CSC242: Intro to AI

Lecture 7
Quiz
Stop Time: 2:15
TA Help Sessions (v2)

- Monday & Tuesday: 17:00–18:00, Hylan 301
- Doodle poll signup before 16:00
- Link on BB:
  http://www.doodle.com/xgxcbxn4knks86sx
Project 1: TTT

- Due in BB Mon Feb 11 23:00
- Late submissions: 10%/day, max 3 days
- Grading: 50% code, 50% writeup
On Implementing TTT

Do I have to use the state-space search framework?

I can code up the optimal strategy for 3x3 TTT in a few if statements

Maybe with a few loops for NxN generality
Adversarial Search
Part II
Adversarial Search
Part II
Games ≠ Toy Problems

$9! = 362880$

$35^{100} = 10^{154}$

$2 \times 10^{170}$
Games

- “ Require the ability to make some decision even when calculating the optimal decision is infeasible”
- “Penalize inefficiency severely”
# Types of Games

<table>
<thead>
<tr>
<th>Deterministic (no chance)</th>
<th>Nondeterministic (dice, cards, etc.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perfect information (fully observable)</td>
<td>Imperfect information (partially observable)</td>
</tr>
<tr>
<td>Zero-sum (total payoff the same in any game)</td>
<td>Arbitrary utility functions</td>
</tr>
</tbody>
</table>
Minimax Algorithm

\[
\text{Minimax}(s) = \begin{cases} 
\text{Utility}(s) & \text{if Terminal-Test}(s) \\
\max_{a \in \text{Actions}(s)} \text{Minimax(\text{Result}(s, a))} & \text{if Player}(s) = \text{max} \\
\min_{a \in \text{Actions}(s)} \text{Minimax(\text{Result}(s, a))} & \text{if Player}(s) = \text{min}
\end{cases}
\]
Minimax Summary

• Computes the optimal move assuming opponent also plays optimally (i.e., worst-case outcome)

• Explores game tree depth-first all the way to terminal states (end of game)

• Backs up utility values through alternating MIN and MAX (what’s best for me is worst for you, and vice-versa)
Minimax Analysis

\[ O(b^m) \] Time Complexity

\[ O(b^{bm}) \] Space Complexity
Heuristic Minimax

- Cutoff search before reaching terminal nodes (time, depth, "quiescence")
- Use heuristic evaluation function to estimate state utility
- Backs up utility values through alternating MIN and MAX (what’s best for me is worst for you, and vice-versa)
Heuristic Minimax

\[ H-\text{MINIMAX}(s) = \begin{cases} 
    h(s) & \text{if CUTOFF-TEST}(s) \\
    \max_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MAX} \\
    \min_{a \in \text{ACTIONS}(s)} \text{MINIMAX}(\text{RESULT}(s, a)) & \text{if PLAYER}(s) = \text{MIN} 
\end{cases} \]
H-Minimax: Cutoff

- When to cutoff search?
  - Time
  - Depth
  - "Quiescence"
- Can adjust dynamically
- AIMA 5.4.2
H-Minimax: Heuristic

• How to evaluate non-terminal state?
  • AIMA 5.4.1
• Chess example
  • Material value
  • Weighted sum
  • Strategic considerations
Today

- How to improve MINIMAX to make it more practical
Minimax Analysis

\[ O(b^m) \quad \text{Time Complexity} \]

\[ O(b^m) \quad \text{Space Complexity} \]
Highest utility seen so far (lower bound on max value)

Lowest utility seen so far (upper bound on min value)
MAX

MIN

3

-∞, +∞
Max

-∞, +∞  A

Min

-∞, 3  B

3  12  8
Value of C is at most 2
Alpha-Beta Pruning
Alpha-Beta Pruning

- During MINIMAX search
- Keep track of
  - \( \alpha \): value of best choice so far for MAX (lower bound on MAX utility)
  - \( \beta \): value of best choice so far for MIN (upper bound on MIN utility)
- Prune when value of node is known to be worse than \( \alpha \) (for MAX) or \( \beta \) (for MIN)
Alpha-Beta Pruning

• Still MINIMAX search
• Optimal (if you search to the terminals)
• Optimal with respect to your heuristic function otherwise
Alpha-Beta Pruning Analysis
Alpha-Beta Pruning Analysis

Ideal case:

Always explore the best successor first: \( O(b^{m/2}) \)

Branching factor: \( b^{1/2} = \sqrt{b} \)

Explore twice as deep a tree in same time
Alpha-Beta Pruning Analysis

Random case:

Explore successors in random order: $O(b^{3m/4})$

Branching factor: $b^{3/4}$

Explore 4/3 as deep a tree in same time
Alpha-Beta Pruning Analysis

“Smart” case:

• e.g., for chess order by:
  • Captures
  • Threats of captures
  • Forward moves
  • Backward moves
Alpha-Beta Summary

- Easy bookkeeping modification of basic MINIMAX algorithm
- Not hard to come up with “useful” node orderings
- Even random gets you 33% deeper search
- Works with other ways of improving game tree search
For Next Time:

Project 1 due
Mon Feb 11 23:00

For Tue
AIMA 5.5-5.6; 5.7-5.9 fyi