Lecture 3: Search 2

Victor R. Lesser

CMPSCI 683

Fall 2010

Today's Lecture

- Finish off Introductory Material on Search
- Brief Review of Blind Search
- How to use heuristics (domain knowledge) in order to accelerate search?
- Reading: Sections 4.1-4.2.

Problem Formalization Issues

- Key issues in defining states:
  - which objects/relations to represent;
  - which configurations of objects/relations need to be mapped into separate states.
- Key issues in defining operators:
  - may have to make explicit, unstated assumptions in the problem description;
  - how state-specific/general should operators be;
  - how much domain-specific knowledge should be “compiled” into the operators.
- Developing an effective state space representation of a problem is choosing an appropriate abstraction.
  - Without abstraction, agents would be swamped by the details of the real-world.

Abstraction

- There are two main aspects of abstraction:
  - removing unnecessary detail from the state descriptions;
  - removing legal operators that are useless or inefficient for achieving goals.
- A good abstraction:
  - removes as much detail as possible to make it easy enough to find a solution;
  - maintains the **validity** of the solutions (for the conceptual goals).
- An *abstract solution* represents a large number of detailed paths.
  - Assumption that there is a valid detailed path that solves the desired problem
- Often there is a trade-off between simplicity and generality (the representation becomes so specific to the given problem that it cannot be used for even very similar problems).

Abstraction Examples

Two standard AI search problems can be used to explore the concept of abstraction.

Missionaries and Cannibals:

Three missionaries and three cannibals are on one side of a river. There is a boat available that can hold up to two people and can be used to cross the river. If the cannibals ever outnumber the missionaries in any location then a missionary will get eaten. Determine how the boat can be used to safely carry all the missionaries and cannibals across the river.

Trip/route Planning:

Determine how to get from one location to another. Assume that you know what city you are in, and have a map and a car.

Missionaries and Cannibals

Straightforward representation of states:

(boat-loc,m1-loc,m2-loc,...,c3-loc)

[loc i {side1,side2,river/boat}].

Results in $3^7 = 2187$ states.

Can you simplify by abstraction?

Missionaries and Cannibals (*cont'd.*)

Abstraction Simplification

- the particular missionaries and cannibals on each side do not matter—only numbers;
- do not have to have explicit states with people in the boat (once in boat will only want to cross to other side);
- once it is known the number of a type on one side know the number on the other side.

Abstract states:

(boat-side1?,#m's-side1,#c's-side1)

Results in $2 \times 4 \times 4 = 32$ states.

Missionaries and Cannibals (*cont'd.*)

- Useless operators can also be removed:

$$(1, m, c) \quad \rightarrow \quad (\mathbf{2}, m-1, c)$$

[single missionary goes to goal side in boat].

- The abstract solution using "move number of people" operators is still a valid solution to the conceptual goal
  - simply have to randomly select particular people when executing).
- Solution $(1,3,3) \rightarrow_{(2c)} (2,3,1) \rightarrow_{(1c)} (1,3,2) \rightarrow_{(2c)} (2,3,0) \rightarrow_{(1c)} (1,3,1) \rightarrow_{(2m)} (2,1,1) \rightarrow_{(mc)} (1,2,2) \rightarrow_{(2m)} (2,0,2) \rightarrow_{(1c)} (1,0,3) \rightarrow_{(2c)} (2,0,1) \rightarrow_{(1c)} (1,0,2) \rightarrow_{(2c)} (2,0,0)$

Simplifying Trip/Route Planning

- In its full generality, states for this problem would be very complex since they would describe "complete" configurations of the world:

  "at latitude and longitude x-y, time is t, radio on, raining, car z meters ahead, etc."

- To simplify, we focus on the problem of finding a sequence of city to city traversals that accomplish the goal.

  In this case, our abstract states simply become: "in city x."

- We can further simplify by identifying important cities (i.e., major cities and cities with road junctions) and identifying the subset of relevant cities (we don't need to include Amherst in the state space if we are trying to get to Boston from Worcester).

Trip/Route Planning (*cont'd.*)

- Likewise, in its full generality, there would be a very large number of operators to be considered and it would take a very large number of operators to achieve a solution:
  - e.g., "go heading *h* at speed *s*," "turn radio on," etc.
- With the abstract states, operators are of the form: "go from city *a* to city *b*" [where there is a road from city *a* to city *b*].
- A solution to the abstract problem solves the basic goal, but does not give us the detail required for, say, a robot vehicle to actually navigate the trip.
  - Still, the abstract problem solution allows us to see if a solution is even possible and to judge its approximate cost.
  - Reactively elaborate high-level plan basic on local conditions encountered

Problem Solving by Search

There are four phases to problem solving :

1. Goal formulation
   - based on current world state, determine an appropriate **goal**;
   - describes desirable states of the world;
   - goal formulation may involve general goals or specific goals;
2. Problem formulation
   - formalize the problem in terms of states and actions;
   - **state space representation**;
3. Problem solution via search
   - find sequence(s) of actions that lead to goal state(s);
   - possibly select "best" of the sequences;
4. Execution phase
   - carry out actions in selected sequence.

Problem-Solving Performance

- Complete search-based problem solving involves both the search process and the execution of the selected action sequence.
  - **Total cost** of search-based problem solving is the sum of the search costs and the path costs (operator sequence cost).
- Dealing with total cost may require:
  - Combining "apples and oranges" (e.g., travel miles and CPU time)
  - Having to make a trade-off between search time and solution cost optimality (resource allocation).
  - These issues must be handled in the performance measure.

Agent vs. Conventional AI View

- A completely autonomous agent would have to carry out all four phases.
- Often, goal and problem formulation are carried out prior to agent design, and the "agent" is given specific goal *instances* (agents perform only search and execution).
  - general goal formulation, problem formulation, specific goal formulation, etc.
- For "non-agent" problem solving:
  - a solution may be simply a specific goal that is achievable (reachable);
  - there may be no execution phase.
- The execution phase for a real-world agent can be complex since the agent must deal with uncertainty and errors.

Problem Types
Vacuum World Domain as an illustration

- Let the world consist of only **2 locations** - Left and Right Box
- Each location may contain **dirt**
- The agent may be in **either box**
- There are **8 possible states**
- The agent can have **3 possible actions** - Left, Right and Suck

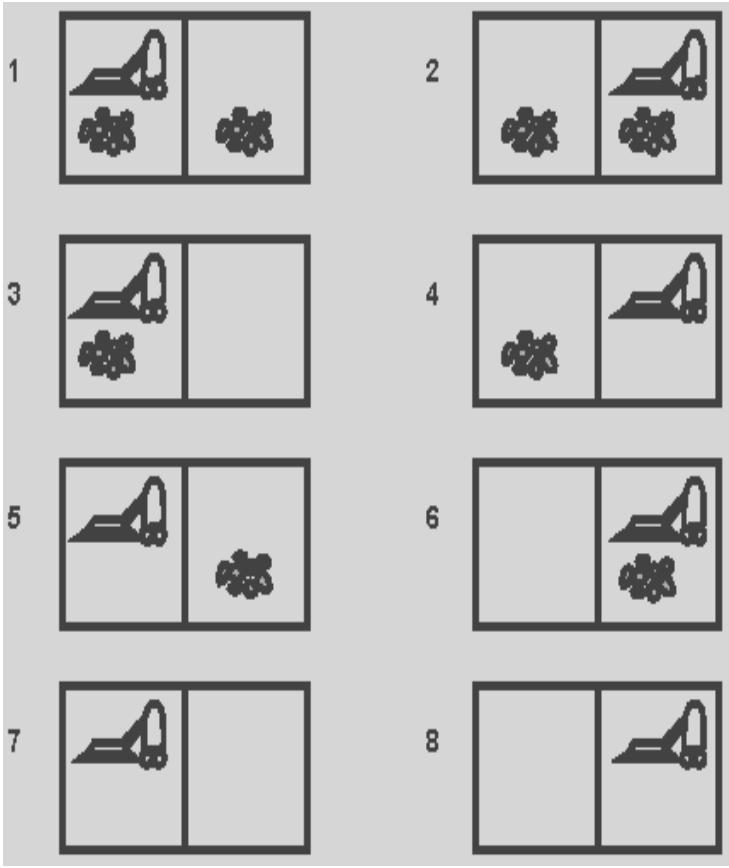


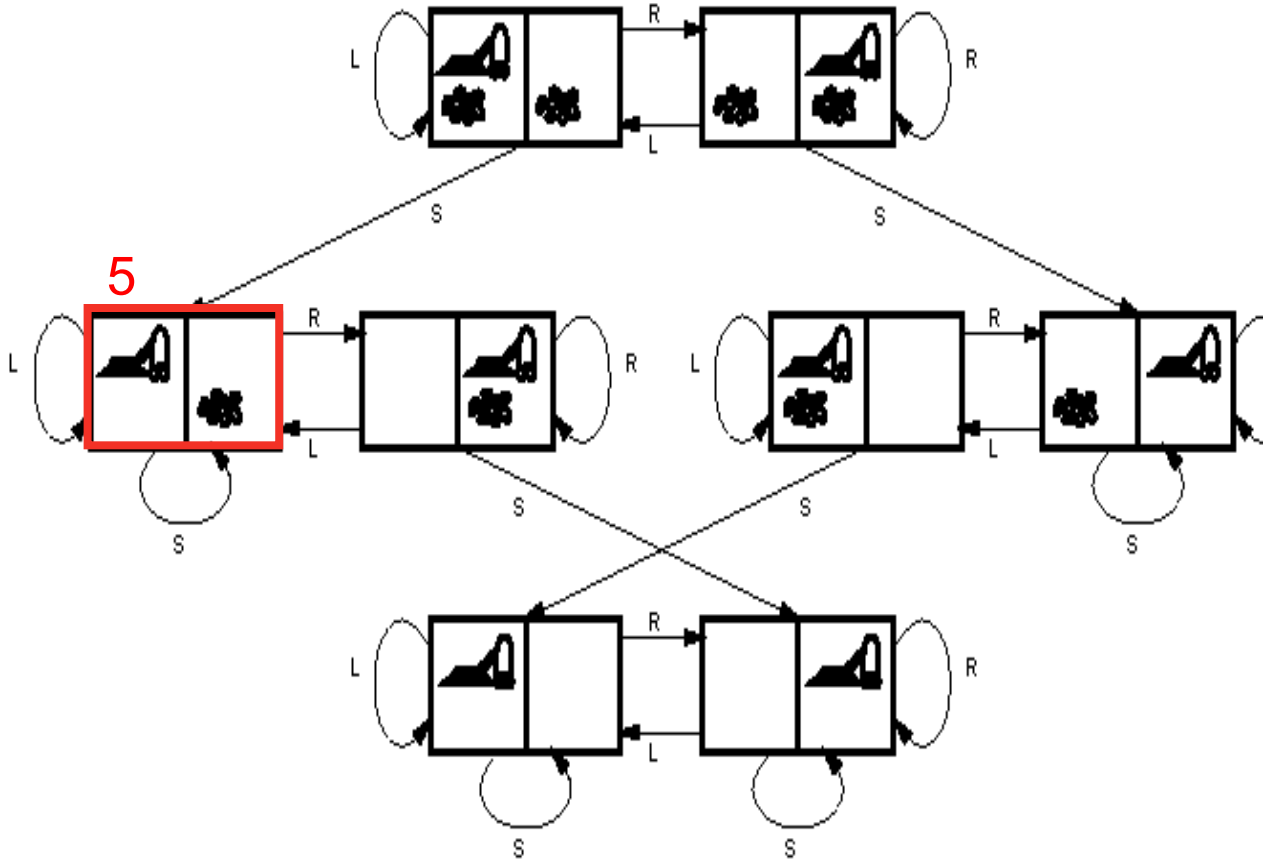
Fig 3.20 The 8 possible states of a Vacuum World

Single-state problems

***Fully observable, Deterministic*=>**

Single-state problems

- **exact** state known
- **effects** of actions **known**
- In vacuum world, if initial state is 5, to achieve the goal, do action sequence [*Right*, *Suck*]



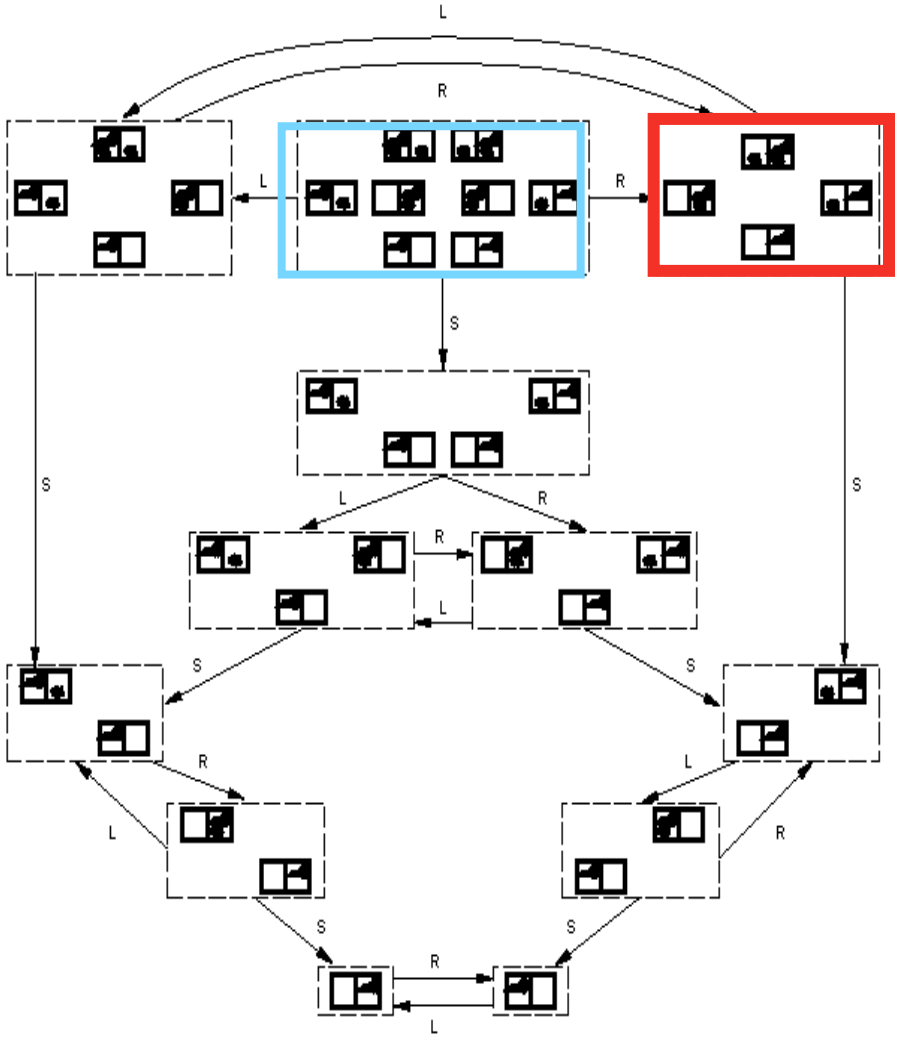
Sensorless (Multiple-state) Problems

Partially observable, deterministic

one *of a set of states*

effects *of actions* ***known***

eg. In vacuum world, where there are **no sensors**, the agent knows that there are 8 initial states, it can be calculated that an action of ***Right*** will achieve state **{2, 4, 6, 8}** and the agent can discover that the action sequence **[*Right, Suck, Left, Suck*]** is guaranteed to reach the **goal**



More Problem Types

- ***Partially observable , stochastic*** **(action is uncertain), => Contingency problems**
  - ***agent can obtain new information after acting***
    - limited sensing
  - **conditional effects** of actions
  - More complex algorithms involving **planning**
  - Must use sensor during execution
  - ***Solution is a tree or policy***
  - Often **interleaving of search and execution**

eg 1. In vacuum world with Murphy's law, adding a simple Sense_Dirt, to use before the action *Suck*.

eg 2. Most of us keep our eyes open while walking,

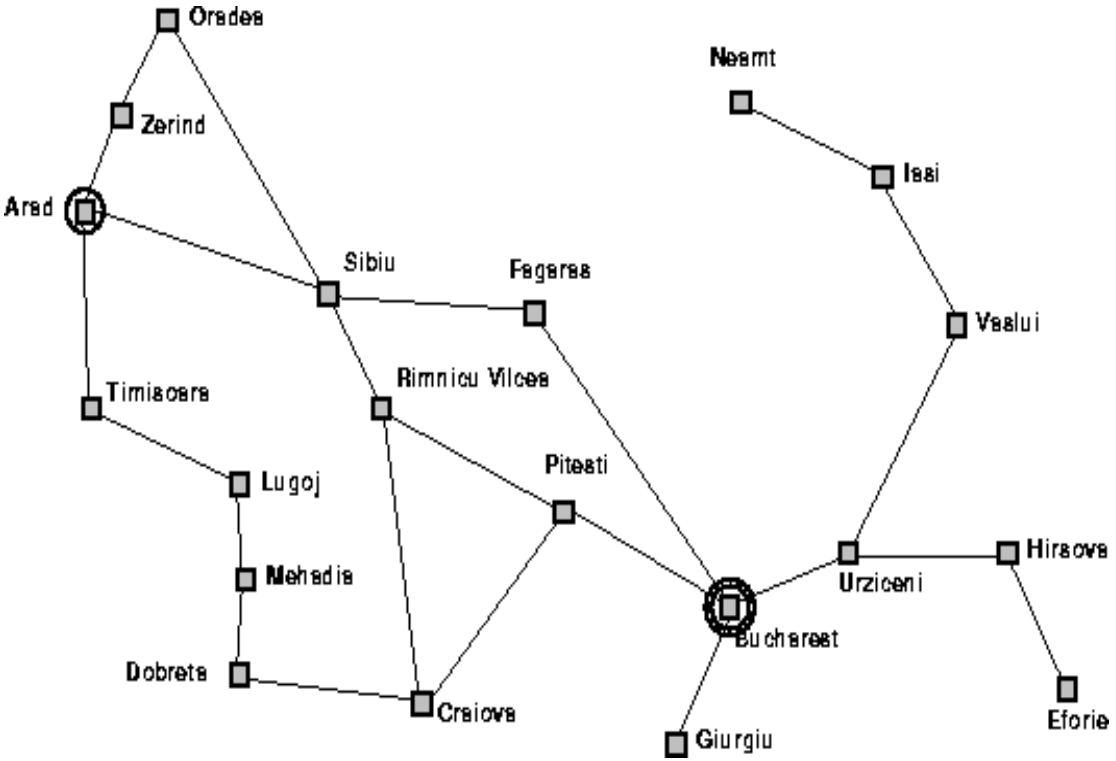
Unknown state space => **Exploration problems**

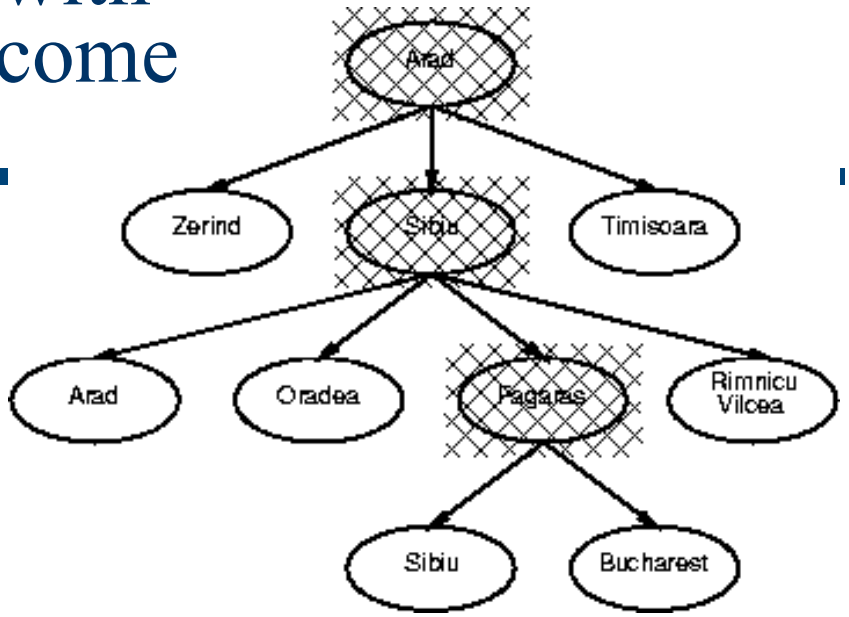
execution 'reveals' states needs to **experiment** in order to survive

Hardest task faced by an intelligent agent, an extreme case of contingency problem need **learning, interleaving of search and execution**

eg 1. Mars Pathfinder

Observable Initial State with Deterministic Operator Outcome Search



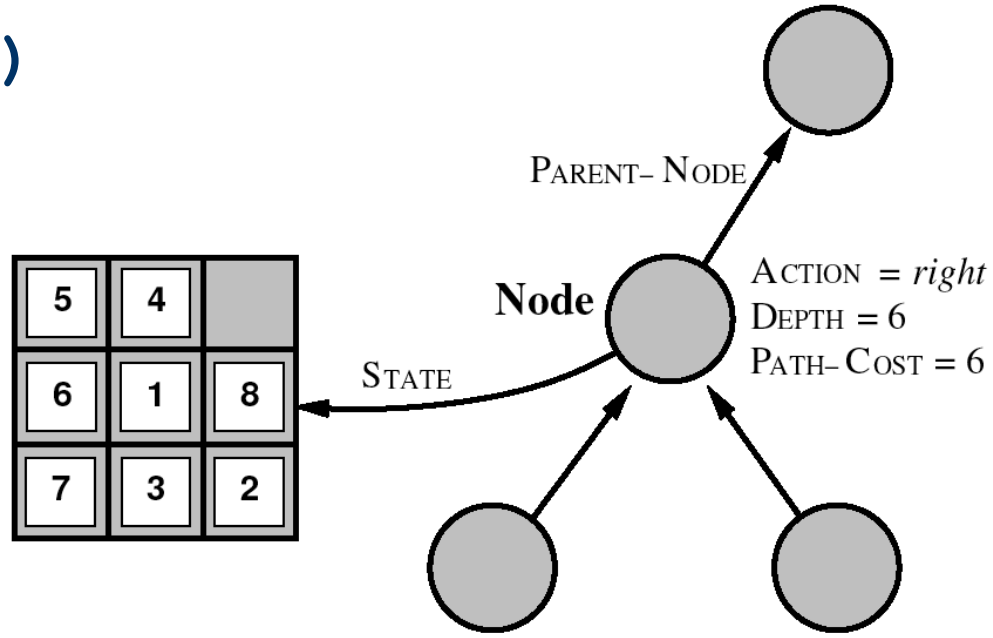


Search Tree Implementation

- **State**: a representation of a physical configuration
- **Node**: a data structure constituting part of a search tree includes parent, children, depth, path cost
- **Expand function**: creates new nodes, filling in the various fields and using the Operators (Successor Function) of the problem to create the corresponding states.
- **Fringe**: A queue used to keep the nodes that are waiting to be visited. How to maintain the order of the nodes in this queue depending on the search strategy.

Representing a node

```
(defstructure node
state
      parent-node
      operator
      depth
      path-cost)
```

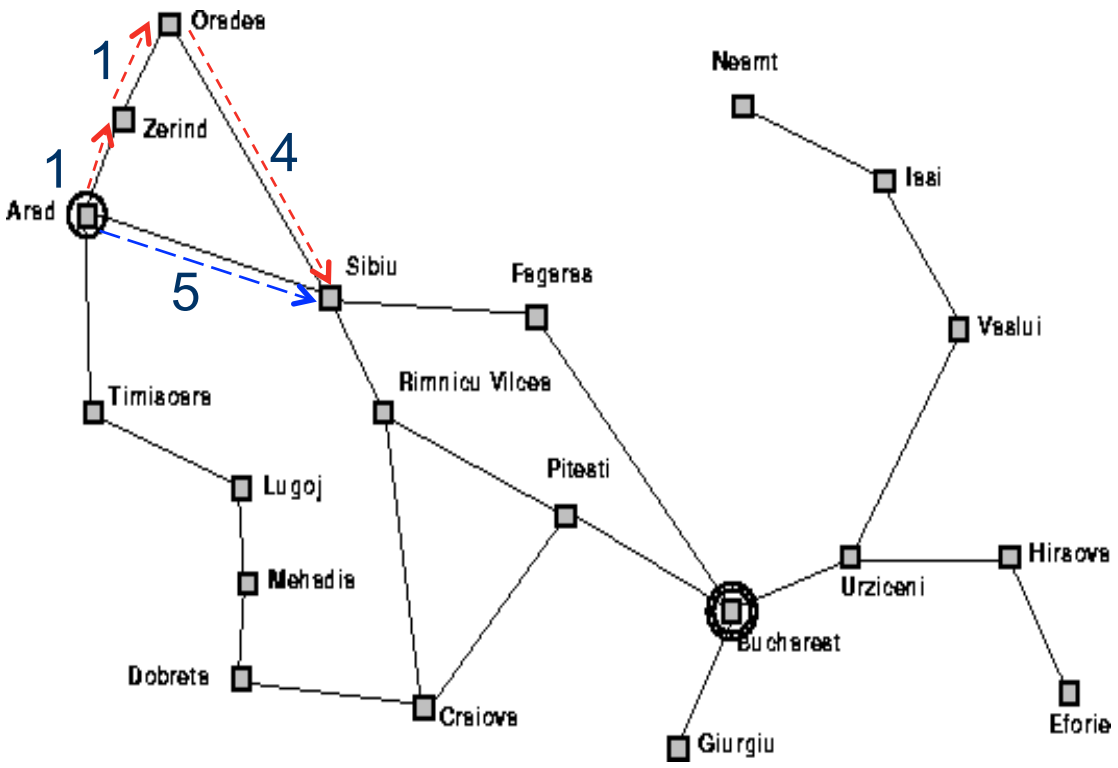


Is this all the information you need to make a decision about how to expand this node?

Is there a one-to-one correspondence between a node and a partial solution

Non-Markov Assumptions

- What would the node Sibiu contain in this situation if you were trying to minimize travel distance



How would you handle the search where you want to minimize travel distance while keeping tolls below a certain level

Next lecture

- Overview of Search Strategies
- Blind Search
- Informed Search
  - How to use heuristics (domain knowledge) in order to accelerate search?
  - A* and IDA*
  - Reading: Sections 4.1-4.2.