Problem Solving Agents & Problem Formulation

AIMA 2.3, 3.1-2

Outline for today's lecture

- ***Defining Task Environments***
- **Environment types**
- **Formulating Search Problems**

Task environments

- **To design a rational agent we need to specify a *task environment***
  - a problem specification for which the agent is a solution
- ***PEAS:*** *to* specify a task environment
  - ***P***erformance measure
  - ***E***nvironment
  - ***A***ctuators
  - ***S***ensors

PEAS: Specifying an automated taxi driver

Performance measure**:**
- ?

Environment**:**
- ?

Actuators**:**
- ?

Sensors**:**
- ?

PEAS: Specifying an automated taxi driver

Performance measure**:**
- safe, fast, legal, comfortable, maximize profits

Environment**:**
- roads, other traffic, pedestrians, customers

Actuators**:**
- steering, accelerator, brake, signal, horn

Sensors**:**
- cameras, sonar, speedometer, GPS

PEAS: Medical diagnosis system

- ***Performance measure*: Healthy patient, minimize costs, lawsuits**
- ***Environment*: Patient, hospital, staff**
- ***Actuators*: Screen display (form including: questions, tests, diagnoses, treatments, referrals)**
- ***Sensors*: Keyboard (entry of symptoms, findings, patient's answers)**

Outline for today's lecture

- Defining Task Environments
- *Environment types*
- Formulating Search Problems

Environment types: Definitions I

- *Fully observable* (vs. partially observable): An agent's sensors give it access to the complete state of the environment at each point in time.
- *Deterministic* (vs. stochastic): The next state of the environment is completely determined by the current state and the action executed by the agent.
  - If the environment is deterministic except for the actions of other agents, then the environment is *strategic*.
- *Episodic* (vs. sequential): The agent's experience is divided into atomic "episodes" during which the agent perceives and then performs a single action, and the choice of action in each episode depends only on the episode itself.

Environment types: Definitions II

- *Static* (vs. dynamic): The environment is unchanged while an agent is deliberating.
  - The environment is *semidynamic* if the environment itself does not change with the passage of time but the agent's performance score does.
- *Discrete* (vs. continuous): A limited number of distinct, clearly defined percepts and actions.
- *Single agent* (vs. multiagent): An agent operating by itself in an environment.

(See examples in AIMA,
however I don't agree with some of the judgments)

Environment Restrictions for Now

- We will assume environment is
  - *Static*
  - *Fully Observable*
  - *Deterministic*
  - *Discrete*

The rational agent designer's goal

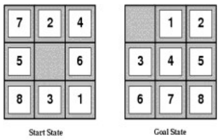
- Goal of AI practitioner who designs rational agents: given a *PEAS* task environment,
  1. Construct agent function f that maximizes (the expected value of) the performance measure,
  2. Design an agent program that implements f on a particular architecture

Outline for today's lecture

- Defining Task Environments
- Environment types
- *Formulating Search Problems (AIMA, 3.1-3.2)*

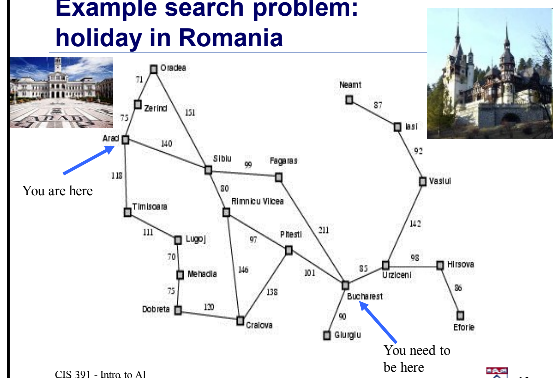
Example search problem: 8-puzzle

- **Formulate *goal***
  - Pieces to end up in order as shown…



- **Formulate *search problem***
  - ***States:*** configurations of the puzzle (9! configurations)
  - ***Actions***: Move one of the movable pieces (≤4 possible)
  - ***Performance measure***: minimize total moves
- **Find *solution***
  - Sequence of pieces moved: 3,1,6,3,1,…

Example search problem: holiday in Romania



Holiday in Romania II

- **On holiday in Romania; currently in Arad**
  - Flight leaves tomorrow from Bucharest
- **Formulate *goal***
  - Be in Bucharest
- **Formulate *search problem***
  - States: various cities
  - Actions: drive between cities
  - Performance measure: minimize distance
- **Find *solution***
  - Sequence of cities; e.g. Arad, Sibiu, Fagaras, Bucharest, …

More formally, a problem is defined by:

1. A set of *states* S
2. An *initial state* $s_i \in S$
3. A *set of actions* A
   - — $\forall s$, *Actions(s)* = *the set of actions that can be executed in s, that are applicable in s.*
4. *Transition Model:* $\forall s \, \forall a \in Actions(s)$, $Result(s, a) \rightarrow s_r$
   - —s_r *is called a successor of s*
   - —$\{s_i\} \cup Successors(s_i)^*$ = *state space*
5. *Goal test Goal(s)*
   - — Can be implicit, e.g. *checkmate(x)*
   - — *s* is a *goal state* if *Goal(s)* is *true*
6. *Path cost* (additive)
   - —e.g. sum of distances, number of actions executed, …
   - —$c(x,a,y)$ is the step cost, assumed ≥ 0
     - (where action a goes from state x to state y)

Solution

A *solution* is a sequence of actions from the *initial state* to a *goal state.*

Optimal Solution:
A solution is *optimal* if no solution has a lower path cost.

Hard subtask: Selecting a state space

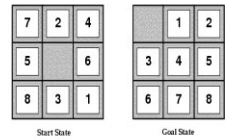
- **Real world is absurdly complex**
  State space must be *abstracted* for problem solving
- **(abstract) *State* = set (equivalence class) of real world states**
- **(abstract) *Action* = complex combination of real world actions**
  - e.g. *Arad* → *Zerind* represents a complex set of possible routes, detours, rest stops, etc
  - The abstraction is valid if the path between two states is reflected in the real world
- **(abstract) *Solution* = set of abstract paths that are solutions in the abstract space**
- **Each abstract action should be "easier" than the real problem**

Formulating a Search Problem

Decide:

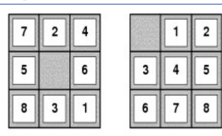
- **Which properties matter & how to represent**
  - *Initial State, Goal State, Possible Intermediate States*
- **Which actions are possible & how to represent**
  - *Operator Set: Actions and Transition Model*
- **Which action is next**
  - *Path Cost Function*

Example: 8-puzzle



- **States??**
- **Initial state??**
- **Actions??**
- **Transition Model??**
- **Goal test??**
- **Path cost??**

Example: 8-puzzle



- **States??** List of 9 locations- e.g., [7,2,4,5,-,6,8,3,1]
- **Initial state??** [7,2,4,5,-,6,8,3,1]
- **Actions??** {Left, Right, Up, Down}
- **Transition Model??** ...
- **Goal test??** Check if goal configuration is reached
- **Path cost??** Number of actions to reach goal

Example: Missionaries & Cannibals

Three missionaries and three cannibals come to a river. A rowboat that seats two is available. If the cannibals ever outnumber the missionaries on either bank of the river, the missionaries will be eaten. ***(problem 3.9)***

How shall they cross the river?

Formulation: Missionaries & Cannibals

- ***How to formalize:***
  - *Initial state*: all M, all C, and boat on one bank
  - *Actions: ??*
  - *Transition Model??*
  - *Goal test*: True if all M, all C, and boat on other bank
  - *Cost*: ??

Remember:

- ***Representation:***
  - *States: Which properties matter & how to represent*
  - *Actions &Transition Model: Which actions are possible & how to represent*
  - *Path Cost: Deciding which action is next*

Missionaries and Cannibals

States: (CL, ML, BL)

Initial 331 Goal 000

Actions:

Travel Across	*Travel Back*
-101	101
-201	201
-011	011
-021	021
-111	111