Lecture Overview

• Basic agents
• Basic vacuum world example
• Need for rational agents
• Task environment properties
• Agents as programs
• Next lecture: types of agents
Basic Agent

- Agent performs an action based on entire percept history
- Simplest agent. Gigantic percept history action table.
- What’s the problem?
- Simplify by having an independent or Markov assumption

Sensors = rewards and state of world
Vacuum World

- Agent perceives (location: A,B and location info: dirty, clean)
- Four actions (left, right, suck, no-op)
- Percept action table
  
  \[
  \begin{array}{c|c}
  \text{[A, Clean]} & \rightarrow \text{Right} \\
  \text{[B, Clean]} & \rightarrow \text{Left} \\
  \text{[*, Dirty]} & \rightarrow \text{Suck}
  \end{array}
  \]

- But what table is the best
- Inverse problem: Given a performance measure, find best action table.
Performance Measures and Rationality

• Consider a performance measure for:
  – Vacuum world
  – Email harvester
  – Chess
  – Immediate reward versus deferred reward
    • Credit allocation problem

• A rational agent for a given percept sequence maximizes its performance measure.

• This is our definition of A.I., not that the agent thinks/reasons like a human.
Environment Properties - 1

- Variations in properties are why there are many A.I. techniques
- Fully versus partially observable
  - Agent perceives all information required to make the optimal action. Does not refer to violation of modeling assumptions.
  - Noisy sensors can make an environment partially observable
- Deterministic versus stochastic
  - From the view of the agent
  - Partially observable, stochastic environment = uncertainty
Environment Properties - 2

• Episodal versus Sequential
  – Most interesting environments are sequential
  – Each episode consists of the agent performing one action. If episodes are independent then episodal. i.e. \( P^*(\text{Action}_t | P_t, P_{t-1}, P^1) = P^*(\text{Action}_t | P_t) \)
Environment Properties - 3

- **Static versus Dynamic**
  - Semi-dynamic

- **Discrete versus continuous**
  - Applies to actions, states

- **Single agent versus multiple agents**
  - Co-operative and competitive agents
Name the Environment Properties

• Chess
• Monopoly
• Robocup simulation league
• Email crawler
Environment Types

Deterministic, accessible $\rightarrow$ single-state problem
Deterministic, inaccessible $\rightarrow$ multiple-state problem

Nondeterministic, inaccessible $\rightarrow$ contingency problem
must use sensors during execution
solution is a *tree* or *policy*
often *interleave* search, execution

Unknown state space $\rightarrow$ exploration problem ("online")

Each problem type motivates a set of A.I. techniques.
Eight State Puzzle

Start State

5 4
6 1 8
7 3 2

Goal State

1 2 3
8 4
7 6 5

states??
operators??
goal test??
path cost??
What is a State and a Node?

A *state* is a (representation of) a physical configuration
A *node* is a data structure constituting part of a search tree
   includes *parent*, *children*, *depth*, *path cost* $g(x)$
*States* do not have parents, children, depth, or path cost!

The **Expand** function creates new nodes, filling in the various fields and
using the **Operators** (or **SuccessorFn**) of the problem to create the
 corresponding states.
A* Search

Idea: avoid expanding paths that are already expensive

Evaluation function $f(n) = g(n) + h(n)$

$g(n) =$ cost so far to reach $n$
$h(n) =$ estimated cost to goal from $n$
$f(n) =$ estimated total cost of path through $n$ to goal

A* search uses an admissible heuristic
i.e., $h(n) \leq h^*(n)$ where $h^*(n)$ is the true cost from $n$.

E.g., $h_{SLD}(n)$ never overestimates the actual road distance

Theorem: A* search is optimal
Competitive Agents

- Zero sum games

Perfect play for deterministic, perfect-information games
Idea: choose move to position with highest minimax value
= best achievable payoff against best play

E.g., 2-ply game:

```
MAX

MIN
```

• Rock, scissors, paper example
• How to generalize for non-zero sum games
Agent Functions as Programs

An agent is completely specified by the **agent function**
mapping percept sequences to actions

(In principle, one can supply each possible sequence to see what it does. Obviously, a lookup table would usually be immense.)

One agent function (or a small equivalence class) is **rational**

**Aim:** find a way to implement the rational agent function concisely

An **agent program** takes a single percept as input, keeps internal state:

```plaintext
function SKELETON-AGENT( percept ) returns action
    static: memory, the agent’s memory of the world
    memory ← UPDATE-MEMORY( memory, percept )
    action ← CHOOSE-BEST-ACTION( memory )
    memory ← UPDATE-MEMORY( memory, action )
    return action
```

CSI 535 - Introduction to A.I. Lecture 2
Agent Types

Four basic types in order of increasing generality:

– simple reflex agents
– reflex agents with state
– goal-based agents
– utility-based agents
Vacuum World

Percepts (<bump> <dirt> <home>))

Actions shutoff forward suck (turn left) (turn right)

Goals (performance measure on environment history)
  - +100 for each piece of dirt cleaned up
  - -1 for each action
  - -1000 for shutting off away from home

Environment
  - grid, walls/obstacles, dirt distribution and creation, agent body
  - movement actions work unless bump into wall
  - suck actions put dirt into agent body (or not)