Adversarial Search

Chapter 6
Section 1 – 4
Outline

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

- "Unpredictable" opponent \(\rightarrow\) specifying a move for every possible opponent reply

- Time limits \(\rightarrow\) 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

function \textsc{Minimax-Decision}(state) returns an action

\[
v \leftarrow \text{Max-Value}(state)
\]

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

function \textsc{Max-Value}(state) returns a utility value

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

\[
v \leftarrow -\infty
\]

for \(a, s\) in \textsc{Successors}(state) do

\[
v \leftarrow \text{Max}(v, \text{Min-Value}(s))
\]

return \(v\)

function \textsc{Min-Value}(state) returns a utility value

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

\[
v \leftarrow \infty
\]

for \(a, s\) in \textsc{Successors}(state) do

\[
v \leftarrow \text{Min}(v, \text{Max-Value}(s))
\]

return \(v\)
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 → exact solution completely infeasible
α-β pruning example
\( \alpha-\beta \) pruning example
α-β pruning example
α-β pruning example
α-β pruning example

```
MAX

MIN

3
12
8
2

3
≤2

14
5
2
```
Properties of $\alpha$-$\beta$

• Pruning **does not** affect final result

• Good move ordering improves effectiveness of pruning

• With "perfect ordering," time complexity $= O(b^{m/2})$
  $\rightarrow$ 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 $\text{max}$

• If $v$ is worse than $\alpha$, $\text{max}$ will avoid it → prune that branch

• Define $\beta$ similarly for $\text{min}$
function **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 **Successors**(state) with value \( v \)

---

function **Max-Value**(state, \( \alpha, \beta \)) returns a utility value

inputs: state, current state in game

\( \alpha \), the value of the best alternative for \( \text{MAX} \) along the path to state

\( \beta \), the value of the best alternative for \( \text{MIN} \) along the path to state

if **Terminal-Test**(state) then return **Utility**(state)

\( v \leftarrow -\infty \)

for \( a, s \) in **Successors**(state) do

\( v \leftarrow \text{Max}(v, \text{Min-Value}(s, \alpha, \beta)) \)

if \( v \geq \beta \) then return \( v \)

\( \alpha \leftarrow \text{Max}(\alpha, v) \)

return \( v \)
The \(\alpha-\beta\) algorithm

function \texttt{Min-Value}(state, \alpha, \beta) \textbf{returns} a utility value

inputs: state, current state in game
\hspace{1cm} \alpha, the value of the best alternative for \texttt{Max} along the path to state
\hspace{1cm} \beta, the value of the best alternative for \texttt{Min} along the path to state

if \texttt{Terminal-Test}(state) then return \texttt{Utility}(state)
\hspace{1cm} v \leftarrow +\infty

for \ a, \ s \ in \texttt{Successors}(state) \ do
\hspace{1cm} v \leftarrow \texttt{Min}(v, \texttt{Max-Value}(s, \alpha, \beta))
\hspace{1cm} \textbf{if} \ v \leq \alpha \ \textbf{then} \ \textbf{return} \ v
\hspace{1cm} \beta \leftarrow \texttt{Min}(\beta, v)

return \ v
Resource limits

Suppose we have 100 secs, explore $10^4$ nodes/sec
→ $10^6$ 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
  \[ \text{Eval}(s) = \sum_{i=1}^{n} w_i f_i(s) \]

• e.g., \( w_1 = 9 \) with
  \( f_1(s) = \text{(number of white queens)} - \text{(number of black queens)} \), 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.

• Chess: Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

• 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