

Game Playing

Preliminaries	2
Search in Game Playing	2
Useful Definitions	3
Minimax	4
MAX-VALUE Procedure	4
MIN-VALUE Procedure	5
Example	6
Alpha-Beta Pruning	7
Idea of Alpha-Beta Pruning	7
α - β MAX-VALUE	8
α - β MIN-VALUE	9
Example	10
Issues	11
Performance of Minimax and Alpha-Beta	11
Other Issues	12

Preliminaries

Search in Game Playing

- In game playing, choosing an action must take the opponent into account.
- A search problem for a game is defined by:
 - s_{current} . Current state/position of the game.
 - $\text{PLAYER}(s)$. Whose turn is it in a given state?
 - $\text{EXPAND}(s)$. A set of states from possible moves.
 - $\text{GAME-OVER}(s)$. Is the game over in a given state?
 - $\text{WHO-WON}(s, p)$. If s is the end of the game, who won?

Useful Definitions

- **max**. The player whose turn it is to move.
- **min**. The other player.
- *Ply*. A synonym for “depth.”
- *Evaluation function*. Estimates **max**'s utility. The state space of many games is too large to search. An alternative is to search as deeply as possible, estimate the values of the fringe states, and combine the values into an overall evaluation.

Minimax

4

MAX-VALUE Procedure

max should maximize evaluation function assuming that **min** minimizes it. Pseudocode assumes it is **max**'s turn to move, and that turns are interleaved.

```

function MAX-VALUE(s, bound)
  if GAME-OVER(s) or bound = 0
    then return EVALUATION(s)
  max ← -∞
  for each ssuccessor in EXPAND(s)
    do eval ← MIN-VALUE(ssuccessor, bound-1)
       if eval > max then max ← eval
  return max
    
```

CS 3793 Artificial Intelligence

Game Playing - 4

MIN-VALUE Procedure

Assume **min** minimizes **max**'s evaluation function.

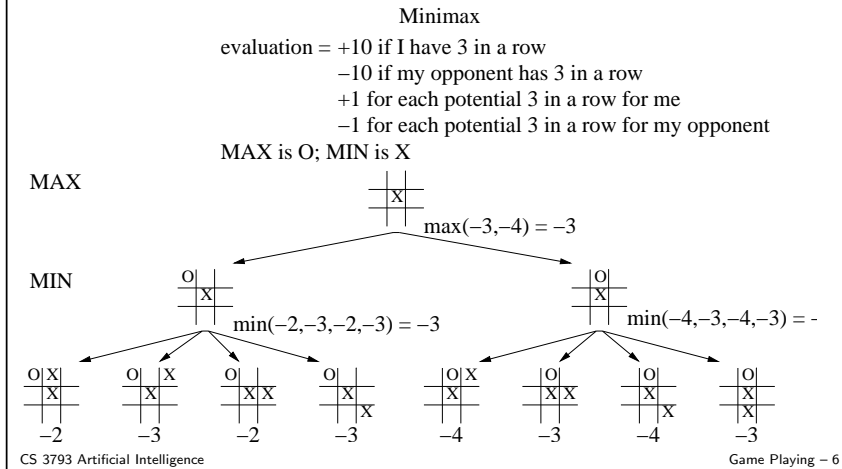
```

function MIN-VALUE(s, bound)
  if GAME-OVER(s) or bound = 0
    then return EVALUATION(s)
  min ← ∞
  for each ssuccessor in EXPAND(s)
    do eval ← MAX-VALUE(ssuccessor, bound-1)
       if eval < min then min ← eval
  return min
    
```

CS 3793 Artificial Intelligence

Game Playing - 5

Example



Alpha-Beta Pruning

7

Idea of Alpha-Beta Pruning

- Alpha-beta pruning avoids search that won't change the minimax evaluation.
- Example: If **max** has a move with value 3, stop searching other moves known to be ≤ 3 .
- General Principle: Consider a state *s*.
 α = largest *max* in ancestors of *s*.
 β = smallest *min* in ancestors of *s*.
 If $\alpha \geq \beta$, processing *s* cannot change eval.
- Proof: Let *v* = *s*'s minimax value.
 $\alpha \geq \beta$ implies $\alpha \geq v$ or $v \geq \beta$.
 $\alpha \geq v$ implies *v* can't change ancestor with α .
 $v \geq \beta$ implies *v* can't change ancestor with β .
 This implies *v* cannot propagate to the top.

CS 3793 Artificial Intelligence

Game Playing - 7

α - β MAX-VALUE Procedure

Parameters:

α is a known *max* value in an ancestor.

β is a known *min* value in an ancestor.

```

function MAX-VALUE(s, bound,  $\alpha$ ,  $\beta$ )
  if GAME-OVER(s) or bound = 0
    then return EVALUATION(s)
  for each  $s_{\text{successor}}$  in EXPAND(s)
    do eval  $\leftarrow$  MIN-VALUE( $s_{\text{successor}}$ , bound-1,  $\alpha$ ,  $\beta$ )
    if eval >  $\alpha$  then  $\alpha \leftarrow$  eval
    if  $\alpha \geq \beta$  then return  $\alpha$ 
  return  $\alpha$ 
    
```

CS 3793 Artificial Intelligence

Game Playing - 8

α - β MIN-VALUE Procedure

Parameters:

α is a known *max* value in an ancestor.

β is a known *min* value in an ancestor.

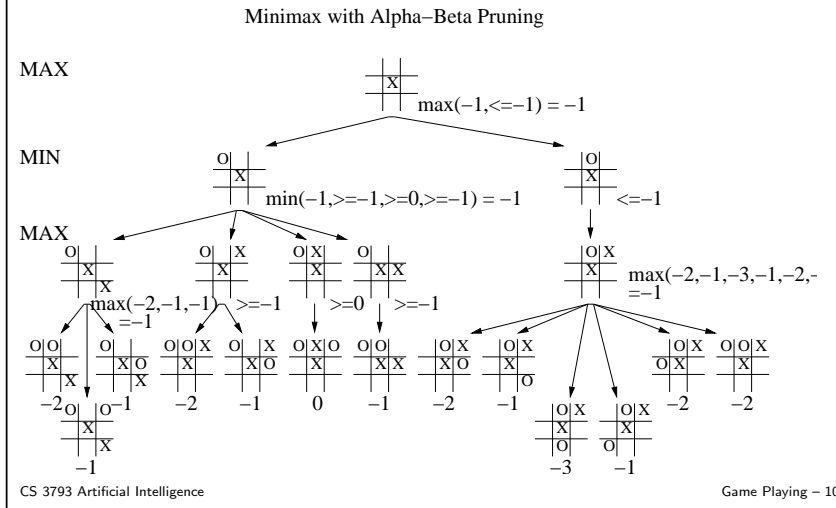
```

function MIN-VALUE(state, bound,  $\alpha$ ,  $\beta$ )
  if TERMINAL(state) or bound = 0
    then return EVALUATION(state)
  for each  $s_{\text{successor}}$  in EXPAND(s)
    do eval  $\leftarrow$  MAX-VALUE( $s_{\text{successor}}$ , bound-1,  $\alpha$ ,  $\beta$ )
    if eval <  $\beta$  then  $\beta \leftarrow$  eval
    if  $\beta \leq \alpha$  then return  $\beta$ 
  return  $\beta$ 
    
```

CS 3793 Artificial Intelligence

Game Playing - 9

Example



Issues

11

Performance of Minimax and Alpha-Beta

- b = branching factor
- d = depth of search
- Minimax visits every state from level 0 to d .

$$\sum_{i=0}^d b^i = \frac{b^{d+1} - 1}{b - 1} \in O(b^d)$$

- Alpha-Beta visits as few as $\Omega(b^{d/2})$ states.
Depends on a good ordering from EXPAND.
Actual programs approach the minimum bound.
- Alpha-beta pruning allows programs to look ahead nearly twice as many moves as minimax.

CS 3793 Artificial Intelligence

Game Playing - 11

Other Issues

- Horizon problem
- Quiescence
- Data bases of openings and end games
- Games of chance