CSE 473
Lecture 7
Playing Games with Minimax and Alpha-Beta Search

Today

• Adversarial Search
  Minimax recap
  α-β search
  Evaluation functions
  State of the art in game playing
Recall: Game Trees

From current position, unwind game into the future

You

MAX(X)

Opponent

MIN(O)

You

MAX(X)

Opponent

MIN(O)

TERMINAL

Utility

Recall: Minimax Search

• Find the best current move for MAX (you) assuming MIN (opponent) also chooses its best move

• Compute for each node $n$:

$\text{MINIMAX-VALUE}(n) =$

\begin{align*}
\text{UTILITY}(n) & \quad \text{if } n \text{ is a terminal} \\
\max_{s \in \text{succ}(n)} \text{MINIMAX-VALUE}(s) & \quad \text{if } n \text{ is a MAX node} \\
\min_{s \in \text{succ}(n)} \text{MINIMAX-VALUE}(s) & \quad \text{if } n \text{ is a MIN node}
\end{align*}
Example: Two-“Ply” Game Tree

Two-Ply Game Tree

(1 ply = 1 move = 1 layer in tree)
Two-Ply Game Tree

What if MIN does not play optimally?

- Definition of optimal play for MAX assumes MIN plays optimally
  
  Maximizes worst-case outcome for MAX

- If MIN does not play optimally, MAX will do even better (utility obtained by MAX will be higher). [Prove it! See Exercise 5.7]
Minimax Algorithm

function Minimax-Decision(state) returns an action
  \( v \leftarrow \text{Max-Value}(state) \)
  return the action in Successors(state) with value \( v \)

function Max-Value(state) returns a utility value
  if Terminal-Test(state) then return Utility(state)
  \( v \leftarrow -\infty \)
  for \( s \) in Successors(state) do
    \( v \leftarrow \text{Max}(v, \text{Min-Value}(s)) \)
  return \( v \)

function Min-Value(state) returns a utility value
  if Terminal-Test(state) then return Utility(state)
  \( v \leftarrow \infty \)
  for \( s \) in Successors(state) do
    \( v \leftarrow \text{Min}(v, \text{Max-Value}(s)) \)
  return \( v \)

Example (4 ply) Which move to choose?
Choose this move
**Extension to Multiplayer Games**

- More than two players
- Single minimax values become *vectors*
- At each node, apply max to appropriate component of minimax vector

**Properties of minimax**

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
  
  Suboptimal opponents: Other strategies may do better but these will do worse for optimal opponents

- **Time complexity?** $O(b^m)$
- **Space complexity?** $O(bm)$ (depth-first exploration)
Is Minimax good enough?

- **Chess:**
  - branching factor $b \approx 35$
  - game length $m \approx 100$
  - search space $b^m \approx 35^{100} \approx 10^{154}$

- **The Universe:**
  - number of atoms $\approx 10^{78}$
  - age $\approx 10^{21}$ milliseconds

Can we search more efficiently?

---

Back to Two-Ply Game Tree

```
MAX

MIN

A_{11} A_{12} A_{13} A_{21} A_{22} A_{23} A_{31} A_{32} A_{33}

3 12 8 2 4 6 14 5 2
```
Pruning trees

Minimax algorithm explores depth-first

Pruning trees

At MIN node:
Current best MAX value $\alpha = 3 > 2$

No need to look at these nodes!! (these nodes can only decrease MIN value from 2)
Pruning trees

MAX

MIN

MAX

MIN

X

X

Pruning trees

MAX

MIN

X

X

MIN
Pruning trees

One more example
One more example

![Tree Diagram]

At MAX node: Current best MIN value $\beta = 10 < 15$

No need to look at these nodes!! (these nodes can only increase MAX value from 15)

This form of tree pruning is known as alpha-beta pruning

$\alpha =$ highest (best) value for MAX along current path from root

$\beta =$ lowest (best) value for MIN along current path from root
The \(\alpha-\beta\) algorithm
(minimax with four lines of added code)

function \textsc{Alpha-Beta-Search}(state) returns an action
inputs: state, current state in game
\[ v \leftarrow \text{Max-Value}(state, -\infty, +\infty) \]
return the action in \text{Successors}(state) with value \(v\)

function \text{Max-Value}(state, \alpha, \beta) returns a utility value
inputs: state, current state in game
\[ \alpha, \text{the value of the best alternative for MAX along the path to state} \]
\[ \beta, \text{the value of the best alternative for MIN along the path to state} \]
if \text{Terminal-Test}(state) then return \text{Utility}(state)
\[ v \leftarrow -\infty \]
for \(s\) in \text{Successors}(state) do
\[ v \leftarrow \text{Max}(v, \text{Min-Value}(s, \alpha, \beta)) \]
if \(v \geq \beta\) then return \(v\)
\[ \alpha \leftarrow \text{Max}(\alpha, v) \]
return \(v\)

The \(\alpha-\beta\) algorithm (cont.)

function \text{Min-Value}(state, \alpha, \beta) returns a utility value
inputs: state, current state in game
\[ \alpha, \text{the value of the best alternative for MAX along the path to state} \]
\[ \beta, \text{the value of the best alternative for MIN along the path to state} \]
if \text{Terminal-Test}(state) then return \text{Utility}(state)
\[ v \leftarrow +\infty \]
for \(s\) in \text{Successors}(state) do
\[ v \leftarrow \text{Min}(v, \text{Max-Value}(s, \alpha, \beta)) \]
if \(v \leq \alpha\) then return \(v\)
\[ \beta \leftarrow \text{Min}(\beta, v) \]
return \(v\)
Properties of α-β

• Pruning does not affect final result

• Good move ordering improves effectiveness of pruning

• With "perfect ordering," time complexity = \(O(b^{m/2})\)
  \(\rightarrow\) allows us to search deeper – doubles depth of search

• α-β search is a simple example of the value of reasoning about which computations are relevant (a form of metareasoning)

Good enough?

▪ Chess:
  ▪ branching factor \(b \approx 35\)
  ▪ game length \(m \approx 100\)
  ▪ \(\alpha-\beta\) search space \(b^{m/2} \approx 35^{50} \approx 10^{77}\)

▪ The Universe:
  ▪ number of atoms \(\approx 10^{78}\)
  ▪ age \(\approx 10^{21}\) milliseconds
Transposition Tables

- Game trees contain repeated states
- In chess, e.g., the game tree may have $35^{100}$ nodes, but there are only $10^{40}$ different board positions
- Similar to *explored set* in graph-search, maintain a transposition table
  - Got its name from the fact that the same state is reached by a transposition of moves.
- $10^{40}$ is still huge!

Can we do better?

- Strategies:
  - search to a fixed depth ("cut off" search)
  - *iterative deepening* (most common)
**Heuristic Evaluation Functions**

- **Motivation:** When search space is too large, create game tree up to a certain depth only.
- **Art is to** estimate utilities of positions that are not terminal states.
- **Example of simple evaluation criteria in chess:**
  - Material worth: pawn=1, knight=3, rook=5, queen=9.
  - Other: king safety, good pawn structure

\[
\text{eval}(s) = \sum \text{weight} \times \text{property}(s) + \ldots
\]

**Idea:** cut off search

Use heuristic evaluation function for these nodes
Cutting off search

- Does it work in practice?
  Suppose $b^m = 10^6$ and $b=35 \Rightarrow m=4$

- 4-ply lookahead is a hopeless chess player!
  - 4-ply $\approx$ human novice
  - 8-ply $\approx$ typical PC, human master
  - 12-ply $\approx$ Deep Blue, Kasparov

Game Playing State-of-the-Art

- **Checkers**: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions. Checkers is now solved!
Game Playing State-of-the-Art

- **Chess**: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue examined 200 million positions per second, used very sophisticated evaluation functions and undisclosed methods for extending some lines of search up to 40 ply. Current programs are even better, if less historic.

- **Othello**: Human champions refuse to play against computers because software is too good.

- **Go**: Human champions refuse to play against computers because software is too bad.
  - In Go, $b > 300$, so need pattern databases and Monte Carlo search
  - Human champions are now beginning to be challenged by machines.

- **Pacman**: The reigning champion is [your CSE 473 program here]
Next Time

• Rolling the dice
• Expectiminimax search

• To Do: Project #1 (due this Sunday!)

Exercise:
Prune this tree!
No, because $-29 > (-37)$ and other children of min can only lower min's value of $-37$ (because of the min operation)
Another pruning opportunity!

Pruning can eliminate entire subtrees!
-43 > (-46)
No need to look at this subtree!