Lecture 5: Minimax, Alpha-Beta pruning & expecti-minimax

AI For Traditional Games
Prof. Nathan Sturtevant
Winter 2011

Analyze strategies

• What is needed to represent a full strategy for a single player?
  • Assume uniform branching factor $b$
  • Assume search tree depth $d$
  • Assume $d$ is even
• This is an AND/OR tree
  • Need single choice at every MAX node
  • Need to consider all choices at every MIN node

Minimax strategy analysis

• Minimax shows that max can get at least $p$
• Minimax shows that min gets no less than $p$
• Requires a strategy for both players
• $b^\lfloor d/2 \rfloor + b^\lceil d/2 \rceil - 1 = O(b^{d/2})$
• Strategies must overlap at a single leaf node
• Assumes that the game isn’t a win at the root
• Otherwise don’t need the min strategy

(Strategy analysis cont’d)

• 1 move at root for MAX
• $b$ moves at depth 2
• 1 move at depth 3 (MAX)
• $b^\lfloor d/2 \rfloor$ total leaf nodes
• Similar for MIN player
  • $b^\lceil d/2 \rceil$
Minimum tree example

- From Korf

![Minimum Depth-Five Binary Minimax Tree](image)

Figure 8.1: Minimal Depth-Five Binary Minimax Tree

Expecti-Minimax

- What if we have chance nodes?
  - We can use minimax
    - We have to average over chance nodes

- Simple game:
  - Choose to roll a 6-sided die (or let opponent go)
    - Then get a 3 to win.
  - Opponent chooses to go first or second,
    - Rolls a die
    - Then we play nim starting from that value

Expecti-Minimax

```plaintext
GetChanceVal(depth)
    if (depth == 0) || Done() return CutoffEval();
    currVal ← 0
    for each successor s in 1... # successors
        ApplyMove(s)
        currVal += Prob(s) * GetMinVal(depth-1)
    UndoMove(s)
return currVal
```

Expected: -0.666

Expected: -0.333
Other details

- When probabilities are involved, can’t just have ordering on payoffs, as they are averaged together by chance nodes
- Can do alpha-beta style pruning, but more complicated
  - Need bounds on payoffs
  - Still have to do a lot of the computation

What depth should I search?

- Cannot know proper search depth \textit{a priori}
- Need a method to dynamically choose search depth
  - Possibilities?
    - Iterative deepening approach
      - Search all depths!

Total nodes expanded by minimax

\[
N(b, d) = 1 + b + b^2 + \ldots + b^{d-1} + b^d
\]
\[
b \cdot N(b, d) = b + b^2 + \ldots + b^{d-1} + b^d + b^{d+1}
\]
\[
b \cdot N(b, d) - N(b, d) = b^{d+1} - 1
\]
\[
N(b, d) \cdot (b - 1) = b^{d+1} - 1
\]
\[
N(b, d) = \frac{b^{d+1} - 1}{b - 1}
\]
\[
N(b, d) \approx \frac{b^{d+1}}{b - 1} = b^d \frac{b}{b - 1}
\]

Nodes expanded by iterative deepening

- Minimax work at depth \(d\): \(b^d \cdot b/(b-1)\)

\[
\frac{b}{b-1} \left( b^d + b^{d-1} + b^{d-2} + \ldots + b + 1 \right)
\]
\[
= b^d \left( \frac{b}{b-1} \right)^2
\]
Advantages of iterative deepening

• How do I order nodes?
  • Hand analysis/heuristics
  • Use information from the previous iterations
    • Start with principal variation
    • History Heuristic

Move ordering

• Can we learn a good move ordering?
  • Killer Heuristic:
    • Find the move that is causing the most cutoffs at each depth of the tree
    • Try it first

Move Ordering

• Killer heuristic just finds a single move
  • Can we generalize to all moves?
• Simplistic “learning”

History Heuristic

• Each move has a score
  • Increment score whenever a move is the best move in a state (or causes a cutoff)
  • Score is $2^d$, where $d$ is the depth of the tree analyzed below the move
• Sort moves by score
Null move

• If one player is in a strong position, they could skip their turn and still win
  • Perform “null” move
  • Search to depth 2/3 ply shallower than required
    • If the value of the game is still better on previous branches, it’s a win
    • Otherwise re-search with the full tree
  • Hop Step?
  • zugzwang -- sometimes it’s better not to move

Horizon Effect

• A result of limited depth knowledge
• Something bad is about to happen, but find a way to delay it until it happens after the search depth
• May turn a minor problem into a catastrophic one
  • Make a bad move now to avoid a worse state by the horizon effect
  • The worse state still happens later

Quiescence search

• Quiescence = quiet
  • Searching to a fixed depth may not be advantageous
  • eg if a capture has just been made, and the capture response hasn’t
  • Extend search until position is quiet
    • eg no captures and no check