Adversarial Search

Chapter 6
Section 1 – 4
Outline

• Optimal decisions
• $\alpha-\beta$ pruning
• Imperfect, real-time decisions
Games vs. search problems

• "Unpredictable" opponent → specifying a move for every possible opponent reply
• Time limits → unlikely to find goal, must approximate
Game tree (2-player, deterministic, turns)
Minimax

- Perfect play for deterministic games
- Idea: choose move to position with highest minimax value
  = best achievable payoff against best play
- E.g., 2-ply game:
Minimax algorithm

\begin{verbatim}
function MINIMAX-DECISION(state) returns an action

    v ← 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 returnUTILITY(state)
    v ← -∞
    for a, s in SUCCESSORS(state) do
        v ← MAX(v, MIN-VALUE(s))
    return v

function MIN-VALUE(state) returns a utility value

    if TERMINAL-TEST(state) then return UTILITY(state)
    v ← ∞
    for a, s in SUCCESSORS(state) do
        v ← MIN(v, MAX-VALUE(s))
    return v
\end{verbatim}
Properties of minimax

- **Complete?** Yes (if tree is finite)
- **Optimal?** Yes (against an optimal opponent)
- **Time complexity?** $O(b^m)$
- **Space complexity?** $O(bm)$ (depth-first exploration)

- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  $\rightarrow$ exact solution completely infeasible
α-β pruning example
α-β pruning example

```
MAX

MIN

3 12 8 2

3 12 8 2

≥3

≤2

X X
```
α-β pruning example
$\alpha$-$\beta$ pruning example

![Diagram of $\alpha$-$\beta$ pruning example]
α-β pruning example
Properties of α-β

• Pruning does not affect final result

• Good move ordering improves effectiveness of pruning

• With "perfect ordering," time complexity = $O(b^{m/2})$
  → doubles depth of search

• A simple example of the value of reasoning about which computations are relevant (a form of metareasoning)
Why is it called $\alpha$-$\beta$?

- $\alpha$ is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for $max$
- If $v$ is worse than $\alpha$, $max$ will avoid it → prune that branch
- Define $\beta$ similarly for $min$
The $\alpha$-$\beta$ algorithm

function $\text{Alpha-Beta-Search}(\text{state})$ returns an action
inputs: $\text{state}$, current state in game

$v \leftarrow \text{Max-Value}(\text{state}, -\infty, +\infty)$

return the action in $\text{Successors}(\text{state})$ with value $v$

function $\text{Max-Value}(\text{state}, \alpha, \beta)$ returns a utility value
inputs: $\text{state}$, current state in game

$\alpha$, the value of the best alternative for $\text{Max}$ along the path to $\text{state}$
$\beta$, the value of the best alternative for $\text{Min}$ along the path to $\text{state}$

if $\text{Terminal-Test}(\text{state})$ then return $\text{Utility}(\text{state})$

$v \leftarrow -\infty$

for $a, s$ in $\text{Successors}(\text{state})$ 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

```plaintext
function Min-Value(state, $\alpha$, $\beta$) returns a utility value
    inputs: state, current state in game
            $\alpha$, the value of the best alternative for MAX along the path to state
            $\beta$, the value of the best alternative for MIN along the path to state
    if Terminal-Test(state) then return Utility(state)
    $v \leftarrow +\infty$
    for $a, s$ in Successors(state) do
        $v \leftarrow \min(v, \max-Value(s, \alpha, \beta))$
        if $v \leq \alpha$ then return $v$
        $\beta \leftarrow \min(\beta, v)$
    return $v$
```
Resource limits

Suppose we have 100 secs, explore $10^4$ nodes/sec
\[ \rightarrow 10^6 \text{ nodes per move} \]

Standard approach:

• **cutoff test:**
  e.g., depth limit (perhaps add quiescence search)

• **evaluation function**
  = estimated desirability of position
Evaluation functions

• For chess, typically linear weighted sum of features

\[ Eval(s) = w_1 \, f_1(s) + w_2 \, f_2(s) + \ldots + w_n \, f_n(s) \]

• e.g., \( w_1 = 9 \) with

\[ f_1(s) = (\text{number of white queens}) - (\text{number of black queens}), \text{ etc.} \]
Cutting off search

MinimaxCutoff is identical to MinimaxValue except

1. Terminal? is replaced by Cutoff?
2. Utility is replaced by Eval

Does it work in practice?

\[ b^m = 10^6, \ b=35 \rightarrow m=4 \]

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
Deterministic games in practice

- Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used a precomputed endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 444 billion positions.
- Othello: human champions refuse to compete against computers, who are too good.
- Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.
Summary

- Games are fun to work on!
- They illustrate several important points about AI
- Perfection is unattainable → must approximate
- Good idea to think about what to think about