

Game Playing

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

Preliminaries

Search in Game Playing

- In game playing, the choice of action must take the opponent into account.
- A search problem for a game is defined by:
 - *Initial state.* Current position and whose turn.
 - *Operators.* The legal moves.
 - *Terminal test.* Is the game over in a given state?
 - *Utility function.* Is a terminal state win, lose, or draw?

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.

Minimax

MAX Procedure

MAX should maximize evaluation function assuming that MIN minimizes it.

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

MIN Procedure

Assume MIN minimizes MAX's evaluation function.

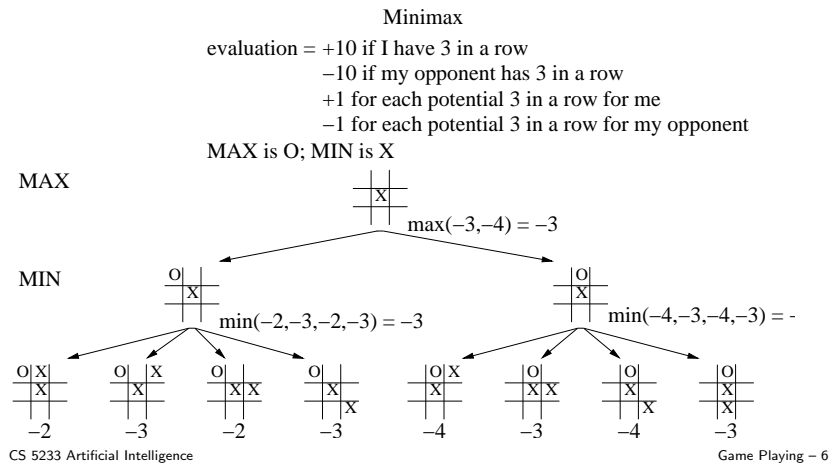
```

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

CS 5233 Artificial Intelligence

Game Playing - 5

Example



Alpha-Beta Pruning

Idea of Alpha-Beta Pruning

- The idea of alpha-beta pruning is to avoid search that won't change the minimax evaluation.
- Example: If MAX has a move which evaluates to 3, then stop searching other moves when their values are known to be ≤ 3 .
- General Example:
 Consider a minimizing node n .
 Let α be the maximum-so-far in an ancestor.
 Let β be the minimum-so-far of n .
 If $\alpha \geq \beta$, value of n cannot change α .

CS 5233 Artificial Intelligence

Game Playing - 7

MAX Procedure

α is a known max-so-far value in an ancestor.
 β is a known min-so-far value in an ancestor.

```

function MAX-VALUE(state, bound,  $\alpha$ ,  $\beta$ )
  if TERMINAL(state) or bound = 0
    then return EVALUATION(state)
  for each next in EXPAND(state)
    do eval ← MIN-VALUE(next, bound - 1,  $\alpha$ ,  $\beta$ )
      if eval >  $\alpha$  then  $\alpha$  ← eval
      if  $\alpha \geq \beta$  then return  $\alpha$ 
  return  $\alpha$ 
  
```

CS 5233 Artificial Intelligence

Game Playing - 8

MIN Procedure

α is a known max-so-far value in an ancestor.
 β is a known min-so-far value in an ancestor.

```

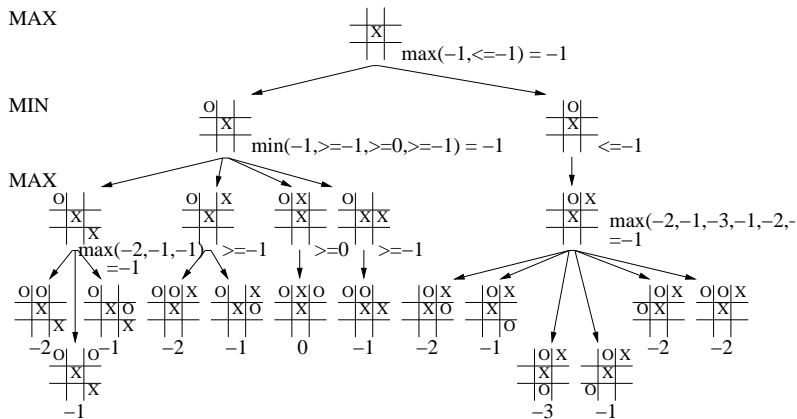
function MIN-VALUE(state, bound,  $\alpha$ ,  $\beta$ )
  if TERMINAL(state) or bound = 0
    then return EVALUATION(state)
  for each next in EXPAND(state)
    do eval  $\leftarrow$  MAX-VALUE(next, bound - 1,  $\alpha$ ,  $\beta$ )
      if eval <  $\beta$  then  $\beta \leftarrow$  eval
      if 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

CS 5233 Artificial Intelligence

Game Playing - 9

Example

Minimax with Alpha-Beta Pruning



CS 5233 Artificial Intelligence

Game Playing - 10

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 5233 Artificial Intelligence

Game Playing - 11

Other Issues

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

CS 5233 Artificial Intelligence

Game Playing - 12