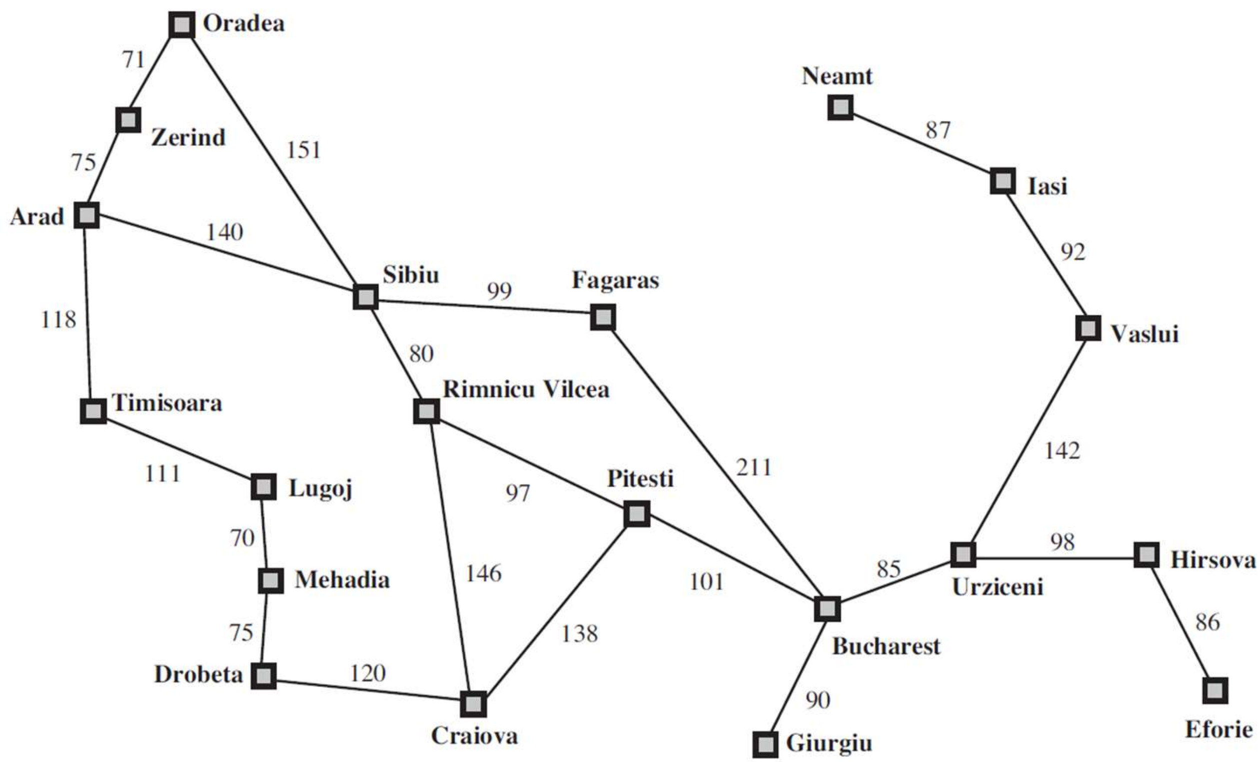
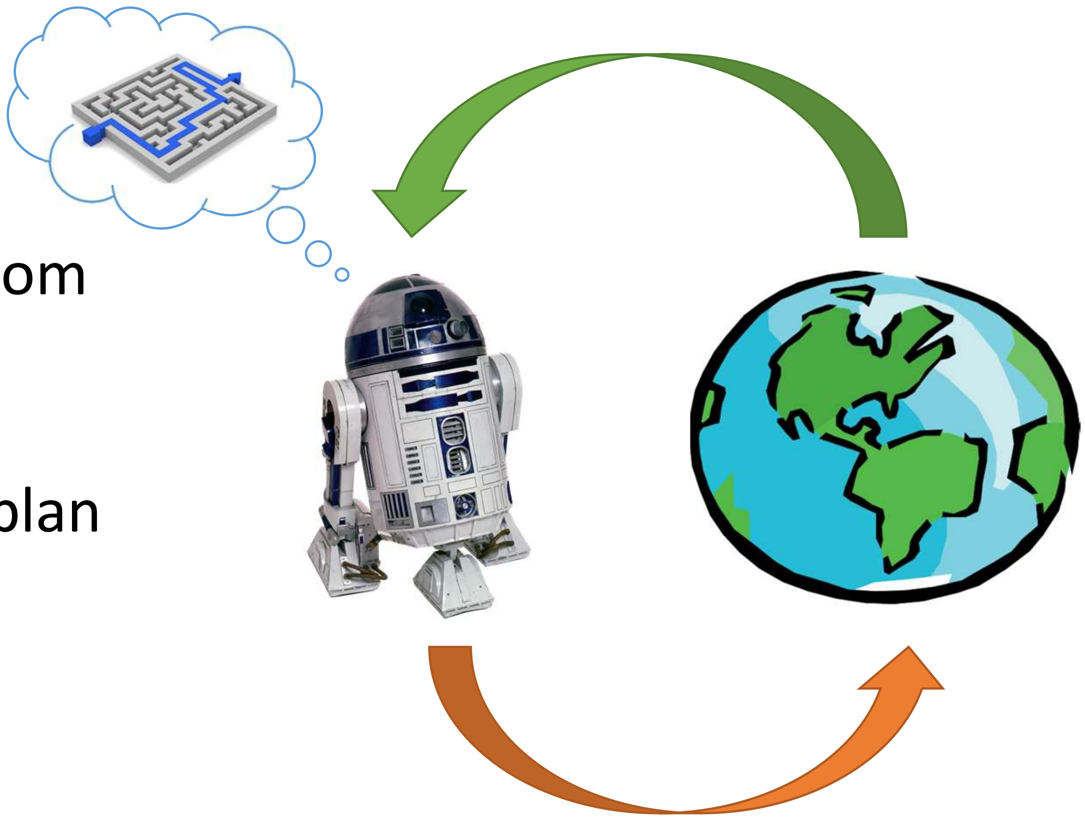
EECE 5550: Mobile Robotics



Lecture 15: Planning

The Central Dogma of Robotics: Sense → Think → Act

- Sense: Process sensor data to construct a model of the world
- Think: Construct a plan to move from the current state to the goal state
- Act: Control actuators to execute plan



The Story So Far

Mathematical foundations →

Computational tools →

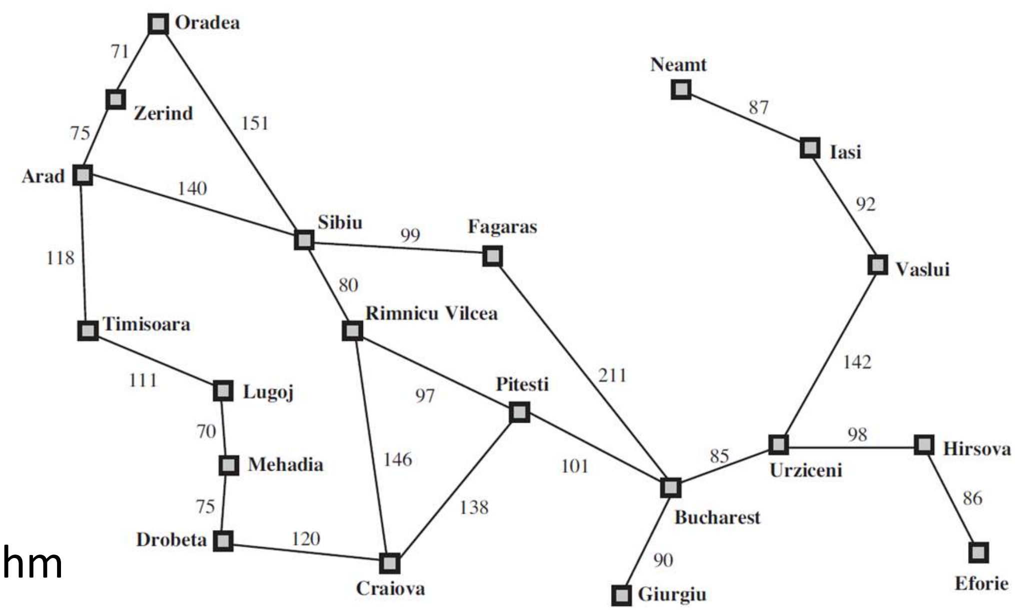
Sense (perception) →

This week: Think (plan) →

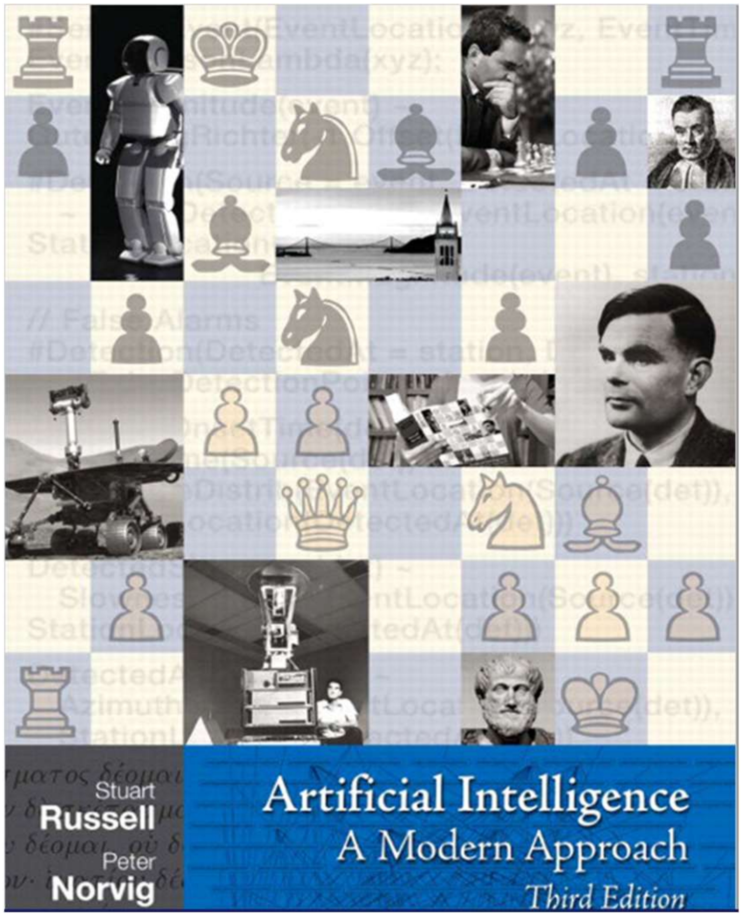
Week	Topics (tentative)
1	Coordinate transformations & geometry
2	Lie groups & probability theory
3	Computational tools: Linux, Git, Ros
4	Sensing, kinematics & computer vision
5	Probabilistic robotics & Bayesian filtering
6	Robotic mapping & localization
7	SLAM & optimization
8	Planning & graph search
9	Feedback, optimal, and model-predictive control
10	Planning under uncertainty
11	Applications: Robotic exploration
12	Guest lectures: research frontiers
13	Final presentations

Plan of the day (☺)

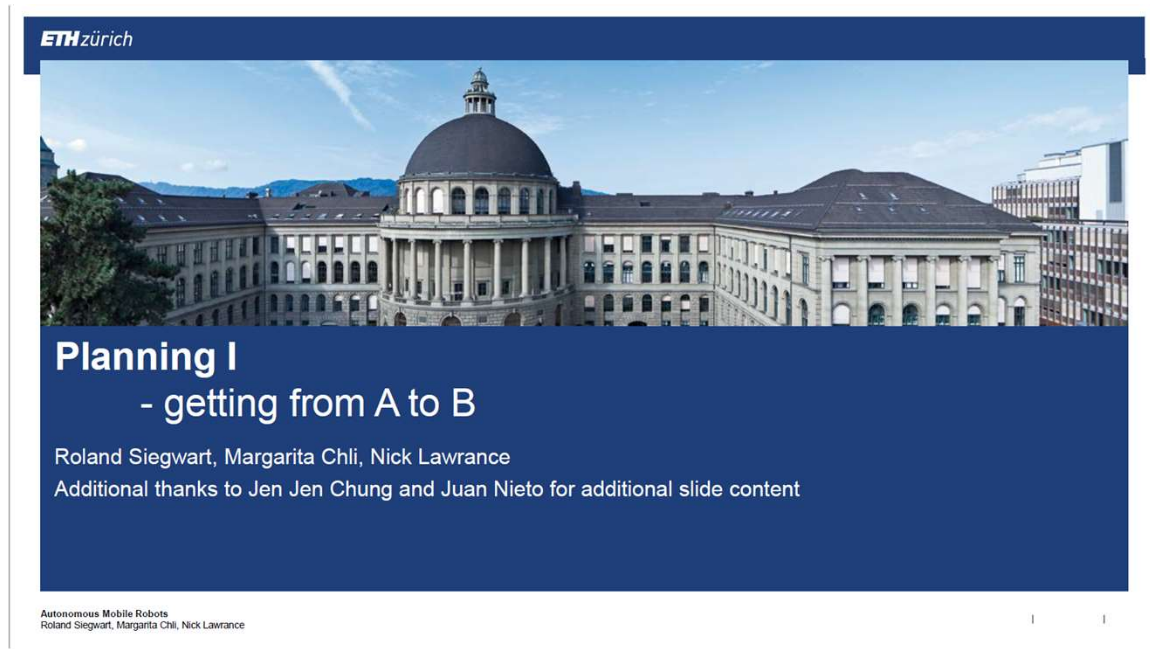
- Introduction to planning
- Formulation of a planning problem
- Planning as graph search
- Four canonical graph search algorithms
  - Breadth-first search
  - Depth-first search
  - Searching for *shortest paths*: Dijkstra's algorithm
  - Informed search: A* search



References



Chapter 3 of “Artificial Intelligence: A Modern Approach”

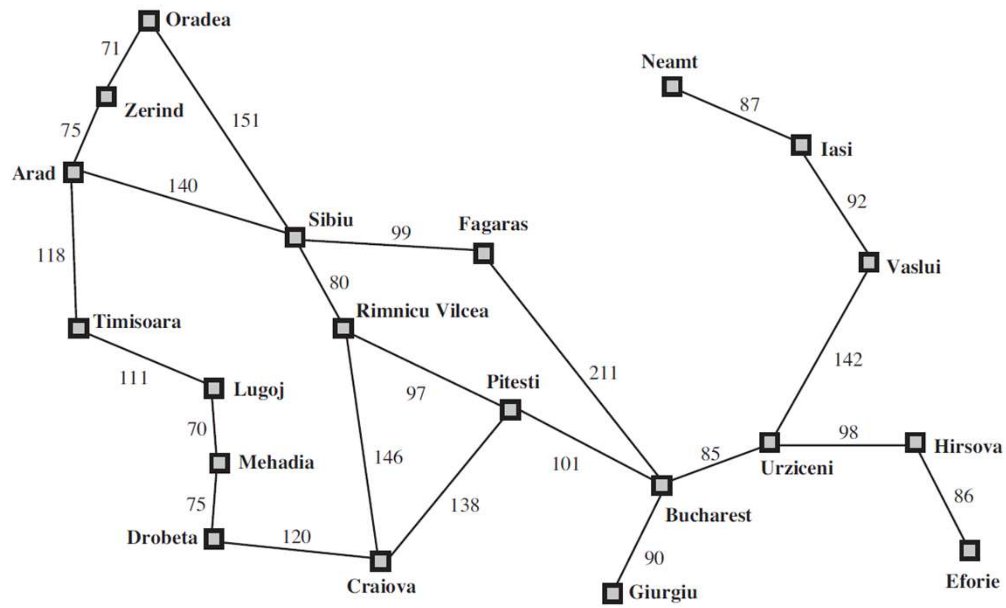


Lecture “Planning I” from ETH Zurich’s Autonomous Mobile Robots course

What is "Planning"?

Intuitively: How do I get from A to B?

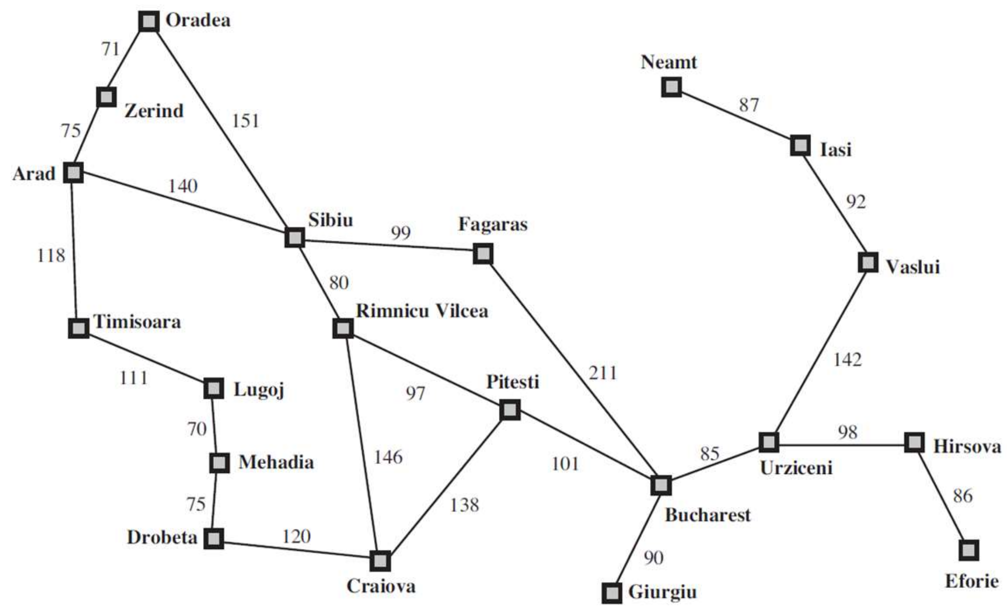
A bit more precisely: Determine a sequence of actions that will drive the world from an *initial state* to a ***goal state.***



Defining a planning problem

We must specify the following elements:

1. State space X: possible states of the world
2. Initial state $x_0 \in X$
3. Actions: function A(x) returns the set of actions available at each state $x \in X$
4. Transition model: function R(x,a) describes the *successor* state achieved by applying action a in state x.
5. Goal test: G(x) returns true/false to indicate whether we have reached a goal
6. Cost: C(x,a) of applying action a in state x.

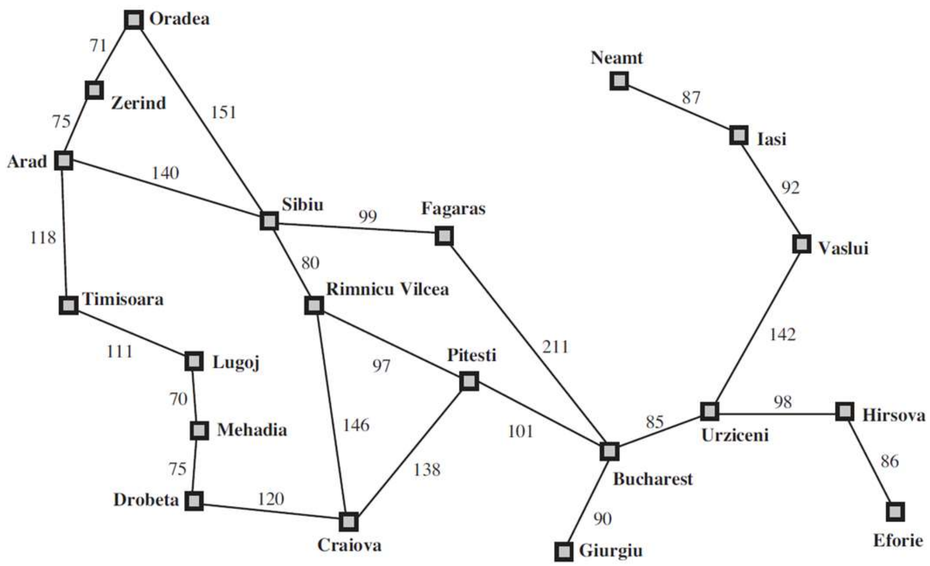


Solutions of a planning problem

Given a planning problem $P = (X, x_0, A, R, G, C)$:

- A *solution* is a *sequence of actions* $a_1, a_2, a_3, \dots$ that leads from the initial state x_0 to a goal state (a state satisfying G(*x*) = true).

- An *optimal solution* is a solution attaining the *minimum possible cost*.



Toy example: Vacuum world

- **States:**
  - 2 locations (each of which may contain dirt)
  - Agent location
- **Initial state:** Any
- **Actions:** Left, right, suck
- **Transition model:** As expected (see diagram)
- **Goal test:** No dirt remains in any square
- **Cost:** Each action costs 1

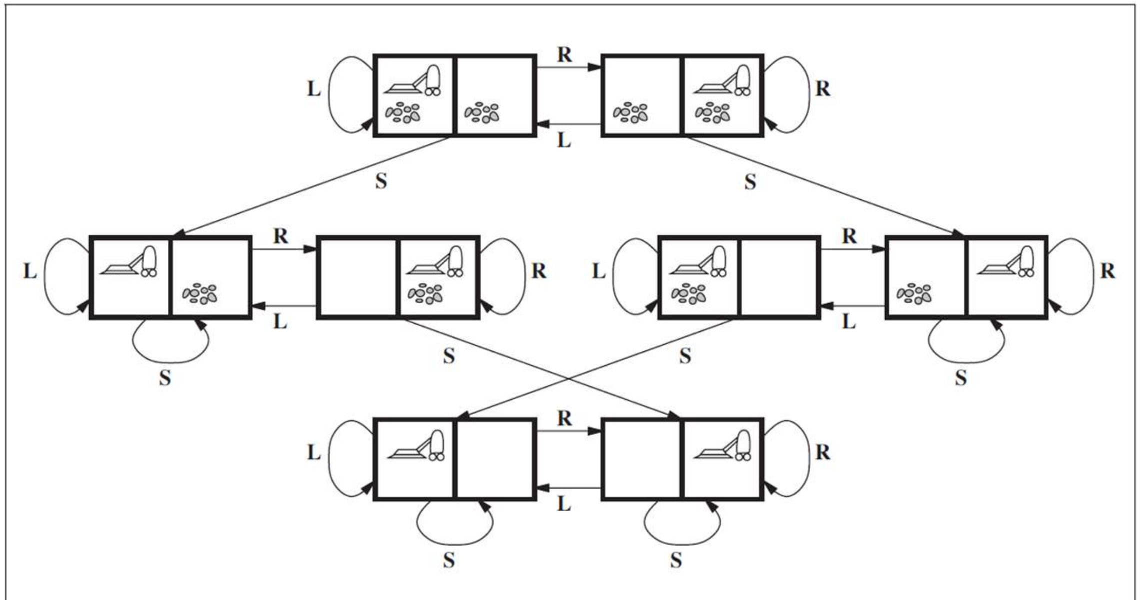
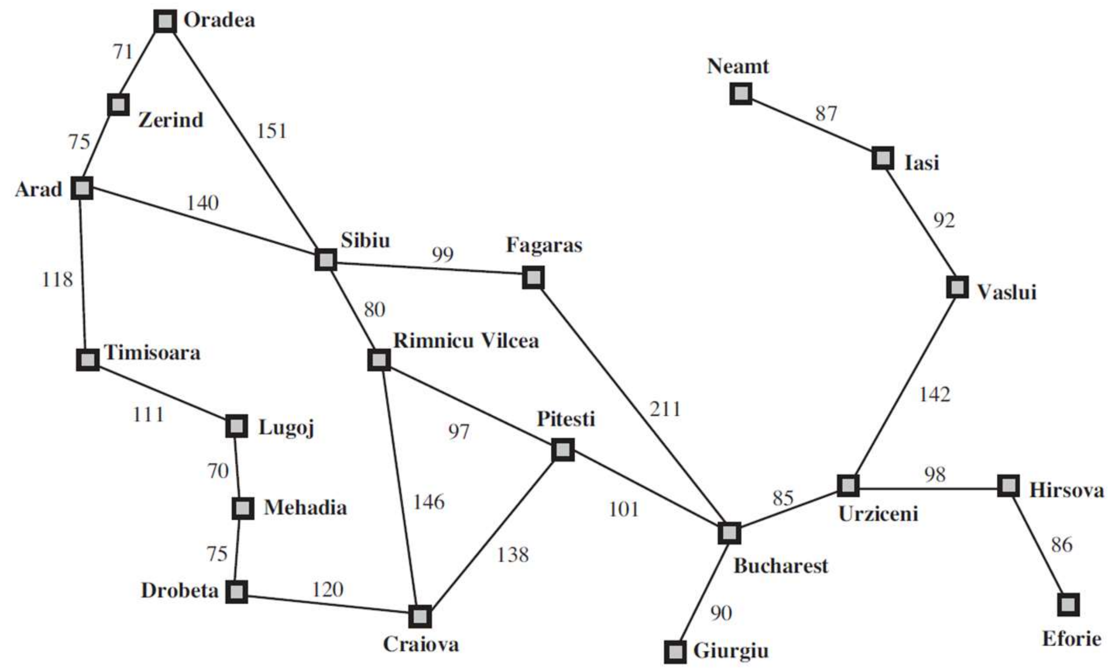


Figure 3.3 The state space for the vacuum world. Links denote actions: L = *Left*, R = *Right*, S = *Suck*.

Real-world example: Route-finding

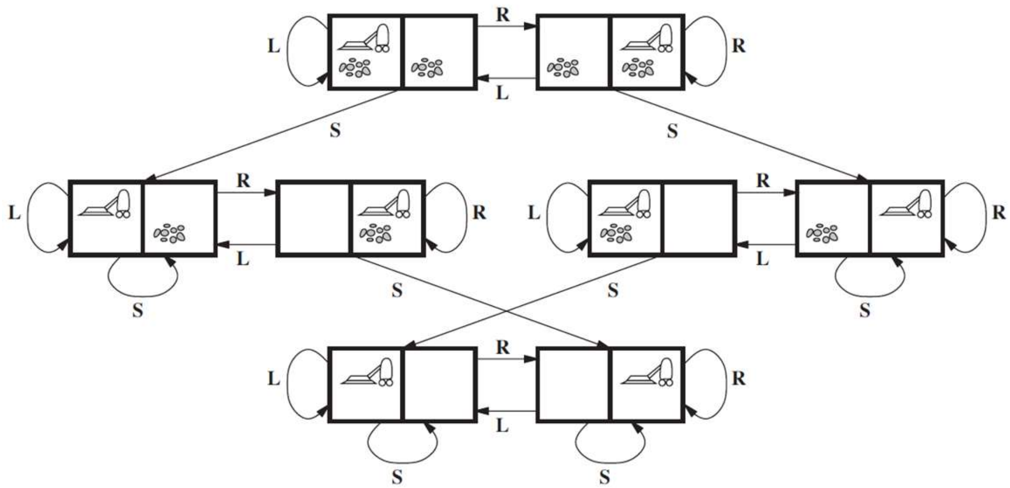
- **States:** Specified set of locations (e.g. cities)
- **Initial state:** Any
- **Actions:** Following any edge (link) between current location and a neighboring location
- **Transition model:** As expected (arrive in the neighboring city)
- **Goal test:** Have you arrived in the goal city?
- **Cost:** Several possible choices:
  - **Money:** Price of tickets, gas, etc.
  - **Time:** Elapsed travel time via the given route



Solving a planning problem

Basic insight: We can model each planning problem $P = (X, x_0, A, R, G, C)$ as a weighted directed graph $G = (X, E, w)$, where:

- Vertices of G are the *states X*
- Edge set *E* contains $i \to j$ iff there is an *action* $a \in A(i)$ for which $j = R(i, a)$.
- The weight $w(i, j)$ of an edge $(i, j) \in E$ is just the cost $C(i, j)$.



Therefore: We can recast *planning* as *graph search*!

- Solution of planning problem *P* ⇔ *path* from initial state x_0 to goal state in *G*
- Optimal solution of *P* ⇔ *minimum-cost path* from x_0 to goal state in *G*

Payoff: We know how to search graphs efficiently [thanks CS ☺!]

Intermission

(over to “Planning I: Graph Search Methods” …)