Search

Dr. Melanie Martin
CS 4480
September 27, 2012

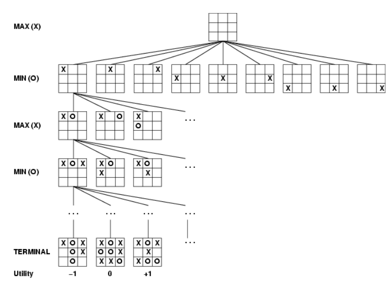
Games vs. search problems

- "Unpredictable" opponent → specifying a move for every possible opponent reply
- Time limits → unlikely to find goal, must approximate

Game as Search Problem

- Initial State
  - Board position and player to move
- Successor Function
  - Returns list of *(state, move)* pairs
    - Legal moves and resulting states
- Terminal (Goal) Test
  - When game is over
- Utility (Objective) Function
  - Assigns numeric outcome to terminal states
    - E.g. +1, -1, 0 for win, lose, draw

Game tree (2-player, deterministic, turns)

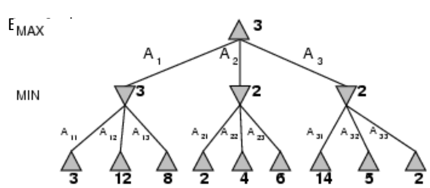


Optimal Strategy

- Leads to outcomes as good as any other strategy when playing an infallible opponent
- Tree where max takes a turn and min takes a turn is ONE MOVE DEEP made up two half-moves - each half move is called a ply

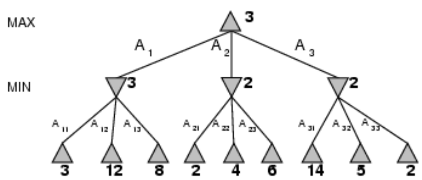
Minimax

- Perfect play for deterministic games
- Idea: choose move to position with highest minimax value = best achievable payoff against best play
- E



Minimax

- Minimax(node) is utility for max of being in corresponding state
- Max prefers a state with maximum value
- Min prefers a state with minimum value



Minimax algorithm

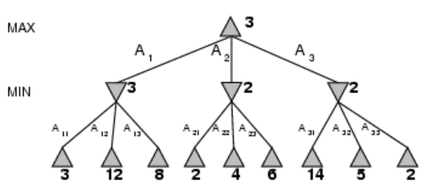
```
function MINIMAX-DECISION(state) returns an action
   v ← MAX-VALUE(state)
   return the action in SUCCESSORS(state) with value v

function MAX-VALUE(state) returns a utility value
   if TERMINAL-TEST(state) then return UTILITY(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
```

Minimax

- Recursion proceeds to leaves and based on utility function assigns minimax values at the level above and so on



Properties of minimax

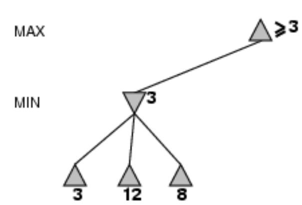
- Complete? Yes (if tree is finite)
- Optimal? Yes (against an optimal opponent)
- Time complexity? $O(b^m)$
- Space complexity? O(bm) (depth-first exploration)

- For chess, b ≈ 35, m ≈100 for "reasonable" games → exact solution completely infeasible

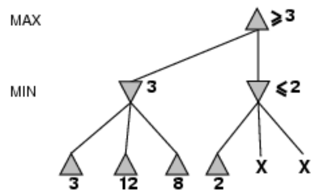
Problem

- Number of game states exponential in number of moves
- Can cut in half with pruning
  - Still exponential
  - Get rid of braches that can't influence final decision

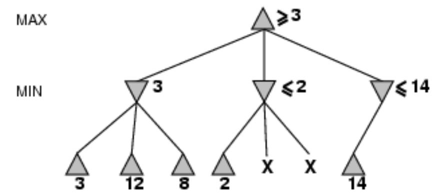
α-β pruning example



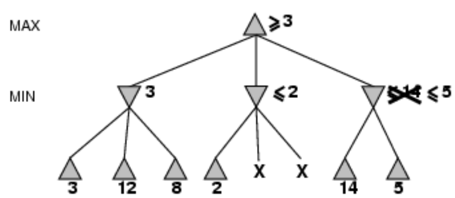
α-β pruning example



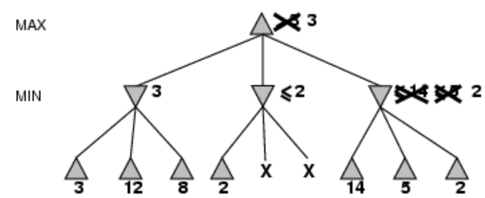
α-β pruning example



α-β pruning example



α-β pruning example

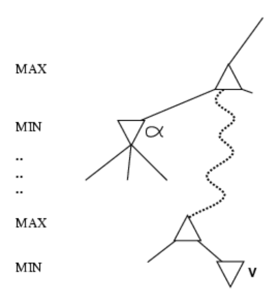


Properties of α-β

- 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 α-β?

- α 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 α, *max* will avoid it
  → prune that branch
- Define β similarly for *min*



The α-β 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, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for MAX along the path to state
           β, 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, α, β))
      if v ≥ β then return v
      α ← MAX(α, v)
   return v
```

The α-β algorithm

```
function MIN-VALUE(state, α, β) returns a utility value
   inputs: state, current state in game
           α, the value of the best alternative for MAX along the path to state
           β, 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, α, β))
      if v ≤ α then return v
      β ← MIN(β, 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
- evaluation function
  = estimated desirability of position

  Usually a heuristic function

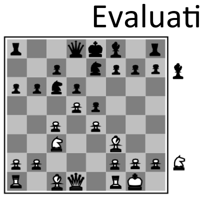
Resource limits

Standard approach:

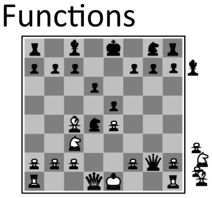
- Use CUTOFF-TEST instead of TERMINAL-TEST
  e.g., depth limit (perhaps add quiescence search)
- Use EVAL instead of UTILITY
  i.e., evaluation function that estimates desirability of position

Suppose we have 100 seconds, explore 10^4 nodes/second
⇒ 10^6 nodes per move ≈ $35^{8/2}$
⇒ α–β reaches depth 8 ⇒ pretty good chess program

Evaluation Functions



Black to move
White slightly better



White to move
Black winning

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

$f_1(s)$ = (number of white queens) – (number of black queens), etc.

Deterministic games in practice

- **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.
- **Othello:** human champions refuse to compete against computers, who are too good.

Deterministic games in practice

- **Chess:** Deep Blue defeated human world champion Gary 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.
- **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.

Getting Better

- http://en.wikipedia.org/wiki/Computer_Go
- http://en.wikipedia.org/wiki/English_draughts
- http://en.wikipedia.org/wiki/Computer_chess
- http://en.wikipedia.org/wiki/Computer_Othello