

1

REVISION

simple reflex agents

reflex agents with state

goal-based agents

utility-based agents

All of these 4 types can be turned into learning agents

2

PROBLEM SOLVING

Problem Solving Agents -> Atomic Representation

Planning Agents -> Factored or Structured Representation

Assumption: Solution of a problem is a fixed sequence of actions and this sequence DOES NOT depend upon future percepts

(Chapter 4 relaxes above mentioned assumption)

3

INFORMED VS UNINFORMED SEARCH

uninformed search algorithms—algorithms that are given no information about the problem other than its definition.

Informed search algorithms, on the other hand, can do quite well given some guidance on where to look for solutions

4

PROBLEM SOLVING AGENT

- 1) Goal Formulation
- 2) Problem Formulation
- 3) Searching Solution
- 4) Executing Solution

5

1- GOAL FORMULATION

Goal formulation, based on the **current situation/state** and the agent's **performance measure**, is the first step in problem solving.

Performance Measure: Visit as many cities as possible, spend as low as possible on fuel.

Current State: In Arad

Possible Goal: In Bucharest

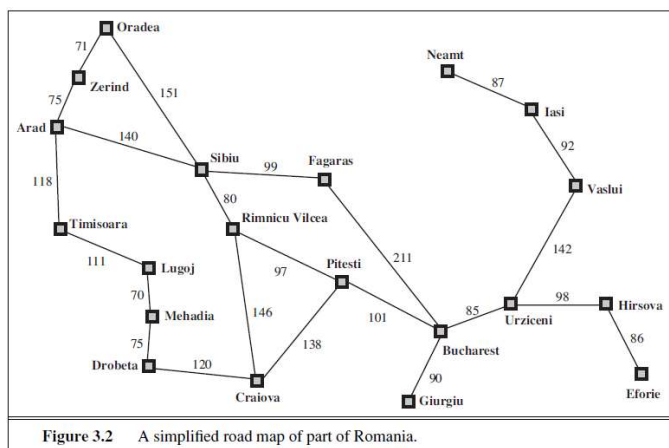


Figure 3.2 A simplified road map of part of Romania.

A set of states in which goal is satisfied

6

2- PROBLEM FORMULATION

Problem formulation is the process of deciding what actions and states to consider, given a goal.

The agent will consider actions at the level of driving from one major town to another. Each state therefore corresponds to being in a particular town.

Go Left, Go Right, Go forward, Go reverse. If these 4 actions are considered agent will never go out of the parking lot let alone reaching Bucharest.

Possible Action: Go Cityname

7

STATE SPACE GRAPH

On the right is the state space graph of our problem

State Space - the set of all states reachable from the initial state by any sequence of actions.

A **path** in the state space is a sequence of states connected by a sequence of actions.

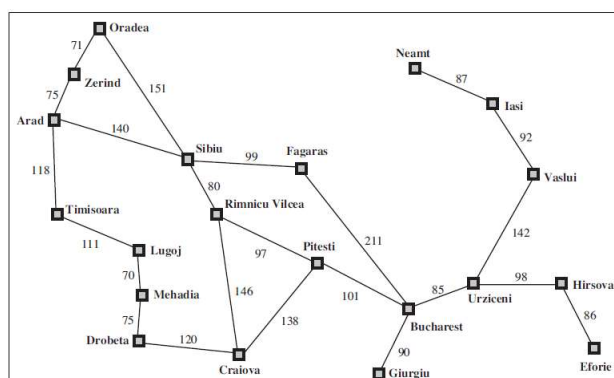
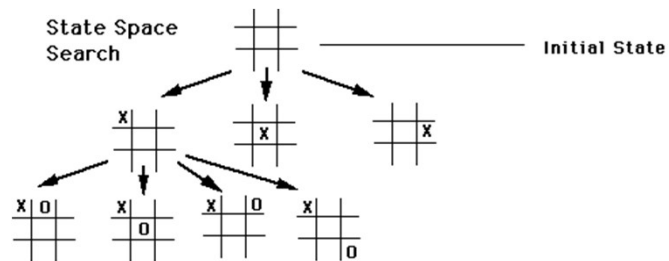


Figure 3.2 A simplified road map of part of Romania.

8

STATE SPACE GRAPH



How would a state space of chess look like?

9

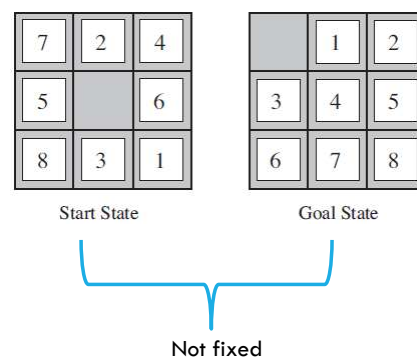
TOY PROBLEM: 8 PUZZLE

Any state can be used as starting and Goal states.

Actions: *Left*, *Right*, *Up*, or *Down*.

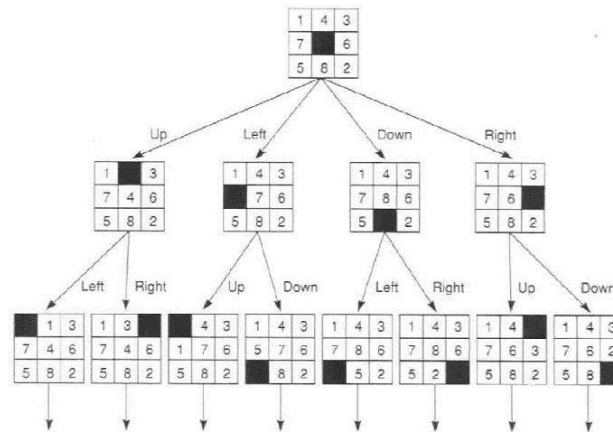
Transition model: Given a state and action, this returns the resulting state; for example, if we apply *Left* to the start state in adjacent figure, the resulting state has the 5 and the blank switched.

Path Cost: Each step costs 1, so the path cost is the number of steps in the path.

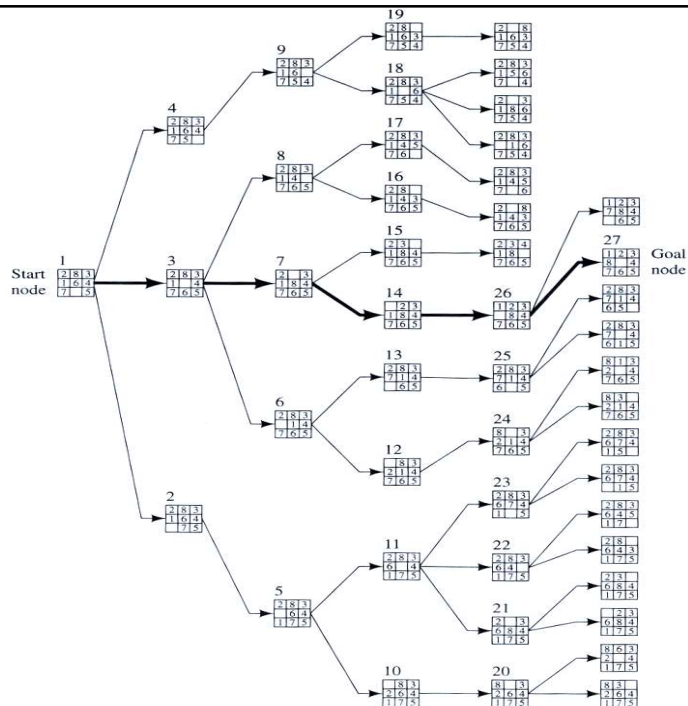


10

TOY PROBLEM: 8 PUZZLE



11



12

PROBLEM DEFINITION

A problem consists of five components,

- Initial State
- Possible Actions
- Transition Model
- Goal Test
- Path Cost

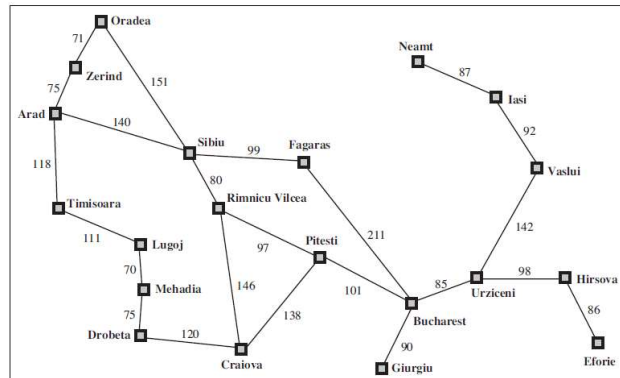


Figure 3.2 A simplified road map of part of Romania.

13

PROBLEM DEFINITION

A problem consists of five components,

- Initial State : in (Arad)
- Possible Actions (given a state): e.g., from Arad possible actions are {Go(Sibiu), Go(Timisoara), Go(Zerind)}.
- Transition Model: specifies the relationship between a state, a possible action and the resulting successor state e.g.,

$$\text{RESULT}(\text{In}(\text{Arad}), \text{Go}(\text{Zerind})) = \text{In}(\text{Zerind})$$

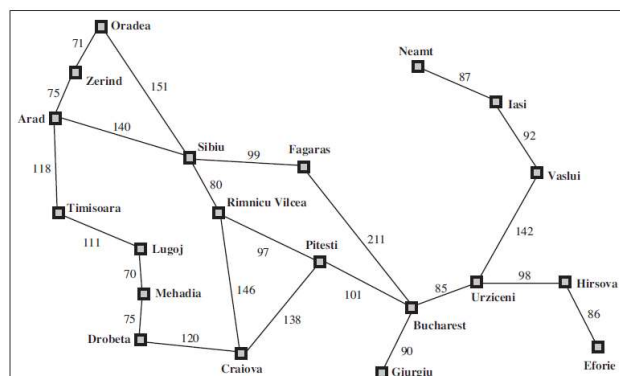


Figure 3.2 A simplified road map of part of Romania.

14

PROBLEM DEFINITION

A problem consists of five components,

- Initial State
- Possible Actions
- Transition Model
- Goal Test (whether a given state is a goal state) e.g. $\{ \text{In}(\text{Bucharest}) \}$.
- Path Cost (based upon agent's performance measure). Two paths shown on the right. **Step cost** (the cost of a single action within a path)

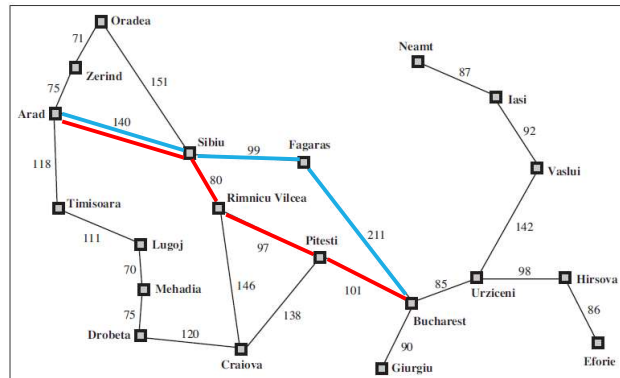


Figure 3.2 A simplified road map of part of Romania.

15

SOLUTION DEFINITION

A **solution** to a problem is an action sequence that leads from the initial state to a goal state.

Solution quality is measured by the path cost function, and an **optimal solution** has the lowest path cost among all solutions.

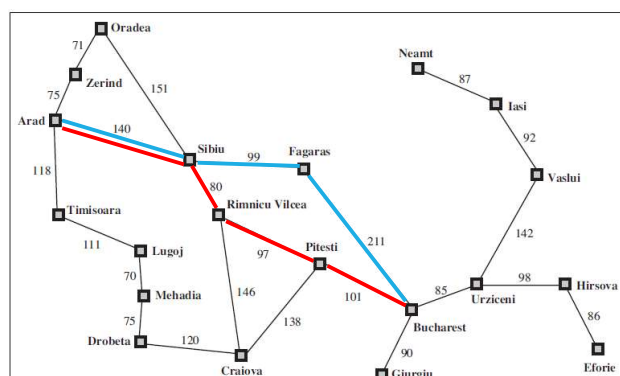
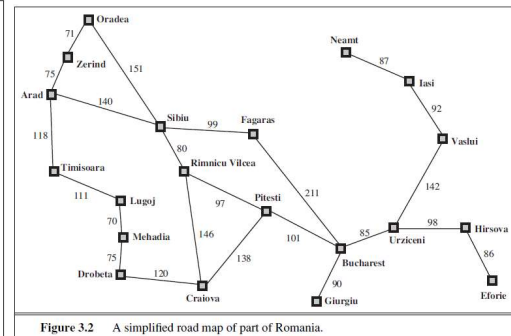
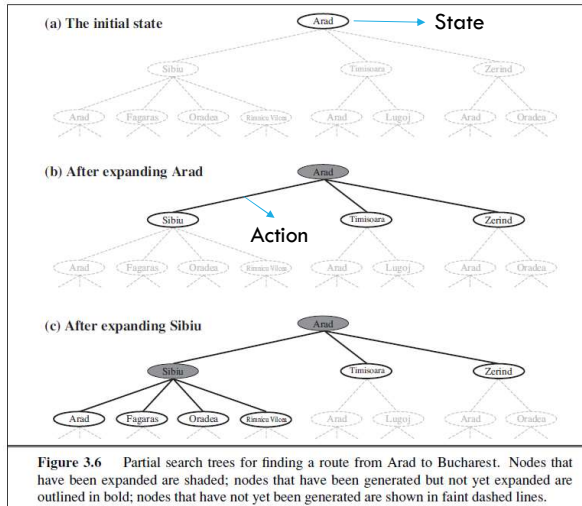


Figure 3.2 A simplified road map of part of Romania.

16

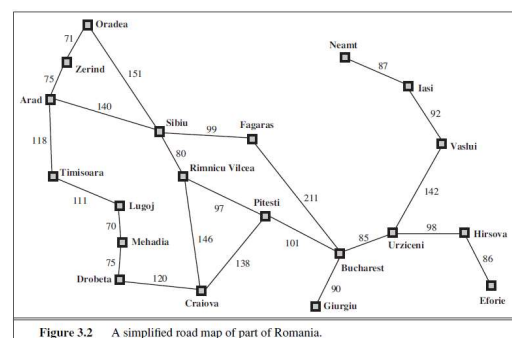
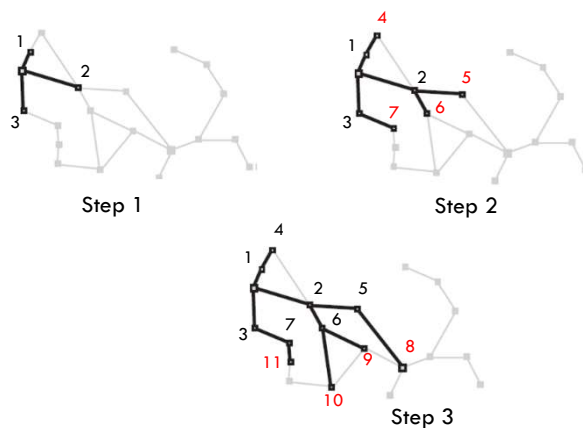
TREE SEARCH



17

GRAPH SEARCH

Use an **explore set** / **closed list** / **frontier** to remember the states already visited



18

INFRASTRUCTURE FOR SEARCH ALGORITHMS

- n .STATE: the state in the state space to which the node corresponds;
- n .PARENT: the node in the search tree that generated this node;
- n .ACTION: the action that was applied to the parent to generate the node;
- n .PATH-COST: the cost, traditionally denoted by $g(n)$, of the path from the initial state to the node, as indicated by the parent pointers.

Frontier / expanded nodes / unexpanded nodes are stored in the form of a queue (can be FIFO, LIFO (stack) or priority queue)

19

MEASURING PROBLEM-SOLVING PERFORMANCE

Completeness: Is the algorithm guaranteed to find a solution when there is one?

Optimality: Does the strategy find the optimal solution.

Time complexity: How long does it take to find a solution?

Space complexity: How much memory is needed to perform the search?

20

MEASURING PROBLEM-SOLVING PERFORMANCE

Complexity in a tree or a graph is measured by:

- 1) **b**, the **branching factor** or maximum number of children of any node
- 2) **d**, the **depth** of the goal node (i.e., the number of steps along the path from the root)
- 3) **m**, the maximum possible depth of entire tree.

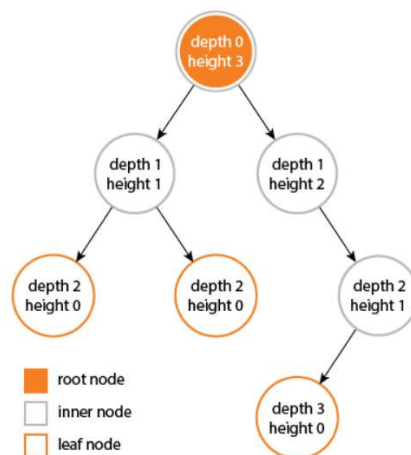
Time is often measured in terms of the number of nodes generated during the search.

Space in terms of the maximum number of nodes stored in memory.

21

MEASURING PROBLEM-SOLVING PERFORMANCE

A binary tree has a branching factor



22

24

24