Adversarial Search and Game Playing

Motivations

- Breadth-first, depth-first, hill-climbing, best-first, and A* assume a non-hostile search space. •

The goals just sit there somewhere in the graph.

- The 8-puzzle game did not try to stop you from reaching the goal.
- But in a real 2-person game, you opponent does try to beat you and make it difficult for you to reach your goal.
- Game Trees complexity :-

  Tic-Tac-Toe – 9! – 362,280 states

  Connect Four – 10^13 states

  Checkers – 10^18 states

  Chess – 10^50 states

  Go – 10^170 states

Typical assumptions in adversial search

- Two agents whose actions alternate
- Utility values for each agent are the opposite of the other
  - creates the adversarial situation

- Fully observable environments

- In game theory terms:
  - “Deterministic, turn-taking, zero-sum games of perfect information”

Search versus Games

- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - can find *optimal* solution
  - Evaluation function: estimate of cost from start to goal through given node
  - Examples: path planning, scheduling activities

- Games – adversary
  - Solution is strategy (strategy specifies move for every possible opponent reply).
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate “goodness” of game position
  - Examples: chess, checkers, Othello, backgammon

Types of Games

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

•Imperfect information:- dont know all actions of other player ,don't recall all past moves of opponent or self .

•Solved games (for which best strategy is found)

Tic-tac-toe
Four In A Line
Checkers

Size of search trees

- b = branching factor

- d = number of moves by both players

- Search tree is $O(b^d)$

- Chess
  - b ~ 35
  - D ~100
    - completely impractical to search this

- Game-playing emphasizes being able to make optimal decisions in a finite amount of time
  - Somewhat realistic as a model of a real-world agent
  - Even if games themselves are artificial

Static (Heuristic) Evaluation Functions

- An Evaluation Function:
  - estimates how good the current board configuration is for a player.
  - Typically, one figures how good it is for the player, and how good it is for the opponent, and subtracts the opponents score from the players
  - Othello: Number of white pieces - Number of black pieces
  - Chess: Value of all white pieces - Value of all black pieces

- Typical values from -infinity (loss) to +infinity (win) or [-1, +1].

- If the board evaluation is X for a player, it's -X for the opponent

- Example:
  - Evaluating chess boards,
  - Checkers
  - Tic-tac-toe

Game Setup

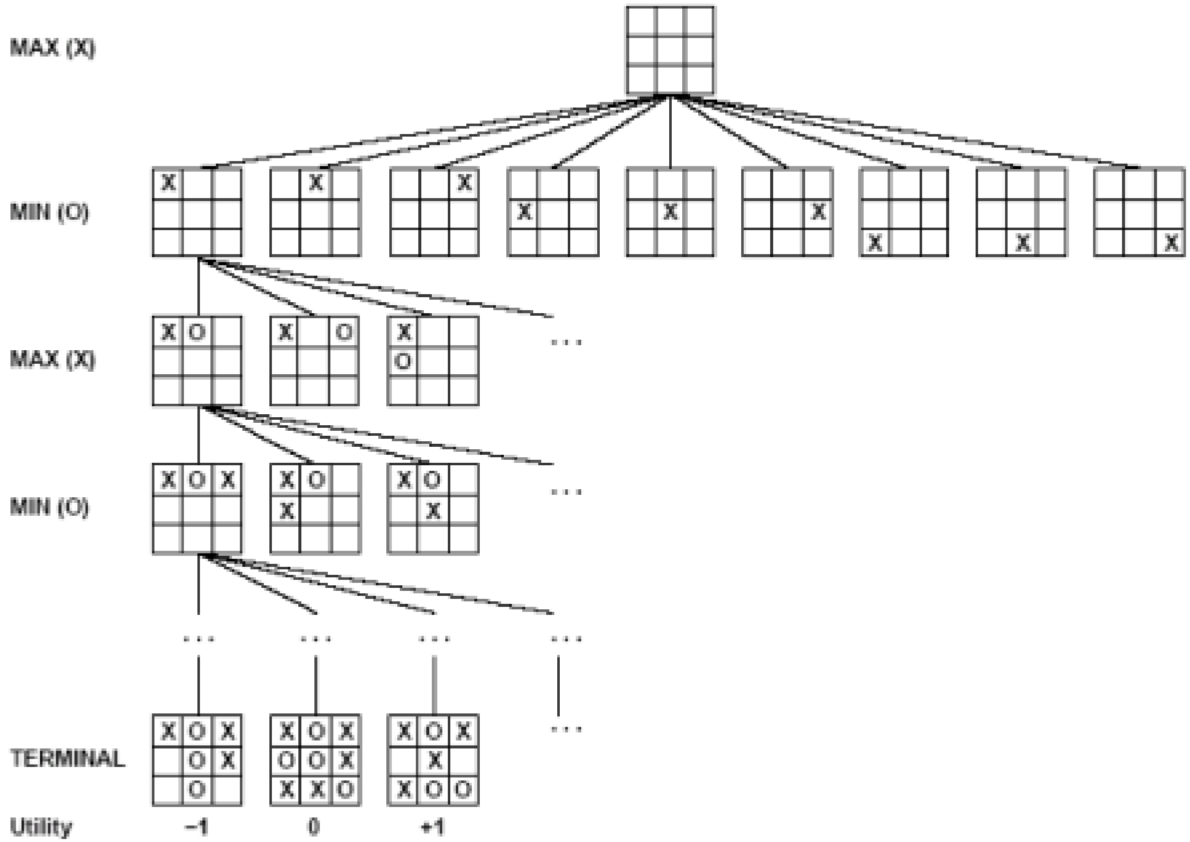
- Two players: MAX and MIN

- MAX moves first and they take turns until the game is over
  - Winner gets award, loser gets penalty.

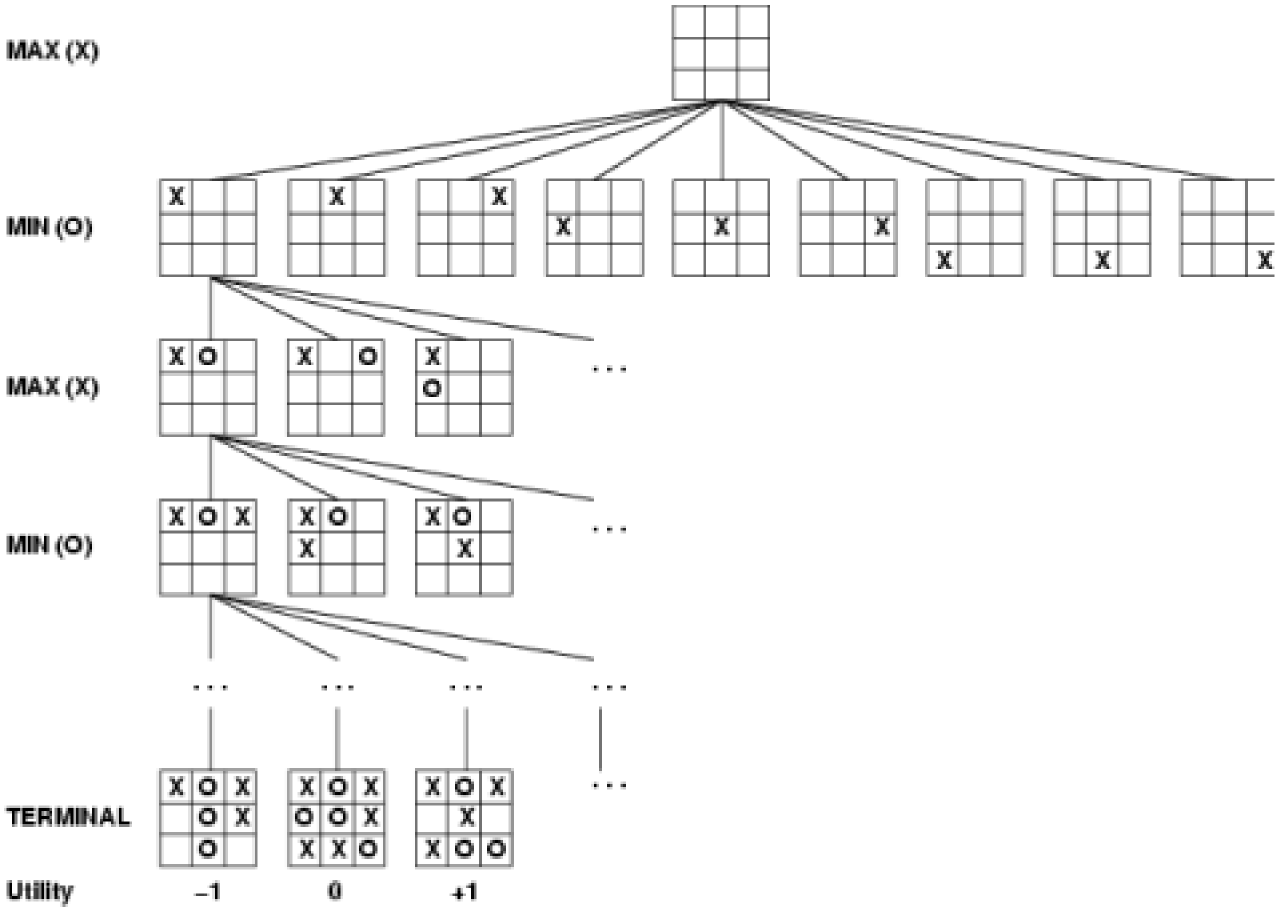
- Games as search:
  - Initial state: e.g. board configuration of chess
  - Successor function: list of (move,state) pairs specifying legal moves.
  - Terminal test: Is the game finished?
  - Utility function: Gives numerical value of terminal states. E.g. win (+1), lose (-1) and draw (0) in tic-tac-toe or chess

- MAX uses search tree to determine next move.

Partial Game Tree for Tic-Tac-Toe

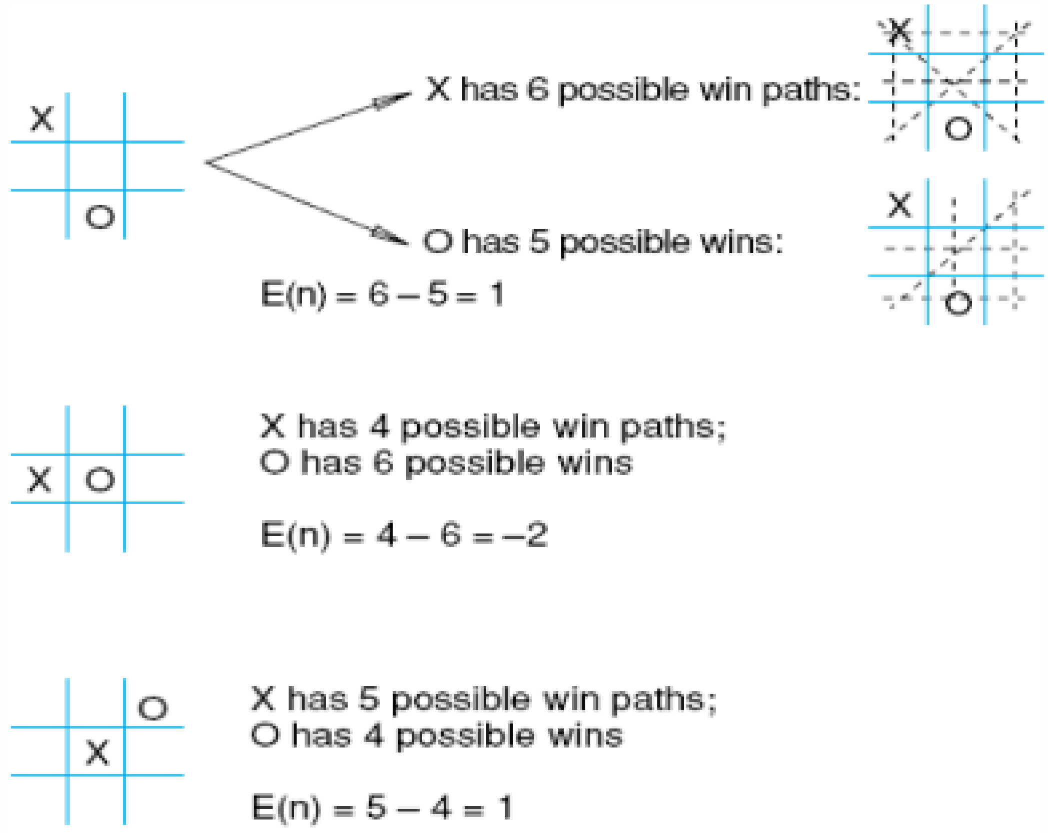


Game tree (2-player, deterministic, turns)



How do we search this tree to find the optimal move?

Heuristic measuring for adversarial tic-tac-toe



Heuristic is E(n) = M(n) – O(n)

where M(n) is the total of My possible winning lines

O(n) is total of Opponent's possible winning lines

E(n) is the total Evaluation for state n

Maximize E(n)

E(n) = 0 when my opponent and I have equal number of possibilities.

Minimax strategy

- Find the optimal *strategy* for MAX assuming an infallible MIN opponent
  - Need to compute this all the down the tree

- Assumption: Both players play optimally!

- Given a game tree, the optimal strategy can be determined by using the minimax value of each node:

MINIMAX-VALUE(*n*)=

{ UTILITY(*n*) If *n* is a terminal OR

max (MINIMAX-VALUE(*s*))from all successors (s) If *n* is a max node OR

min (MINIMAX-VALUE(*s*) from all successors (s) If *n* is a min node }

Simple game for example:
Minimax decision

MAX (player)

MIN
(opponent)

3 12 8 2 4 6 14 5 2

Simple game for example: Minimax decision

MAX (player)

MIN
(opponent)
3

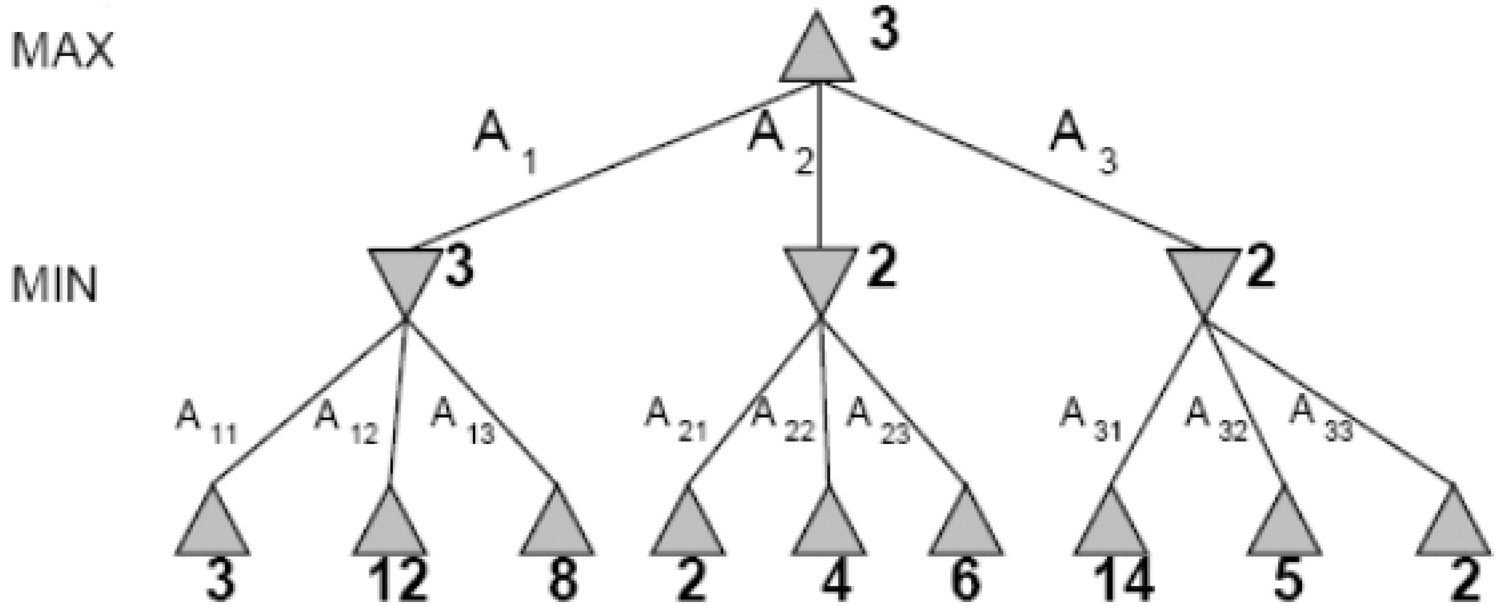
3 2 2

3 12 8 2 4 6 14 5 2

Example 2:- Two-Ply Game Tree

Minimax maximizes the utility for the worst-case outcome for max

The minimax decision



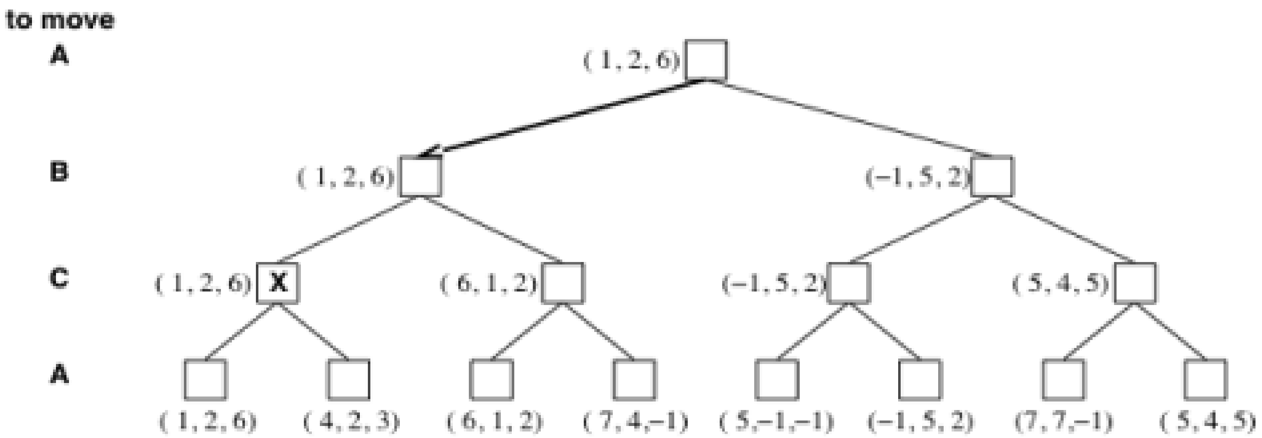
What if MIN does not play optimally?

- Definition of optimal play for MAX assumes MIN plays optimally:
  - maximizes worst-case outcome for MAX
- But if MIN does not play optimally, MAX will do even better

Multiplayer games

- Games allow more than two players
- Single minimax values become vectors

- standard minimax analysis assumes that each player operates to maximize only their own utility
- If game is not zero-sum (i.e., utility(A) = - utility(B) then alliances can be useful even with 2 players
  - e.g., both cooperate to maximum the sum of the utilities



Practical problem with minimax search

- Number of game states is exponential in the number of moves.
  - Solution: Do not examine every node

    => pruning
    - Remove branches that do not influence final decision

Alpha-Beta Pruning: Example

3

MAX (player)

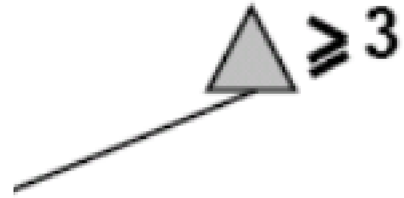
MIN (opponent)

3

3 12 8 2

Stop right here when evaluating this node: •opponent takes minimum of these nodes, •player will take maximum of nodes above

Alpha-Beta Example



Do DF-search until first

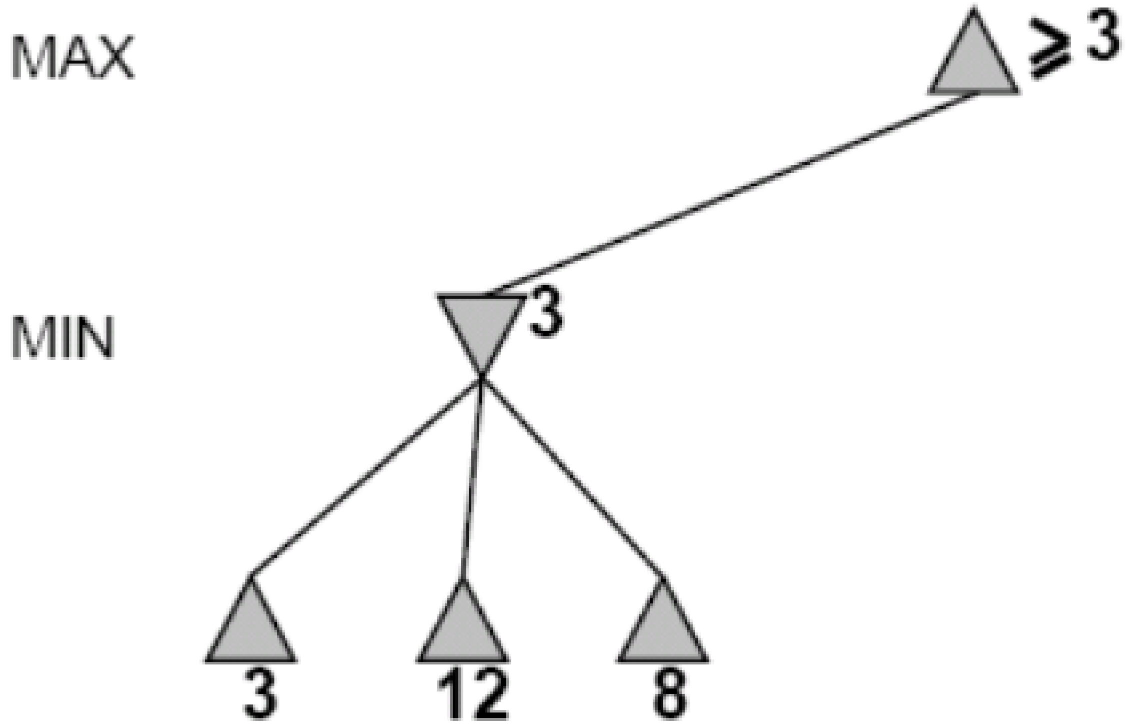
leaf

Range of possible values

$[-\infty,+\infty]$

$[-\infty, +\infty]$

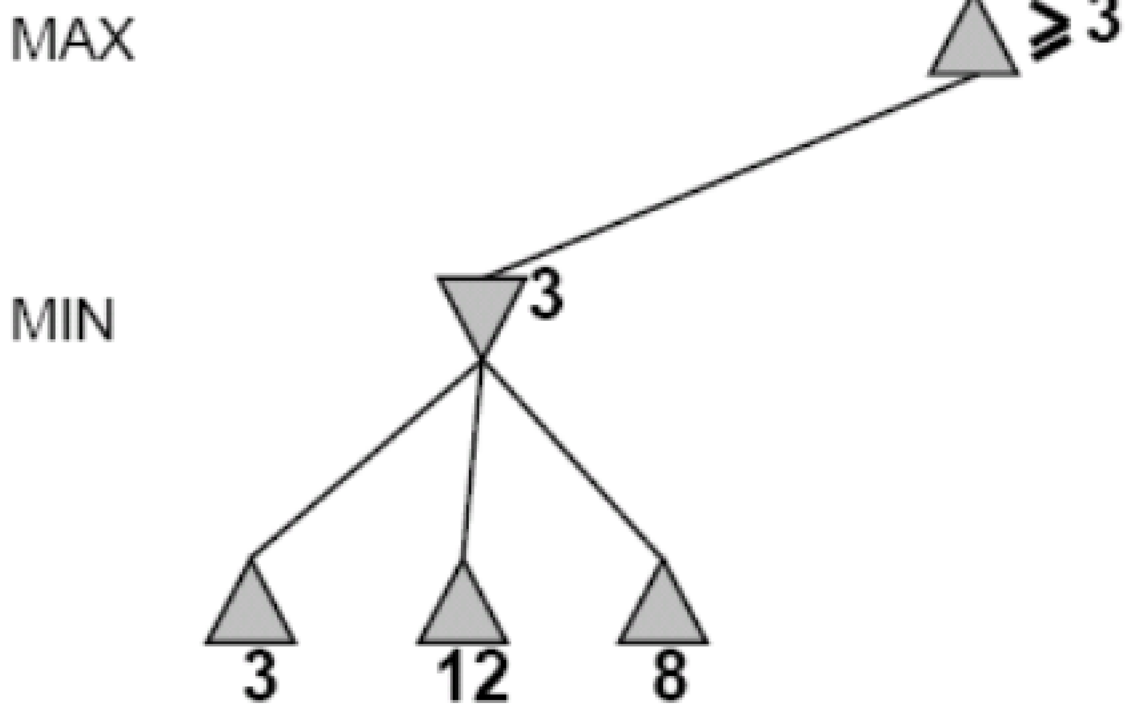
Alpha-Beta Example (continued)



$[-\infty, 3]$

$[-\infty, +\infty] \leqslant 3$

Alpha-Beta Example (continued)



MAX
≥3
MIN
3
3
12
8

[-∞,3]

[-∞,+∞] ≤ 3

ple (continued)

[3,+∞]

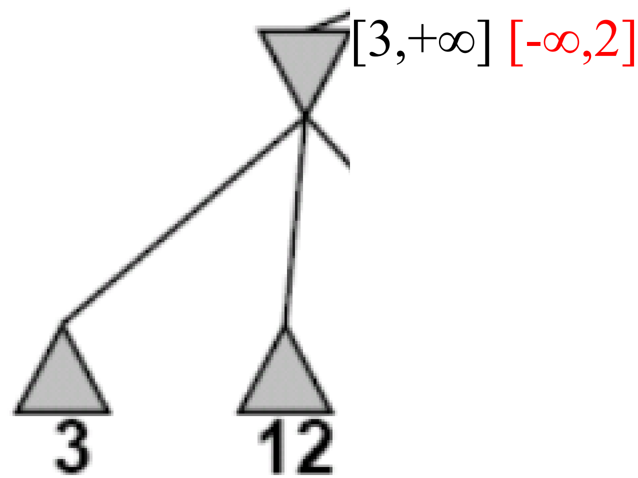
[3,3]

Alpha-Beta Example

MAX

(continued)

MIN



[3,3]

This node is worse for MAX

Alpha-Beta Pruning: Concept

There are two rules for terminating search:

• Search can be stopped below any MIN node having a beta value less than or equal to the alpha value of any of its MAX ancestors.

• Search can be stopped below any MAX node
having an alpha value greater than or equal to the beta value of any of its MIN ancestors.

Alpha-Beta Example (continued)

,

[3,14]

[3,3] [-∞,14] [-∞,2]

Alpha-Beta Example (continued)

,

[3,5]

[3,3] [-∞,5] [−∞,2]

Alpha-Beta Example (continued)

$[3,3]$

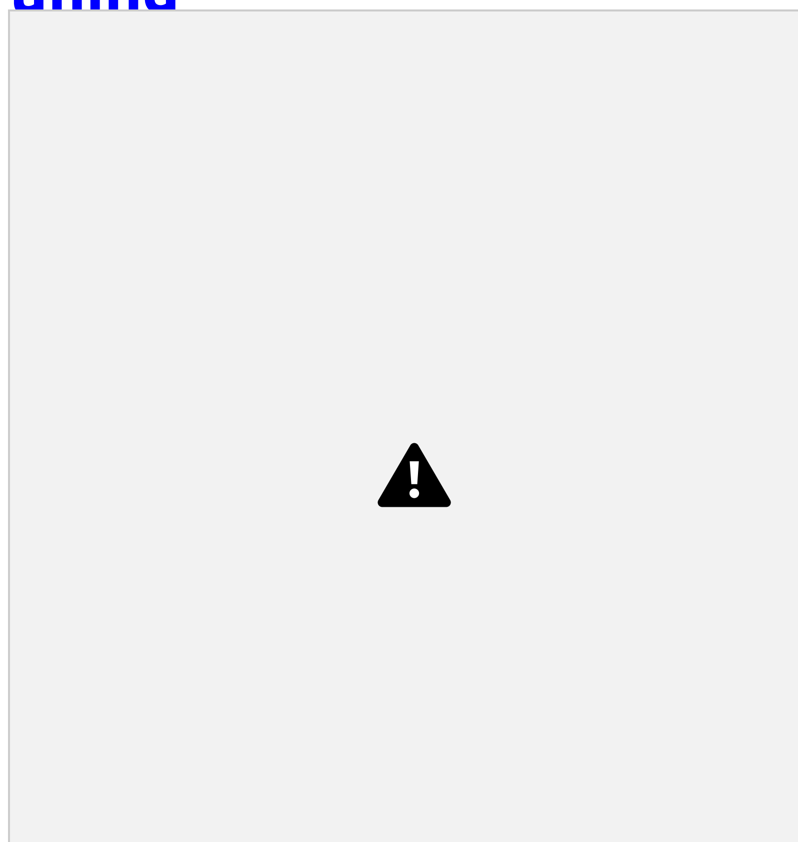
$[3,3]$ $[-\infty,2]$ $[2,2]$

Alpha-Beta Example (continued)

[3,3]

[3,3] [-∞,2] [2,2]

General alpha-beta pruning



- Consider a node *n* somewhere in the tree

- If player has a better choice at – Parent node of n
  – Or any choice point further up

- *n* will **never** be reached in actual play.

- Hence when enough is known about *n*, it can be pruned.

Alpha-beta Algorithm

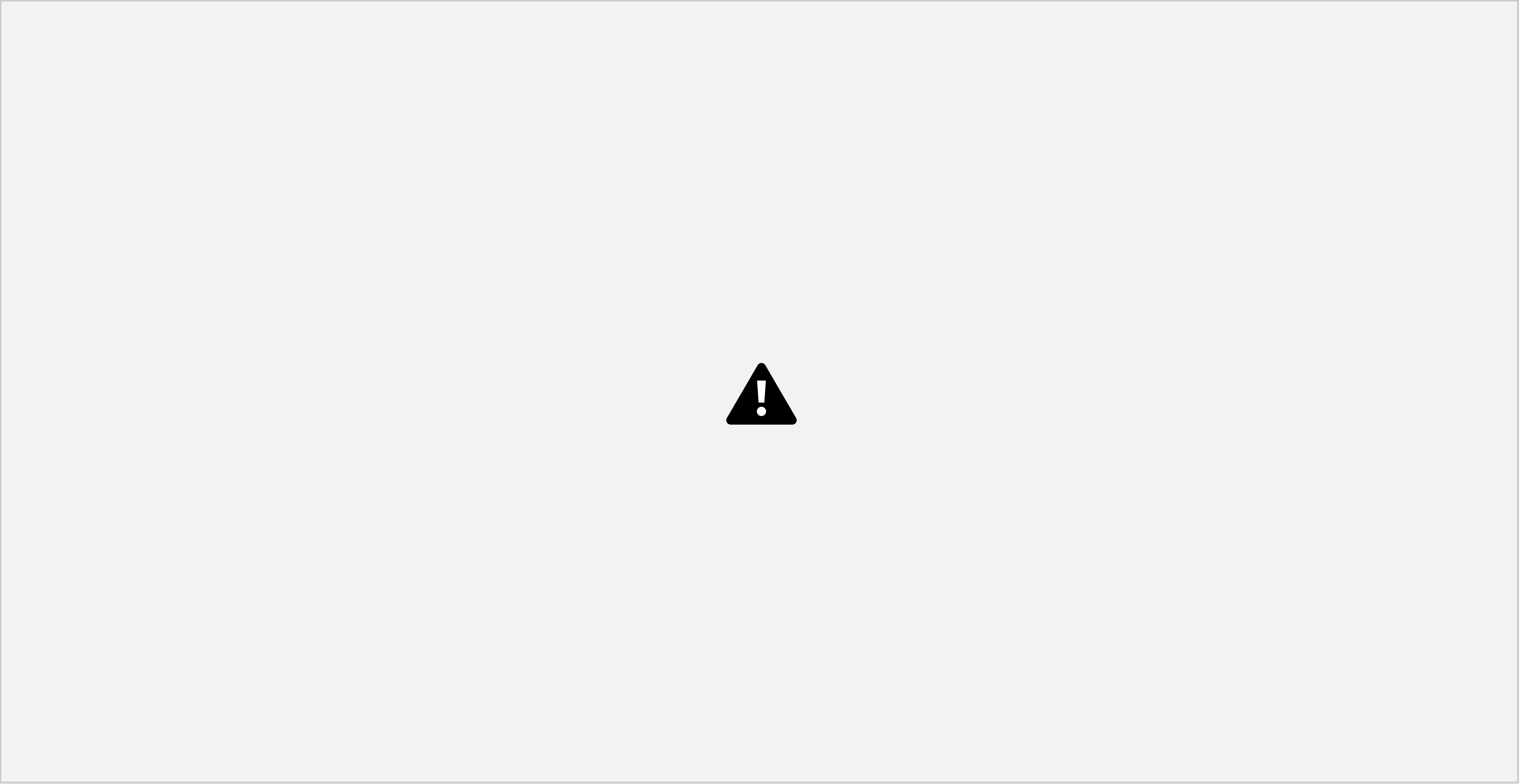
- Depth first search – only considers nodes along a single path at any time

α = highest-value choice we have found at any choice point along the path for MAX

β = lowest-value choice we have found at any choice point along the path for MIN

- update values of α and β during search and prunes remaining branches as soon as the value is known to be worse than the current α or β value for MAX or MIN

Minimax without pruning



Alpha-beta pruning example

F

G H I J

Unevaluated nodes

Effectiveness of Alpha-Beta Search

- Worst-Case
  - branches are ordered so that no pruning takes place. In this case alpha-beta gives no improvement over exhaustive search
- Best-Case
  - each player’s best move is the left-most alternative (i.e., evaluated first)
  - in practice, performance is closer to best rather than worst-case

- In practice often get $O(b^{(d/2)})$ rather than $O(b^d)$ – this

  is the same as having a branching factor of sqrt(b), •

  since $(\text{sqrt}(b))^d = b^{(d/2)}$

  - i.e., we have effectively gone from b to square root of b

  – e.g., in chess go from b ~ 35 to b ~ 6

  - this permits much deeper search in the same amount of time

Final Comments about Alpha-Beta

Pruning • Pruning does not affect final

results

- Entire subtrees can be pruned.

- Good move *ordering* improves effectiveness of pruning

Iterative (Progressive) Deepening

- In real games, there is usually a time limit T on making a move

- In practice, iterative deepening search (IDS) is used – IDS runs depth-first search with an increasing depth-limit – when the clock runs out we use the solution found at the previous depth limit

The State of Play

- Checkers:
  - Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994.
- Chess:
  - Deep Blue defeated human world champion Garry Kasparov in a six-game match in 1997.
- Othello:
  - human champions refuse to compete against computers: they are too good.
- Go:
  - human champions refuse to compete against computers: they are

too bad

– b > 300 (!)

Deep Blue

- 1957: Herbert Simon
  - "within 10 years a computer will beat the world chess champion"
- 1997: Deep Blue beats Kasparov
- Parallel machine with 30 processors for "software" and 480 VLSI processors for "hardware search"
- Searched 126 million nodes per second on average
  - Generated up to 30 billion positions per move
  - Reached depth 14 routinely
- Uses iterative-deepening alpha-beta search with transpositioning
  - Can explore beyond depth-limit for interesting moves

Example :- Complete game tree for Nim-7

- ✓ 7 coins are placed on a table between the two opponents
- ✓ A move consists of dividing a pile of coins into two nonempty piles of different sizes
- ✓ For example, 6 coins can be divided into piles of 5 and 1 or 4 and 2, but not 3 and 3
- ✓ The first player who can no longer make a move loses the game

P1

P2

P1

P2 loses

P1 starts

P2 replies

P2 losesP1 loses

MIN vs. MAX in a Nim game

The best that MIN (Player1) can do is to lose unless Player2 makes a mistake.

moves first to minimize

to maximize

Node score = 0 means MIN wins.

1 means MAX wins.

Bold edges indicate forced win for MAX, Player2.

MIN wins

MAX wins

What should be the winning statergy ?

MIN wins

Summary

- Game playing can be effectively modeled as a search problem
- Game trees represent alternate computer/opponent moves
- Evaluation functions estimate the quality of a given board configuration for the Max player.
- Minimax is a procedure which chooses moves by assuming that the opponent will always choose the move which is best for them

- Alpha-Beta is a procedure which can prune large parts of the search tree and allow search to go deeper

- For many well-known games, computer algorithms based on heuristic search match can out-perform human world experts.