Adversarial Search

Dr. Steven Bethard

Computer and Information Sciences
University of Alabama at Birmingham

28 Jan 2016

Outline

Outline

Games as Search

Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Games as Search

Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Games as Search

Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Games as Search

Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Games as Search

Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Games as Search

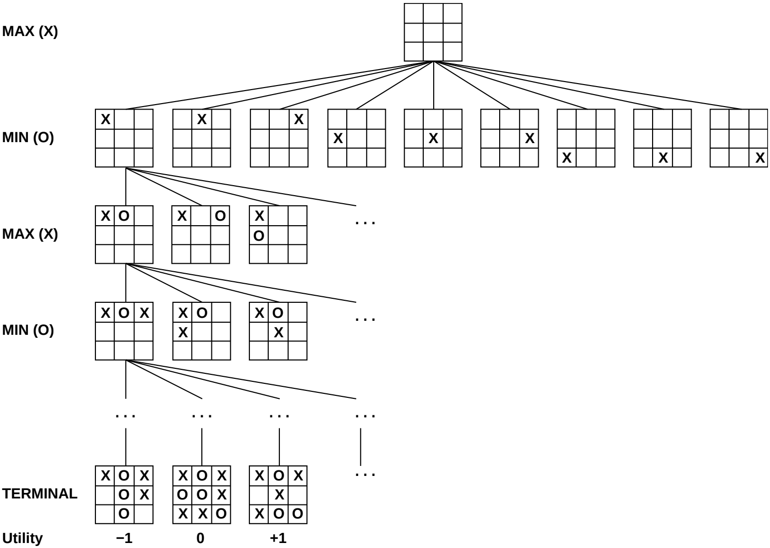
Games as Search

Initial	A board position and which player is to move
Actions	Moves and resulting states
Goal	A terminal state, where the game has ended
Score	Based on utility function, e.g. +1 win, -1 lose

Game Types

	Deterministic	Stochastic
Perfect information	chess, checkers, go, reversi	backgammon, monopoly
Imperfect information	battleship, blind tic-tac-toe	bridge, poker, scrabble

Tic-Tac-Toe Game Tree



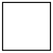
Outline

Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility

Max Player



Min Player

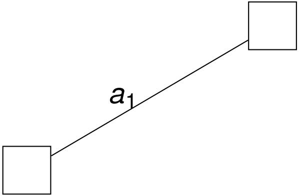
Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility

Max Player

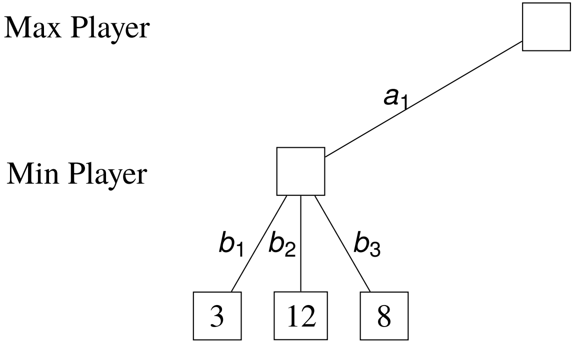
Min Player



Minimax

Idea

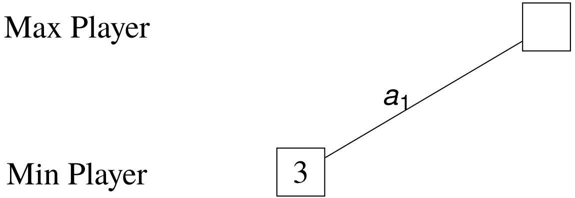
- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

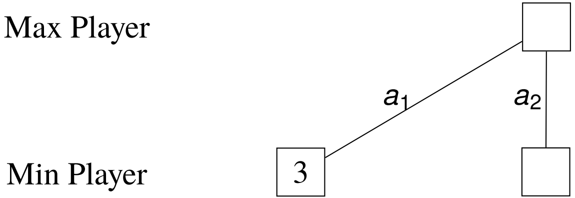
- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



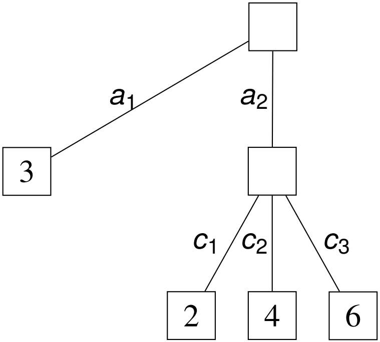
Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility

Max Player

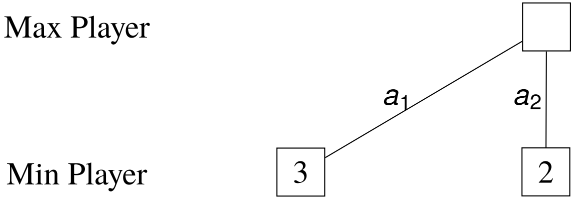
Min Player



Minimax

Idea

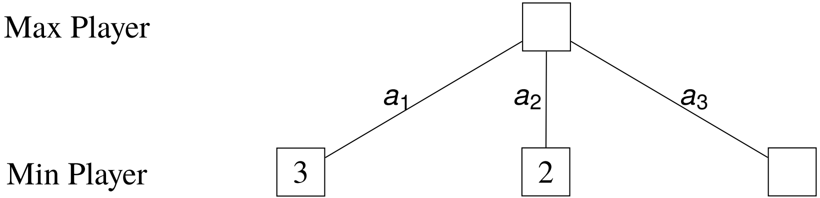
- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

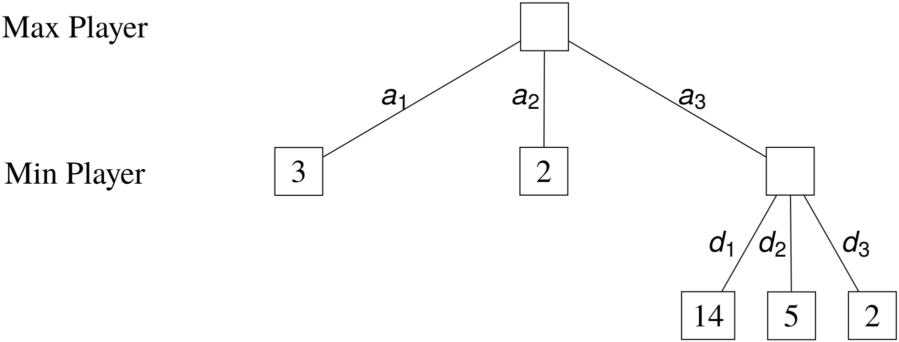
- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

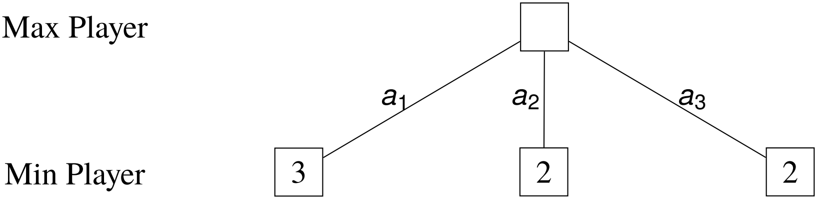
- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility



Minimax

Idea

- Expect other player to minimize your utility
- Choose action that maximizes the minimized utility

Max Player $\boxed{3}$

Min Player

Minimax Exercise

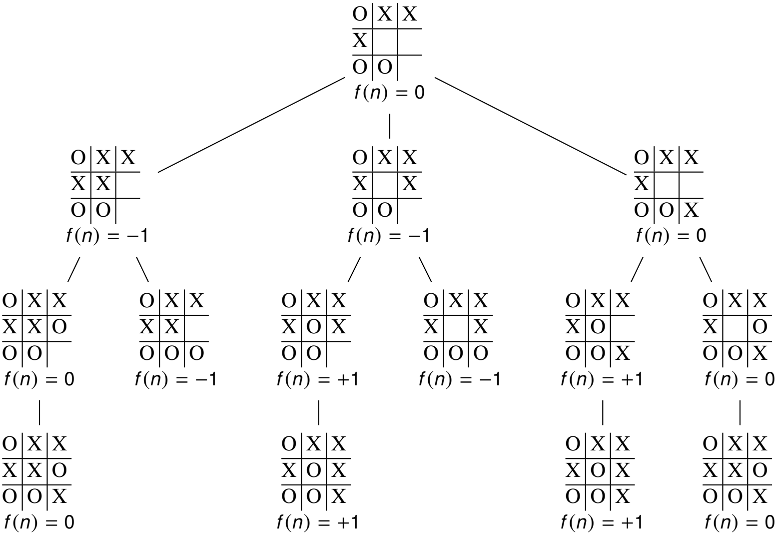
Build Minimax Tree

- It is X's turn
- Scoring:

X Wins	+1
X Loses	-1
X and O Tie	0

O	X	X
X		
O	O	

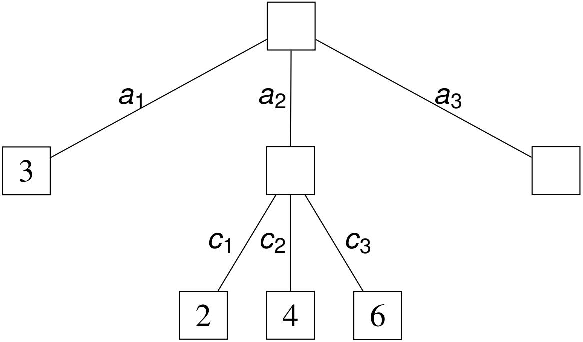
Minimax Exercise



Minimax Properties

Max Player

Min Player



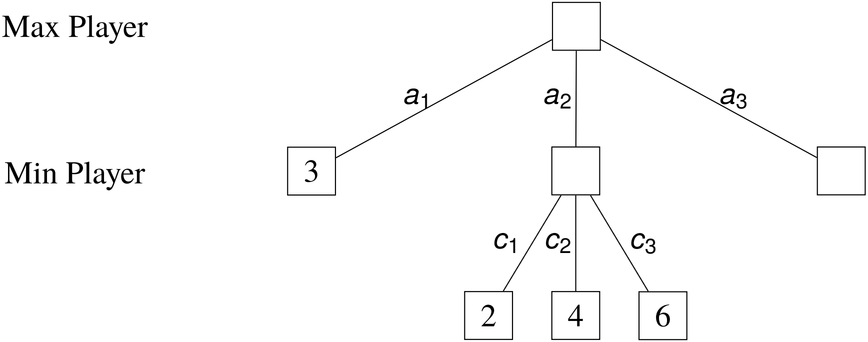
Complete? Yes, if tree is finite

Optimal? Yes, for optimal opponent

Time? $O(b^m)$, all nodes in the tree

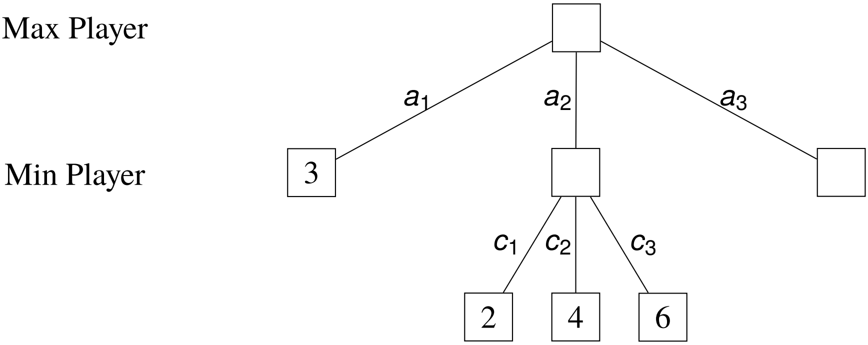
Space? $O(bm)$, like depth first search

Minimax Properties



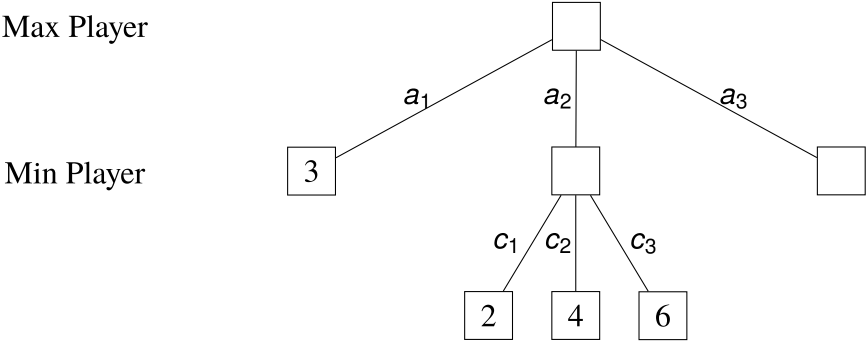
Complete?	Yes, if tree is finite
Optimal?	Yes, for optimal opponent
Time?	$O(b^m)$, all nodes in the tree
Space?	$O(bm)$, like depth first search

Minimax Properties



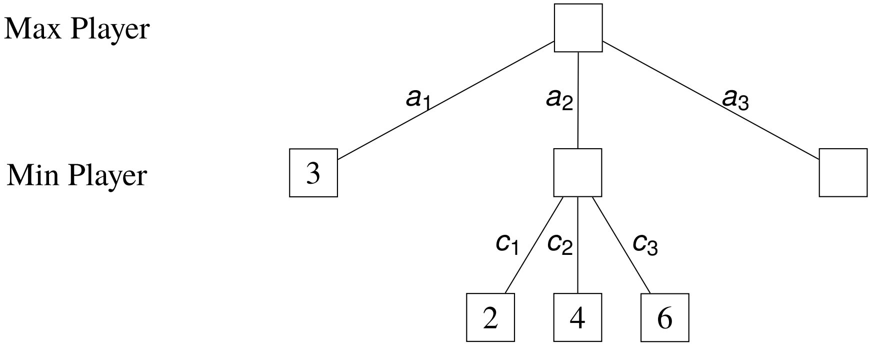
Complete?	Yes, if tree is finite
Optimal?	Yes, for optimal opponent
Time?	$O(b^m)$, all nodes in the tree
Space?	$O(bm)$, like depth first search

Minimax Properties



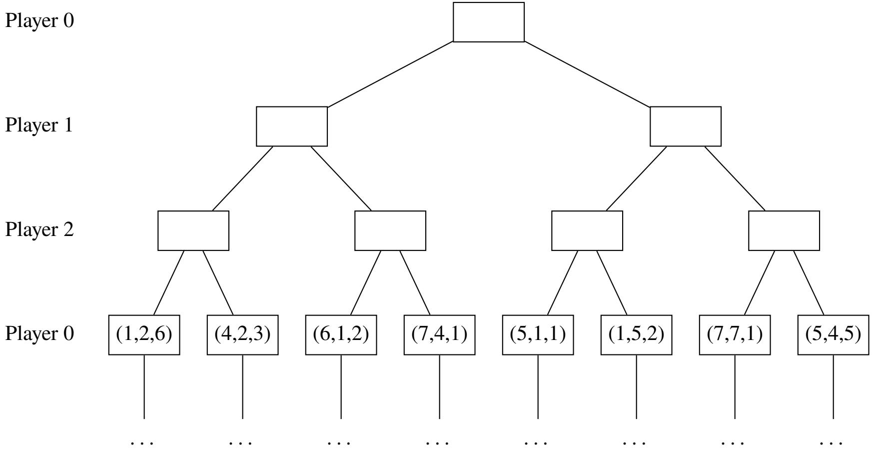
Complete?	Yes, if tree is finite
Optimal?	Yes, for optimal opponent
Time?	$O(b^m)$, all nodes in the tree
Space?	$O(bm)$, like depth first search

Minimax Properties

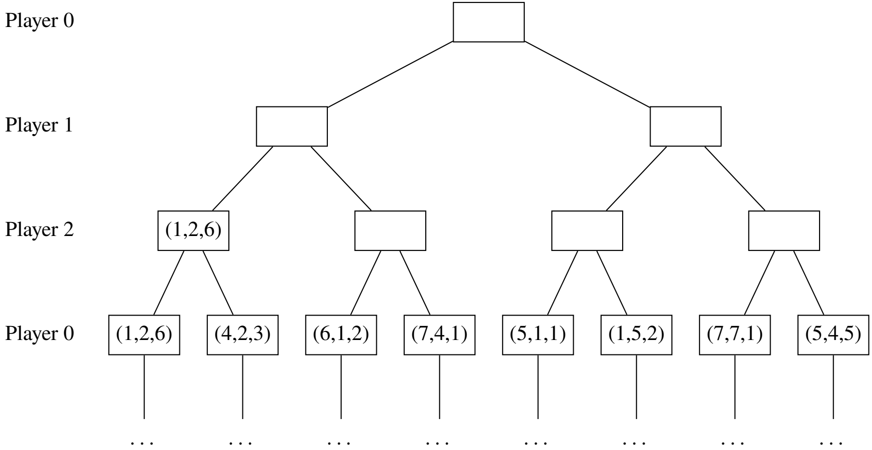


Complete?	Yes, if tree is finite
Optimal?	Yes, for optimal opponent
Time?	$O(b^m)$, all nodes in the tree
Space?	$O(bm)$, like depth first search

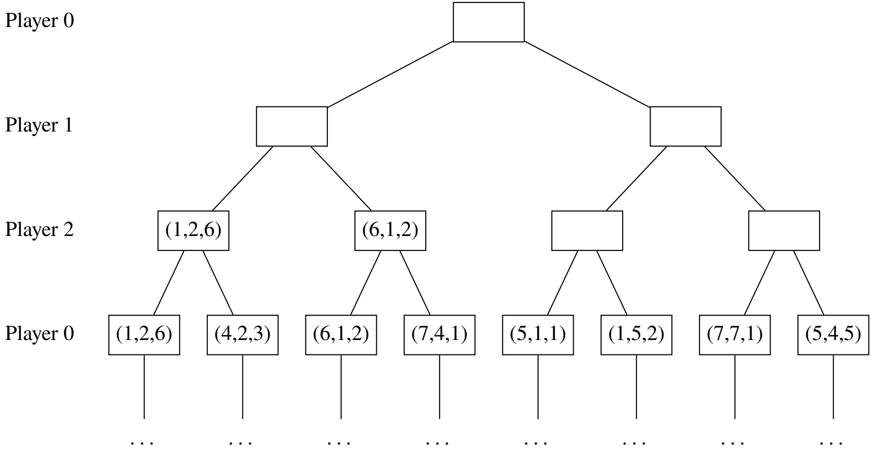
Multiplayer Minimax



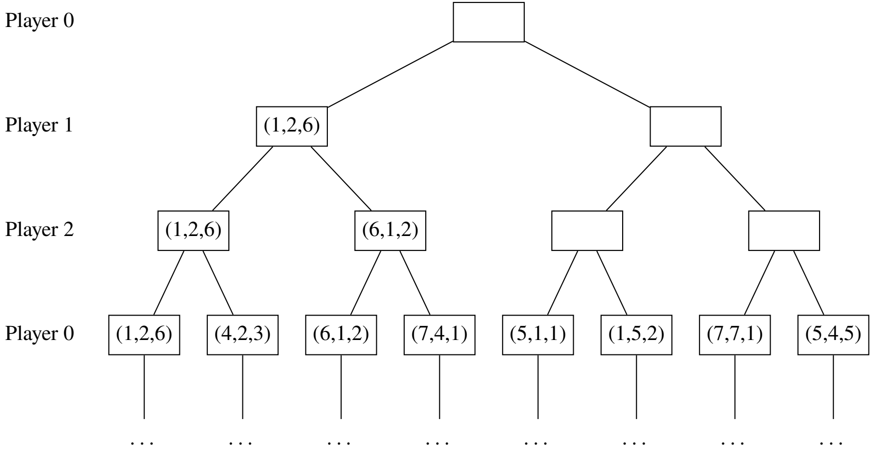
Multiplayer Minimax



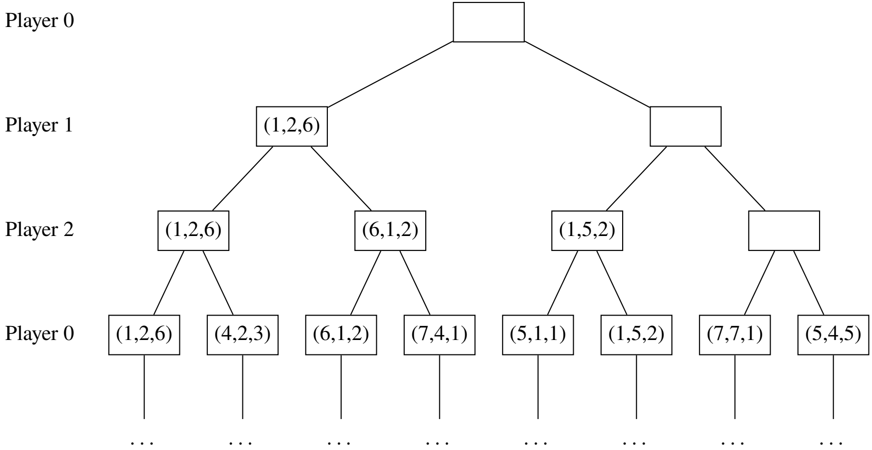
Multiplayer Minimax



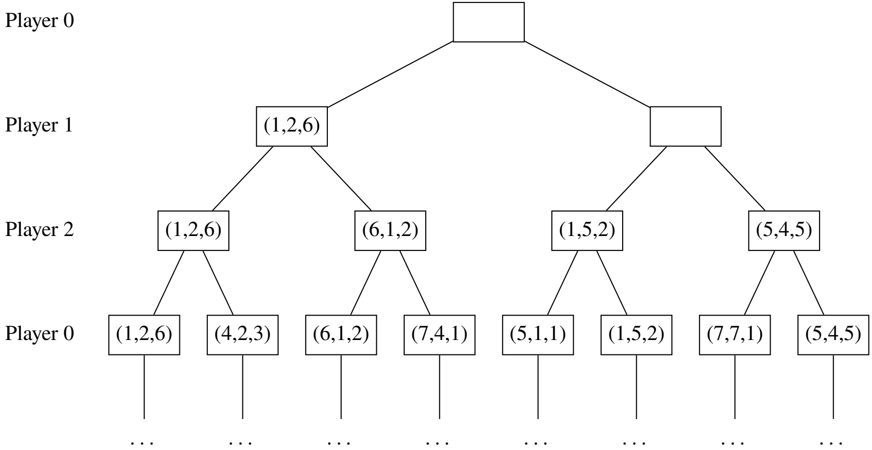
Multiplayer Minimax



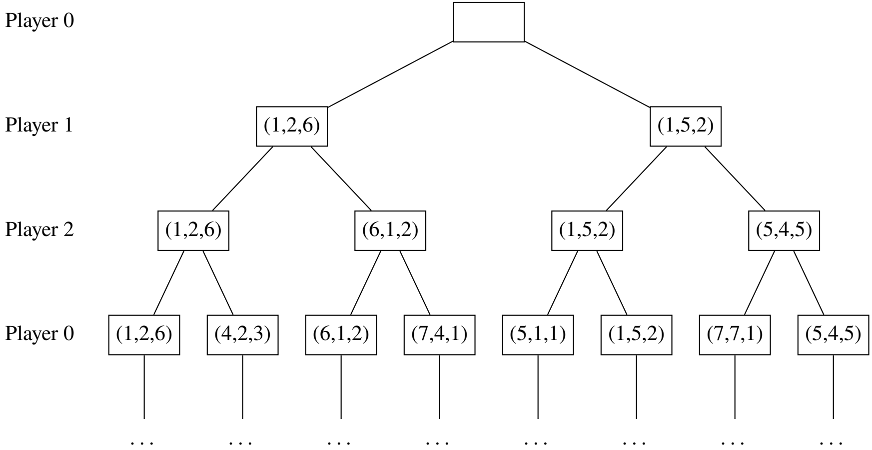
Multiplayer Minimax



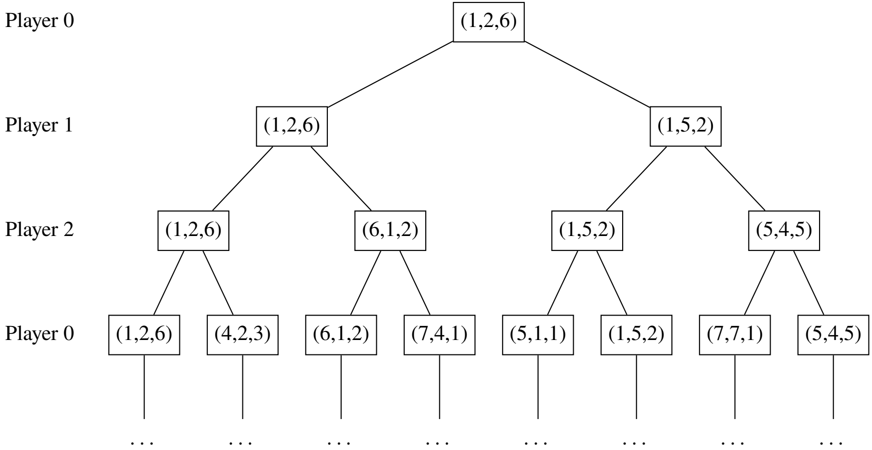
Multiplayer Minimax



Multiplayer Minimax



Multiplayer Minimax



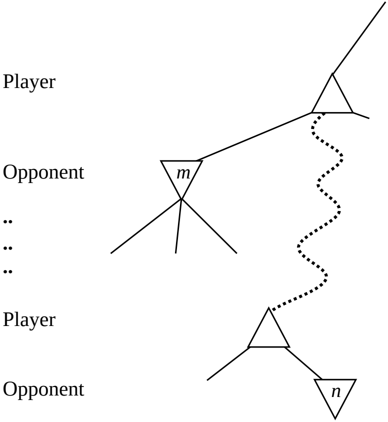
Alpha-Beta Pruning

Idea

If m is a better choice, then n will never be reached

Alpha-Beta Bookkeeping

- α Best value for Player (the highest value)
- β Best value for Opp (the lowest value)



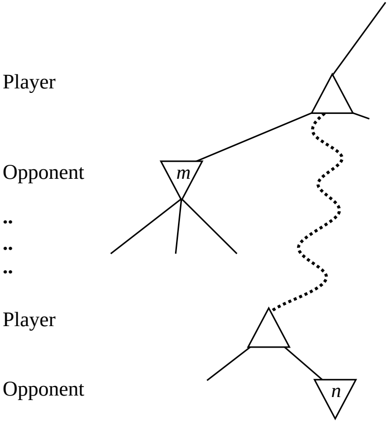
Alpha-Beta Pruning

Idea

If m is a better choice, then n will never be reached

Alpha-Beta Bookkeeping

- α Best value for Player (the highest value)
- β Best value for Opp (the lowest value)



Alpha-Beta Pruning

Max

$\geq -\infty$
$\leq +\infty$

Min

Max

Min

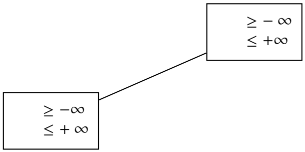
Alpha-Beta Pruning

Max

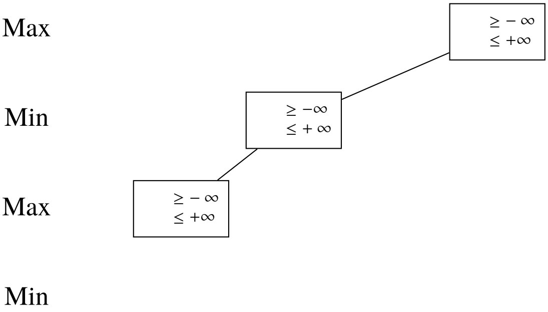
Min

Max

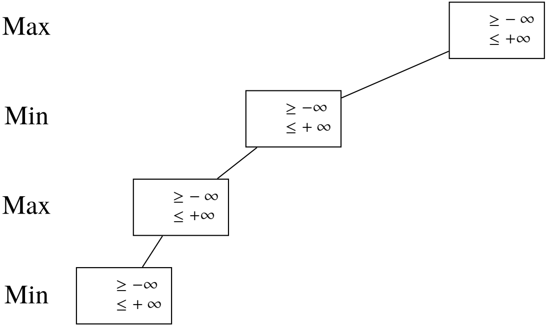
Min



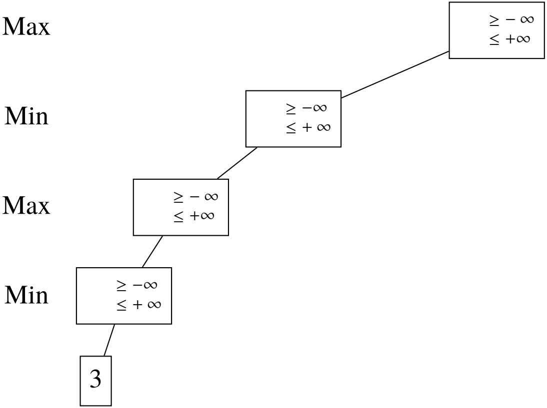
Alpha-Beta Pruning



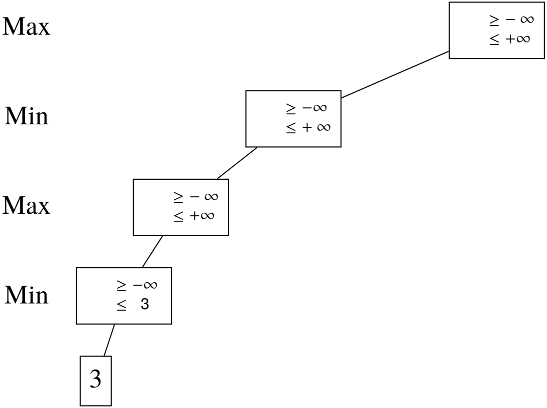
Alpha-Beta Pruning



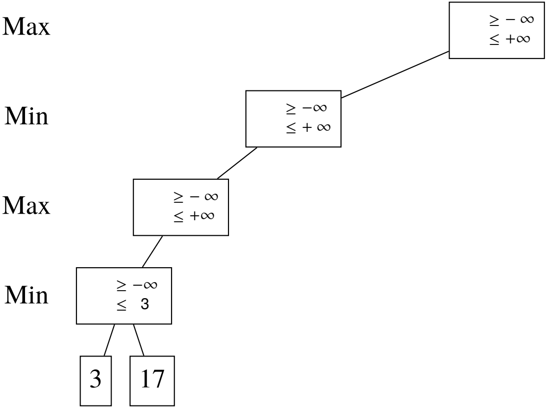
Alpha-Beta Pruning



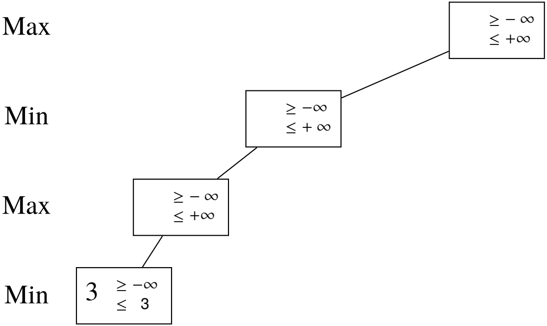
Alpha-Beta Pruning



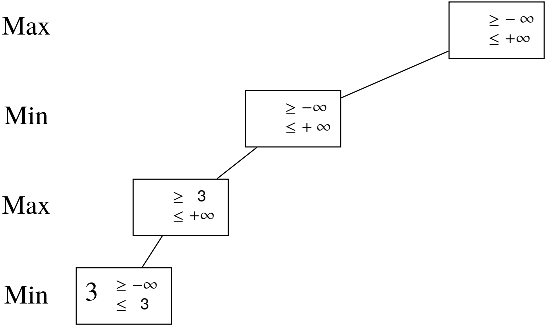
Alpha-Beta Pruning



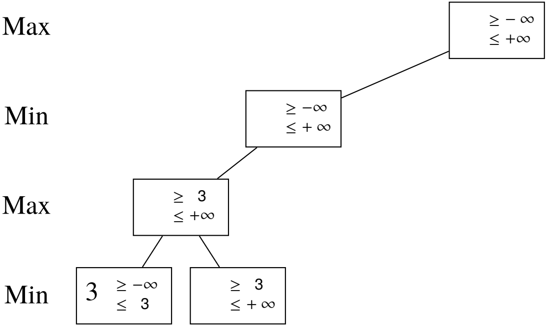
Alpha-Beta Pruning



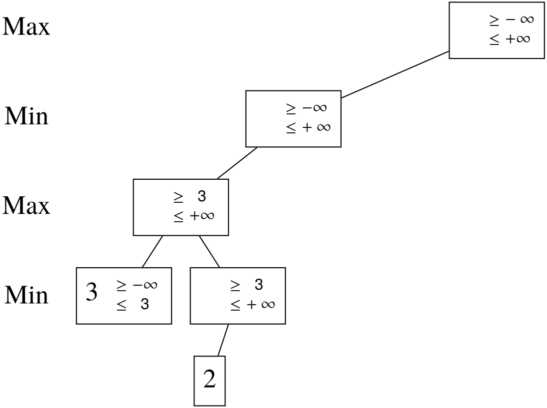
Alpha-Beta Pruning



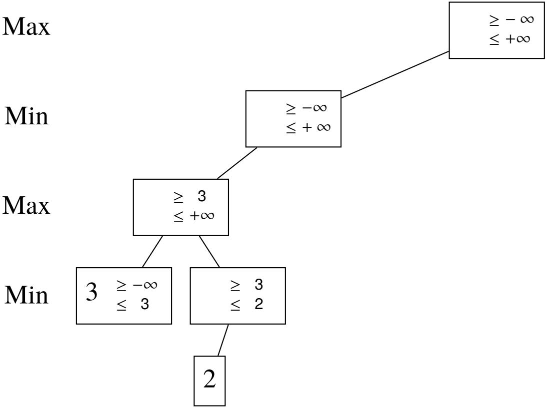
Alpha-Beta Pruning



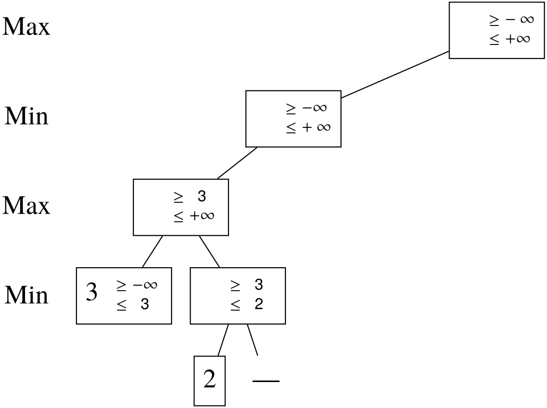
Alpha-Beta Pruning



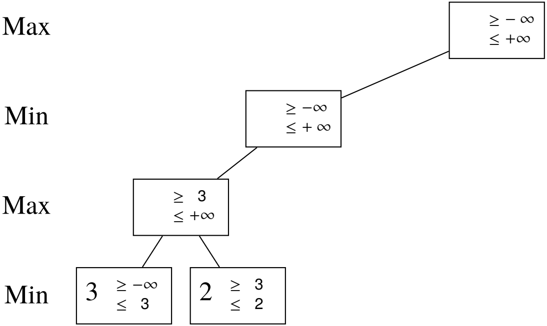
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



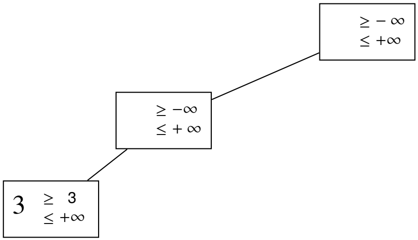
Alpha-Beta Pruning

Max

Min

Max

Min



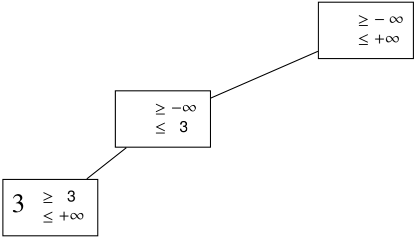
Alpha-Beta Pruning

Max

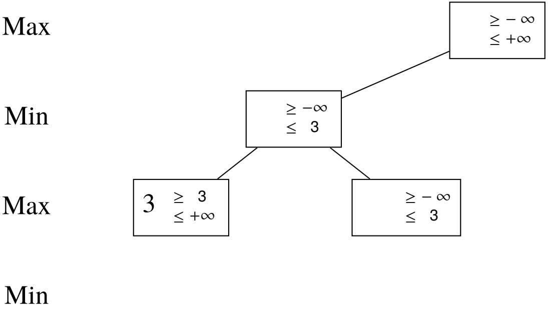
Min

Max

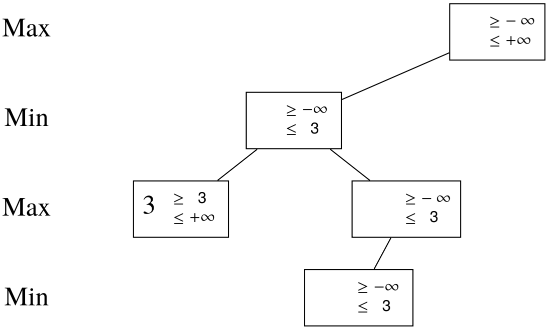
Min



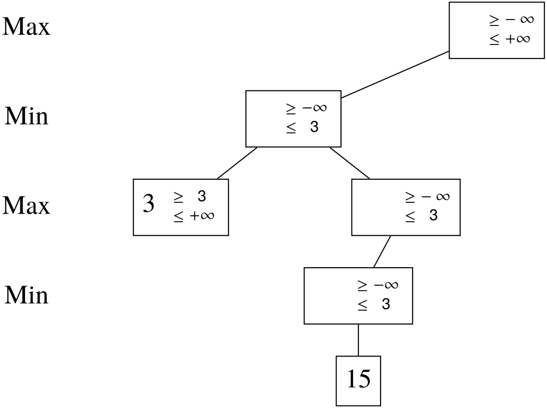
Alpha-Beta Pruning



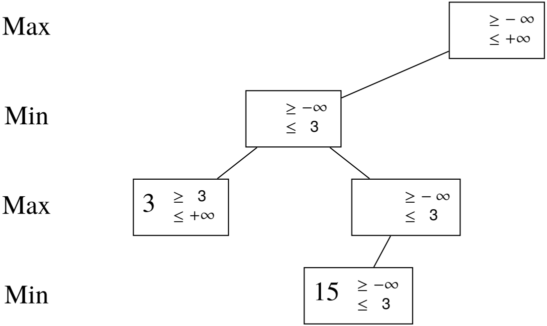
Alpha-Beta Pruning



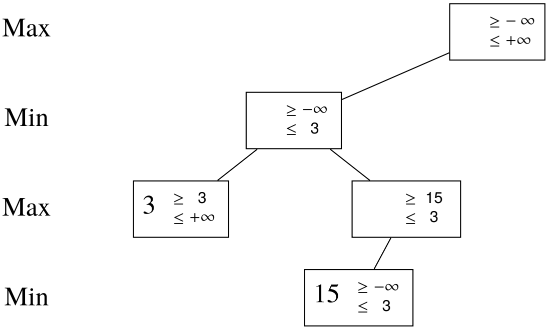
Alpha-Beta Pruning



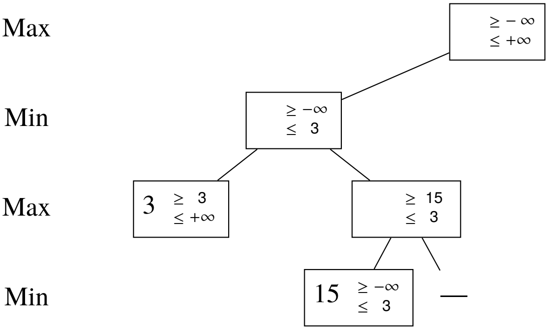
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



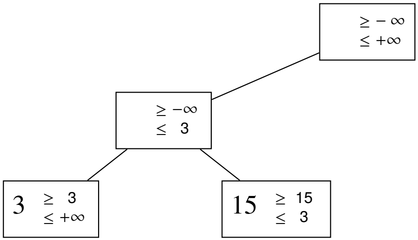
Alpha-Beta Pruning

Max

Min

Max

Min



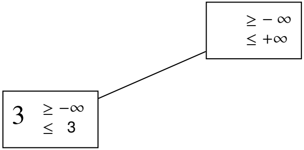
Alpha-Beta Pruning

Max

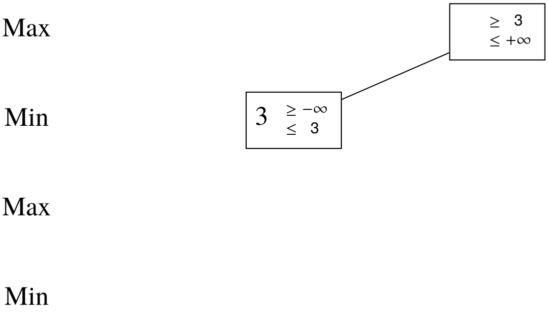
Min

Max

Min



Alpha-Beta Pruning



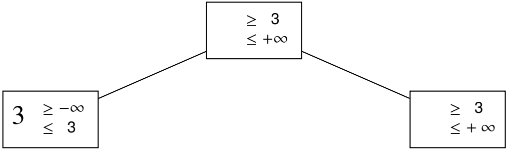
Alpha-Beta Pruning

Max

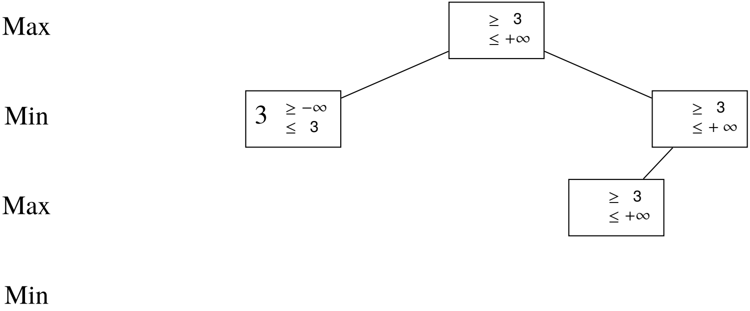
Min

Max

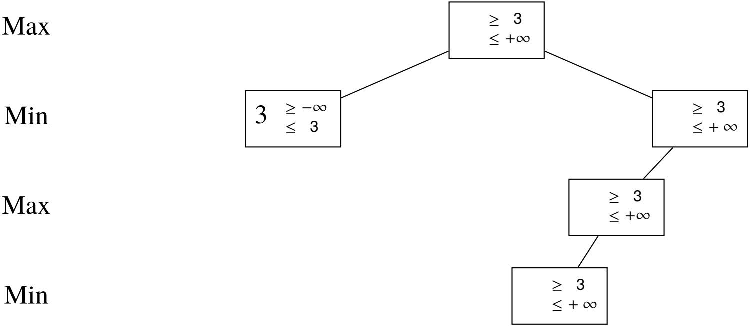
Min



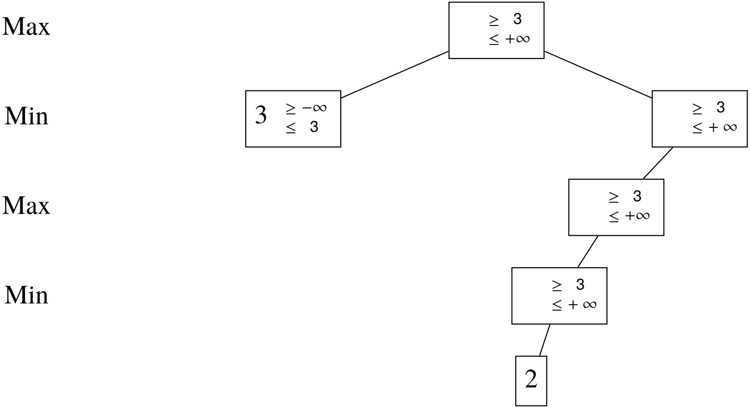
Alpha-Beta Pruning



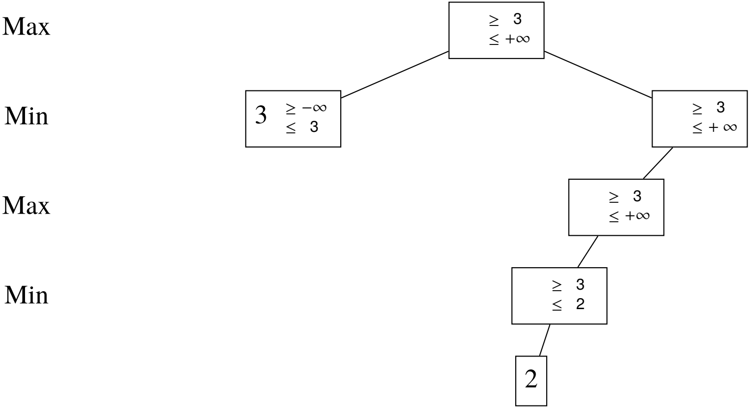
Alpha-Beta Pruning



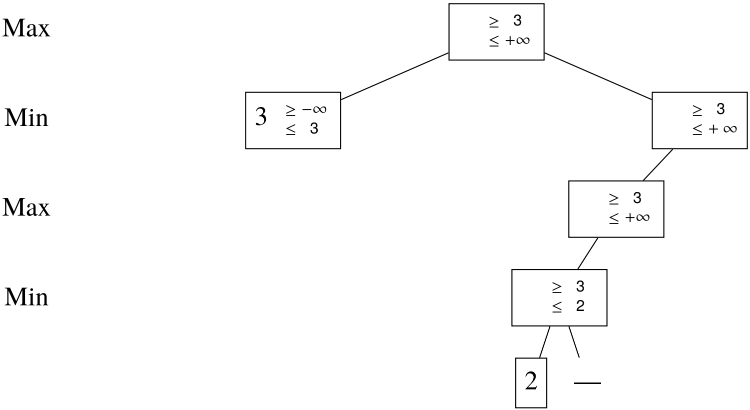
Alpha-Beta Pruning



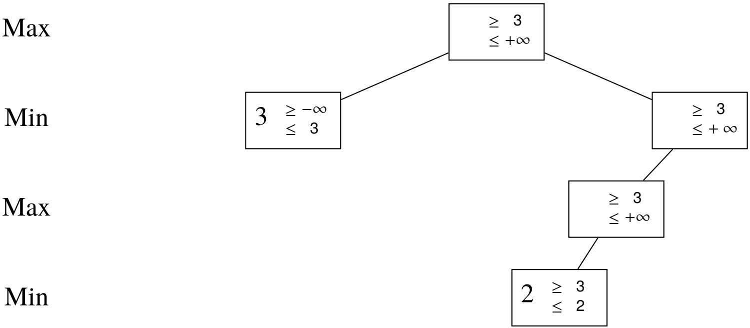
Alpha-Beta Pruning



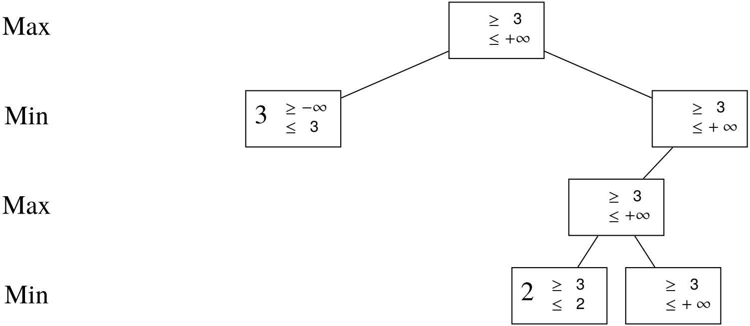
Alpha-Beta Pruning



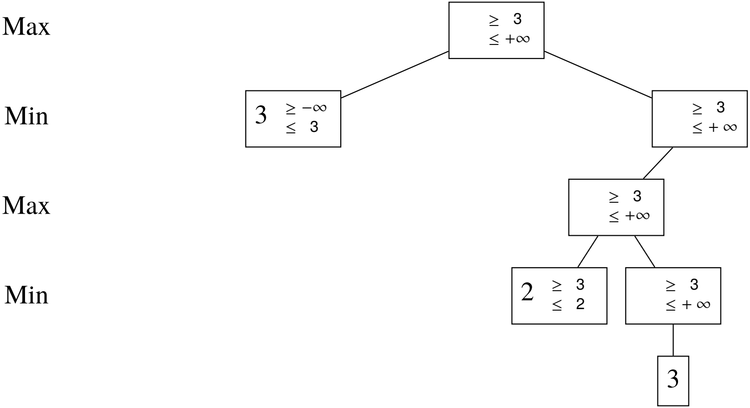
Alpha-Beta Pruning



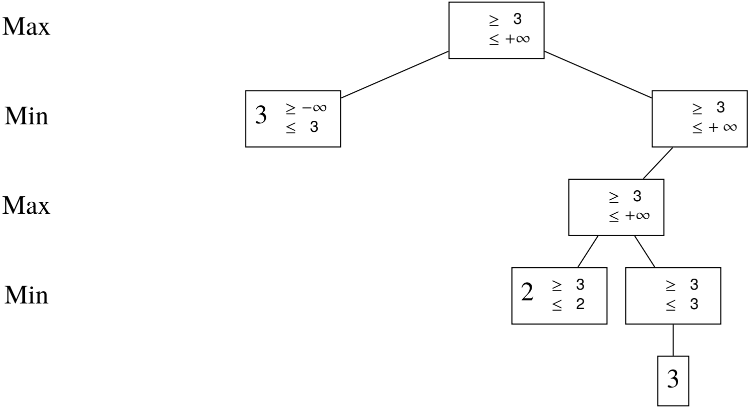
Alpha-Beta Pruning



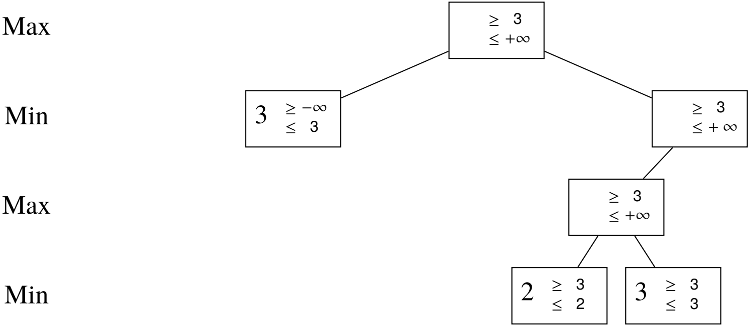
Alpha-Beta Pruning



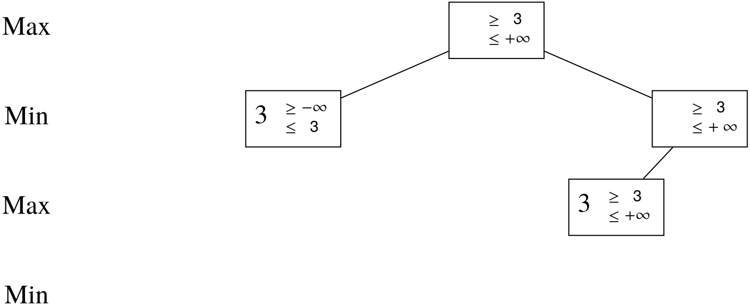
Alpha-Beta Pruning



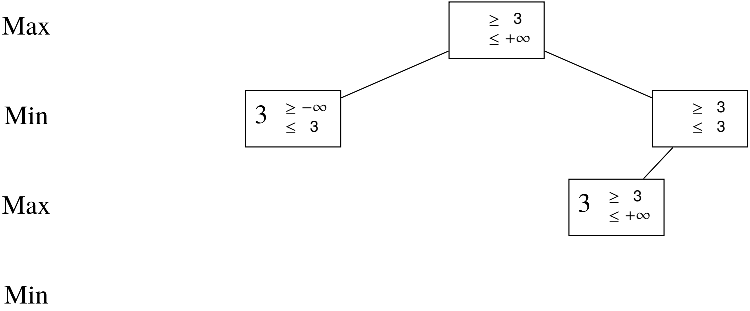
Alpha-Beta Pruning



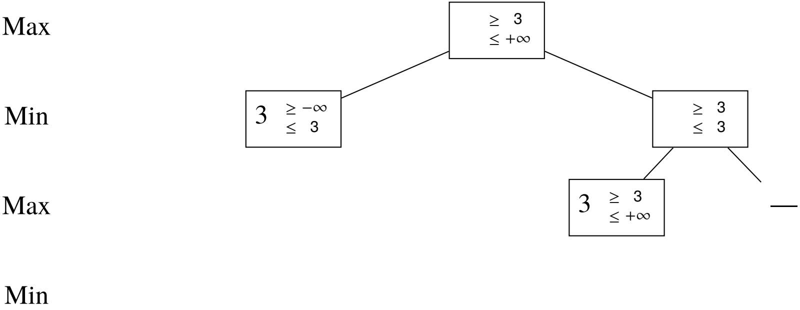
Alpha-Beta Pruning



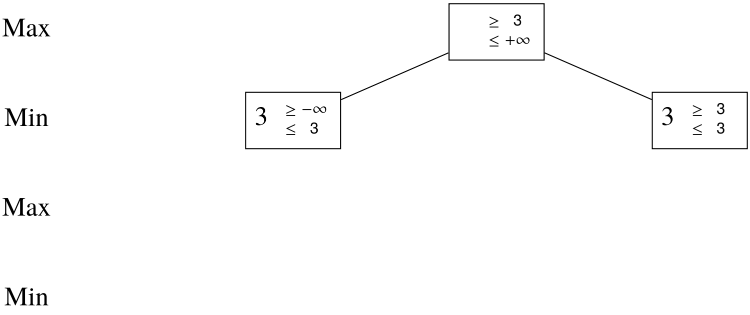
Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning



Alpha-Beta Pruning

Max

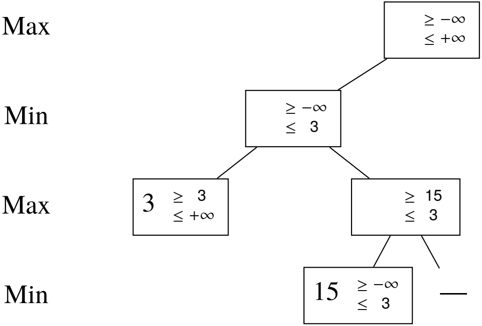
$\boxed{3 \quad \begin{matrix} \geq & 3 \\ \leq & +\infty \end{matrix}}$

Min

Max

Min

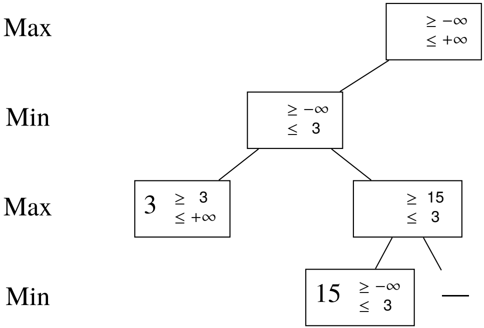
Alpha-Beta Pruning Properties



Time Complexity

- "Perfectly" ordered branches gives $O(b^{m/2})$
- Still exponential, but doubles solvable depth

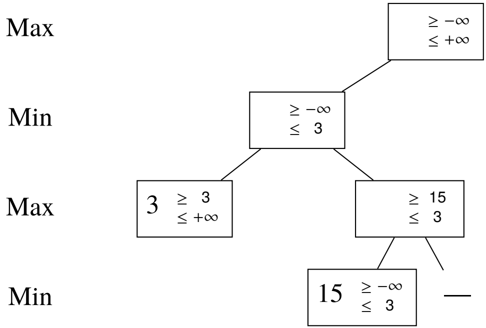
Alpha-Beta Pruning Properties



Time Complexity

- "Perfectly" ordered branches gives $O(b^{m/2})$
- Still exponential, but doubles solvable depth

Alpha-Beta Pruning Properties



Time Complexity

- “Perfectly” ordered branches gives $O(b^{m/2})$
- Still exponential, but doubles solvable depth

Approximate Solutions

Problem

Even with Alpha-Beta Pruning, chess space is still 35^{50}

Solution

Don't search the entire tree:

```
if problem.is_terminal(state):
    return problem.get_utility(state)
```

Example

- Given 100 seconds
- Given ability to explore 10^4 nodes/second
- Can explore to depth ≈ 8 ($10^6 \approx 35^{8/2}$)

Approximate Solutions

Problem

Even with Alpha-Beta Pruning, chess space is still 35^{50}

Solution

Don't search the entire tree:

```
if problem.is_terminal(state):
    return problem.get_utility(state)
```

Example

- Given 100 seconds
- Given ability to explore 10^4 nodes/second
- Can explore to depth ≈ 8 ($10^6 \approx 35^{8/2}$)

Approximate Solutions

Problem

Even with Alpha-Beta Pruning, chess space is still 35^{50}

Solution

Don't search the entire tree:

```
if problem.is_past_cutoff(state):
    return problem.get_estimated_utility(state)
```

Example

- Given 100 seconds
- Given ability to explore 10^4 nodes/second
- Can explore to depth ≈ 8 ($10^6 \approx 35^{8/2}$)

Approximate Solutions

Problem

Even with Alpha-Beta Pruning, chess space is still 35^{50}

Solution

Don't search the entire tree:

```
if problem.is_past_cutoff(state):
    return problem.get_estimated_utility(state)
```

Example

- Given 100 seconds
- Given ability to explore 10^4 nodes/second
- Can explore to depth ≈ 8 ($10^6 \approx 35^{8/2}$)

Evaluation Functions

Necessary Properties

- Quickly computable
- For terminals, $\textsc{Eval}(s)$ orders by utility
- For nonterminals, $\textsc{Eval}(s)$ correlates with winning

Typical Approach

Weighted linear combination of features:

$$\textsc{Eval}(s) = w_1 f_1(s) + w_1 f_2(s) + \ldots + w_n f_n(s)$$

Evaluation Functions

Necessary Properties

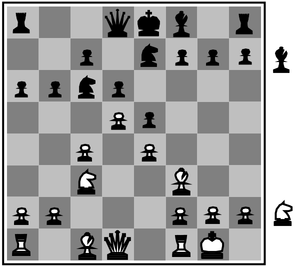
- Quickly computable
- For terminals, $\textsc{Eval}(s)$ orders by utility
- For nonterminals, $\textsc{Eval}(s)$ correlates with winning

Typical Approach

Weighted linear combination of features:

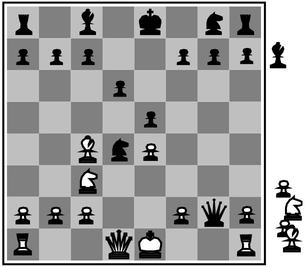
$$\textsc{Eval}(s) = w_1 f_1(s) + w_1 f_2(s) + \ldots + w_n f_n(s)$$

Example Evaluation Function



Black to move

White slightly better



White to move

Black winning

Weighted Features

$$
\begin{aligned}
w_{\text{pawn}} \cdot f_{\text{pawn}}(s) &= 1 \cdot (\text{white pawns} - \text{black pawns}) \\
\ldots &= \ldots \\
w_{\text{queen}} \cdot f_{\text{queen}}(s) &= 9 \cdot (\text{white queens} - \text{black queens})
\end{aligned}
$$

Design an Evaluation Function

Extended Tic-Tac-Toe

- $N \times N$ board
- K in a row wins

Must satisfy:

- Quickly computable
- $\forall x, t, o \in \textsc{Terminals},$
  $$\textsc{Util}(x) = +1 \wedge \textsc{Util}(t) = 0 \wedge \textsc{Util}(o) = -1 \implies \textsc{Eval}(x) > \textsc{Eval}(t) > \textsc{Eval}(\textsc{o})$$
- $\forall s \in \textsc{Nonterminals}, \textsc{Eval}(s)$ correlates with chance of $+1$

Deterministic Games in Practice

Checkers

- 1995: Chinook "Man-Machine World Champion" (1-0-31)
- 2007: Chinook's creators "proved" it cannot lose
- End-game database for all ≤ 8 piece states

Chess

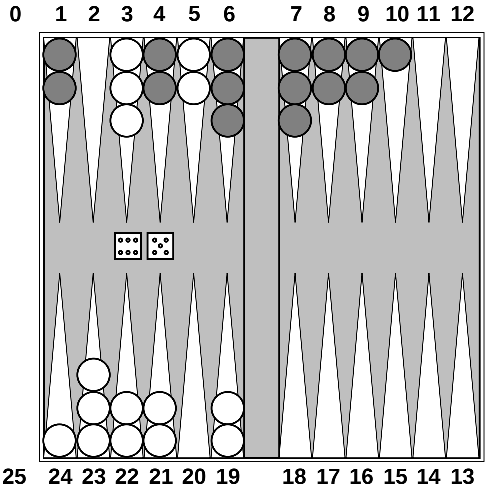
- 1997: Deep Blue defeated Garry Kasparov (2-1-3)
- Searches 6-40 plies; end-game database for ≤ 5 piece states
- 2002,'04,'06,'09,'11,'13: Junior wins World Computer Chess

Go

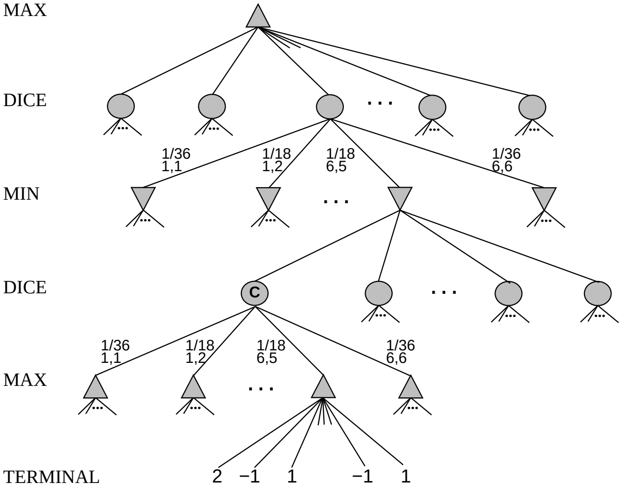
- Branching factor starts at 361 $\Rightarrow$ Monte Carlo methods
- Computers still rank as advanced amateurs (6 dan)

Outline

Nondeterministic Game: Backgammon



Backgammon Game Tree

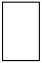


Expectiminimax

Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$

Max



Chance

Min

Expectiminimax

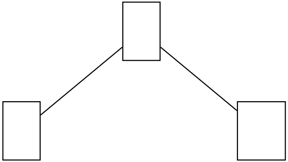
Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$

Max

Chance

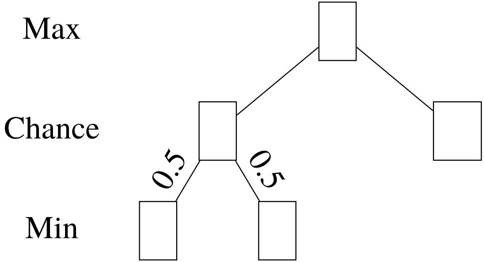
Min



Expectiminimax

Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

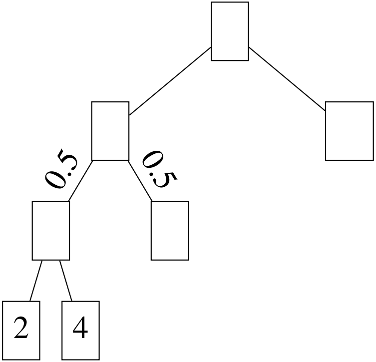
Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$

Max

Chance

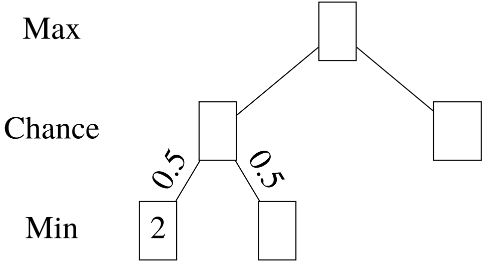
Min



Expectiminimax

Expected Values for Chance Nodes

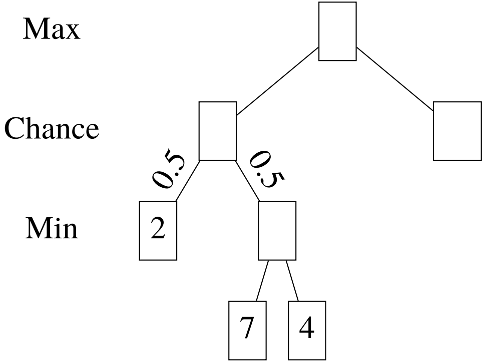
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

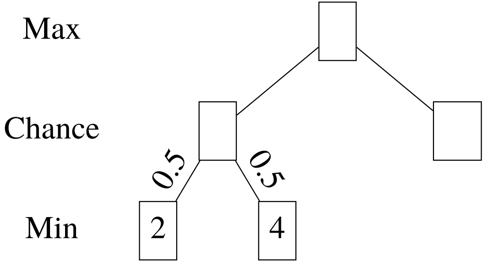
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

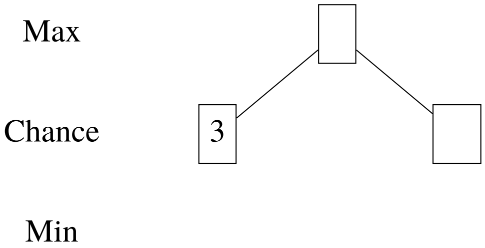
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

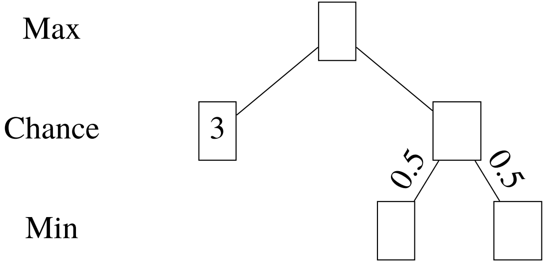
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

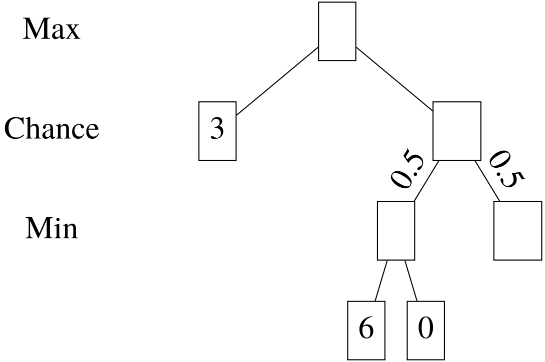
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

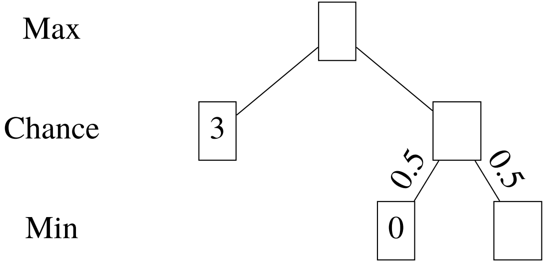
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

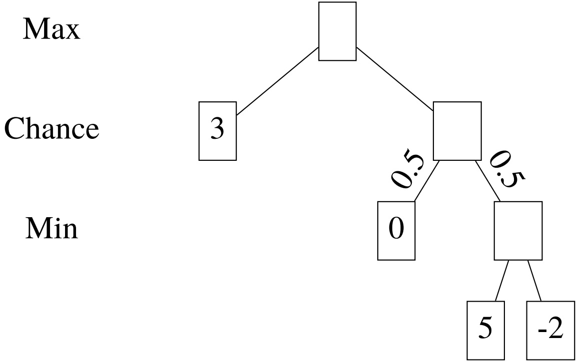
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

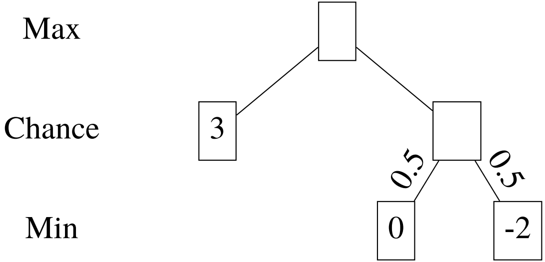
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

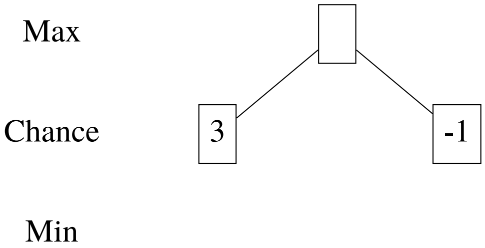
$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$



Expectiminimax

Expected Values for Chance Nodes

$$f(n) = \sum_{s \in \textsc{Successors}(n)} P(s) \cdot f(s)$$

Max $\boxed{3}$

Chance

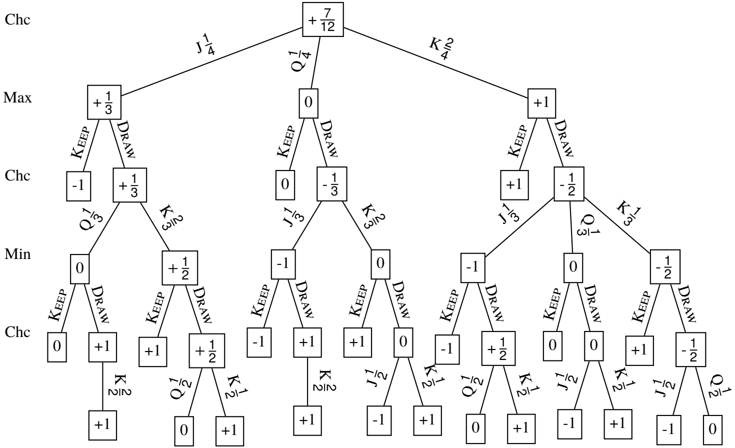
Min

Expectiminimax Practice

A Simple Game

- Deck contains J, Q, K, K in a random order
- A card is drawn and Player 1 either:
  - Ends the game, or
  - Another card is drawn and Player 2 either:
    - Ends the game, or
    - Another card is drawn and the game ends
- Utility is determined by the last card drawn
- J scores -1, Q scores 0, and K scores +1

Expectiminimax Practice



Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and "rolls")
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all "roll" sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and “rolls”)
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all “roll” sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and "rolls")
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all "roll" sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and “rolls”)
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all “roll” sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and "rolls")
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all "roll" sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and "rolls")
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all "roll" sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and "rolls")
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all "roll" sequences

Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

Using Expectiminimax

Expectiminimax Properties

Complete?	Yes, if tree is finite (both moves and “rolls”)
Optimal?	Yes
Time?	$O(b^m n^m)$, all nodes, all “roll” sequences

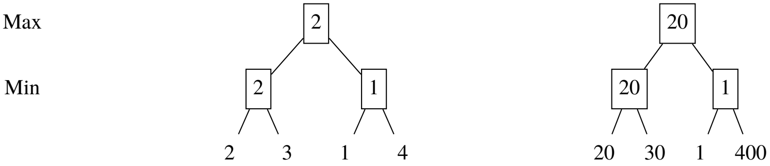
Consequences

Node Likelihood?	Decreases with depth
Alpha-Beta Pruning?	Effectiveness decreased

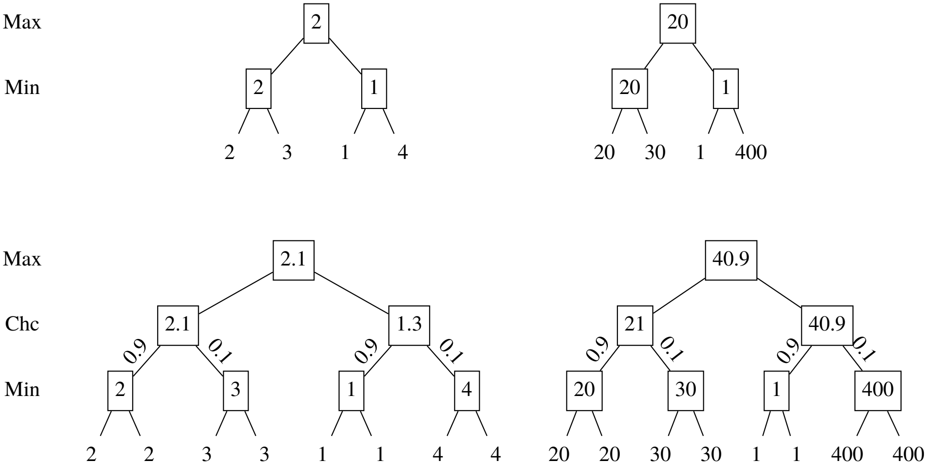
Real-World Example: TD-Gammon

- Depth 2-3 search, no Alpha-Beta pruning
- Neural network eval function trained by self-play

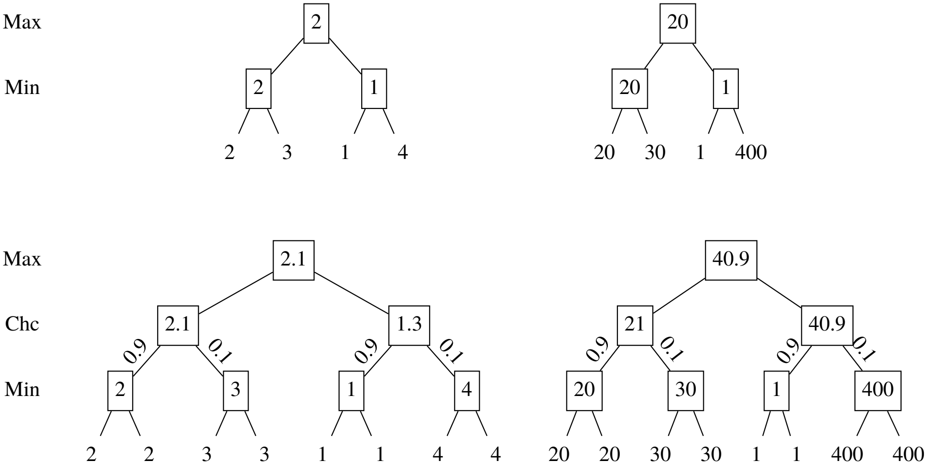
Utilities: Minimax vs. Expectiminimax



Utilities: Minimax vs. Expectiminimax



Utilities: Minimax vs. Expectiminimax



Evaluation function must be linear transform of true utility

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Take the left fork and you'll find a large heap of gold
  - Take the right fork and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Take the left fork and you'll find a large heap of gold
  - Take the right fork and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
    - Take the left fork and you'll find a large heap of gold
    - Take the right fork and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Take the left fork and you'll find a large heap of gold
  - Take the right fork and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Take the left fork and you'll be run over by a bus
  - Take the right fork and you'll find a large heap of gold

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
    - Take the left fork and you'll be run over by a bus
    - Take the right fork and you'll find a large heap of gold

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Guess correctly and you'll find a large heap of gold
  - Guess incorrectly and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns?

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Guess correctly and you'll find a large heap of gold
  - Guess incorrectly and you'll be run over by a bus

Partially Observable Games

Missing knowledge on opponent state

- Poker, Blackjack, Bridge, etc.
- Just average over all possible unknowns? No!

Problems with *averaging over clairvoyance*

- Road A leads to a small heap of gold
- Road B leads to a fork:
  - Guess correctly and you'll find a large heap of gold
  - Guess incorrectly and you'll be run over by a bus

Partially Observable Games

Key Point

Value of an action is not the average across all states

Should be searching through a tree of belief states, and:

- Acting to obtain information
- Signaling to one's partner
- Acting randomly to minimize information disclosure

But in the Real World. . .

Most programs use Monte-Carlo estimation:

- Generate 100+ deals consistent with bidding
- Pick action that wins most on average

Partially Observable Games

Key Point

Value of an action is not the average across all states
Should be searching through a tree of belief states, and:

- Acting to obtain information
- Signaling to one's partner
- Acting randomly to minimize information disclosure

But in the Real World. . .

Most programs use Monte-Carlo estimation:

- Generate 100+ deals consistent with bidding
- Pick action that wins most on average

Partially Observable Games

Key Point

Value of an action is not the average across all states
Should be searching through a tree of belief states, and:

- Acting to obtain information
- Signaling to one's partner
- Acting randomly to minimize information disclosure

But in the Real World. . .

Most programs use Monte-Carlo estimation:

- Generate 100+ deals consistent with bidding
- Pick action that wins most on average

Key Points

Representing Games

- Multiple plies per round, one per player
- Stochastic games introduce chance nodes

Optimal Solutions

- (Expecti-)Minimax produces optimal actions
- Search belief states when information is incomplete

Approximate Solutions

- (Expecti-)Minimax explores the whole tree
- Approximations use utility estimates and cutoffs
- Chance dramatically reduces the depth explored