Artificial Intelligence

Chapter 6

Stuart RUSSEL

reorganized by L. Aszalós

April 4, 2016

Outline

- Games
- Perfect play
  - minimax decisions
  - α–β pruning
- Resource limits and approximate evaluation
- Games of chance
- Games of imperfect information

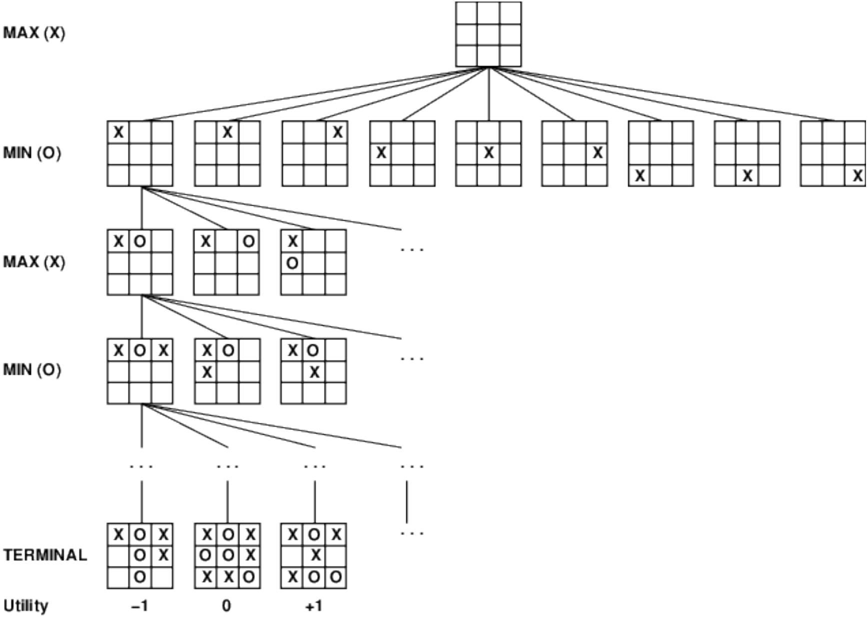
Games vs. search problems

- "Unpredictable" opponent ⇒ solution is a **strategy**
  - specifying a move for every possible opponent reply
- Time limits ⇒ unlikely to find goal, must approximate
- Plan of attack:
  - Computer considers possible lines of play (Babbage, 1846)
  - Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
  - Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
  - First chess program (Turing, 1951)
  - Machine learning to improve evaluation accuracy (Samuel, 1952–57)
  - Pruning to allow deeper search (McCarthy, 1956)

Types of games

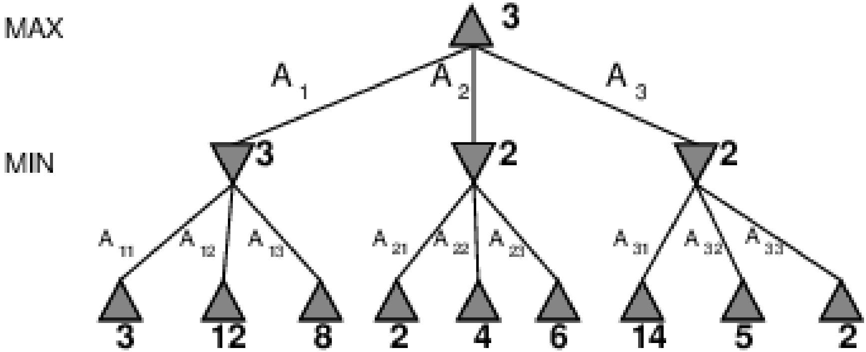
	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

Game tree (2-player, deterministic, turns)



Minimax

- Perfect play for deterministic, perfect-information games
- **Idea**: choose move to position with highest **minimax value**
  - = best achievable payoff against best play
- E.g., 2-ply game:



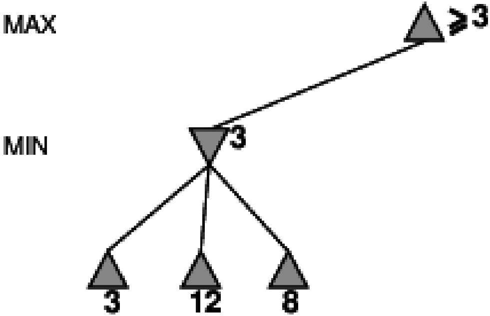
Minimax algorithm

```
function Minimax-Decision(state) returns an action
  state: current state in game

  return a in Actions(state) maximizing Min-Value(Result(a, state))

function Max-Value(state) returns a utility value

  if Terminal-Test(state) then return Utility(state)
  v := -infinity
  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 := infinity
  for (a, s) in Successors(state) do
    v := Min(v, Max-Value(s))
  return v
```

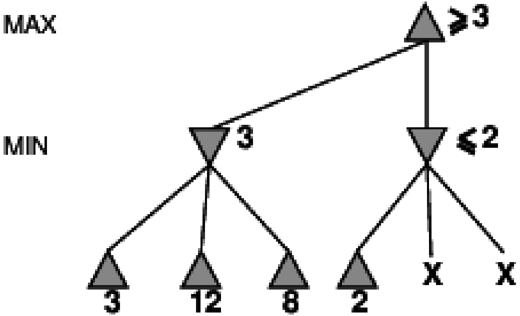
Properties of minimax

- Complete
  - Yes, if tree is finite (chess has specific rules for this)
- Optimal
  - Yes, against an optimal opponent. Otherwise??
- Time complexity
  - $O(b^m)$
- Space complexity
  - $O(bm)$ (depth-first exploration)
- For chess, $b \approx 35$, $m \approx 100$ for "reasonable" games
  - $\Rightarrow$ exact solution completely infeasible
- But do we need to explore every path?

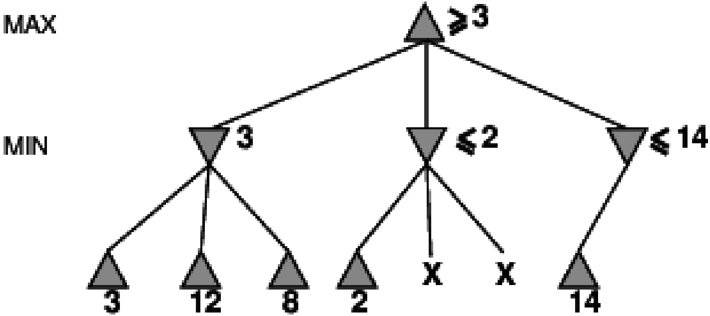
α–β pruning example



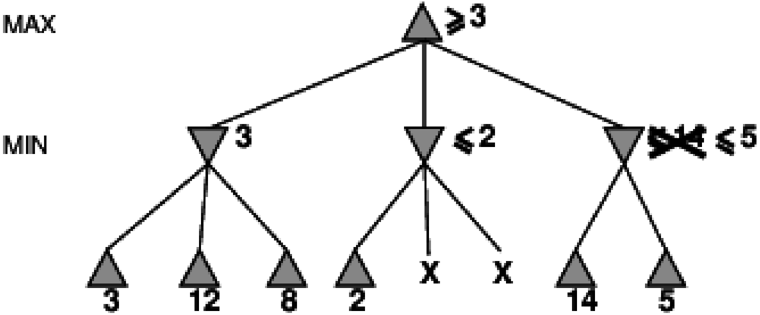
α–β pruning example



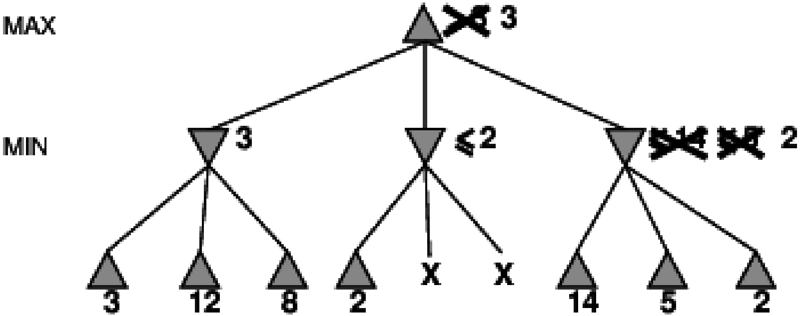
α–β pruning example



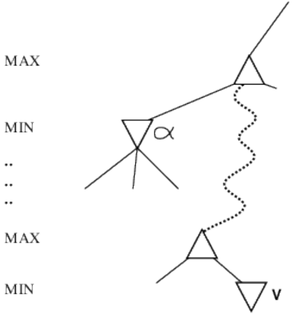
α–β pruning example



α–β pruning example



Why is it called α–β?



- α is the best value (to *MAX*) found so far off the current path
- If V is worse than α, *MAX* will avoid it
  - $\Rightarrow$ prune that branch
- Define β similarly for *MIN*

The α–β algorithm

```
function Alpha-Beta-Decision(state) returns an action
  return a in Actions(state) maximizing
              Min-Value(Result(a, state), -infinity, infinity)

function Max-Value(state, alpha, beta) returns a utility value
  state: current state in game
  alpha: the value of the best alternative for MAX along the path to state
  beta:  the value of the best alternative for MIN along the path to state

  if Terminal-Test(state) then return Utility(state)
  v:= -infinity
  for (a, s) in Successors(state) do
    v := Max(v, Min-Value(s, alpha, beta))
    if v >= beta then return v
    alpha := Max(alpha, v)
  return v
```

function `Min-Value(state, alpha, beta)` same as `Max-Value` but with roles of alpha and beta reversed

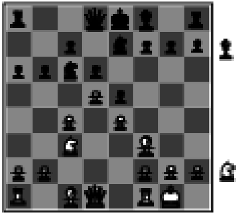
Properties of α–β

- Pruning *does not* affect final result
- Good move ordering improves effectiveness of pruning
- With "perfect ordering," time complexity $= O(b^{m/2})\}$
  - $\Rightarrow$ *doubles* solvable depth
- A simple example of the value of reasoning about which computations are relevant (a form of **metareasoning**)
- Unfortunately, 35^{50} is still impossible!

Resource limits

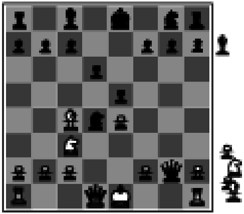
- Standard approach:
  - Use *Cutoff-Test* instead of *Terminal-Test*
  - e.g., depth limit (perhaps add **quiescence search**)
- Use *Eval* instead of *Utility*
  - i.e., **evaluation function** that estimates desirability of position
- Suppose we have 100 seconds, explore 10^4 nodes/second
  - $\Rightarrow 10^6$ nodes per move $\approx 35^{8/2}$
  - $\Rightarrow \alpha$–β reaches depth 8 $\Rightarrow$ pretty good chess program

Evaluation functions



Black to move

White slightly better



White to move

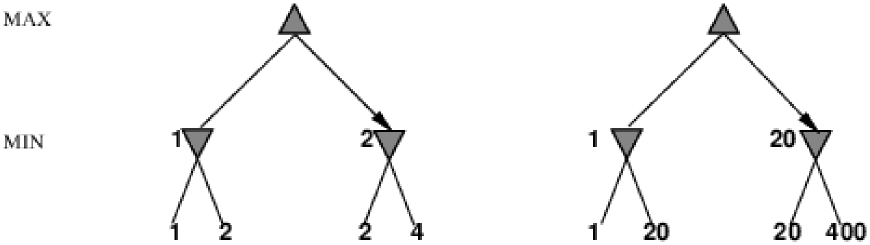
Black winning

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.

Digression: Exact values don't matter



- Behaviour is preserved under any *monotonic* transformation of *Eval*
- Only the order matters:
  - payoff in deterministic games acts as an **ordinal utility** function

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.

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.

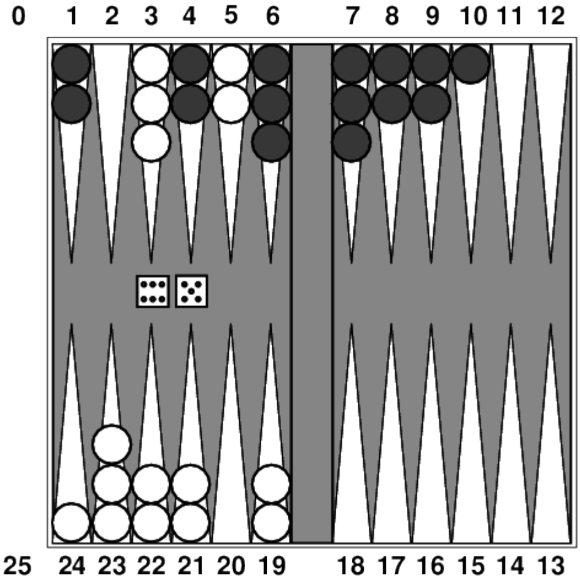
Othello: human champions refuse to compete against computers, who are too good.

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. (*from 2004*)

Deterministic games in practice

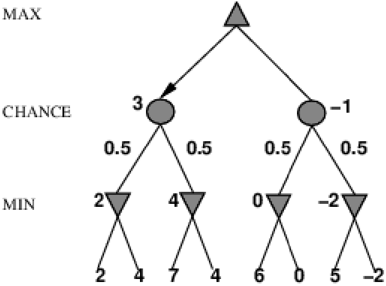
Go: update: In March 2016, AlphaGo beat Lee Sedol in a five-game match, the first time a computer Go program has beaten a 9-dan professional without handicaps. (final score of 4 games to 1 in favour of AlphaGo)

Nondeterministic games: backgammon



Nondeterministic games in general

- In nondeterministic games, chance introduced by dice, card-shuffling
- Simplified example with coin-flipping:



Algorithm for nondeterministic games

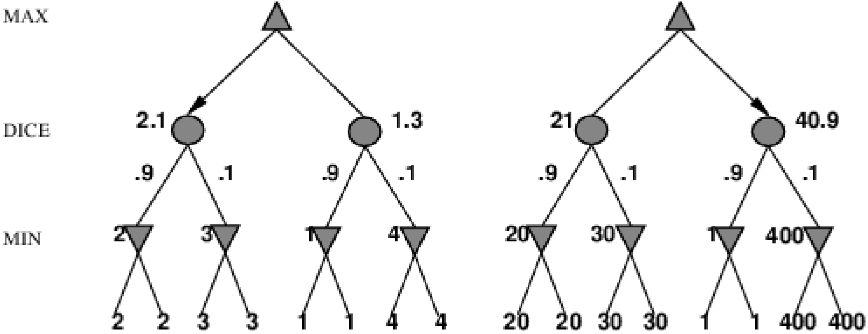
- *Expectiminimax* gives perfect play
- Just like *Minimax*, except we must also handle chance nodes:

```
...
if state is a MAX node then
   return the highest ExpectiMinimax-Value of Successors(state)
if state is a MIN node then
   return the lowest ExpectiMinimax-Value of Successors(state)
if state is a chance node then
   return average of ExpectiMinimax-Value of Successors}(state)
...
```

Nondeterministic games in practice

- Dice rolls increase *b*: 21 possible rolls with 2 dice
  - Backgammon $\approx$ 20 legal moves (can be 6,000 with 1-1 roll)
  - $$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$
- As depth increases, probability of reaching a given node shrinks
  - $\Rightarrow$ value of lookahead is diminished
- α–β pruning is much less effective
  - *TDGammon* uses depth-2 search + very good *Eval*
  - $\approx$ world-champion level

Digression: Exact values DO matter



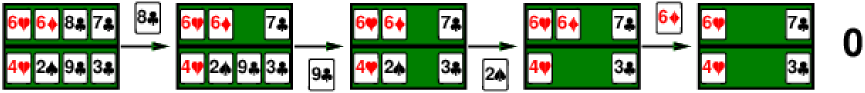
- Behaviour is preserved only by **positive linear** transformation of *Eval*
- Hence *Eval* should be proportional to the expected payoff

Games of imperfect information

- E.g., card games, where opponent's initial cards are unknown
- Typically we can calculate a probability for each possible deal
- Seems just like having one big dice roll at the beginning of the game
- **Idea**: compute the minimax value of each action in each deal then choose the action with highest expected value over all deals
- Special case: if an action is optimal for all deals, it's optimal.
- GIB, current best bridge program, approximates this idea by
  1. generating 100 deals consistent with bidding information
  2. picking the action that wins most tricks on average

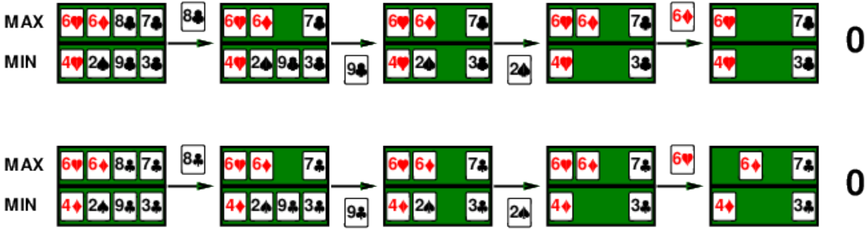
Example

Four-card bridge/whist/hearts hand, *MAX* to play first



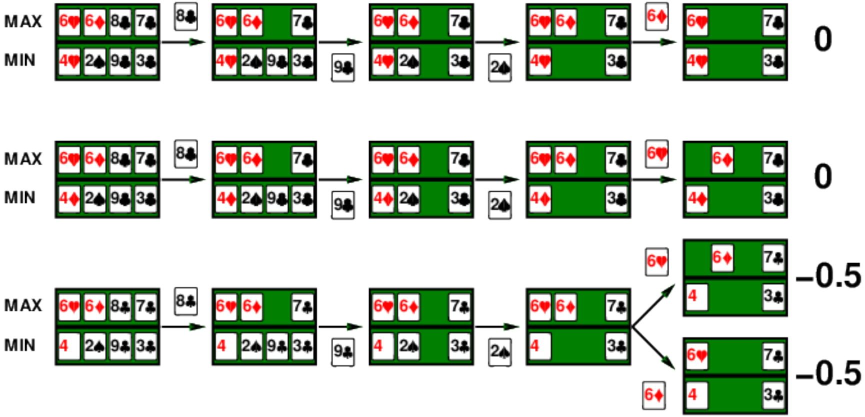
Example

Four-card bridge/whist/hearts hand, *MAX* to play first



Example

Four-card bridge/whist/hearts hand, *MAX* to play first



Commonsense example

- Road A leads to a small heap of gold pieces
- Road B leads to a fork:
  - take the left fork and you'll find a mound of jewels;
  - take the right fork and you'll be run over by a bus.
- Road A leads to a small heap of gold pieces
- 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 mound of jewels.
- Road A leads to a small heap of gold pieces
- Road B leads to a fork:
  - guess correctly and you'll find a mound of jewels;
  - guess incorrectly and you'll be run over by a bus.

Proper analysis

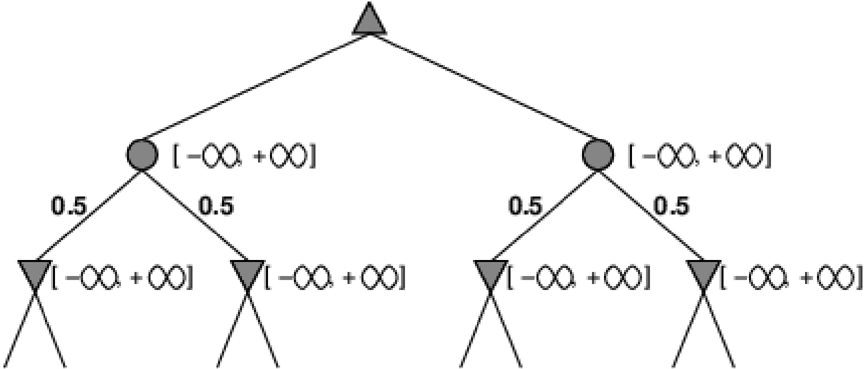
- Intuition that the value of an action is the average of its values
  - in all actual states is *WRONG*
- With partial observability, value of an action depends on the **information state** or **belief state** the agent is in
- Can generate and search a tree of information states
- Leads to rational behaviors such as
  - Acting to obtain information
  - Signalling to one's partner
  - Acting randomly to minimize information disclosure

Summary

- Games are fun to work on! (and dangerous)
- They illustrate several important points about AI
  - perfection is unattainable $\Rightarrow$ must approximate
  - good idea to think about what to think about
  - uncertainty constrains the assignment of values to states
  - optimal decisions depend on information state, not real state
- Games are to AI as grand prix racing is to automobile design

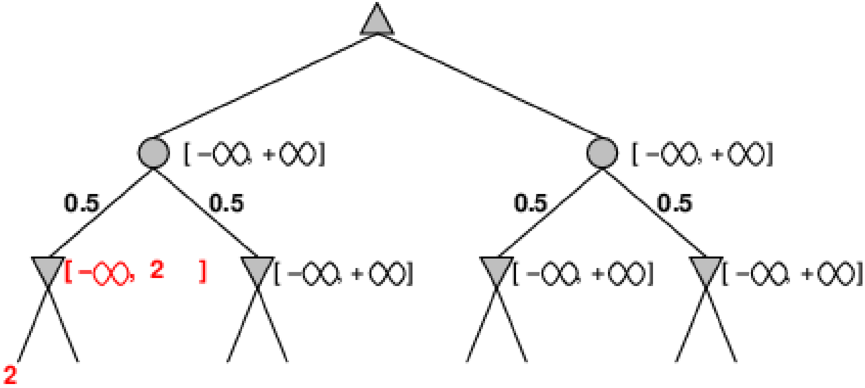
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



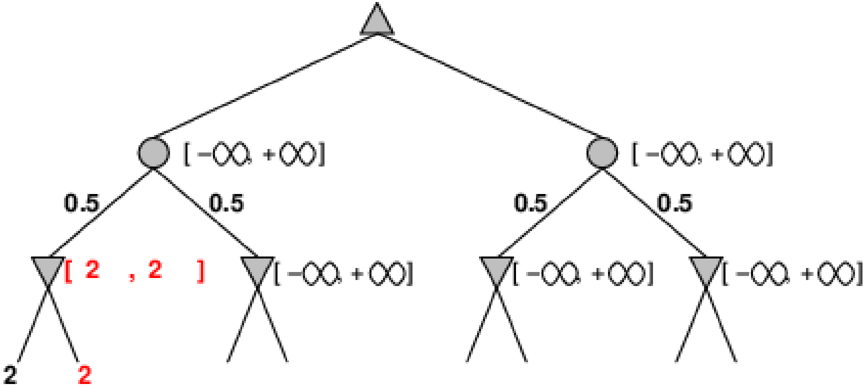
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



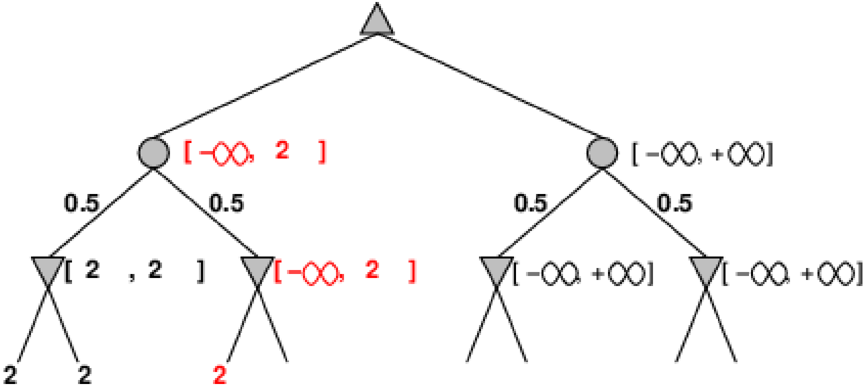
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



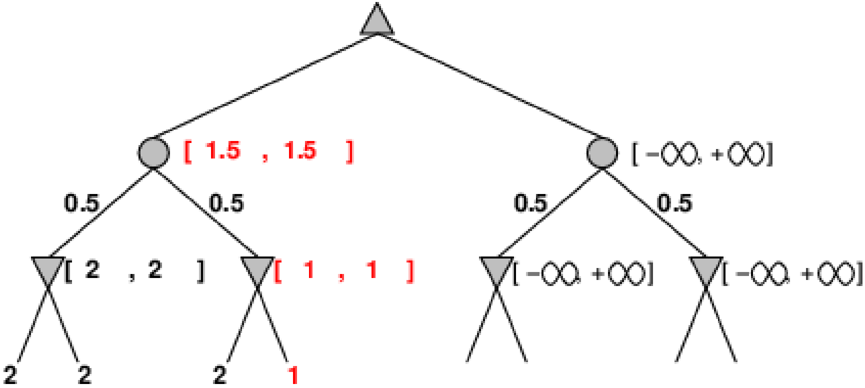
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



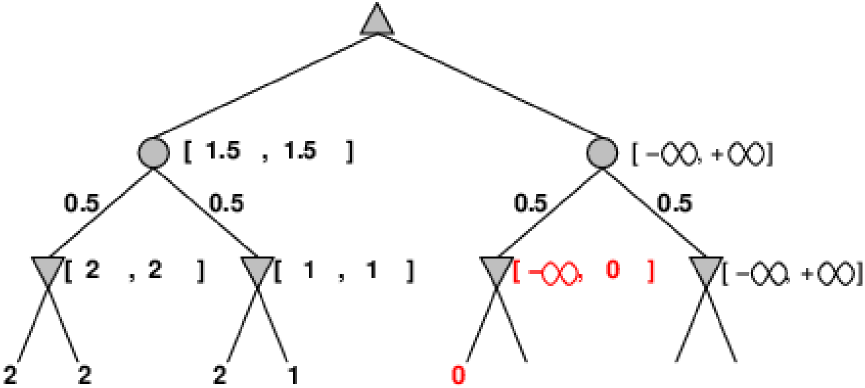
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



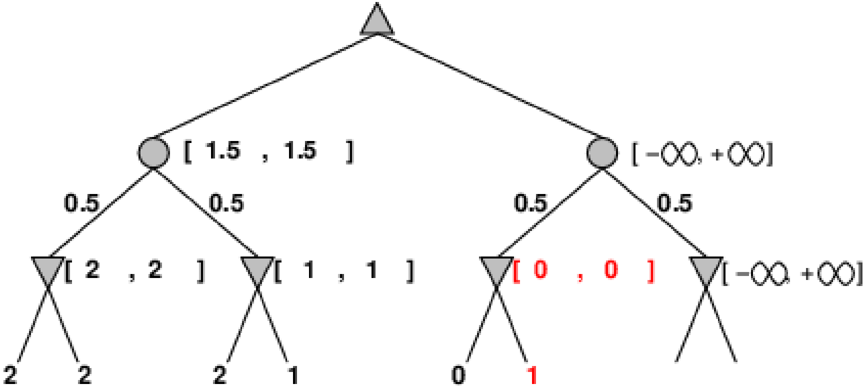
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



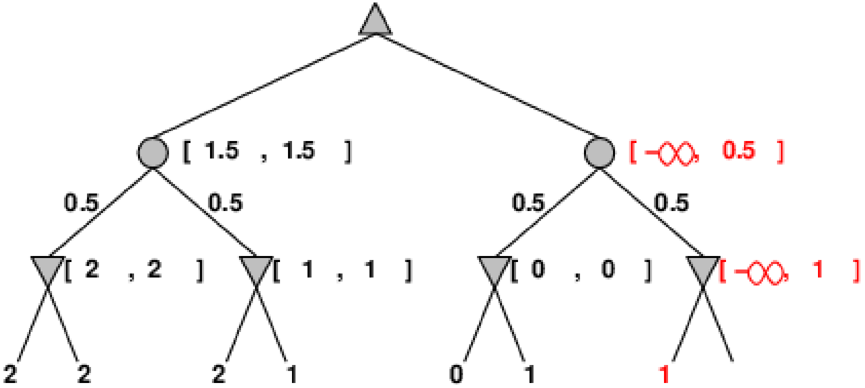
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



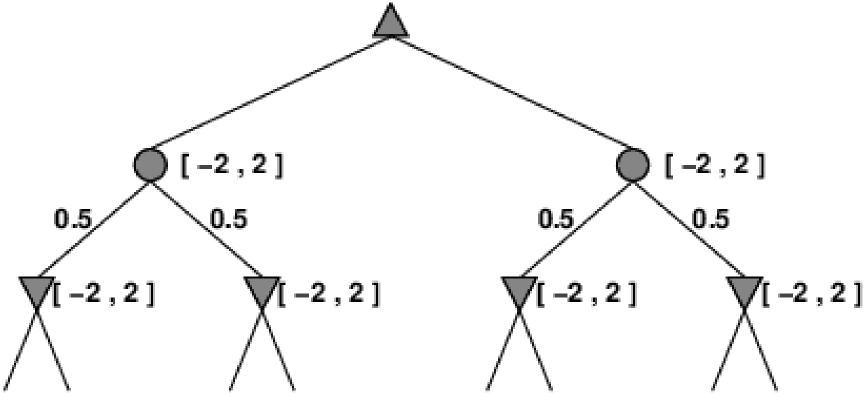
Pruning in nondeterministic game trees

A version of α-β pruning is possible:



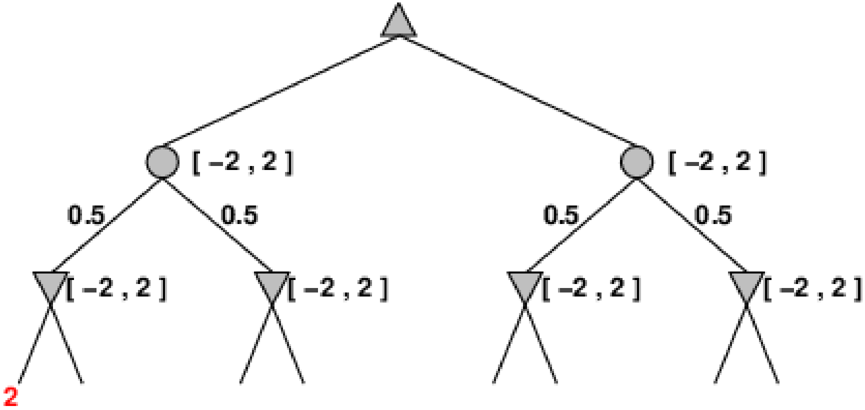
Pruning contd.

More pruning occurs if we can bound the leaf values



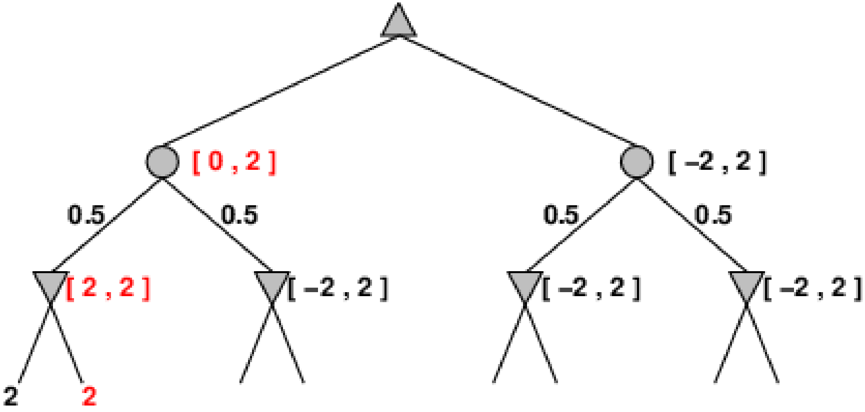
Pruning contd.

More pruning occurs if we can bound the leaf values



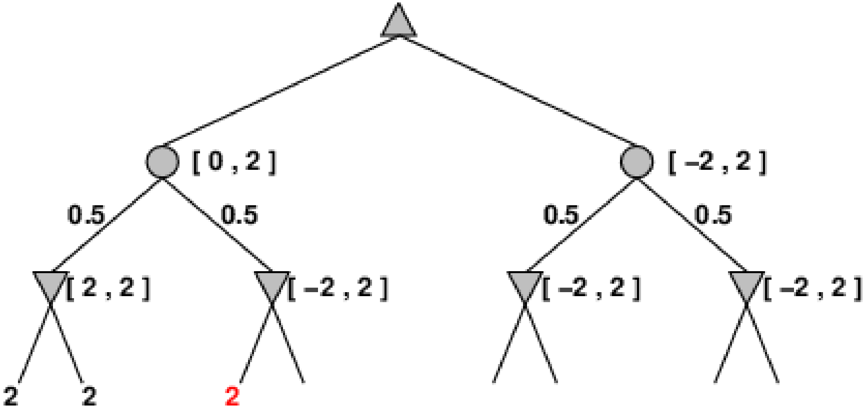
Pruning contd.

More pruning occurs if we can bound the leaf values



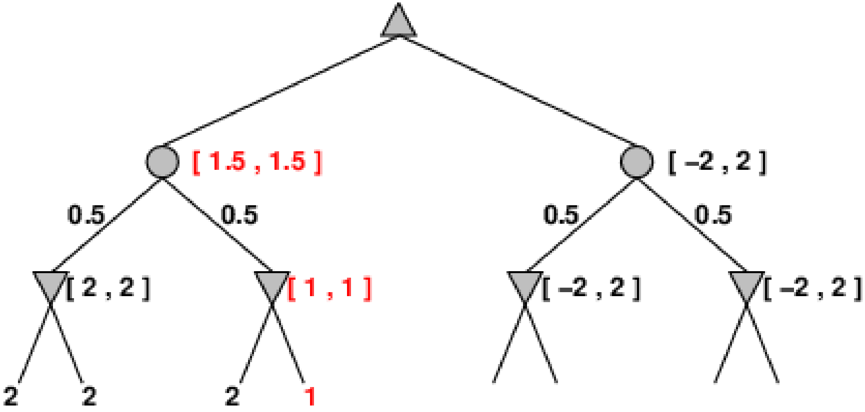
Pruning contd.

More pruning occurs if we can bound the leaf values



Pruning contd.

More pruning occurs if we can bound the leaf values



Pruning contd.

More pruning occurs if we can bound the leaf values