Lecture 1: Introduction

Intelligent versus Automated

- *Automated*: run within a well-defined set of parameters and are very restricted in what tasks they can perform
- *Intelligent, autonomous systems*: self-governing, adapts to changes in the environment

Systems	Autonomous or Automated?
ATM	Automated
Disease outbreak detection	Intelligent
Kettle with automatic shut off	Automated
Self-driving cars	Intelligent
Warehouse robots	Insufficient Information

Types of Intelligent Decisions

Single Step Decisions
- Deciding which is the "best" action to select for a given input
- Output does not affect future input or output

Sequential Decision Making: Deciding which action to select for a given input or situation, considering future actions

Lecture 2: Agents

Agent: anything that perceives its environment through sensors and acts on that environment through actuators

PEAS Descriptions of Task Environments

Performance, Environment, Actuators, Sensors

Example: Medical Diagnosis Agent

Performance Measure	Environment	Actuators	Sensors
Healthy patient, minimize costs, lawsuits	Patient, hospital, staff,	Display questions, tests, diagnoses, treatments, referrals	Keyboard entry of symptoms, findings, patient's answers

Task Environment	Observable	Deterministic	Episodic	Static	Discrete	Agents
Crossword puzzle	Fully	Deterministic	Sequential	Static	Discrete	Single
Poker	Partially	Stochastic	Sequential	Static	Discrete	Multi
Backgammon	Fully	Stochastic	Sequential	Static	Discrete	Multi
Taxi driving	Partially	Stochastic	Sequential	Dynamic	Continuous	Multi
Refinery controller	Partially	Stochastic	Sequential	Dynamic	Continuous	Single
Interactive English tutor	Partially	Stochastic	Sequential	Dynamic	Discrete	Multi

Properties of Environments

Fully observable: can access complete state of environment at each point in time	Partially observable: could be due to noisy, inaccurate or incomplete sensor data
Deterministic: next state of the environment completely determined by current state and agent's action	Stochastic: when actions have multiple outcomes, each prescribed by a probability
Episodic: agent's experience divided into independent, atomic episodes in which agent perceives and performs a single action in each episode.	Sequential: current decision affects all future decisions
Static: agent doesn't need to keep sensing while deciding what action to take, doesn't need to worry about time	Dynamic: environment changes while agent is thinking (changes with time)
Discrete: (note: applies to states, time, percepts, or actions)	Continuous: continuous values of states and/or actions
Single agent: single decision-making and executing entity	Multiagent: multiple decision-making/executing entities; cooperative or competitive

Rational Agent

Rational agent: for each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has

Rationality depends on 4 things:
1. Performance measure of success
2. Agent's prior knowledge of environment
3. Actions agent can perform
4. Agent's percept sequence to date

Types of Agents

Simple Reflex Agent
- Selects actions using only the current percept
- Works on condition-action rules:
  if *condition* **then** *action*

Model-based Reflex
- Maintain some internal state that keeps track of the part of the world it can't see now
- Needs model

Utility-directed Agents
- Utility measures which states are preferable to other states
- Assign numeric values to each possible outcome (utility or "happiness")
- Multidimensional utility (quality, failure rate, etc.)
- Time-dependent utility (hard/soft deadlines)
- Subjective vs. objective utility functions

Goal-directed Agents
- Goal information guides agent's actions (looks to the future)
- Sometimes achieving goal is simple e.g. from a single action
- Other times, goal requires reasoning about long sequences of actions
- Flexible: simply reprogram the agent by changing goals

Learning Agents

Successful agents split task of computing policy in 3 periods
1. Initially, designers compute some prior knowledge to include in policy
2. When deciding its next action, agent does some computation
3. Agent learns from experience to modify its behavior

Learn from experience to compensate for partial or incorrect prior knowledge

Lecture 3: Uninformed Search

Control Types

Open loop control
- Decision depends on start and goal state
- No sensing at each state

Closed loop control
- Decision depends on each state
- Sensing at each state

Evaluating BFS

	BFS	DFS
Complete?	Yes (if branching factor is finite)	Yes
Optimal?	Yes, if step costs are identical. Not guaranteed to be optimal in general	No
Time Complexity	$O(b^{d+1})$ b – maximum branching factor d – depth of least-cost solution	$O(b^m)$
Space Complexity	Terrible memory usage for large graphs	$O(bm)$

Uninformed Search

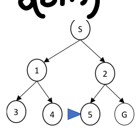
- Breadth-first search (open list is FIFO queue)
- Depth-first search (open list is a LIFO queue)
- Uniform-cost search (shallowest node first)
- Depth-limited search (DFS with cutoff)
- Iterative-deepening search (incrementing cutoff)
- Bidirectional search (forward and backward)

BFS

Expanded Nodes	Frontier List
	{S}
{S}	{1,2}
{S,1,2}	{3,4,5,G}
{S,1,2,3}	{4,5,G}
{S,1,2,3,4}	{5,G}
{S,1,2,3,4,5}	{G}
{S,1,2,3,4,5,G}	{}

DFS

Expanded Nodes	Frontier List
	{S}
{S}	{1,2}
{S,1}	{3,4,2}
{S,1,3}	{4,2}
{S,1,3,4}	{2}
{S,1,3,4,2}	{5,G}
{S,1,3,4,2,5}	{G}
{S,1,3,4,2,5,G}	{}



Lecture 4: Uninformed and Informed Search

Depth-limited Search

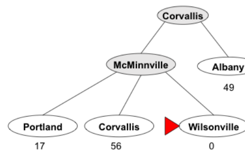
Complete?	No (If shallowest goal node beyond depth limit)
Optimal?	No (If depth limit > depth of shallowest goal node and we expand a much longer path than the optimal one first)
Time Complexity	$O(b^l)$
Space Complexity	$O(bl)$

- Solves infinite path problem by using predetermined depth limit l
- Nodes at depth l are treated as if they have no successors
- Can use knowledge of the problem to determine l (but in general you don't know this in advance)

Informed Search

Greedy Best-First Search

Greedy Best-First Search Example

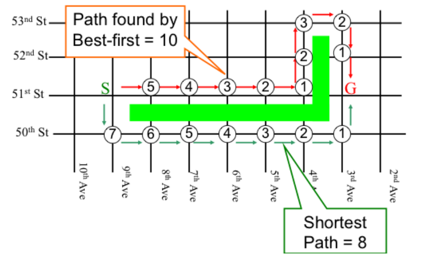


Corvallis	56
Albany	49
Salem	28
Portland	17
McMinnville	18

Corvallis →McMinnville→ Wilsonville = 74 miles

Greedy Best-First Search: Navigating in Manhattan

Complete, but not optimal



Evaluating Greedy Best-First Search

Complete?	Yes if the graph is finite and the heuristic function is informative (i.e. not 0 at all nodes).
Optimal?	No
Time Complexity	$O(b^m)$
Space Complexity	$O(b^m)$

Greedy Best-First search results in lots of unnecessary nodes being expanded

$g(n)$ = cost of path from the initial state to n

$h(n)$ = estimate of the remaining distance

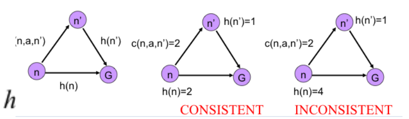
Heuristic Search: A*

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

Admissibility and Consistency

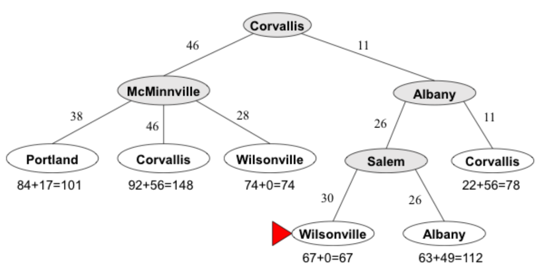
- Admissible heuristic: never overestimates the actual cost to reach a goal.
- Consistent (or monotone) heuristic:

$$h(n) \le c(n, a, n') + h$$



- Every consistent heuristic is also admissible

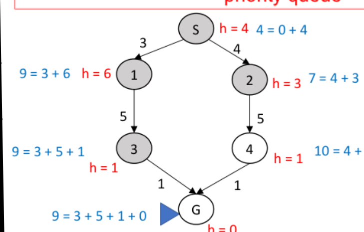
A* Search Example



Heuristic: h(n)

Corvallis	56
Albany	49
Salem	28
Portland	17
McMinnville	18

Proper termination: Stop when you pop a goal state from the priority queue



Queue:
S-1-3-G
S-2-4

Comparing Heuristics

Given two heuristics, how to evaluate which one is better?

- If a heuristic dominates another heuristic, it is strictly better

1. If h_1 and h_2 are admissible, is $\min\{h_1, h_2\}$ admissible? Yes
   - Is it better than h_1 and h_2? No
2. If h_1 and h_2 are admissible, is $\max\{h_1, h_2\}$ admissible? Yes
   - Is it better than h_1 and h_2? Yes
3. If h_1 and h_2 are admissible and h_1 strictly dominates h_2, $(i.e\ h_{1(n)} \ge h_{2(n)})$, is h_1 better than h_2? Yes

Summary of A* Search

Complete?	Yes if h(n) is admissible, b is finite, and all step costs exceed some finite ε[1]
Optimal?	Yes if h(n) is admissible
Time Complexity	$O(b^d)$ (In the worst case but a good heuristic can reduce this significantly)
Space Complexity	$O(b^d)$

Local Search

6

1. Hill-climbing
2. Simulated Annealing
3. Beam Search
4. Genetic Algorithms
5. Gradient Descent

Hill-climbing (Intuitively)

- Starting at initial state X, keep moving to the neighbor with the highest objective function value greater than X's.

Hill Climbing Search

- Hill-climbing also called greedy local search
- Greedy because it takes the best immediate move
- Greedy algorithms often perform quite well

Problems:

- Can get stuck at a local maximum.
- Unable to find its way off a plateau.
- Cannot climb along a narrow ridge when each possible step goes down.

- Disadvantage: all k states can become stuck in a small region of the state space
- To fix this, use stochastic beam search
- Stochastic beam search:
  - Doesn't pick best k successors
  - Chooses k successors at random, with probability of choosing a given successor being an increasing function of its value

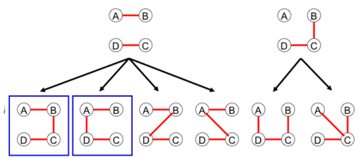
Simulated Annealing

- Hill-climbing never makes a downhill move
- What if we added some random moves to hill-climbin help it get out of local maxima?
- This is the motivation for simulated annealing
- Generate successors randomly
- Allow "bad" moves with some probability
- How to select p?

Local Beam Search

Local Beam Search Example

Travelling Salesperson Problem (k=2)



Select the best k successors from the complete list and repeat the process

Genetic Algorithms

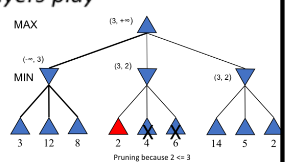
- Like natural selection in which an organism creates offspring according to its fitness for the environment
- Essentially a variant of stochastic beam search that combines two parei states
- Over time, population contains individuals with high fitness

- An individual program is represented by a sequence of "genes".
- The selection strategy is randomized with probability of selection proportional to "fitness".
- Individuals selected for reproduction are randomly paired, certain genes are crossed-over, and some are mutated.

Lecture 7: Adversarial Search

The Minimax Value of a Node

The minimax value of a node is the utility for MAX of being in the corresponding state, *assuming that both players play optimally* from there to the end of the game



Minimax_value(n) =

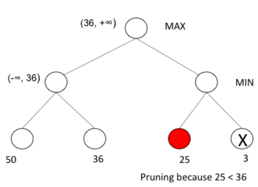
$$\begin{cases} \text{UTILITY}(n) & \text{If n is a terminal state} \\ Max_{s \in Successors(n)} \text{Minimax\_value}(s) & \text{If n is a MAX node} \\ Min_{s \in Successors(n)} \text{Minimax\_value}(s) & \text{If n is a MIN node} \end{cases}$$

Minimax value maximizes worst-case outcome for MAX

Properties

- Computes minimax decision from the current state
- Depth-first exploration of the game tree
- Complete? Yes, if the graph is finite
- Optimal? Yes, against an optimal opponent
- Time Complexity: $O(b^m)$ where b=# of legal moves, m=maximum depth of tree
- Space Complexity:
  - O(bm) if all successors generated at once
  - O(m) if only one successor generated at a time (each partially expanded node remembers which successor to generate next)

ıss Exercise #1: Alpha-Beta Pruning



Alpha-Beta Pruning: Intuition

```
function MAX-VALUE(state, α, β) returns a utility valu
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

max: $v \geq \beta$

```
function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

min: $v < \alpha$

Lecture 8: Game Theory

Dominant Strategies

- Normal form game: form of representing the game
- Strategies: different actions/ decision options
- Payoffs: utility of each decision
- Pure strategy: deterministic strategy selection
- Mixed strategy: probabilistic strategy selection

Suppose a player has two strategies S and S'. We say S dominates S' if choosing S always yields at least as good an outcome as choosing S'.

- S strictly dominates S' if choosing S always gives a better outcome than choosing S' (no matter what the other player does)
- S weakly dominates S' if there is one set of opponent's actions for which S is superior, and all other sets of opponent's actions give S and S' the same payoff.

Example of Dominant Strategies

	Bob: testify	Bob: refuse
Alice: testify	A = -5, B = -5	A = 0, B = -10
Alice: refuse	A = -10, B = 0	A = -1, B = -1

If Bob testifies, "testify" strongly dominates "refuse" strategy for Alice

	Bob: testify	Bob: refuse
Alice: testify	A = -5, B = -5	A = 0, B = -10
Alice: refuse	A = -10, B = 0	A = 0, B = -1

If Bob refuses, "testify" weakly dominates "refuse" for Alice

Note

Lecture 9: Game Theory

Strategies and Equilibria

- Dominant strategy: A player's best move, regardless of what other players do.
- Pareto optimality: A state where no one can be made better off without making someone else worse off (i.e. the best for all players)
- Nash equilibrium: A situation where no player can improve their outcome by changing their strategy, while other players keep theirs the same.
- Dominant strategy equilibrium: A Nash equilibrium where all players have a dominant strategy.

How to Spot a Nash Equilibrium

B

A	S1	S2	S3
S1	A = 0, B = 4	A = 4, B = 0	A = 5, B = 3
S2	A = 4, B = 0	A = 0, B = 4	A = 5, B = 3
S3	A = 3, B = 5	A = 3, B = 5	A = 6, B = 6

A won't change her Strategy of S3

Payoff of 6 > 5 (S2) and 6 > 5 (S1)

B won't change his Strategy of S3

Payoff of 6 > 5 (S2) and 6 > 5 (S1)

Dominant Strategy Equilibrium

	Bob: testify	Bob: refuse
Alice: testify	A = -5, B = -5	A = 0, B = -10
Alice: refuse	A = -10, B = 0	A = -1, B = -1

- (testify,testify) is a dominant strategy equilibrium
- It's an equilibrium because no player can benefit by switching strategies given that the other player sticks with the same strategy
- An equilibrium is a local optimum in the space of policies

	B: S1	B: S2	B: S3
A: S1	A=0, B=2	A=5, B=3	A=2, B=1
A: S2	A=1, B=-3	A=-2, B=1	A=7, B=4
A: S3	A=10, B=10	A=3, B=8	A=1, B=6

- Does Player A have a strictly dominant strategy? If so, which one? No

- Pareto optimality: A state where no one can be made better off without making someone else worse off (i.e. the best for all players)

	Bob: testify	Bob: refuse
Alice: testify	A = -5, B = -5	A = 0, B = -10
Alice: refuse	A = -10, B = 0	A = -1, B = -1

	Best: cloud	Best: VR
ACME: cloud	A = 9, B = 9	A = -3, B = -1
ACME: VR	A = -4, B = -1	A = 5, B = 5

There are two Nash Equilibria in this game. In general, you can have multiple Nash Equilibria.

Nash equilibrium: A situation where no player can improve their outcome by changing their strategy, while other players keep theirs the same.

	Bob: testify	Bob: refuse
Alice: testify	A = -5, B = -5	A = 0, B = -10
Alice: refuse	A = -10, B = 0	A = -1, B = -1

Mixed Strategies

- Recall that a pure strategy is a deterministic policy i.e. you pick a strategy and play it all the time
- A mixed strategy is a randomized policy i.e. you select your strategy based on a probability distribution
- E.g. Select strategy S1 with probability p and strategy S2 with probability (1-p)
- Is there a mixed strategy Nash Equilibrium in 2 Fingered Morra?