CMSC 372
Artificial Intelligence

Fall 2017

Game Playing

• 2-person, zero-sum games
• Tic-Tac-Toe, checkers, chess, etc.
• Base algorithm: Minimax
• Improvised with $\alpha$-$\beta$ Pruning
Minimax Game Tree

The Minimax Algorithm

**Function** minimax(n) returns (pbv, move)

  if n at depth bound
    return (e(n), move(n))
  expand n to \( n_1, n_2, \ldots, n_b \) successors

  if n is a MAX node:
    cbv = -\( \infty \), bestMove = \( \emptyset \)
    for each \( n_i \text{ in } n_1, n_2, \ldots, n_b \)
      bv, move = minimax(\( n_i \))
      if bv > cbv
        cbv = bv, bestMove = move
    return (cbv, bestMove)

  if n is a MIN node:
    cbv = \( \infty \), bestMove = \( \emptyset \)
    for each \( n_i \text{ in } n_1, n_2, \ldots, n_b \)
      bv, move = minimax(\( n_i \))
      if bv < cbv
        cbv = bv, bestMove = move
    return (cbv, bestMove)
Which Move? What Value?

Minimax with $\alpha$-$\beta$ Pruning
Minimax with $\alpha$-$\beta$ Pruning

- $\alpha$ = value of the best (highest value) choice we have at any choice point along path for MAX
- $\beta$ = value of the best (lowest value) choice we have at any choice point along path for MIN
- $\alpha$- value of a MAX node can never decrease
- $\beta$- value of a MIN node can never increase

$\alpha$-$\beta$ Pruning Rules

- $\alpha$-cutoff: discontinue search below any MIN node whose $\beta \leq \alpha$ of its MAX ancestors.
- $\beta$-cutoff: discontinue search below any MAX node whose $\alpha \geq \beta$ of its MIN ancestors.
Minimax with α-β Pruning

Function minimax-α-β(n, α, β) returns (pbv, move)
   if n at depth bound
       return (e(n), move(n))
   expand n to \( n_1, n_2, ..., n_b \) successors

if n is a MAX node:
   bestMove = Ø
   for each \( n_i \) in \( n_1, n_2, ..., n_b \)
      bv, move = minimax-α-β \( n_i, α, β \)
      if bv > α
         α = bv, bestMove = move
      if α >= β
         return (β, bestMove)
   return (α, bestMove)

if n is a MIN node:
   bestMove = Ø
   for each \( n_i \) in \( n_1, n_2, ..., n_b \)
      bv, move = minimax-α-β \( n_i, α, β \)
      if bv < β
         β = bv, bestMove = move
      if β <= α
         return (α, bestMove)
   return (β, bestMove)
α-β Pruning

Depth limit = 3

(α = -∞, β = ∞)
**α-β Pruning**

$\alpha = -\infty$, $\beta = \infty$

Depth limit = 3

![Diagram of α-β Pruning with depth limit 3](image-url)
\[\alpha - \beta \text{ Pruning} \]

\((\alpha = -\infty, \beta = \infty)\)

\[
\begin{align*}
&3 & 12 & 8 \\
\end{align*}
\]
\( \alpha - \beta \) Pruning

\((\alpha = -\infty, \beta = \infty)\)

\[(\alpha = -\infty, \beta = 3)\]

\[3 \quad 12 \quad 8\]
α-β Pruning

(\(\alpha = -\infty, \beta = \infty\))

(\(\alpha = -\infty, \beta = 3\))

3 12 8

3

α-β Pruning

(\(\alpha = -\infty, \beta = \infty\))

(\(\alpha = 3, \beta = \infty\))

3 12 8

3
\( \alpha - \beta \) Pruning

\( (\alpha = -\infty, \beta = \infty) \)

\( (\alpha = -\infty, \beta = 3) \)

\( (\alpha = 3, \beta = \infty) \)

\( (\alpha = 3, \beta = 2) \)

\( (\alpha = 3, \beta = \infty) \)
\( \alpha - \beta \) Pruning

\[ \begin{aligned}
\left( \alpha = -\infty, \beta = \infty \right) \\
\left( \alpha = -\infty, \beta = 3 \right) \\
\left( \alpha = 3, \beta = \infty \right) \\
\left( \alpha = 3, \beta = 2 \right)
\end{aligned} \]

\( \alpha - \) cutoff
\(\alpha-\beta\) Pruning

(\(\alpha = -\infty, \beta = \infty\))

(\(\alpha = -\infty, \beta = 3\))

(\(\alpha = 3, \beta = \infty\))

(\(\alpha = 3, \beta = 3\))

(\(\alpha = 3, \beta = \infty\))

\(\alpha\)-cutoff
\( \alpha - \beta \) Pruning

\[
\begin{align*}
\alpha &= -\infty, \quad \beta = 3 \\
\alpha &= 3, \quad \beta = \infty
\end{align*}
\]

\( \alpha \)-cutoff
Minimax with α-β Pruning

**Function** minimax-α-β(n, α, β) returns (pbv, move)

if n at depth bound
   return (e(n), move(n))
expand n to \(n_1, n_2, ..., n_b\) successors

if n is a MAX node:
   bestMove = Ø
   for each \(n_i\) in \(n_1, n_2, ..., n_b\)
      bv, move = minimax-α-β \((n_i, \alpha, \beta)\)
      if bv > \(\alpha\)
         \(\alpha = bv\), bestMove=move
      if \(\alpha = \beta\)
         return (\(\beta\), bestMove)
   return (\(\alpha\), bestMove)

if n is a MIN node:
   bestMove = Ø
   for each \(n_i\) in \(n_1, n_2, ..., n_b\)
      bv, move = minimax-α-β \((n_i, \alpha, \beta)\)
      if bv < \(\beta\)
         \(\beta = bv\), bestMove=move
      if \(\beta = \alpha\)
         return (\(\alpha\), bestMove)
   return (\(\beta\), bestMove)
Game Strategies

- Base algorithm: Minimax
- Improved Efficiency: \( \alpha-\beta \) Pruning
- Progressive Deepening
- Move Ordering
- Game specific heuristics

Another Example

\((\alpha = -\infty, \beta = \infty)\)

```
8 7 3 9

9 8 2 4
```

\(\text{Move ordering:}
8 7 3 9
9 8 2 4
\)