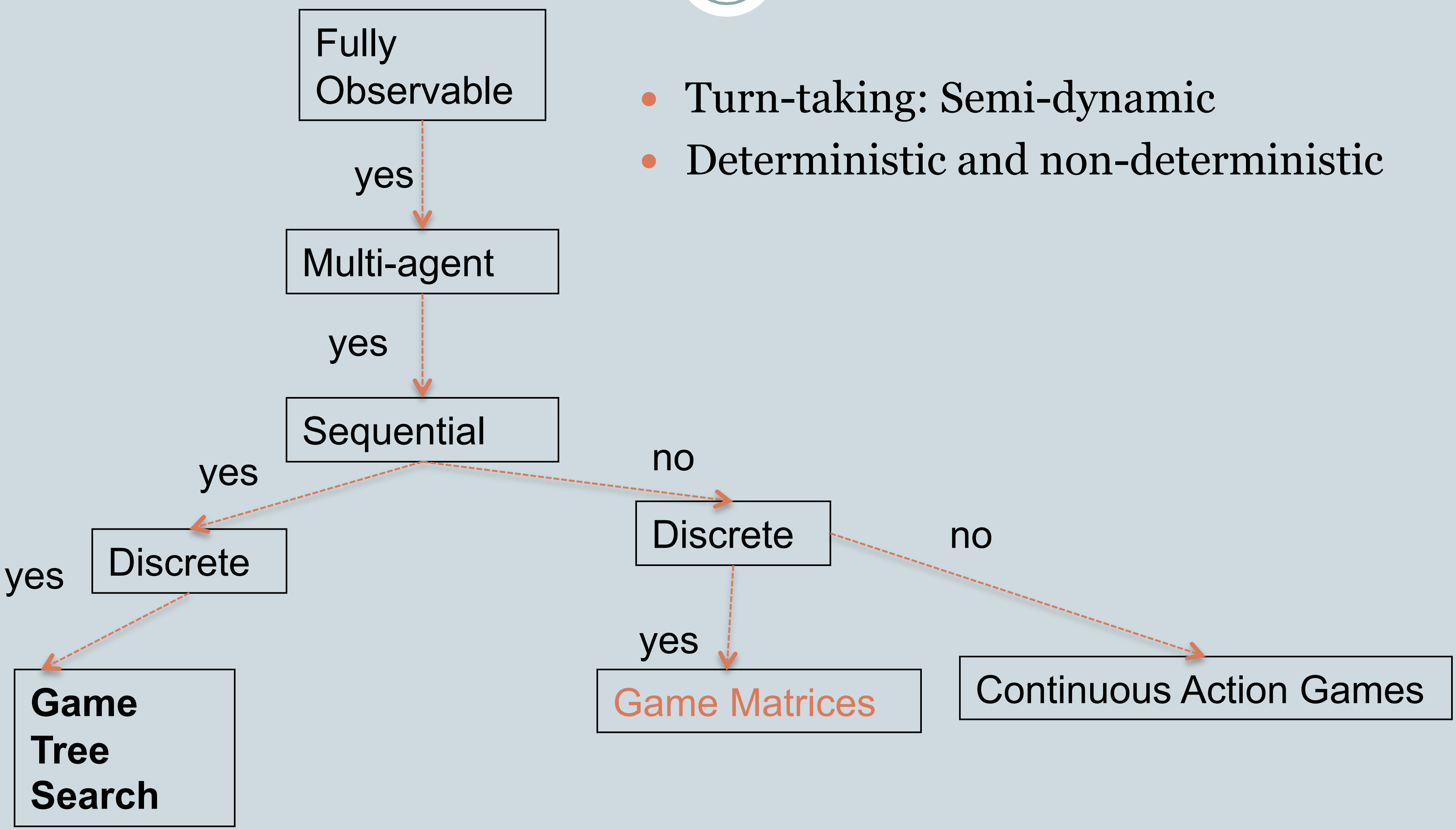
Adversarial Search and Game-Playing

CHAPTER 5
CMPT 310: Summer 2011
Oliver Schulte

Environment Type Discussed In this Lecture

- Turn-taking: Semi-dynamic
- Deterministic and non-deterministic



Adversarial Search

- Examine the problems that arise when we try to plan ahead in a world where other agents are planning against us.
- A good example is in board games.
- Adversarial games, while much studied in AI, are a small part of game theory in economics.

Typical AI assumptions

- 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: Zero-sum games of perfect information.
- We'll relax these assumptions later.

Search versus Games

- Search – no adversary
  - Solution is (heuristic) method for finding goal
  - Heuristic techniques 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).
  - **Optimality depends on opponent.** Why?
  - Time limits force an *approximate* solution
  - Evaluation function: evaluate "goodness" of game position
  - Examples: chess, checkers, Othello, backgammon

Types of Games

	deterministic	Chance moves
Perfect information	Chess, checkers, go, othello	Backgammon, monopoly
Imperfect information (Initial Chance Moves)	Bridge, Skat	Poker, scrabble, blackjack

- on-line backgammon
- on-line chess
- tic-tac-toe

- **Theorem of Nobel Laureate Harsanyi:** Every game with chance moves during the game has an equivalent representation with initial chance moves only.
- A deep result, but computationally it is more tractable to consider chance moves as the game goes along.
- This is basically the same as the issue of full observability + nondeterminism vs. partial observability + determinism.

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.

Size of search trees

- b = branching factor

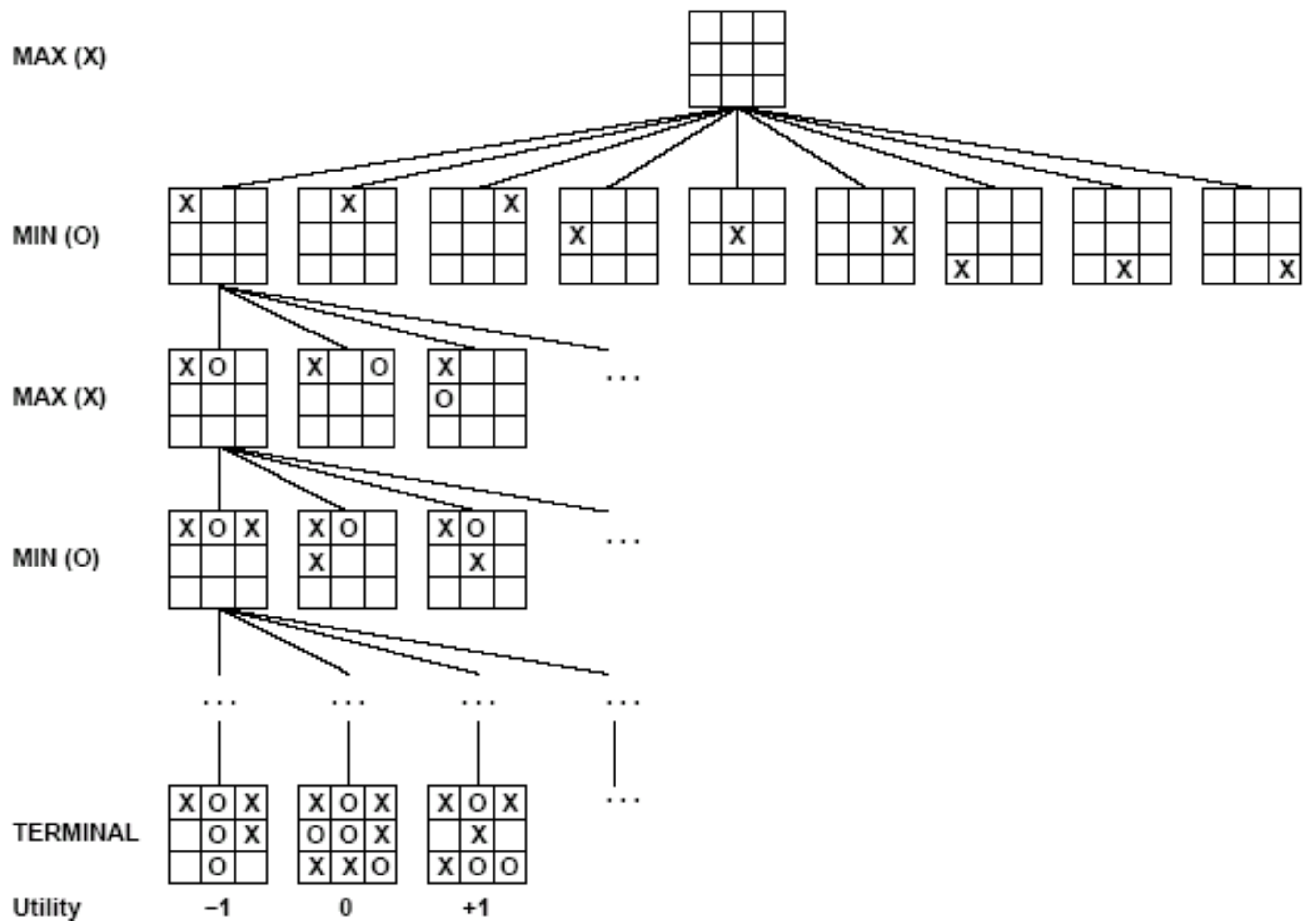
- d = number of moves by both players

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

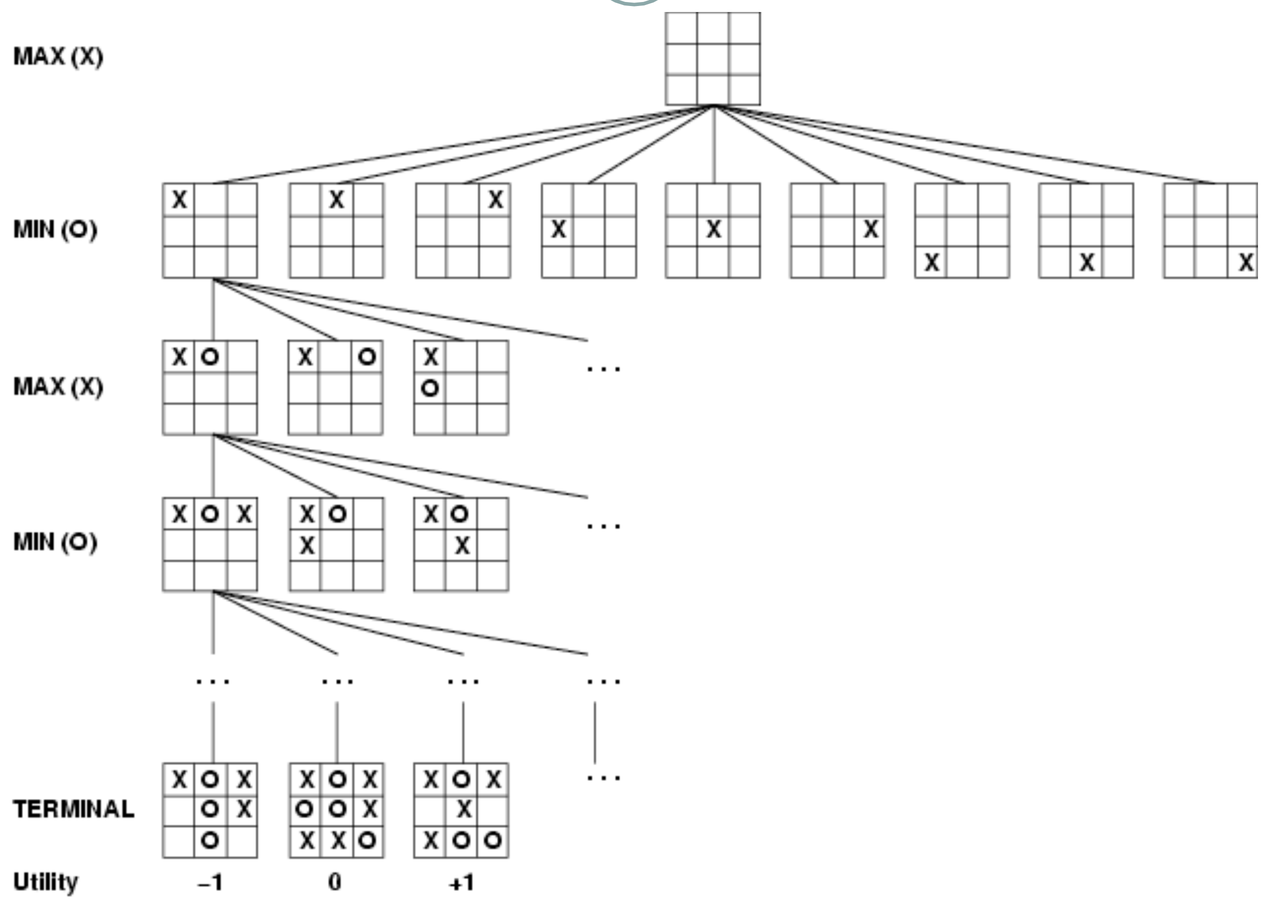
- Chess
  - b ~ 35
  - D ~100
    - search tree is ~ 10^{154} (!!)
    - completely impractical to search this

- 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

Partial Game Tree for Tic-Tac-Toe



Game tree (2-player, deterministic, turns)



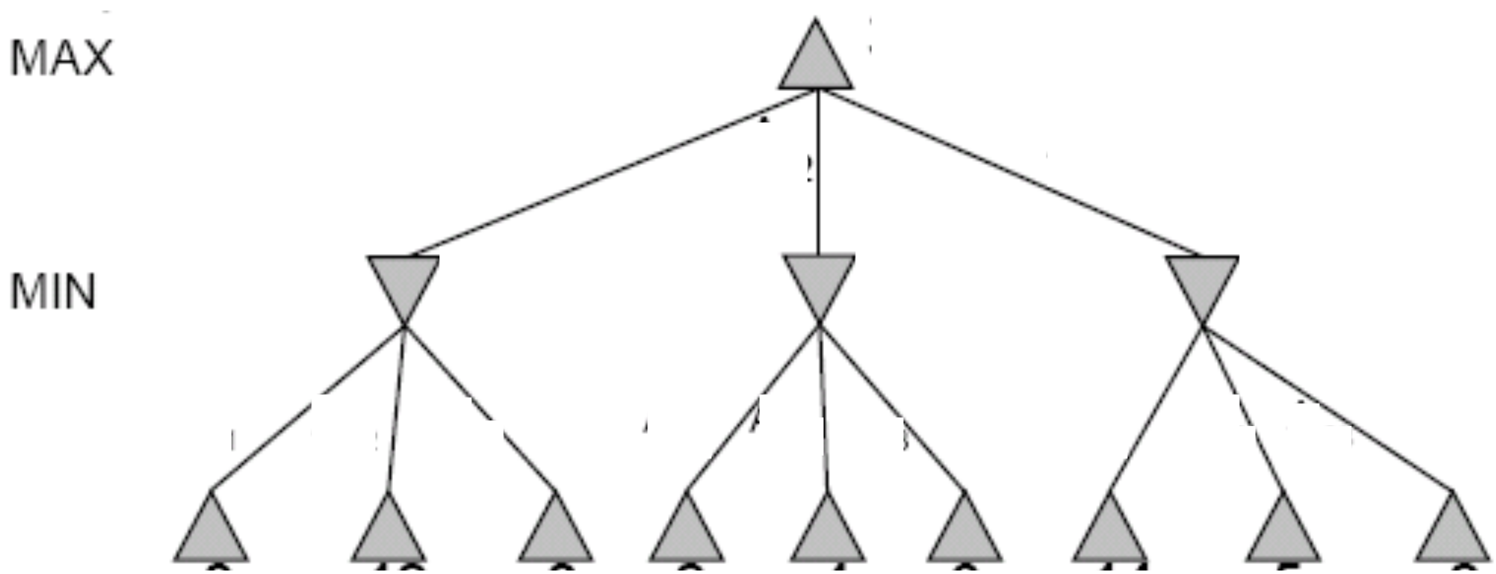
How do we search this tree to find the optimal move?

Minimax strategy: Look ahead and reason backwards

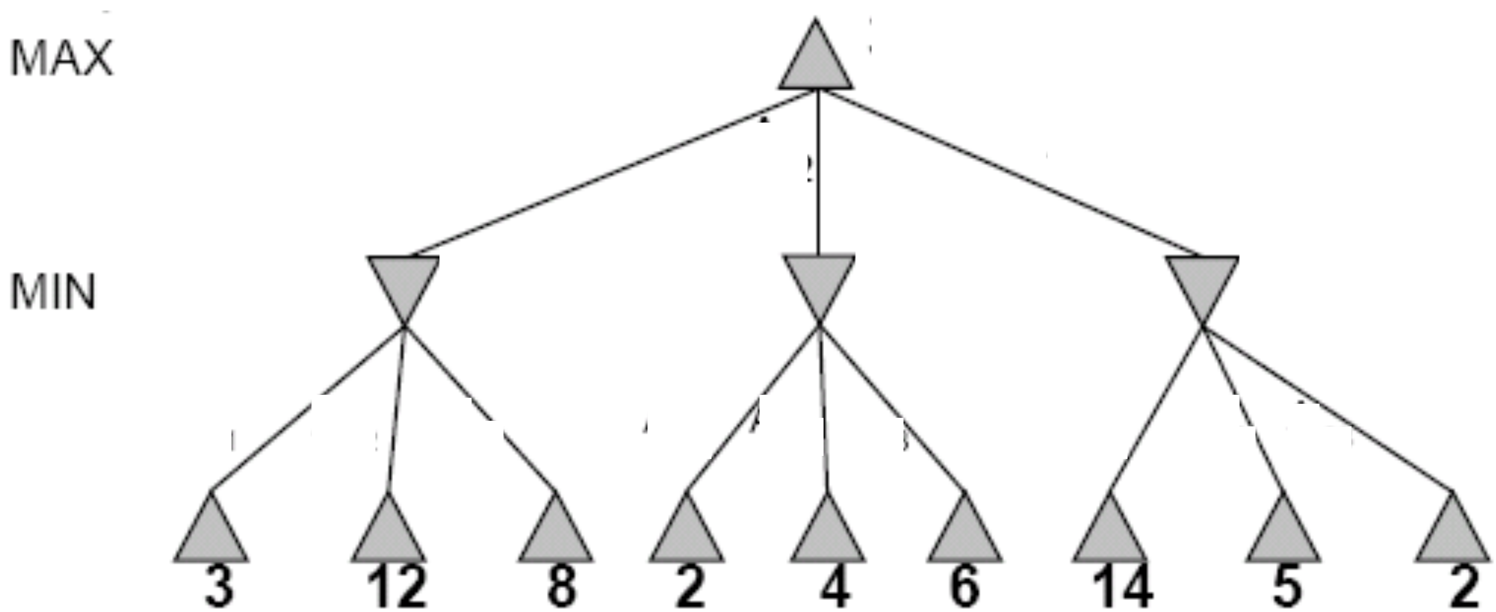
- Find the optimal *strategy* for MAX assuming an infallible MIN opponent
  - Need to compute this all the down the tree
  - Game Tree Search Demo

- Assumption: Both players play optimally!
- Given a game tree, the optimal strategy can be determined by using the **minimax value of each node.**
- Zermelo 1912.

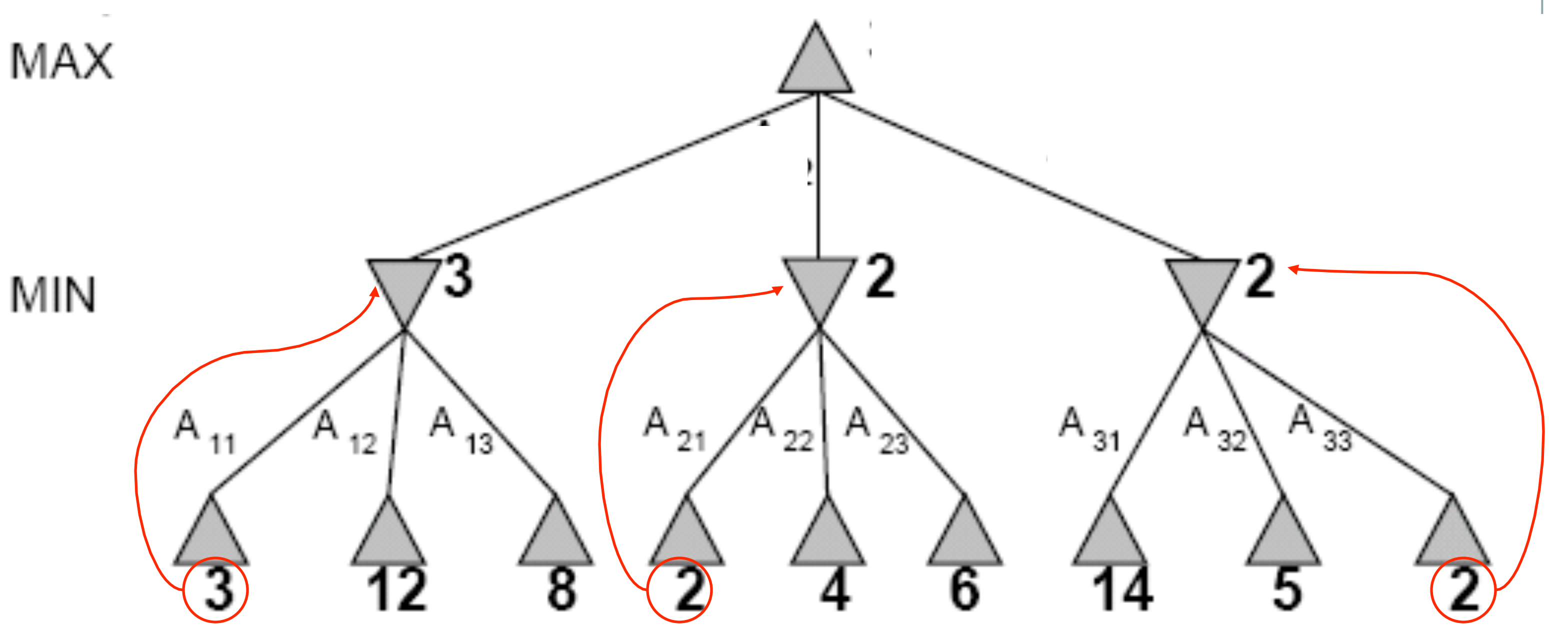
Two-Ply Game Tree



Two-Ply Game Tree

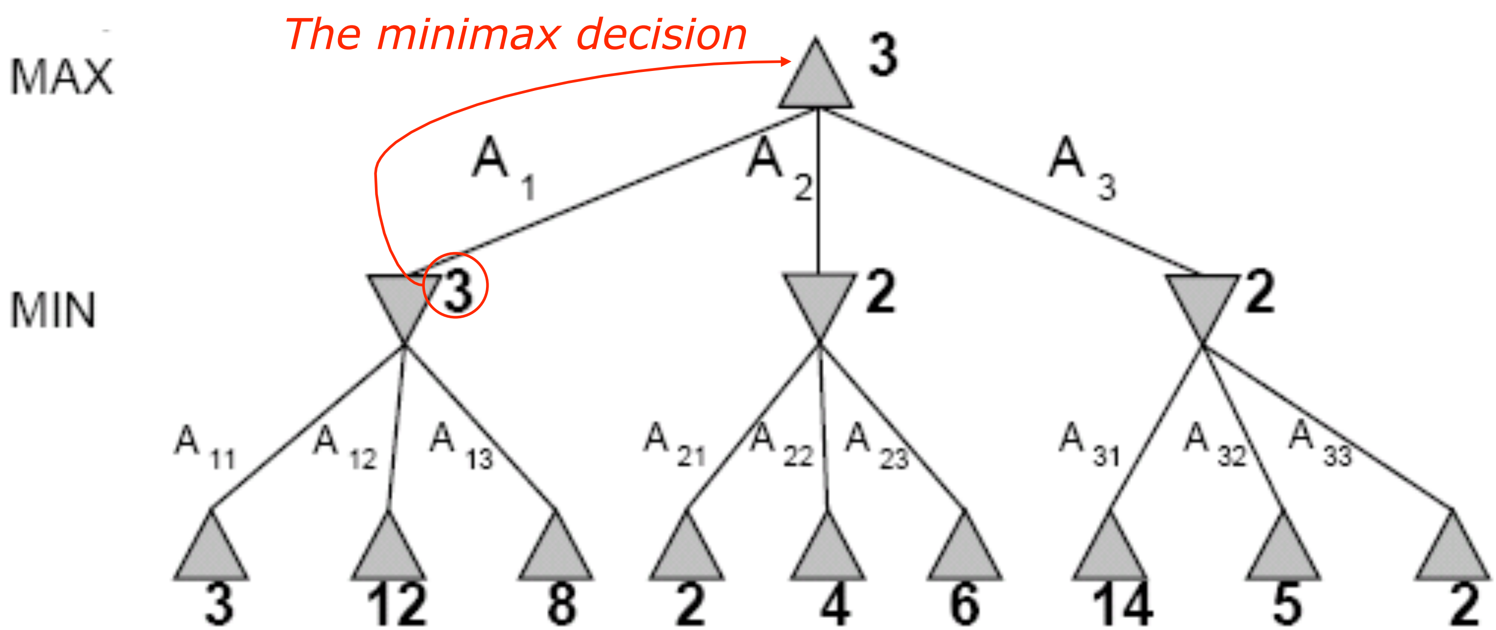


Two-Ply Game Tree



Two-Ply Game Tree

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



Pseudocode for Minimax Algorithm

function MINIMAX-DECISION(*state*) **returns** *an action*
  inputs: *state*, current state in game
  v←MAX-VALUE(*state*)
  return the *action* in SUCCESSORS(*state*) with value *v*

---

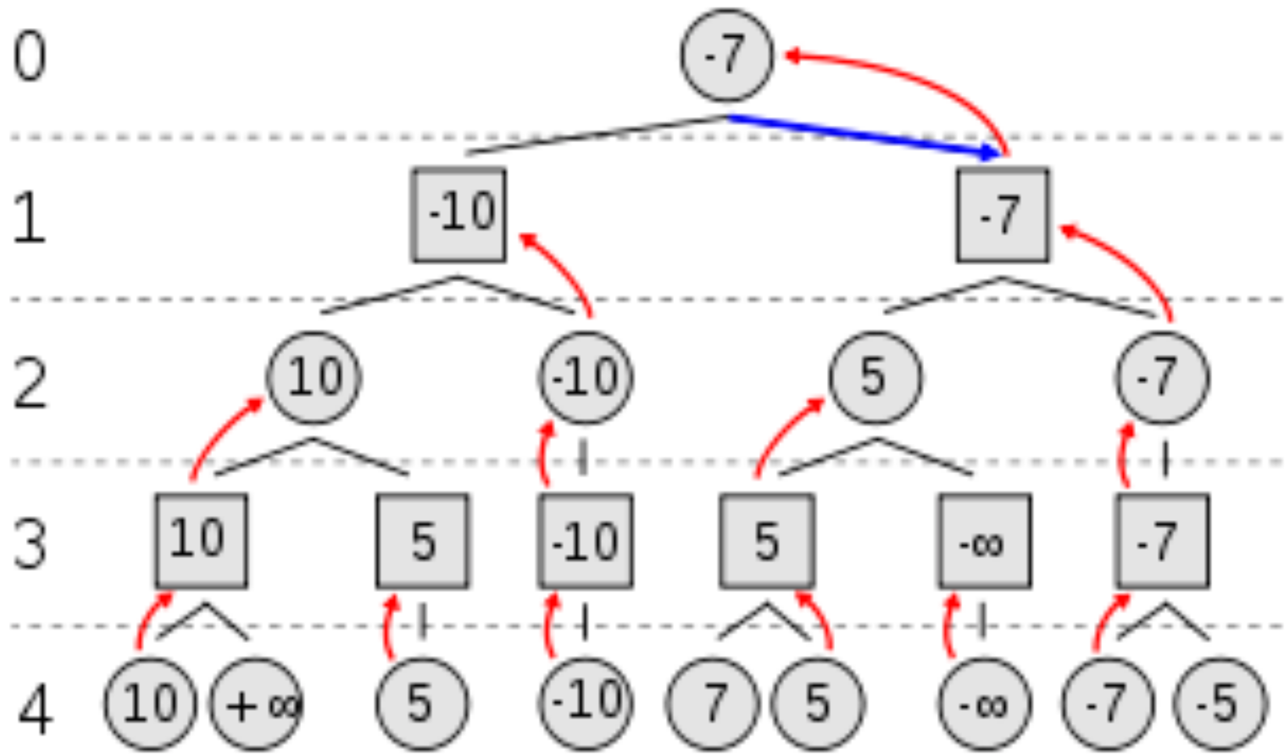
function MAX-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow -\infty$
  for *a,s* in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MAX(*v*,MIN-VALUE(*s*))
  return *v*

---

function MIN-VALUE(*state*) **returns** *a utility value*
  if TERMINAL-TEST(*state*) **then return** UTILITY(*state*)
  $v \leftarrow \infty$
  for *a,s* in SUCCESSORS(*state*) **do**
    $v \leftarrow$ MIN(*v*,MAX-VALUE(*s*))
  return *v*

Example of Algorithm Execution

MAX to move



Minimax Algorithm

- Complete depth-first exploration of the game tree
- Assumptions:
  - Max depth = d, b legal moves at each point
  - E.g., Chess: d ~ 100, b ~35

Criterion	Minimax
Time ☹	$O(b^d)$
Space ☺	$O(bd)$

Practical problem with minimax search

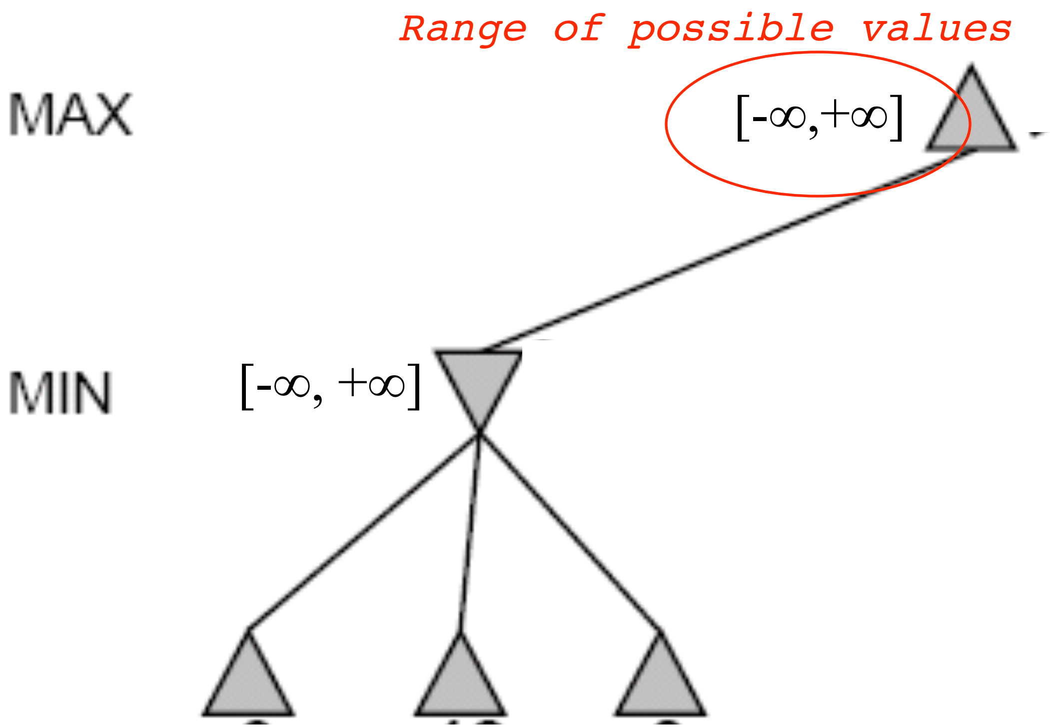
- Number of game states is exponential in the number of moves.
  - Solution: Do not examine every node

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

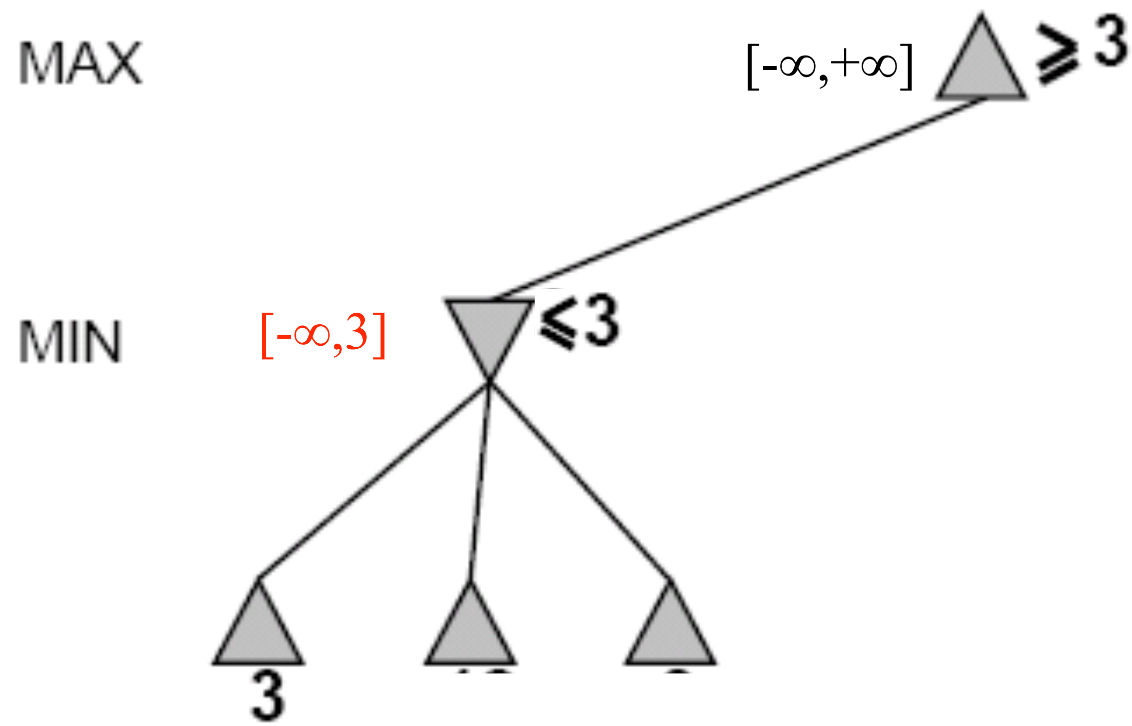
- Revisit example …

Alpha-Beta Example

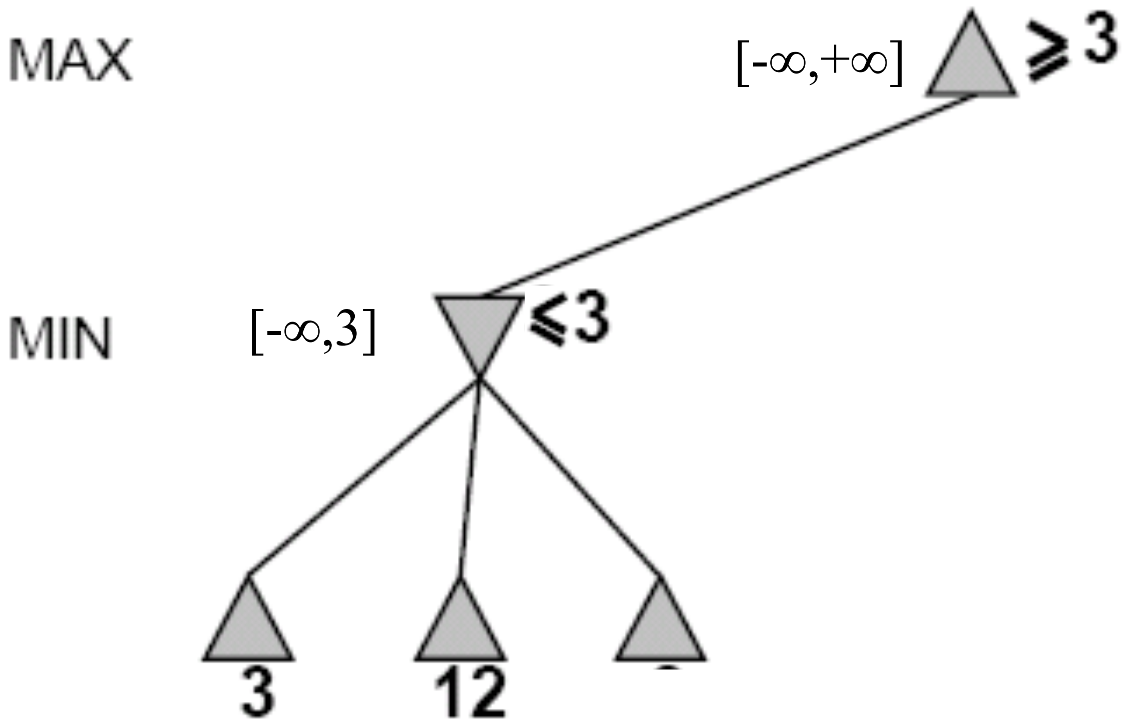
Do DF-search until first leaf



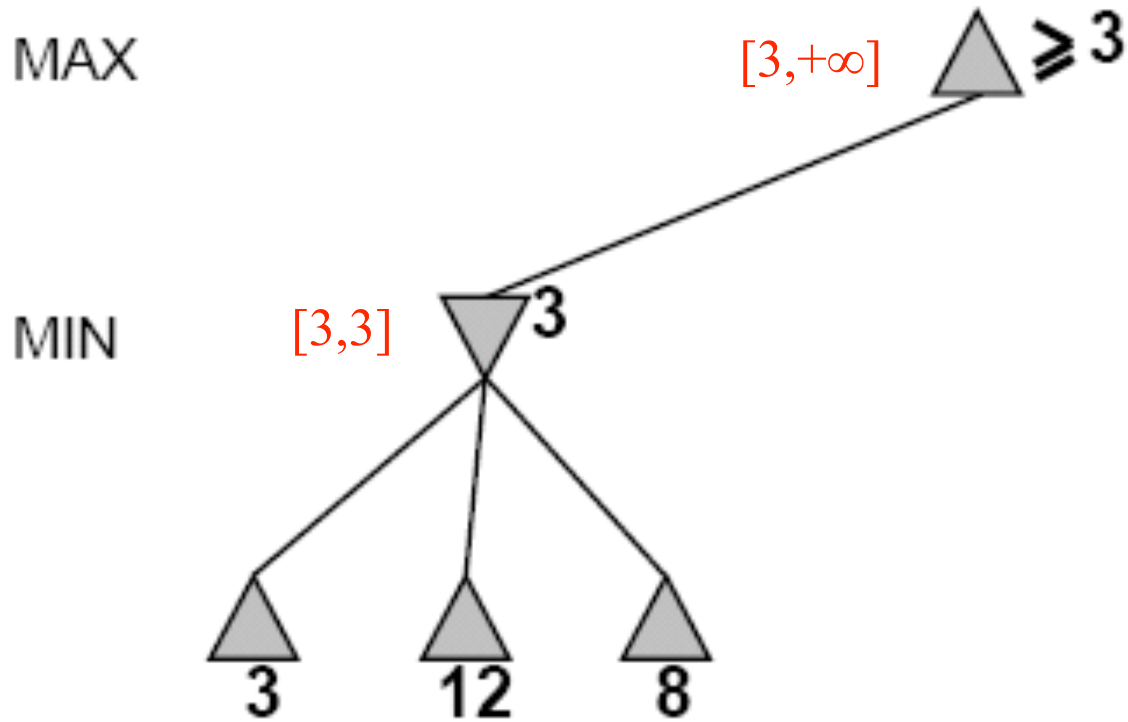
Alpha-Beta Example (continued)



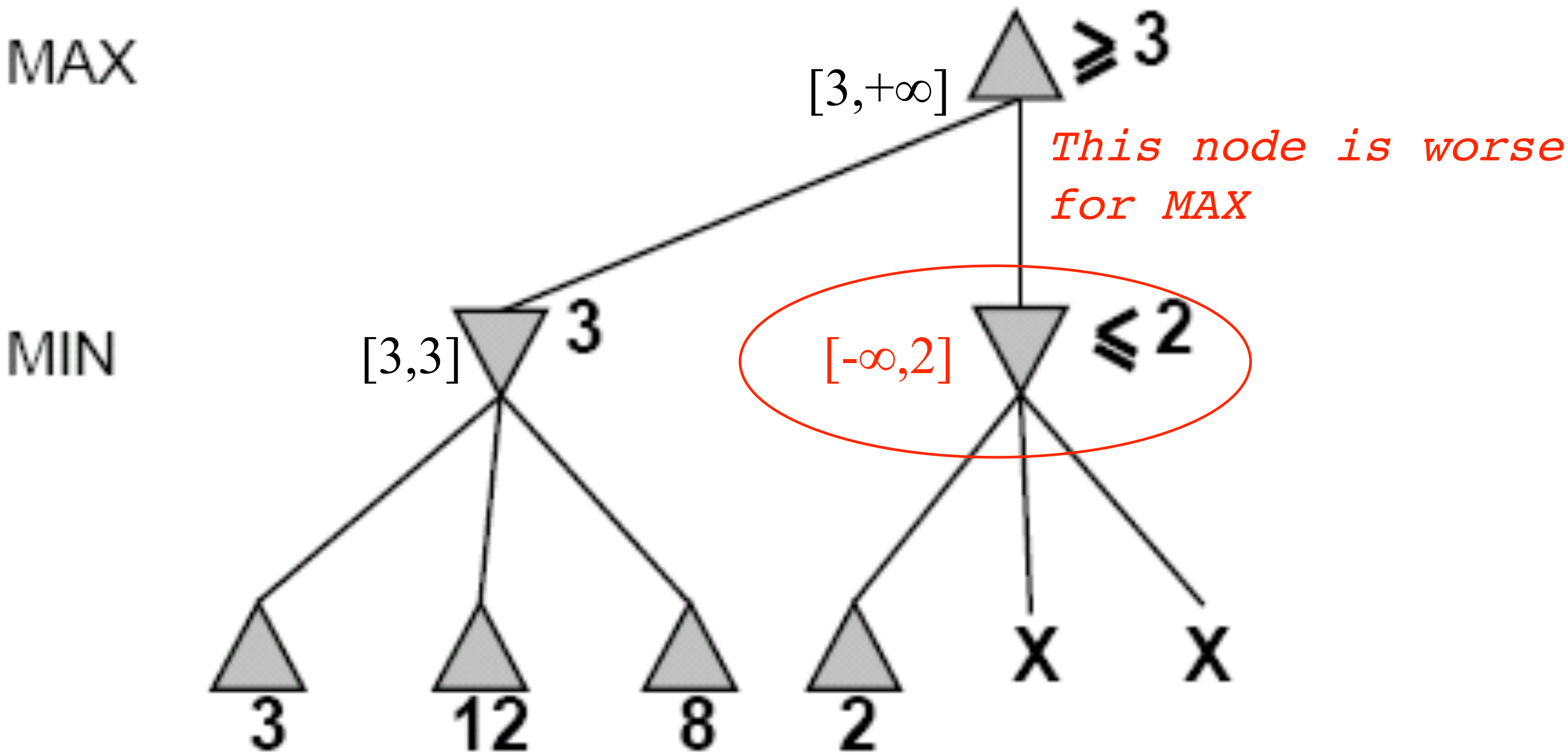
Alpha-Beta Example (continued)



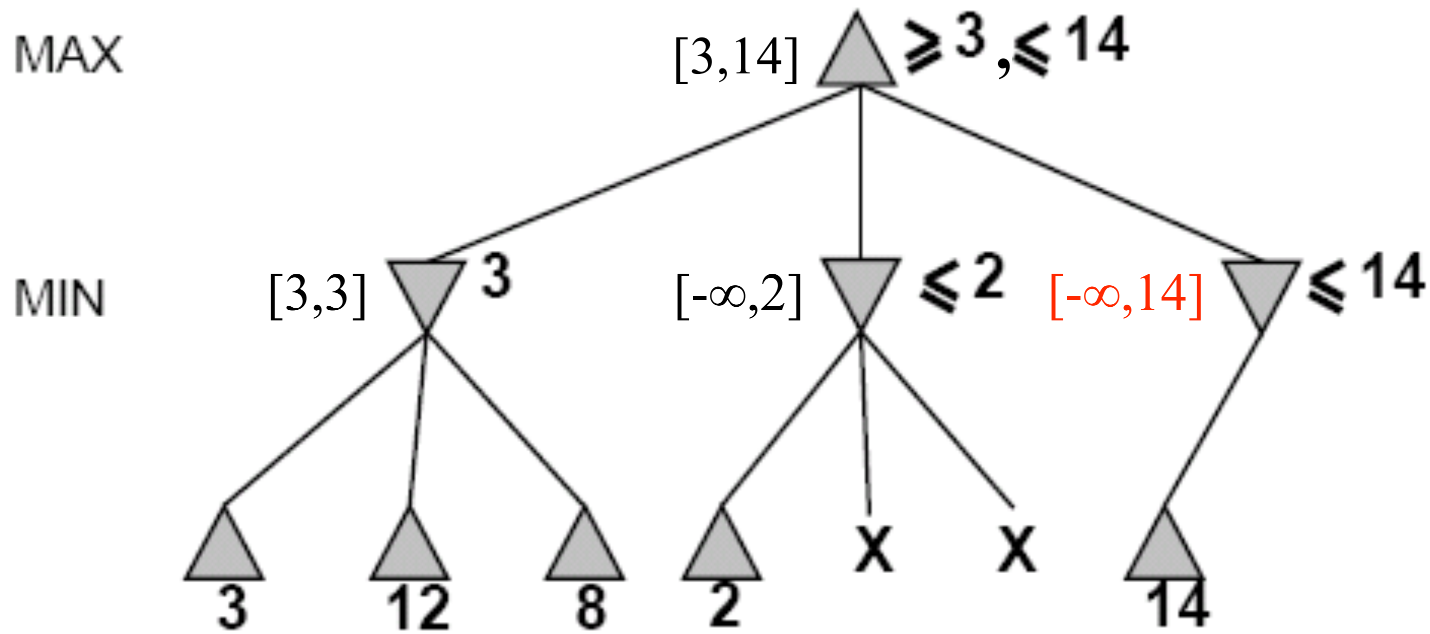
Alpha-Beta Example (continued)



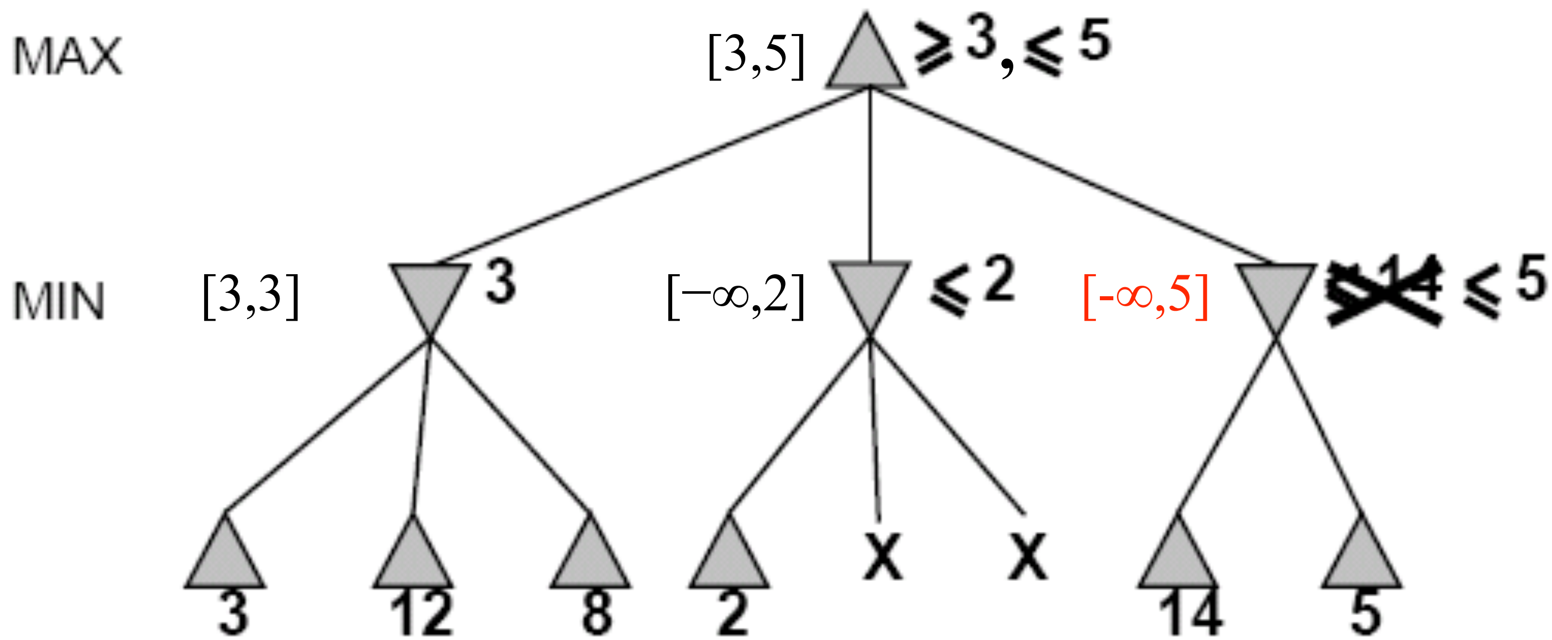
Alpha-Beta Example (continued)



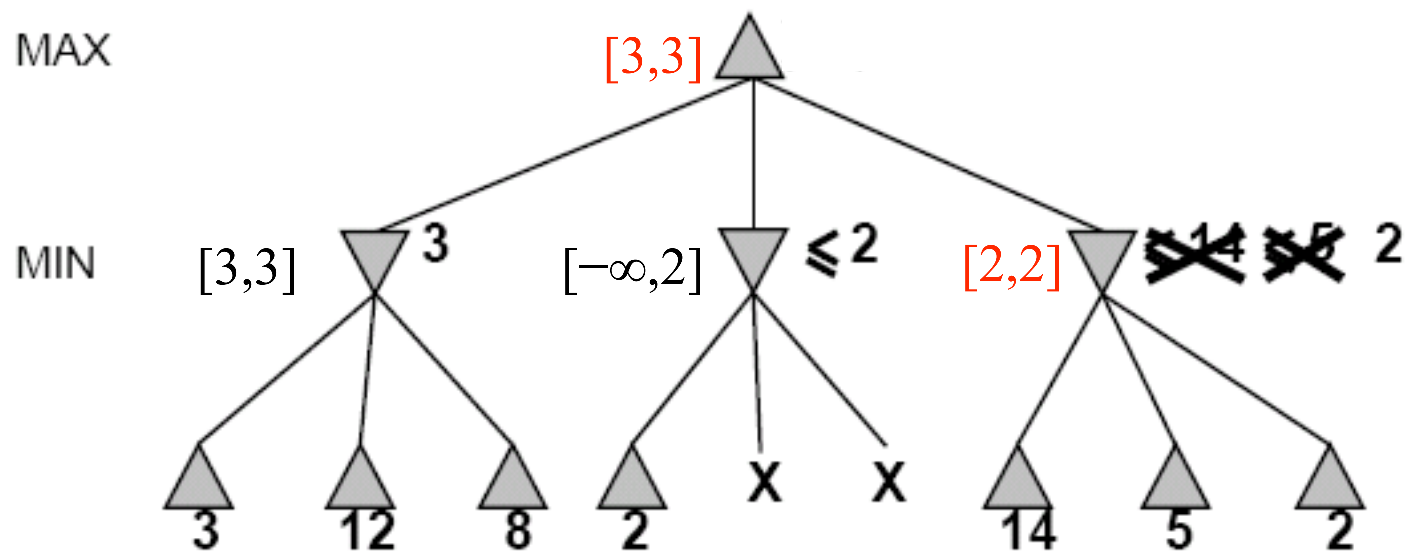
Alpha-Beta Example (continued)



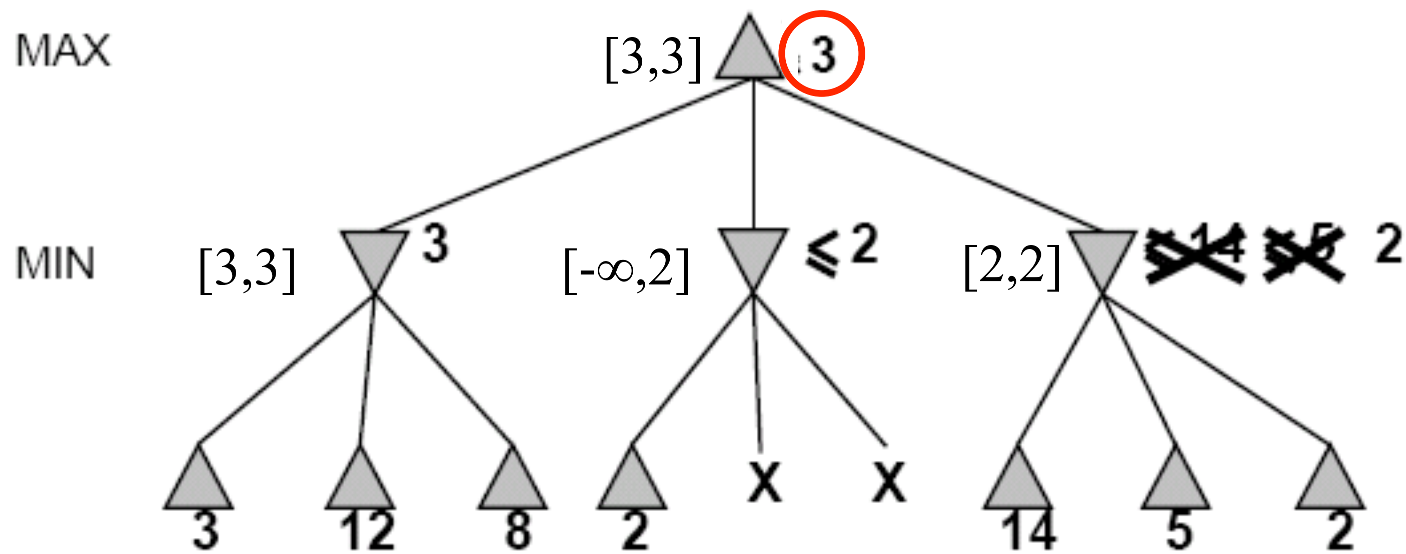
Alpha-Beta Example (continued)



Alpha-Beta Example (continued)



Alpha-Beta Example (continued)



Alpha-beta Algorithm

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

α = highest-value choice that we can guarantee for MAX so far in the current subtree.

β = lowest-value choice that we can guarantee for MIN so far in the current subtree.

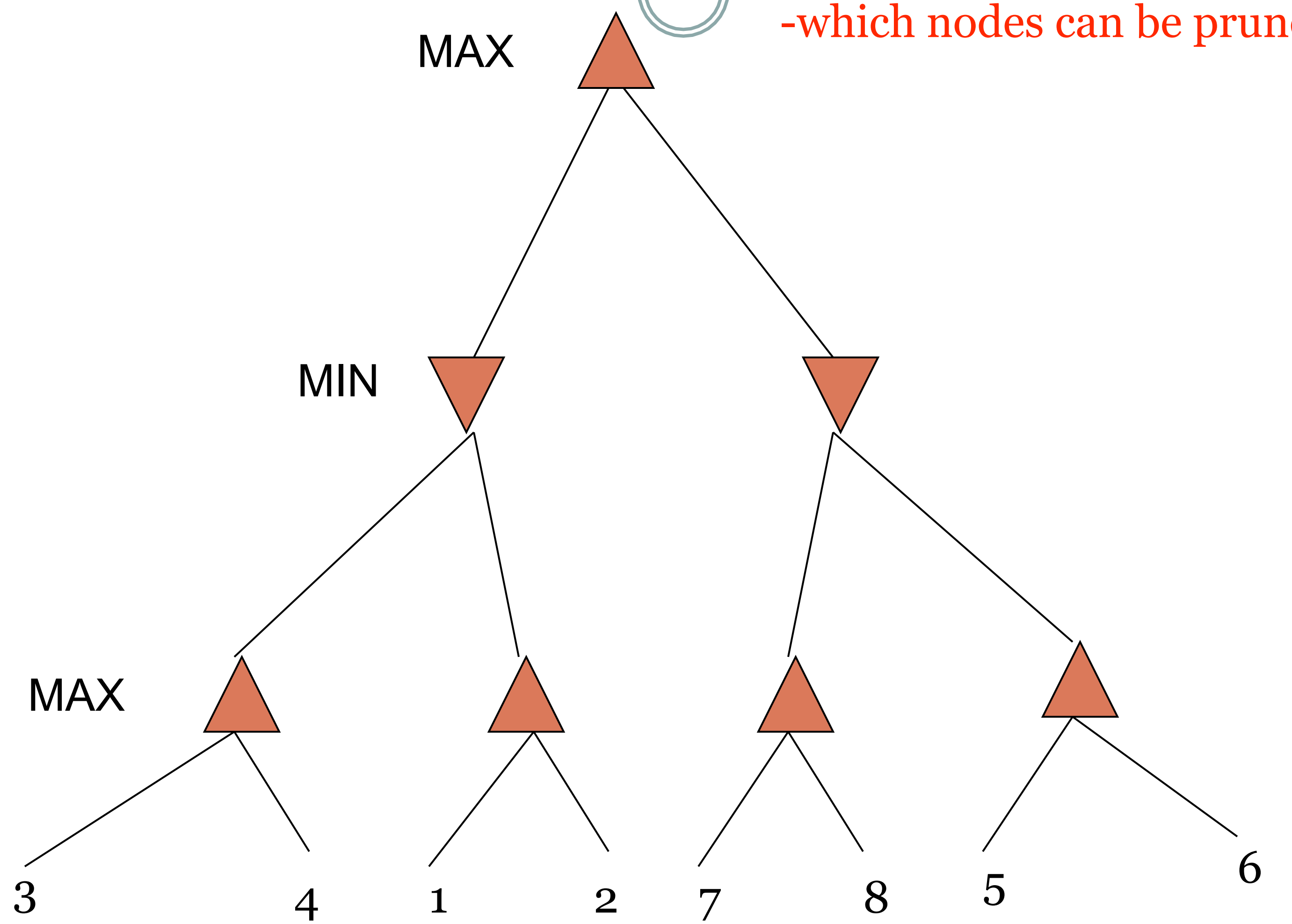
- update values of α and β during search and prunes remaining branches as soon as the value is known to be worse than the current α or β value for MAX or MIN.
- Alpha-beta Demo.

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 $(\mathrm{sqrt}(b))^d = b^{(d/2)}$
    - 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
    - Typically twice as deep.

Example

-which nodes can be pruned?



Final Comments about Alpha-Beta Pruning

- Pruning does not affect final results
- Entire subtrees can be pruned.
- Good move *ordering* improves effectiveness of pruning
- Repeated states are again possible.
  - Store them in memory = transposition table

Practical Implementation

How do we make these ideas practical in real game trees?

Standard approach:

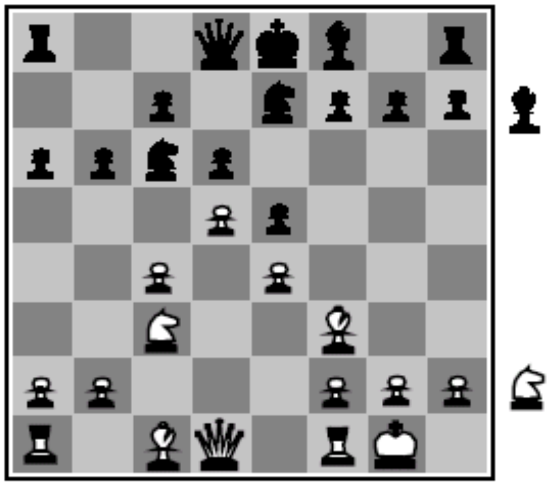
- cutoff test: (where do we stop descending the tree)
  - depth limit
  - better: iterative deepening
  - cutoff only when no big changes are expected to occur next (quiescence search).
- evaluation function
  - When the search is cut off, we evaluate the current state by estimating its utility using **an evaluation function**.

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

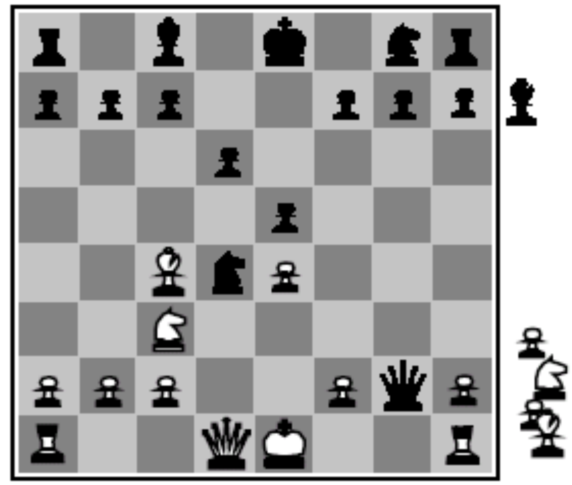
- Typical values from -infinity (loss) to +infinity (win) or [-1, +1].
- If the board evaluation is X for a player, it's -X for the opponent.
- Many clever ideas about how to use the evaluation function.
  - e.g. null move heuristic: let opponent move twice.
- Example:
  - Evaluating chess boards,
  - Checkers
  - Tic-tac-toe

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.

Iterative (Progressive) Deepening

- In real games, there is usually a time limit T on making a move
- How do we take this into account?
  - using alpha-beta we cannot use "partial" results with any confidence unless the full breadth of the tree has been searched
  - So, we could be conservative and set a conservative depth-limit which guarantees that we will find a move in time < T
    - disadvantage is that we may finish early, could do more search
- 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 $b > 300$ (!)
- See (e.g.) http://www.cs.ualberta.ca/~games/ for more information


U of A GAMES Group Home Page - Mozilla Firefox
File Edit View History Bookmarks Tools Help
http://www.cs.ualberta.ca/~games/
Web of Science Gmail - Inbox Smyth CS175 CS271 UCI Soccer News CML Center ToDo List TopicSeek Traffic GScholar cs540 Intro to AI CLIVAR Calendar
Gmail - Inbox (1543) Gmail - Inbox (1542) Mail :: Inbox: RE: latest ... nogoods.pdf (applicatio... ToolBox: Evaluations U of A GAMES Group ... fulltext.pdf (application/...
The University of Alberta GAMES Group
Game-playing,
Analytical methods,
Minimax search and
Empirical
Studies
Projects | News | What We Do | People | Links | Publications |
Announcements
Weekly GAMES group meetings are from 4-5pm on Thursdays at CSC333. You can check the schedule here.
The University of Alberta GAMES Group has an opening for a postdoctoral fellow in the area of Artificial Intelligence in Commercial (Video) Games. Check here for details.
Projects
Checkers
Chinook is the official world checkers champion.
Play
Poker
Poki is the strongest poker AI in the world.
Play
Lines of Action
YL & Mona are two of the best LoA programs in the world.
Play
Hex
Queenbee is one of the best Hex programs in the world.
Play
Go
The Computer Go group has developed two programs, Explorer and NeuroGo.
Real-Time Stategy
We are trying to apply AI to real-time stategy games.
Othello
Logistello defeated the human world Othello champion, 6-0, in 1997. Keyano is another strong program.
Sokoban
Rolling Stone pushes the boundaries of single agent search.
RoShamBo
Home of the International RoShamBo Programming Competition
Amazons
Three programs, and several theoretical contributions.
Spades & Hearts
Spades & Hearts are the test beds for the research on multi-player games.
Shogi
ISshogi has won the world computer Shogi championship many times (currently inactive).
Previous Projects:
Chess
Awari
Play
Chinese Chess
Play
Post's Correspondence Problem
Domineering
Done

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

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 or out-perform human world experts.

AI Games vs. Economics Game Theory

- Seminal Work on Game Theory:
  Theory of Games and Economic Behavior, 1944, by von Neumann and Morgenstern.

- Agents can be in **cooperation as well as in conflict.**

- Agents may move simultaneously/independently.

Example: The Prisoner's Dilemma

	Column Player	
Row Player	C	D
C	2, 2	0, 3
D	3, 0	1, 1

Other Famous Matrix Games:
- Chicken
- Battle of The Sexes
- Coordination

Solving Zero-Sum Games

• Perfect Information: Use Minimax Tree Search.
• Imperfect Information: Extend Minimax Idea with **probabilistic actions.**
⇨ von Neumann’s **Minimax Theorem**: there exists an essentially unique optimal probability distribution for randomizing an agent’s behaviour.

Matching Pennies

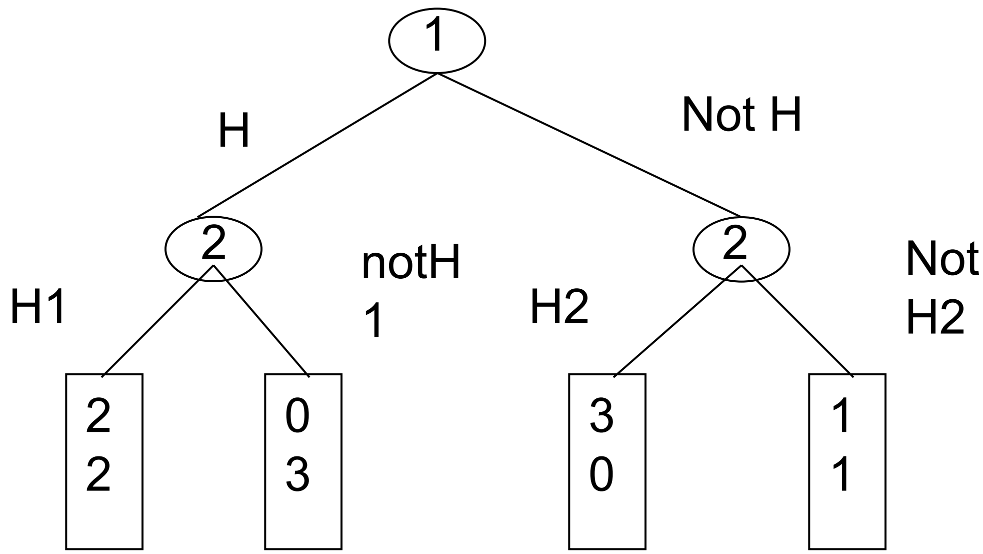
- Why should the players randomize?
- What are the best probabilities to use in their actions?

	Heads	Tails
Heads	1,-1	-1,1
Tails	-1,1	1,-1

Nonzero Sum Game Trees

- The idea of “look ahead, reason backward” works for any game tree with perfect information.
  - I.e., also in cooperative games.
- In AI, this is called **retrograde analysis**.
- In game theory, it is called **backward induction** or **subgame perfect equilibrium.**
- Can be extended to many games with imperfect information (sequential equilibrium).

Backward Induction Example: Hume's Farmer Problem



Summary: Solving Games

	Zero-sum	Non zero-sum
Perfect Information	Minimax, alpha-beta	Backward induction, retrograde analysis
Imperfect Information	Probabilistic minimax	Nash equilibrium

Nash equilibrium is beyond the scope of this course.

Single Agent vs. 2-Players

- *Every single agent problem can be considered as a special case of a 2-player game*. How?
  1. Make one of the players the Environment, with a constant utility function (e.g., always 0).
     1. The Environment acts but does not care.
  2. An adversarial Environment, with utility function the negative of agent's utility.
     1. In minimization, Environment's utility is player's costs.
     2. Worst-Case Analysis.
     3. E.g., program correctness: no matter what input user gives, program gives correct answer.
- So agent design is a subfield of game theory.

Single Agent Design = Game Theory

Von Neumann-Morgenstern Games

Decision Theory = 2-player game, 1st player the "agent", 2nd player "environment/nature"
(with constant or adversarial utility function)

Markov Decision Processes

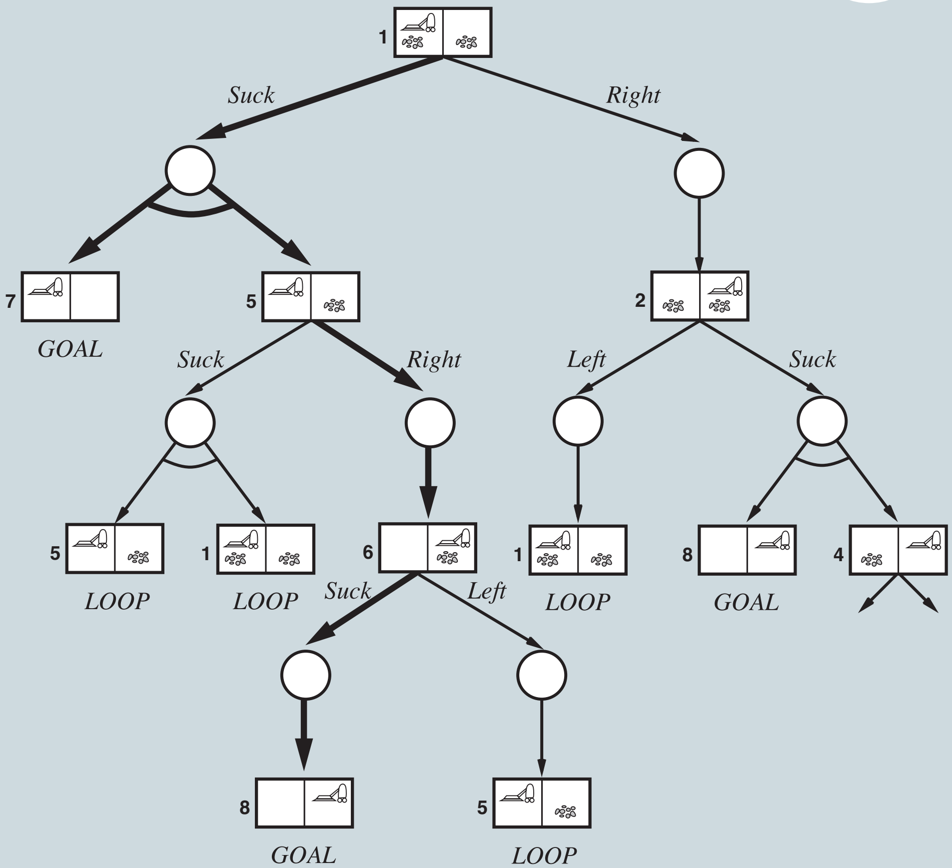
Planning Problems

From
General
To
Special
Case

Example: And-Or Trees

- If an agent's actions have nondeterministic effects, we can model worst-case analysis as a zero-sum game where the environment chooses the effects of an agent's actions.
- Minimax Search ≈ And-Or Search.
- Example: The Erratic Vacuum Cleaner.
  - When applied to dirty square, vacuum cleans it and sometimes adjacent square too.
  - When applied to clean square, sometimes vacuum makes it dirty.
  - Reflex agent: same action for same location, dirt status.

And-Or Tree for the Erratic Vacuum



- The agent "moves" at labelled OR nodes.
- The environment "moves" at unlabelled AND nodes.
- The agent wins if it reaches a goal state.
- The environment "wins" if the agent goes into a loop.

Summary

- Game Theory is a very general, highly developed framework for multi-agent interactions.
- Deep results about equivalences of various environment types.
- See Chapter 17 for more details.