

Lecture 5 and 6: Multiagent Search

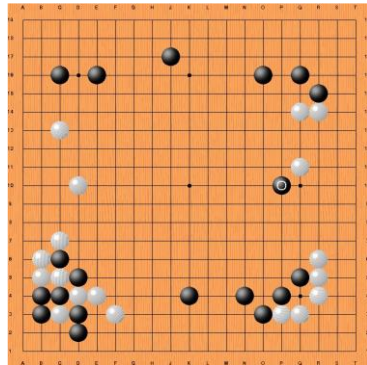
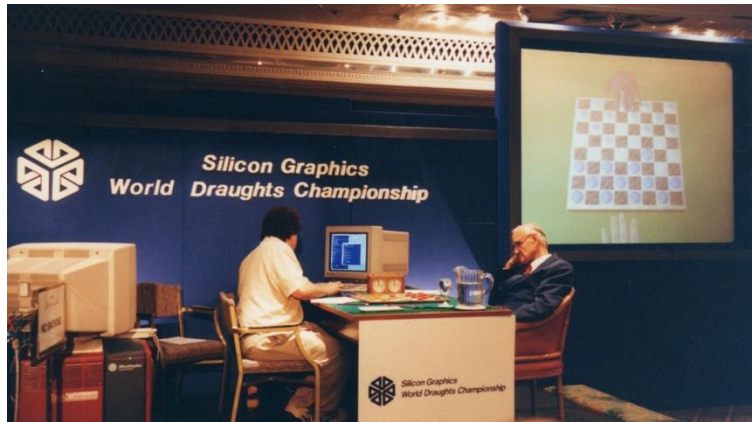
CS 486/686: Introduction to Artificial Intelligence

Plan for Today

- Games and their role in AI progress
- Basic Game Theory
- Minimax Search and Alpha-Beta Pruning
- Evaluation Functions
- Expectiminimax
- Monte Carlo Tree Search (MCTS)

Games

- Games are the oldest and, arguably, the most well-studied domain in AI
 - Chinook
 - Deep Blue
 - AlphaGo
 - Pluribus
 - Cicero



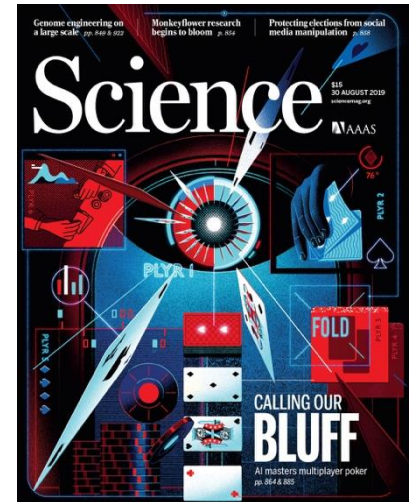
RESEARCH ARTICLE | COMPUTER SCIENCE



Human-level play in the game of *Diplomacy* by combining language models with strategic reasoning

META FUNDAMENTAL AI RESEARCH DIPLOMACY TEAM (FAIR), ANTON BAKHTIN, NOAM BROWN, EMILY DINAN, GABRIELE FARINA, COLIN FLAHERTY,

DANIEL FRIED, ANDREW GOFF, JONATHAN GRAY, [...], AND MARKUS ZIJLSTRA, +17 authors [Authors Info & Affiliations](#)



Games

- And they continue to be relevant today

Game Arena

Measuring the frontier of model capabilities through dynamic game environments.

[Read Our Blog](#)

[? Q&A](#)



#	Replay	Model	↓ Equilibrium Rating	Avg. Tokens/Turn	Avg. Cost/Turn
1		GPT-5.4	0.08% ±0.07%	6599	4.38¢
2		Gemini 3.1 Pro Preview	-0.38% ±0.31%	5866	2.59¢
3		Gemini 3 Flash Preview	-1.15% ±0.43%	5827	0.67¢
4		Claude Opus 4.6	-1.25% ±0.59%	6416	4.22¢
5		Claude Sonnet 4.6	-2.15% ±0.59%	6054	2.37¢

Why Games?

- There are many reasons why games have been some prominent in AI progress
 - They are easy to represent and the rules are clear (e.g. “simple environments”)
 - State spaces can be very large
 - In chess, the search tree has $\sim 10^{154}$ nodes
 - Like in the “real world”, decisions have to be made, often quickly
 - Easy to evaluate progress (winning is usually well defined)
 - They are fun!

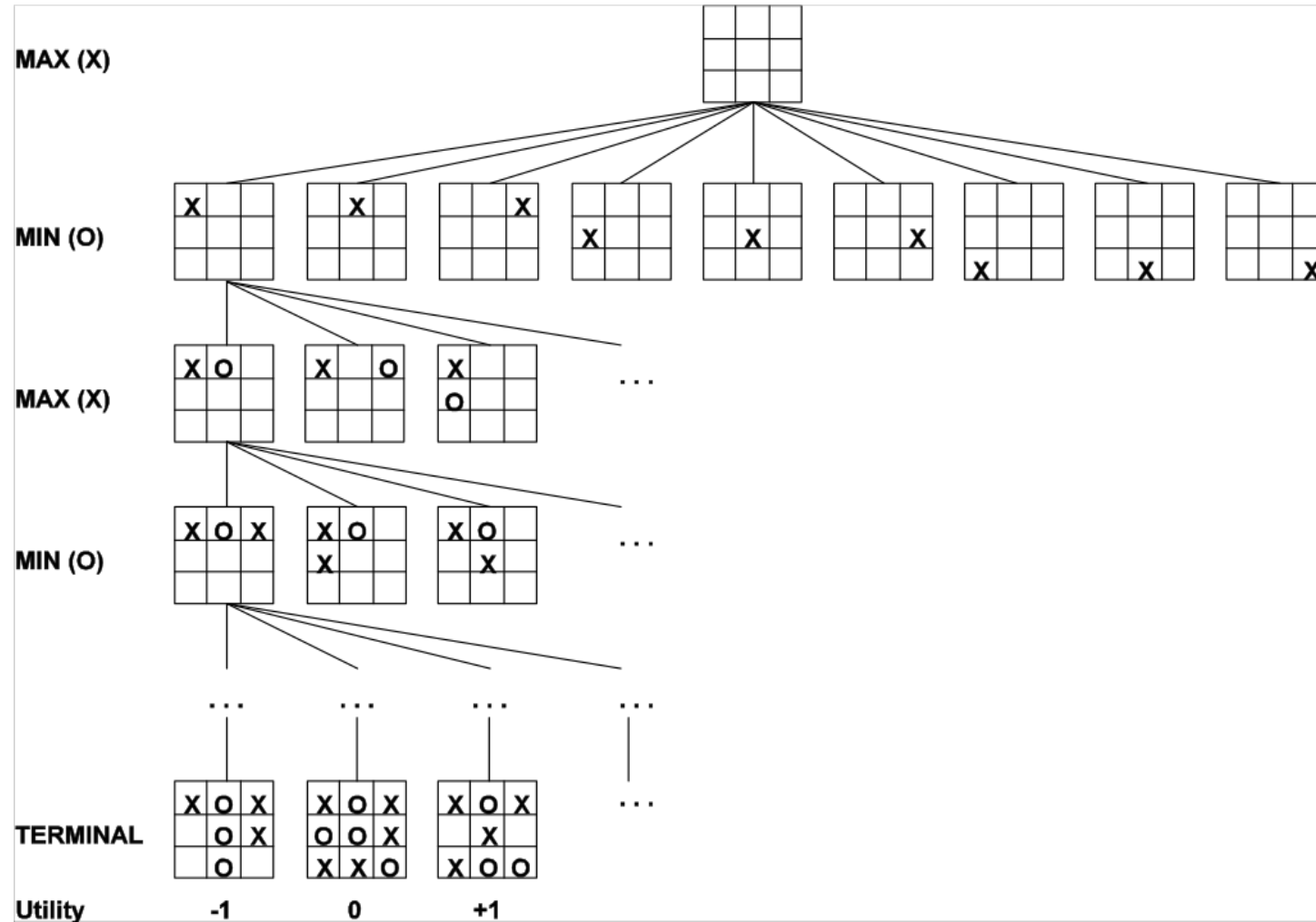
Types of Games

- We distinguish between types of games, just like we do with environments (you can think of games as environments)
- Perfect vs Imperfect Information
 - Perfect information: players see the entire state of the game
 - Imperfect information: they don't
- Deterministic vs Stochastic
 - Deterministic: change in state is fully controlled by players
 - Stochastic: change in state is partially determined by chance

Game Search Challenge

- What makes game search challenging?
 - There is an opponent
 - The opponent is malicious (it wants to win, by making you lose)
- We need to take this into account when choosing moves/actions
- Notation:
 - MAX player wants to maximize its utility
 - MIN player wants to minimize MAXs utility

Example



MAX's job is to use the search tree to determine the best move

What does it mean to be “optimal”?

- In standard search
 - The optimal solution is a sequence of moves leading to a goal state
- In games, we talk about **strategies**
 - From MAX's perspective
 - Specify a move from the initial state
 - Specify a move for all possible states arising from MIN's response
 - Then all possible responses to MIN's response to MAX's previous move
 -

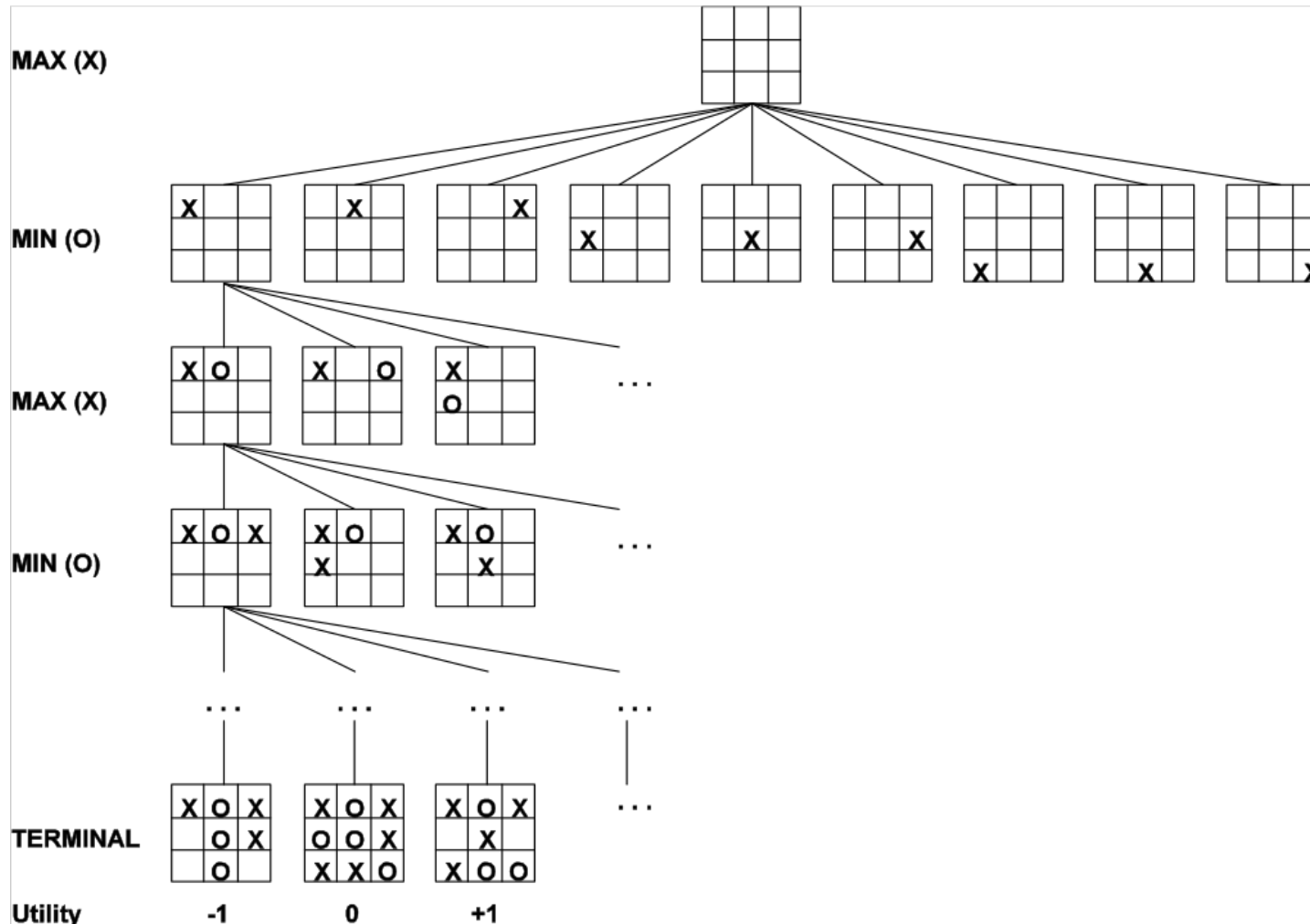
What does it mean to be optimal?

- An optimal strategy is one that leads to outcomes at least as good as any other strategy, *assuming that MIN is playing optimally*
 - Equilibrium (game theory)
- Today we focus on zero-sum games of perfect information
 - “Easy games” according to game theory

Game Theory Detour

- Game theory is a formal way of reasoning about interactions between multiple agents
 - To define a game we need the following components
 - The players, N
 - Their possible strategies (or actions) S_i
 - Their utility functions $(u_i(s_1, \dots, s_n))$
- (Nash) equilibrium: Mutual best-response
 - “given the action/strategy taken by my opponent, this is what I should do”
 - We will formally revisit these ideas later in the course

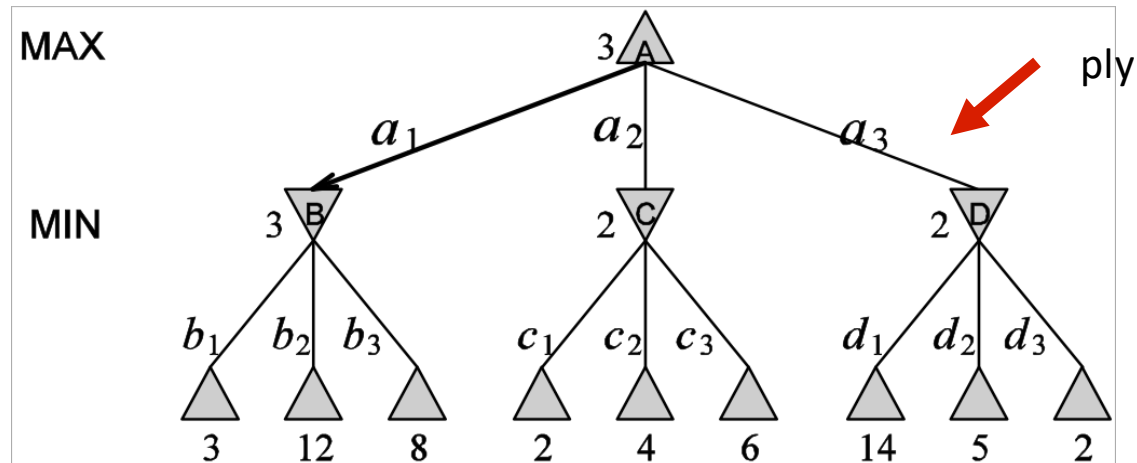
Example



Theorem (Kuhn): (Informally)
Games like Tic-Tac-Toe have
an equilibrium that we can
find using backward
induction.

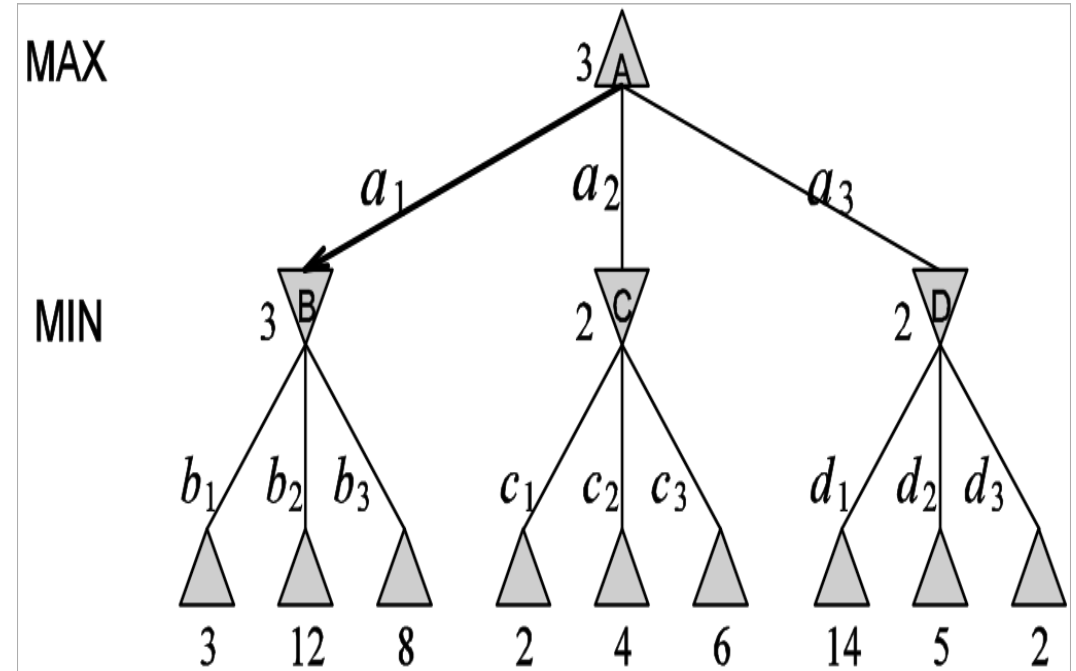
Minimax Value

$$\text{Minimax-value}(n) = \begin{cases} \text{Utility}(n) & \text{if } n \text{ is a terminal state} \\ \text{Max}_{s \in \text{Succ}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MAX node} \\ \text{Min}_{s \in \text{Succ}(n)} \text{MINIMAX-VALUE}(s) & \text{if } n \text{ is a MIN node} \end{cases}$$

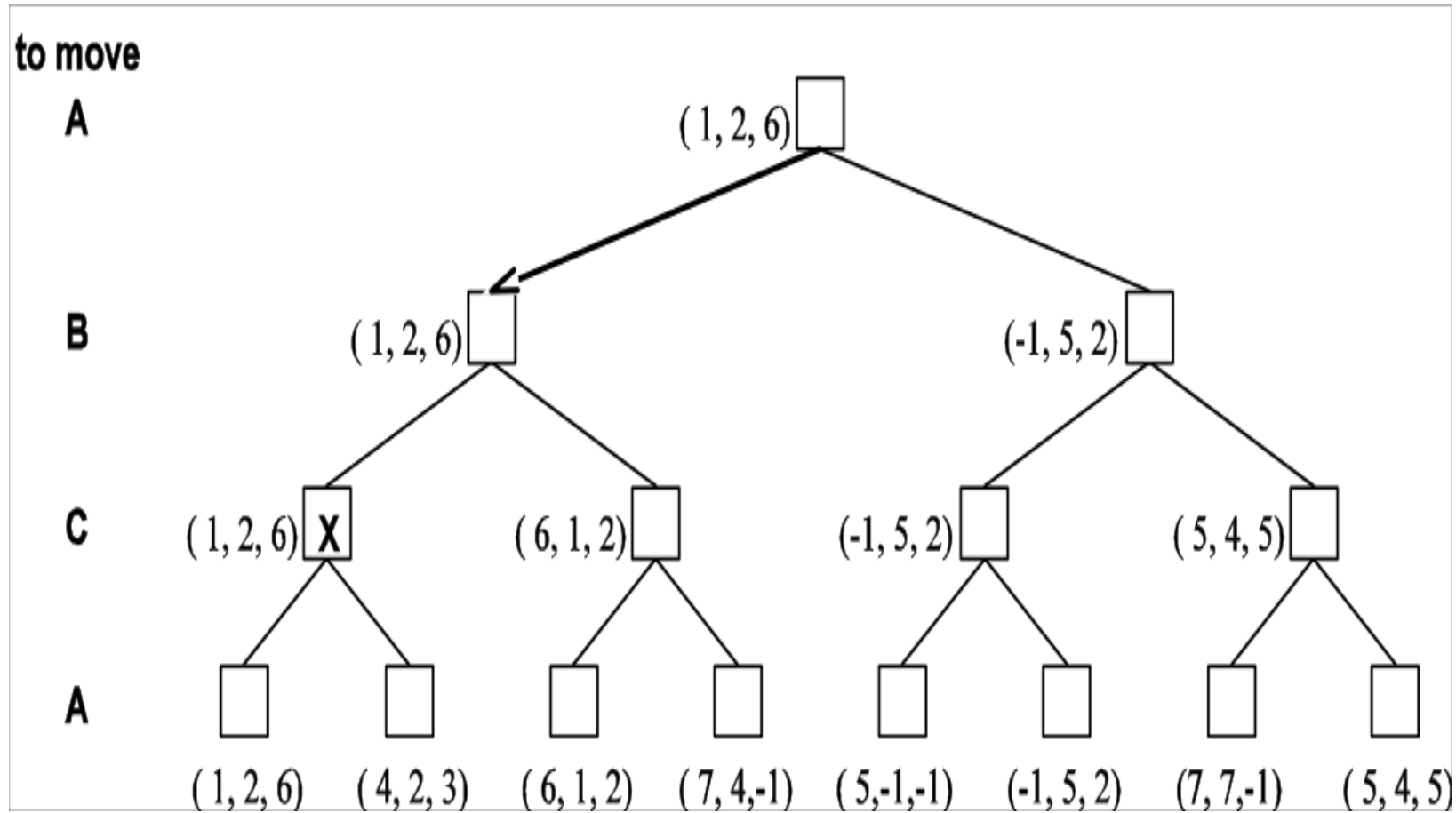


Properties of Minimax

- Complete?
- Optimal?
- Space Complexity?
- Time Complexity?



Minimax and Multi-Player Games



Question

- Chess is a deterministic two player game. Can we use minimax search to build an agent that can play chess reasonably well?

Question

- Chess is a deterministic two player game. Can we use minimax search to build an agent that can play chess reasonably well?

The search tree for chess is $O(b^m)$ where $b \sim 35$ and $m \sim 100$

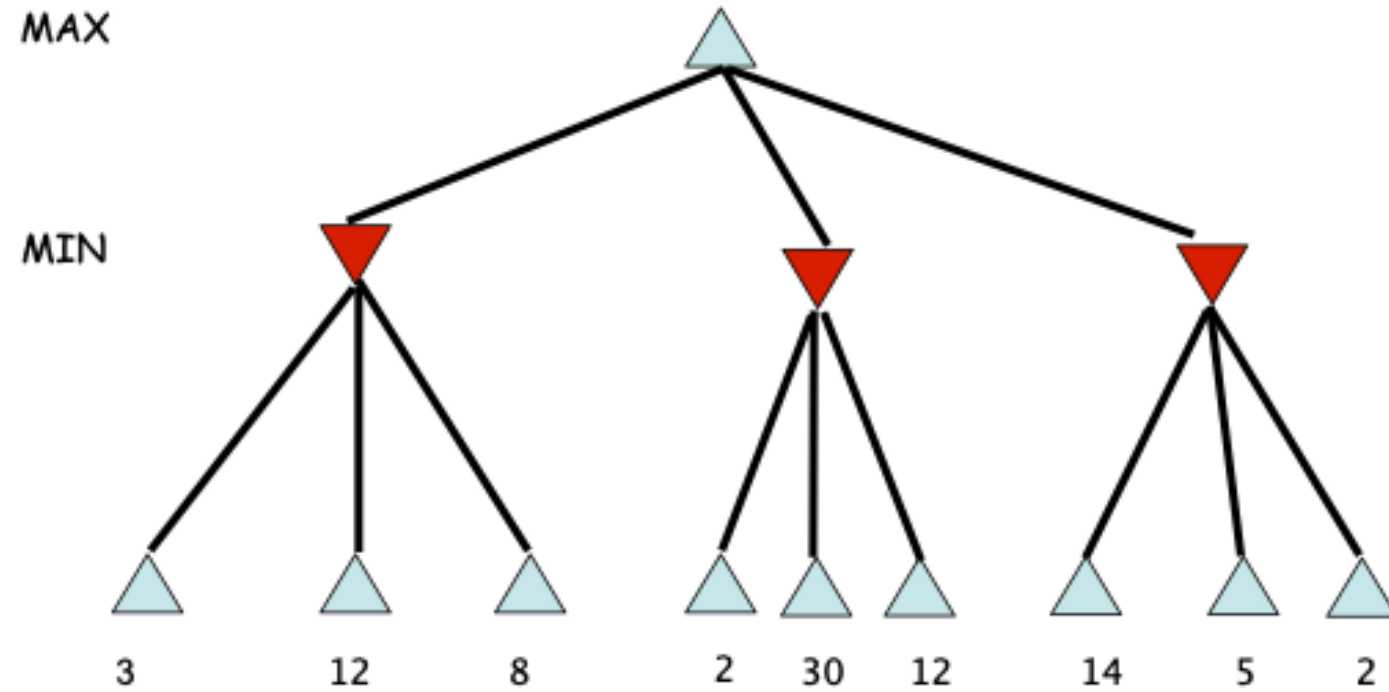
Alpha-Beta Pruning

- We can't realistically run minimax search on many games since we need to traverse the entire game tree.
- But
 - If we are smart (and lucky) we can **prune** the tree
 - Eliminate large parts of the tree from consideration
 - **Alpha-beta pruning** applied to minimax search

Alpha-Beta Pruning (Core idea)

- Alpha:
 - Value of the best (highest value) choice we have found so far on path for MAX
- Beta:
 - Value of the best (lowest value) choice we have found so far on path for MIN
- Update alpha and beta as search continues
- Prune (stop search along the path) as soon as value of current node is known to be worse than current alpha and beta values for MAX or MIN

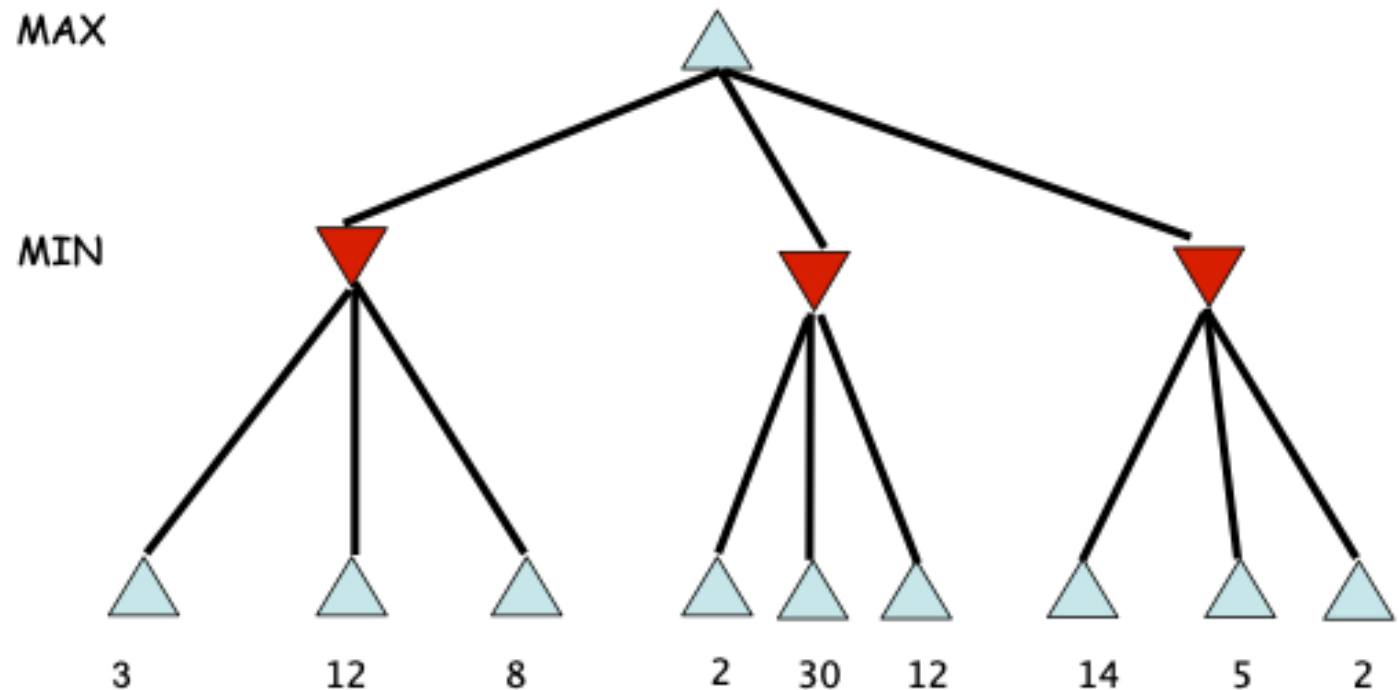
Example



Questions

Can pruning result in a different outcome than minimax search?

How much can be pruned when searching?



Real-Time Decisions

- Alpha-beta can be a huge improvement over minimax search but it is often not enough
 - Need to search to terminal states for at least part of the search space
 - Need to make decisions quickly
- Solution
 - Heuristic evaluation functions + cutoff tests

Evaluation Functions

- You apply evaluation functions to game states
 - If the game state is a terminal state, the function should return the actual utility
 - If non-terminal, the function should return an estimate of the expected utility
 - Function must be fast to compute!

Evaluation Functions

- How do we get evaluation functions?
 - Expert knowledge
 - Learned from experience
- State features
 - Weighted linear function $Eval(s) = \sum_i w_i f_i(s)$
 - These can be learned (we will talk about this later in the course), "function approximation" $Eval(s) = g(s, w)$ where w are weight parameters

Cutting Off Search Early

- Do we have to search to terminal states?
 - No! Cut search early and apply evaluation functions
- When?
 - Arbitrarily (but deeper is better)
 - Quiescent states
 - States that are not “stable”
 - Singular extensions
 - Searching deeper when you have a move that is “clearly better”
 - Can be used to avoid the horizon effect

Cutting Off Search Early?

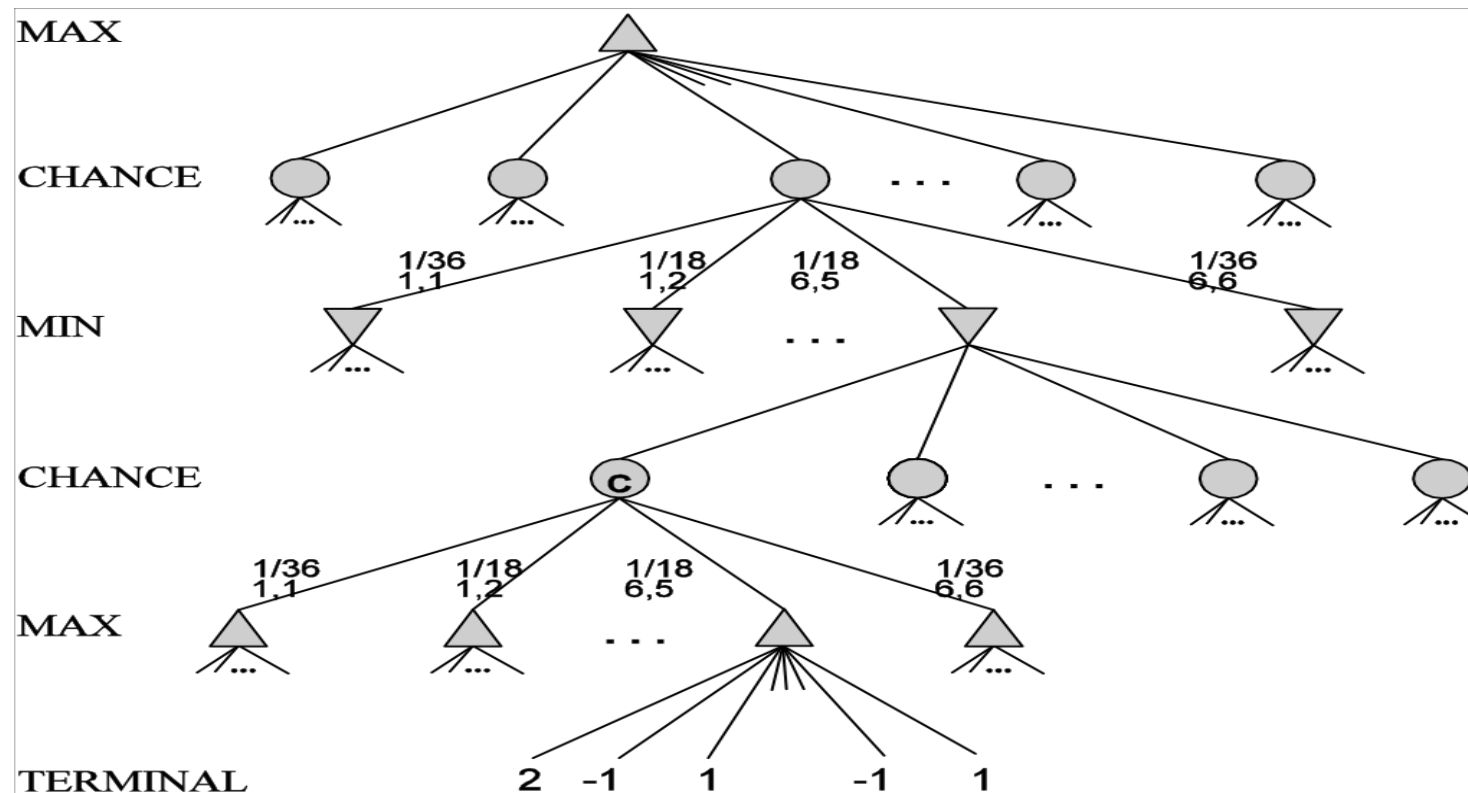
- How deep?
 - Novice player
 - 5 ply (minimax search)
 - Master player
 - 10 ply (alpha-beta pruning)
 - Grandmaster
 - 14 ply + fantastic evaluation functions, opening and endgame databases,...

Stockfish

- The Stockfish chess engine is one of the strongest engines out there
 - Elo rating of 3653 (September 2026) (Grandmaster level)
- Minimax search + alpha-beta pruning
- **Very powerful** static evaluation function (neural network trained on billions of game positions)
- Opening book + endgame databases

Stochastic Games

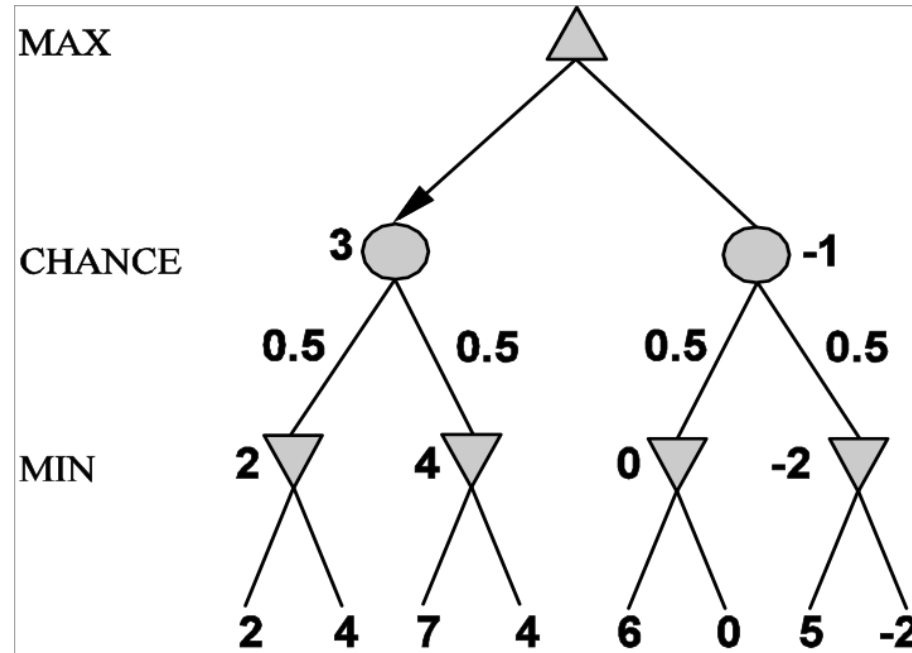
- What is there are stochastic aspects to the game (e.g. dice rolls, shuffled cards?)



Stochastic Games

- Randomness adds a few complications into the search but the basic idea remains the same.
 - You now need to consider the best/worse outcomes + the probability that these outcomes will occur
 - Recall: $E[X] = \sum_{x \in X} P(x)$
- Expectiminimax:
 - ~minimax search but at chance nodes compute the expected value given the children

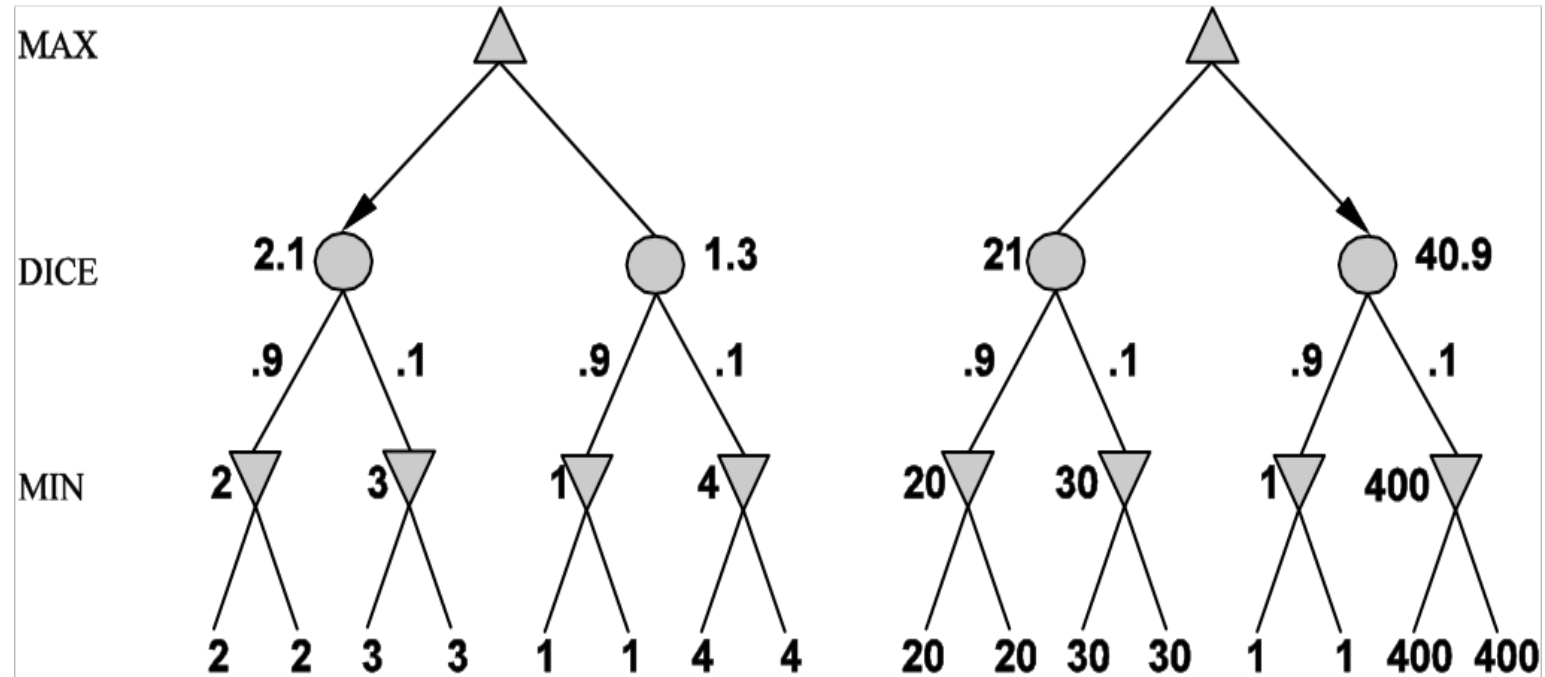
Expectiminimax



$$\text{EMMVal}(n) = \begin{cases} \text{Utility}(n) & \text{if } n \text{ is a terminal state} \\ \max_{s \in \text{Succ}(n)} \text{EMMVal}(s) & \text{if } n \text{ is a Max node} \\ \min_{s \in \text{Succ}(n)} \text{EMMVal}(s) & \text{if } n \text{ is a Min node} \\ \sum_{s \in \text{Succ}(n)} P(s) \text{EMMVal}(s) & \text{if } n \text{ is a Chance node} \end{cases}$$

Expectiminimax Search

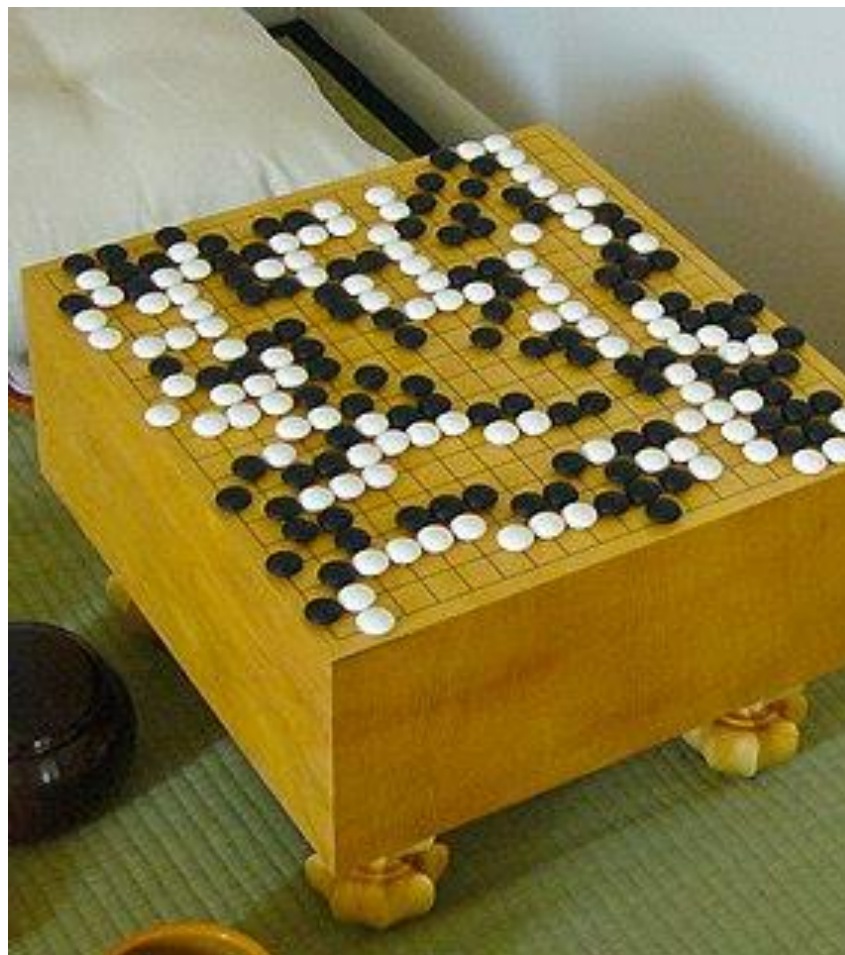
- Be careful!



WARNING: exact values do matter! Order-preserving transformations of the evaluation function can change the choice of moves. Must have positive linear transformations only

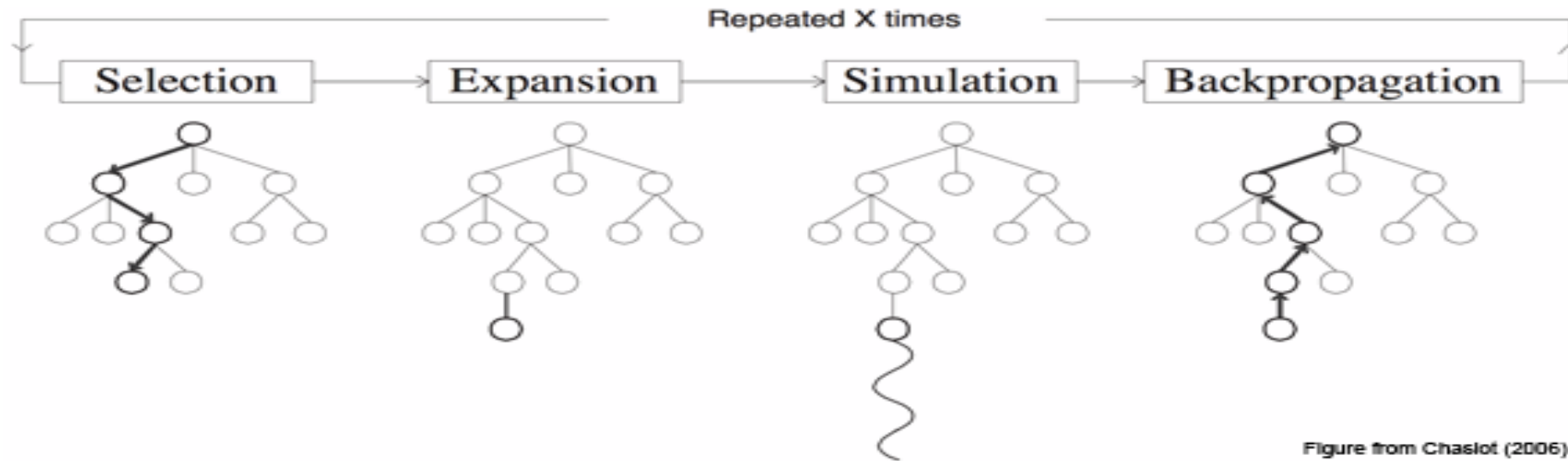
Now what about Go?

- Go is an enormous game
 - $b \sim 250$
 - $d \sim 150$



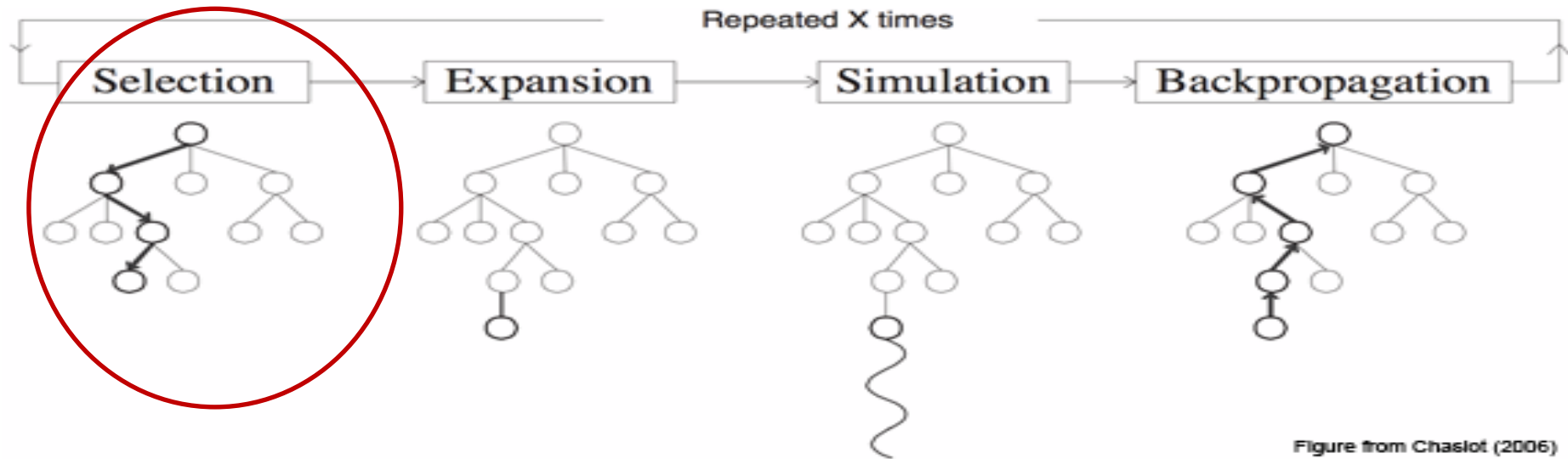
Monte Carlo Tree Search

- Expand a search tree based on random sampling of the search space



You store information in the nodes including the current value and the number of times you have visited the node.

Monte Carlo Tree Search: Selection



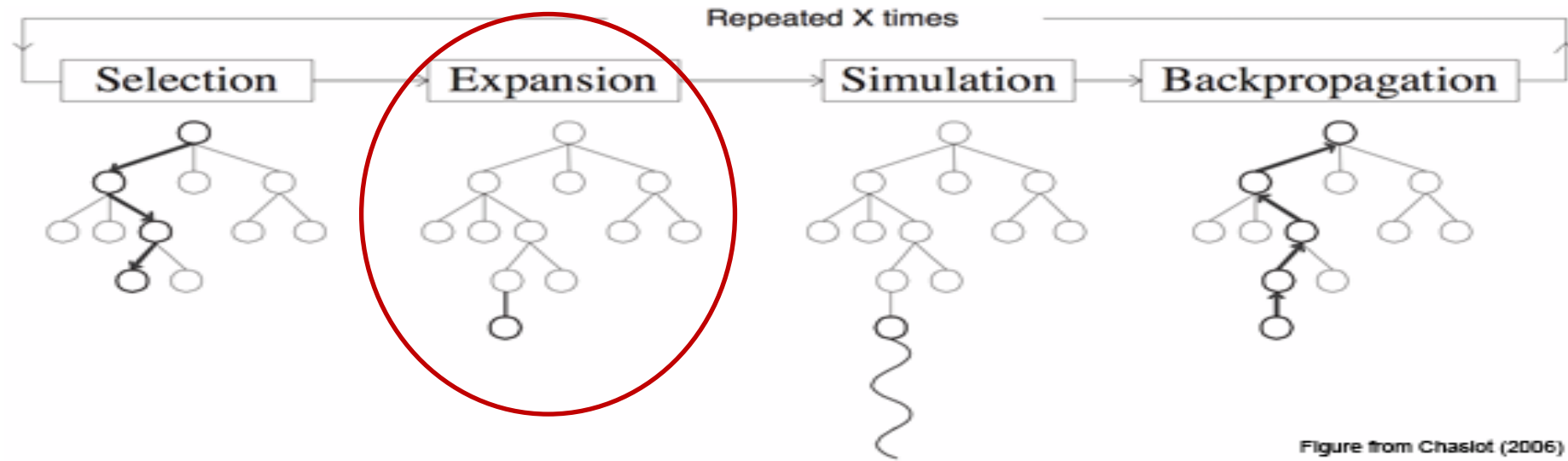
- Starting at the root node, traverse the current tree
- How?
 - Use Upper Confidence Bounds (UCB) as your selection strategy

$$v_i + C\sqrt{\frac{\ln N}{n_i}}$$

Balances the value and the number of times you have seen a particular node.

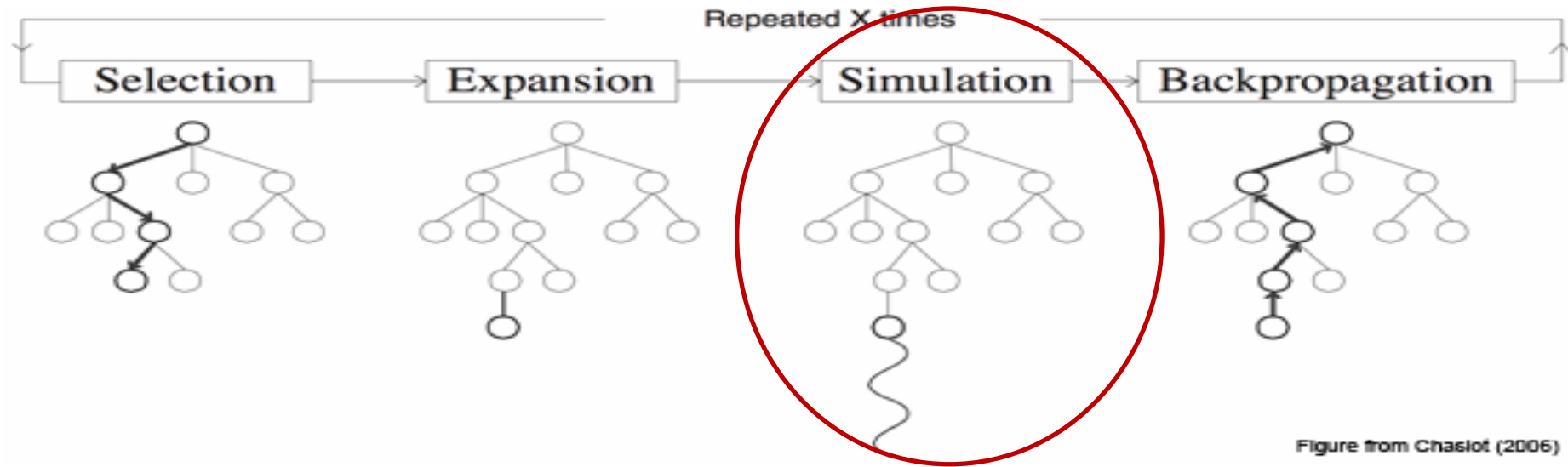
We will see this idea again (and derive the formula) later in the term when we cover Bandits)

Monte Carlo Tree Search: Expansion



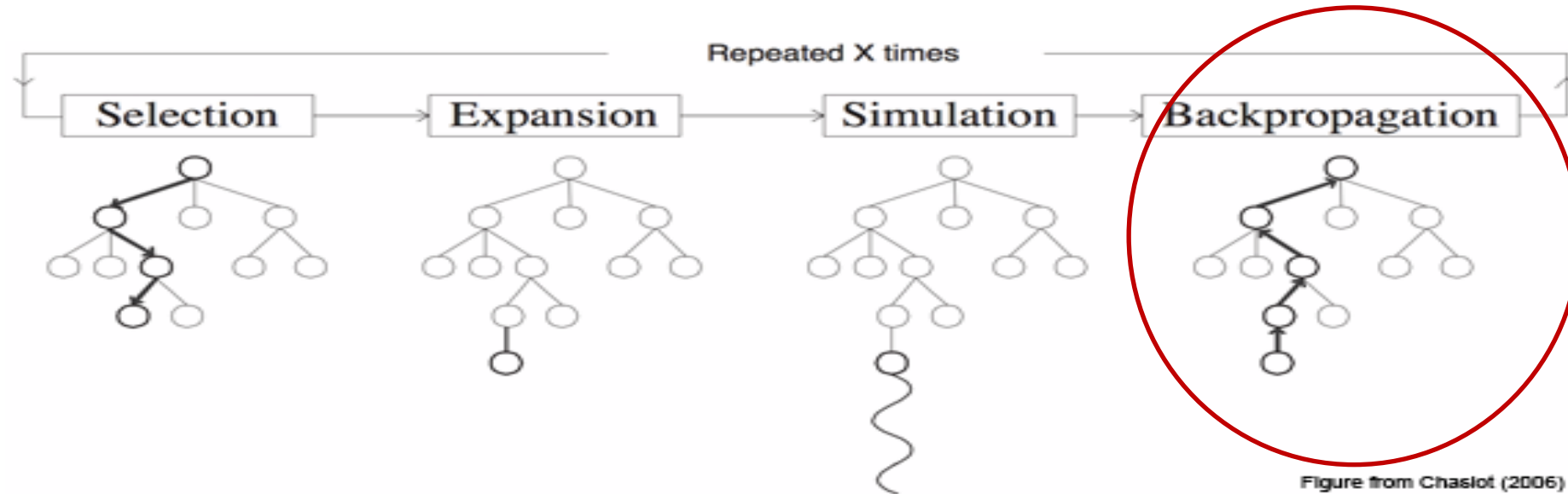
- Once you reach a node where you have no statistics, expand it and randomly select a child

Monte Carlo Tree Search: Simulation



- Using a “rollout policy” play out the game until a terminal node
 - This policy should be “lightweight”. Often a random policy is used.

Monte Carlo Tree Search: Backpropagation



- Use the results from the rollout to update the statistics in the tree, for all nodes from the child you generated through expansion to the root node.

Key Take-Aways

- Games form interesting testbeds for AI
- Minimax search allows an agent to play optimally against an optimal opponent
- Alpha-beta pruning allows us to scale minimax search to larger problems
- MCTS combines tree search with random sampling
- A good evaluation function is critical
- Games are fun!