Artificial Intelligence

Lesson 6
(From Russell & Norvig)

Games- 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 Diagram]

Properties of minimax

- **Complete? (=will not run forever)** Yes (if tree is finite)
- **Optimal? (=will find the optimal response)** 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

Minimax algorithm

```
function Minimax-Decision(state) returns an action
  ε ← Max-Value(state)
  return the action in Successors(state) with value ε

function Max-Value(state) returns a utility value
  if Terminal-Test(state) then return Utility(state)
  ε ← −∞
  for $a_i$ in Successors(state) do
    $ε ← Max(ε, Min-Value(a))$
  return $ε$

function Min-Value(state) returns a utility value
  if Terminal-Test(state) then return Utility(state)
  $ε ← ∞$
  for $a_i$ in Successors(state) do
    $ε ← Min(ε, Max-Value(a))$
  return $ε$
```

$α$-$β$ pruning example

```
MAX
  ▲  > 3
min
  ▼  3
  ▼  3
```

For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
$\rightarrow$ exact solution completely infeasible
Properties of $\alpha$-$\beta$

- Pruning does not affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering" on binary tree, 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
- $\beta$ is the value of the best alternative for min along the path to state
- If $v$ is worse than $\alpha$, max will avoid it → prune that branch
- Define $\beta$ similarly for min

The $\alpha$-$\beta$ algorithm

```
function Alpha-Beta-Search(state) returns an action
inputs: state, current state in game
    v ← MAX-VALUE(state, −∞, +∞)
    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 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 ← −∞
    for $a,s$ in SUCCESSORS(state) do
        $v ← MAX(v, MIN-VALUE(s, $\alpha$, $\beta$))$
        if $v ≥ \beta$ then return $v$
        $\alpha ← MAX(\alpha, v)$
    return $v$
```

The $\alpha$-$\beta$ algorithm

```
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)
    $v ← +∞$
    for $a,s$ in SUCCESSORS(state) do
        $v ← MIN(v, MAX-VALUE(s, $\alpha$, $\beta$))$
        if $v ≤ \alpha$ then return $v$
        $\beta ← MIN(\beta, v)$
    return $v$
```
Resource limits

Suppose we have 100 secs, explore $10^4$ nodes/sec $\rightarrow 10^6$ nodes per move

Standard approach:
- cutoff test:
  e.g., depth limit (perhaps add quiescence search)
- evaluation function
  $= \text{estimated desirability of position}$

Evaluation functions

- For chess, typically linear weighted sum of features
  $$\text{Eval}(s) = w_1 f_1(s) + w_2 f_2(s) + \ldots + w_n f_n(s)$$
- e.g., $w_1 = 9$ with
  $f_1(s) = \text{(number of white queens)} - \text{(number of black queens)}$, etc.

Cutting off search

$\text{MinimaxCutoff}$ is identical to $\text{MinimaxValue}$ except
1. $\text{Terminal?}$ is replaced by $\text{Cutoff?}$
2. $\text{Utility}$ is replaced by $\text{Eval}$

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