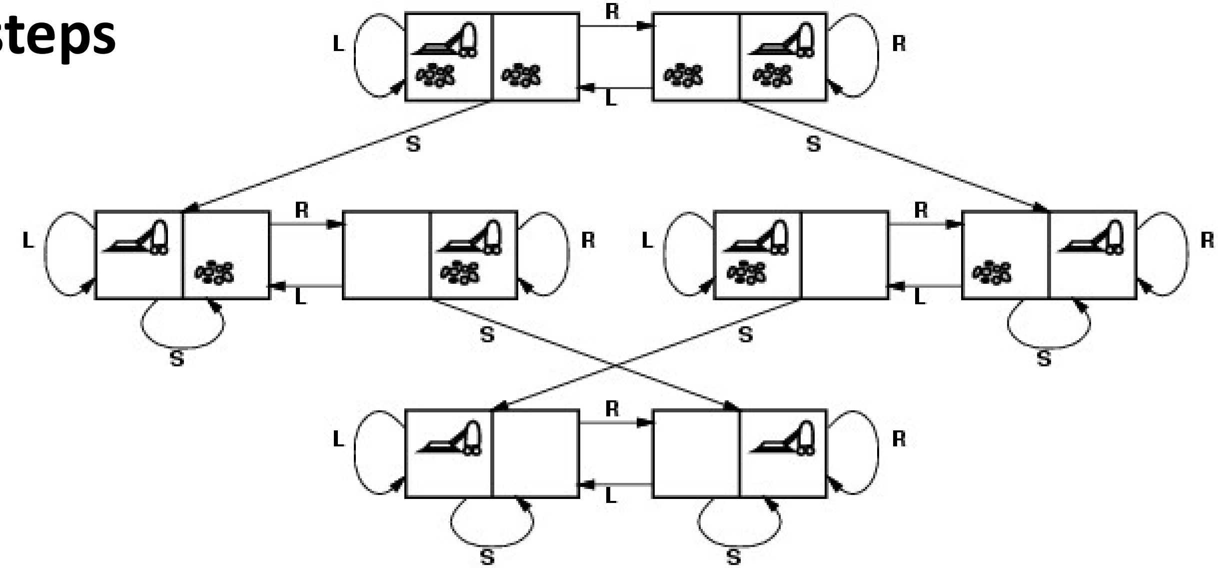
C_1 C_2 C_3 C_4

```
   S E N D
+ M O R E
-----------------
M O N E Y
```

```
  1 0 1 1
   9 5 6 7
+ 1 0 8 5
-----------------
1 0 6 5 2
```

Example: Vacum Cleaner Problem

- State: 8 states represented (defined by the position of the robot and garbage)
- Initial state: Any
- Operators: left, right, vacuum
- Objective Test: There is no trash in any of the squares
- Solution Cost: Each action costs 1 (total cost = number of solution steps



Example: Vacum Cleaner without Sensors!

- **Multiple State Problem**
  - Now we have in every instant a set of possible states!
- **Problem Formulation**
  - State Set: Subset of represented states
  - Operators: left, right and suck
  - Objective Test : All states in the set cannot have garbage
  - Solution Cost: Each action costs 1

Example of Real World Problems

- **Routes/Path Finding Problem**
  - Find the best route from one point to another (applications: google maps, computer networks, military planning, air travel)
  - Visit each point at least once in a given space (Ex: traveling salesman visit each city exactly once, find the shortest route)

Exercise: Missionaries and Cannibals

Missionaries and Cannibals Problem

3 missionaries and 3 cannibals are on one side of the river with a boat that only takes 2 people.

The objective is to find a way to take the 6 to the other side of the river without ever leaving more cannibals than missionaries on one of the banks during the process!

a) Formulate this problem as a search problem, defining the representation of the state, initial state, the operators (and respective preconditions and effects), the objective test and the cost of the solution.

b)Solve the problem through a tree search

Exercise: Bucket Filling Problem

Bucket Filling Problem

Two buckets, with capacities c1 (ex: 4 liters) and c2 (ex: 3 liters), respectively, are initially empty. The buckets have no intermediate markings.

The only operations you can perform are:

– empty a bucket
– fill (completely) a bucket
– pour one bucket into the other until the second is full
– pour one bucket into the other until the first is empty

The objective is to determine which operations to carry out so that the first bucket contains n liters (example: 2 liters)?

a) Formulate the Problem as a search problem

b) Solve the problem through a tree search

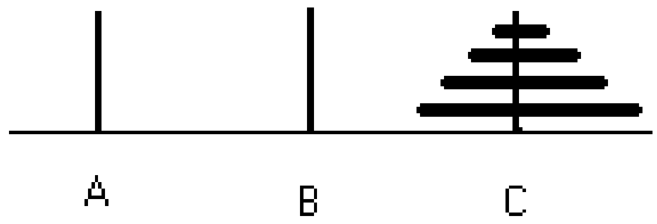
Exercise: Hanoi Towers

Hanoi Towers Problem

a) Formulate the problem of the Towers of Hanoi as a search problem.

Notes:

- **In this version of the problem you have 3 towers (A, B and C) and 4 disks (D1 to D4).**
- **Initially the disks are in tower C and the objective is to transfer them to tower A.**



- **In each move, the player can move a disk from one tower to another tower, if he does not place that disk on a smaller disk.**

b) Suppose that the number of disks and the number of towers can be different (n disks and m towers) and formulate this generic version of the problem as a search problem.

IART - Artificial Intelligence

Lecture 2a: Search Problems

Luís Paulo Reis

lpreis@fe.up.pt

Director of LIACC – Artificial Intelligence and Computer Science Lab.

Associate Professor at DEI/FEUP – Informatics Engineering Department, Faculty of Engineering of the University of Porto, Portugal

President of APPIA – Portuguese Association for Artificial Intelligence