CSC242: Intro to AI

Lecture 7
Games of Imperfect Knowledge &
Constraint Satisfaction
What is This?
Moral

- Many people cannot learn from lectures
- Do the homework!
  - If you do, exams will be easy
  - If you don’t, exams will be impossible
- First exam: February 20
The Road Ahead

- Complete calendar for the semester now online
- Topics
- Exam dates
- Project dates
- 3 programming projects
- 3 in-class exams + final exam
Othello

- Phase I due Feb 25
- Project page updated, (re)check details!
- Generating legal moves is not trivial!
  - A legal move must capture some pieces!
- My own solution: 125 lines of Python
  - == 125 lines of C == 250 lines of Java
Stochastic Games of Perfect Information
Stochastic Games of Perfect Information

Examples:
- Backgammon
- Roulette
- Candyland
- Parcheesi

Why stochastic?

Why perfect information?
Stochastic Games of Perfect Information

Examples:
- Backgammon
- Roulette
- Candyland
- Parcheesi

Why stochastic? Contains a random element

Why perfect information?
Stochastic Games of Perfect Information

- **Examples:**
  - Backgammon
  - Roulette
  - Candyland
  - Parcheesi

- Why stochastic? Contains a random element
- Why perfect information? *No hidden state!*
Expecti-Minimax

- Same as MINIMAX for MIN and MAX nodes
- Same backing up utilities from terminal nodes
- Take expectation over chance nodes
- Weighted average of possible outcomes
Expecti-Minimax

\[
EMinimax(s) = \begin{cases} 
  Utility(s) & \text{if } Terminal-Test(s) \\
  \max_a EMinimax(Result(S, a)) & \text{if } Player(s) = \text{max} \\
  \min_a EMinimax(Result(S, a)) & \text{if } Player(s) = \text{min} \\
  \sum_r P(r) EMinimax(Result(S, r)) & \text{if } Player(s) = \text{chance}
\end{cases}
\]
Partial Observability

• Some of the state of the world is hidden (unobservable)
Partially-Observable Games

• Some of the state of the game is hidden from the player(s)

• Interesting because:
  • Valuable real-world games like poker
  • Partial observability arises all the time in real-world problems
Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - No element of randomness
  - Examples?
Partially-Observable Games

- Deterministic partial observability
  - Opponent has hidden state
  - Battleship, Stratego
Partially-Observable Games

• Deterministic partial observability
  • Opponent has hidden state
  • Battleship, Stratego

• Stochastic partial observability
  • Hidden information is random
  • Examples?
Stochastic Partially Observable Games
<table>
<thead>
<tr>
<th>Hand</th>
<th>Frequency</th>
<th>Approx. Probability</th>
<th>Approx. Cumulative</th>
<th>Approx. Odds</th>
<th>Mathematical expression of absolute frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Royal flush</td>
<td>4</td>
<td>0.000154%</td>
<td>0.000154%</td>
<td>649,739 : 1</td>
<td>( \binom{4}{1} )</td>
</tr>
<tr>
<td>Straight flush (excluding royal flush)</td>
<td>36</td>
<td>0.00139%</td>
<td>0.00154%</td>
<td>72,192.33 : 1</td>
<td>( \binom{10}{1} \binom{4}{1} - \binom{4}{1} )</td>
</tr>
<tr>
<td>Four of a kind</td>
<td>624</td>
<td>0.0240%</td>
<td>0.0256%</td>
<td>4,164 : 1</td>
<td>( \binom{13}{1} \binom{12}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>Full house</td>
<td>3,744</td>
<td>0.144%</td>
<td>0.170%</td>
<td>683.2 : 1</td>
<td>( \binom{13}{1} \binom{4}{1} \binom{12}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>Flush (excluding royal flush and straight flush)</td>
<td>5,108</td>
<td>0.197%</td>
<td>0.367%</td>
<td>507.8 : 1</td>
<td>( \binom{13}{5} \binom{4}{1} - \binom{10}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>Straight (excluding royal flush and straight flush)</td>
<td>10,200</td>
<td>0.392%</td>
<td>0.76%</td>
<td>253.8 : 1</td>
<td>( \binom{10}{1} \binom{4}{5} - \binom{10}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>Three of a kind</td>
<td>54,912</td>
<td>2.11%</td>
<td>2.87%</td>
<td>46.3 : 1</td>
<td>( \binom{13}{1} \binom{4}{1} \binom{12}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>Two pair</td>
<td>123,552</td>
<td>4.75%</td>
<td>7.62%</td>
<td>20.03 : 1</td>
<td>( \binom{13}{2} \binom{4}{2} \binom{11}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>One pair</td>
<td>1,098,240</td>
<td>42.3%</td>
<td>49.9%</td>
<td>1.36 : 1</td>
<td>( \binom{13}{1} \binom{4}{2} \binom{12}{1} \binom{4}{1} )</td>
</tr>
<tr>
<td>No pair / High card</td>
<td>1,302,540</td>
<td>50.1%</td>
<td>100%</td>
<td>.995 : 1</td>
<td>( \binom{13}{5} - 10 \binom{4}{5} - 4 )</td>
</tr>
<tr>
<td>Total</td>
<td>2,598,960</td>
<td>100%</td>
<td>100%</td>
<td>1 : 1</td>
<td>( \binom{52}{5} )</td>
</tr>
</tbody>
</table>


Weighted Minimax

- For each possible deal $s$:
  - Assume $s$ is the actual situation
  - Compute Minimax or H-Minimax value of $s$
  - Weight value by probability of $s$
  - Take move that yields highest expected value over all the possible deals
Weighted Minimax

$$\text{argmax}_a \sum_s P(s) \text{Minimax}(\text{Result}(s, a))$$
Weighted Minimax

\[
\operatorname{argmax}_a \sum_s P(s) \operatorname{MINIMAX}(\operatorname{RESULT}(s, a))
\]

2-Player Hearts: \[
\begin{pmatrix} 52 - 13 \\ 13 \end{pmatrix} = 8 \times 10^9
\]

4-Player Hearts: \[
\begin{pmatrix} 39 \\ 13 \end{pmatrix} \begin{pmatrix} 26 \\ 13 \end{pmatrix} \begin{pmatrix} 13 \\ 13 \end{pmatrix} = 8 \times 10^{16}
\]

4-Player Poker: \[
\begin{pmatrix} 47 \\ 5 \end{pmatrix} \begin{pmatrix} 42 \\ 5 \end{pmatrix} \begin{pmatrix} 37 \\ 5 \end{pmatrix} = 1 \times 10^{17}
\]
Monte Carlo Methods

- Use a “representative” sample to approximate a large, complex distribution
Monte-Carlo Minimax

\[
\arg\max_a \frac{1}{N} \sum_{i=1}^{N} \text{Minimax}(\text{Result}(s_i, a))
\]

- Can also sample during minimax search
- Equivalently: expand a random sample of children at each level
- Used in champion card playing programs
- Bridge, Poker
Monte Carlo MiniMax

• Useful even for deterministic games of perfect information that have very high branching factors!
Summary

- Stochastic games
  - Expecti-MINIMAX: Compute expected MINIMAX value over chance nodes
- Partially observable games
  - Weighted MINIMAX: Compute expected value over possible hidden states
  - When tree becomes too large, sample branches rather than explore exhaustively
Constraint Satisfaction
What is Constraint Satisfaction?

In most of the search problems we have discussed up to now, a solution corresponds to a path or the initial step in a path through a state space.

- Route-finding
- Solving the 8 Puzzle
- Game playing
What is Constraint Satisfaction?

In many other problems, however, a solution is a goal state itself. We don't care what the path is to the goal. We can easily test if a state is a goal. But we must search to find a state that makes the test true.

What problem did we see that was like this?
The Problem With State-Space Search

- State representation is specific to a given problem (or domain of problems)
- Functions on states (successor generation, goal test) are specific to the state representation
- Heuristic functions are both problem-specific and dependent on the state representation
- Many design choices, many opportunities for coding errors
The CSP Approach

• Impose a structure on the representation of states

• Using that representation, successor generation and goal tests are problem- and domain-independent

• Can also develop effective problem- and domain-independent heuristics
Bottom Line

Represent State This Way

Write No Code!

No Bugs!
Assign a color to each region such that no two neighboring regions have the same color.
enum Color = red, green, blue
Color WA, NT, Q, NSW, V, SA, T

enum Color = red, green, blue

State: assignment of colors to regions
Successor function: pick an unassigned region and assign it a color
Goal test: All regions assigned and no adjacent regions have the same color
Color WA, NT, Q, NSW, V, SA, T

def enum Color = red, green, blue

WA = red, NT = green, Q = red, NSW = green, V = red, SA = blue, T = red
Constraint Satisfaction Problem (CSP)

X: Set of variables \{ X_1, ..., X_n \}
D: Set of domains \{ D_1, ..., D_n \}
  Each domain \( D_i \) = set of values \{ v_1, ..., v_k \}
C: Set of constraints \{ C_1, ..., C_m \}
Australia Map CSP

\[ X: \{ X_i \} = \{ WA, NT, Q, NSW, V, SA, T \} \]

\[ D: \text{Each } D_i = \{ \text{red, green, blue} \} \]

\[ C: \{ SA \neq WA, SA \neq NT, SA \neq Q, SA \neq NSW, \]
\[ \quad \text{SA} \neq V, \text{WA} \neq NT, \text{NT} \neq Q, Q \neq NSW, \]
\[ \quad \text{VSW} \neq V \} \]
More CSP Terminology

- Assignment: \{ X_i = v_i, X_j = v_j, \ldots \}
- Consistent: does not violate any constraints
- Partial: some variables are unassigned
- Complete: every variable is assigned
- Solution: consistent, complete assignment
Constraints

• Unary constraint: one variable
  • e.g., NSW $\neq$ red, $X_i$ is even, $X_i = 2$

• Binary constraint: two variables
  • e.g., NSW $\neq$ WA, $X_i > X_j$, $X_i + X_j = 2$

• “Global” constraint: more than two vars
  • e.g., $X_i$ is between $X_j$ and $X_k$, $\text{AllDiff}(X_i, X_j, X_k)$

• Can be reduced to set of binary constraints (possibly inefficiently)
• Faster to search (solve)
• Problem-independent (no code!)
• Constraint propagation
<table>
<thead>
<tr>
<th>State</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R, G, B</td>
</tr>
<tr>
<td>NT</td>
<td>R, G, B</td>
</tr>
<tr>
<td>SA</td>
<td>R, G, B</td>
</tr>
<tr>
<td>Q</td>
<td>R, G, B</td>
</tr>
<tr>
<td>NSW</td>
<td>R, G, B</td>
</tr>
<tr>
<td>V</td>
<td>R, G, B</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

Possibilities: $3^7 = 2,187$
<table>
<thead>
<tr>
<th>State</th>
<th>Color Options</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R, G, B</td>
</tr>
<tr>
<td>NT</td>
<td>R, G, B</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R, G, B</td>
</tr>
<tr>
<td>NSW</td>
<td>R, G, B</td>
</tr>
<tr>
<td>V</td>
<td>R, G, B</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

Make choice: color SA blue

Remaining possibilities: $3^6 = 729$
<table>
<thead>
<tr>
<th></th>
<th>WA</th>
<th>NT</th>
<th>SA</th>
<th>Q</th>
<th>NSW</th>
<th>V</th>
<th>T</th>
</tr>
</thead>
</table>

Simplify: remove B from adjacent regions

Remaining possibilities:

$$2^5 \times 3 = 96$$
<table>
<thead>
<tr>
<th>State</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R, G</td>
</tr>
<tr>
<td>NT</td>
<td>R, G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>R, G</td>
</tr>
<tr>
<td>V</td>
<td>R, G</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

Make choice: color Q red

Remaining possibilities: $2^4 \times 3 = 48$
<table>
<thead>
<tr>
<th>Region</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R, G</td>
</tr>
<tr>
<td>NT</td>
<td>G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R, G</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

Simplify: remove R from adjacent regions

Remaining possibilities: \(2^2 \times 3 = 12\)
<table>
<thead>
<tr>
<th>State</th>
<th>Colors</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R, G</td>
</tr>
<tr>
<td>NT</td>
<td>G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R, G</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

NT and NSW are forced G

Remaining possibilities: \(2^2 \times 3 = 12\)
<table>
<thead>
<tr>
<th>Region</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R</td>
</tr>
<tr>
<td>NT</td>
<td>G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

Simplify: remove G from adjacent regions

Remaining possibilities: 3
<table>
<thead>
<tr>
<th>State</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R</td>
</tr>
<tr>
<td>NT</td>
<td>G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R</td>
</tr>
<tr>
<td>T</td>
<td>R, G, B</td>
</tr>
</tbody>
</table>

WA and V are forced red

Remaining possibilities: 3
<table>
<thead>
<tr>
<th>Region</th>
<th>Color</th>
</tr>
</thead>
<tbody>
<tr>
<td>WA</td>
<td>R</td>
</tr>
<tr>
<td>NT</td>
<td>G</td>
</tr>
<tr>
<td>SA</td>
<td>B</td>
</tr>
<tr>
<td>Q</td>
<td>R</td>
</tr>
<tr>
<td>NSW</td>
<td>G</td>
</tr>
<tr>
<td>V</td>
<td>R</td>
</tr>
<tr>
<td>T</td>
<td>R</td>
</tr>
</tbody>
</table>

Choose: any color for T

Solved!
Constraint Propagation

• Using the constraints to reduce the set of legal values of a variable, which can in turn reduce the legal values of another variable, and so on

• Not a search process itself!

• Part of state update in state-space search

• A type of inference: making implicit information explicit
Arc-Consistency

- The particular kind of constraint propagation we just saw is called **arc-consistency**

- Why? Because it involves considering 2 nodes at a time (the ends of an arc)

- There are other kinds of constraint propagation, but arc-consistency is usually the most practical
Constraint Propagation

- Can be used as pre-processing step for any kind of search
- Including local search
- Can be interleaved with any kind of search over partial assignments, where the action is “assign a value to an unassigned variable”
- Popular choice: depth-first search
Domain-Independent Heuristics

• There are good heuristics for deciding which variable to assign next

• Choose one with the smallest domain
  • Maximizes likelihood of making a correct choice!

• Choose one involved in largest number of constraints
  • Likely to lead to lots of constraint propagation!
Check Your Understanding

- Why can’t you use constraint propagation after each step of local search?
Check Your Understanding

• Why can’t you use constraint propagation after each step of local search?

• Because local search is over complete states

• Every variable has a particular value

• You can’t therefore remove a value from the domain of a variable
CSPs Summary

- Impose a structure on the representation of states: Variables, Domains, Constraints
- Backtracking search for complete, consistent assignment of values to variables
- Inference (constraint propagation) can reduce the domains of variables
- Preprocessing or interleaved with search