

Lecture 6:

Game Theory and Adversarial Games

- ECS7002P – Artificial Intelligence in Games
- Raluca D. Gaina – r.d.gaina@qmul.ac.uk
- Office: CS.335



<http://gameai.eecs.qmul.ac.uk>

Queen Mary University of London

Outline

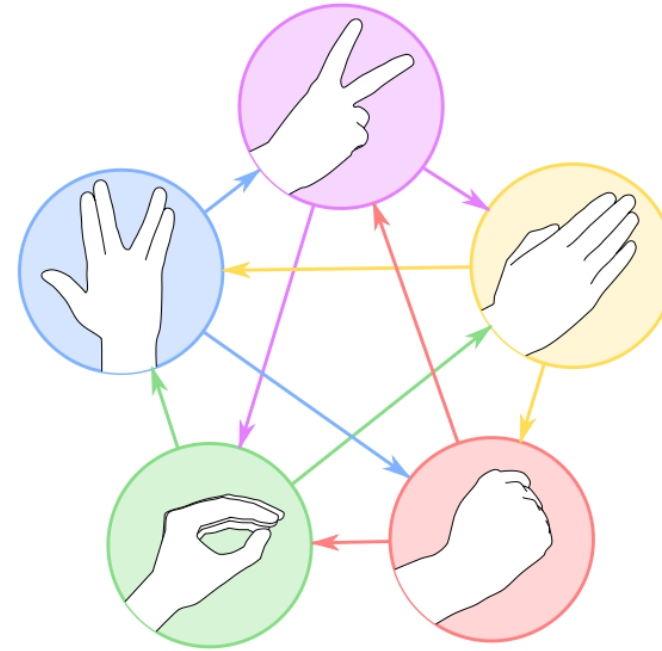
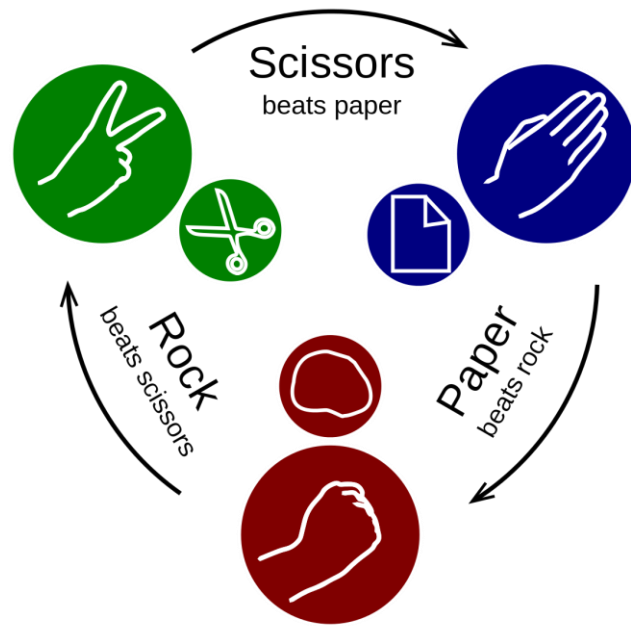
- Introduction

- Minimax and Alpha-Beta Search

Game Theory and Adversarial Games

Introduction

2-(N-) Player Games



- What is the best strategy? Why?
- How can you change the strategy over time?
- In previous MDP, the optimal strategy or policy π is to maximize the cumulative reward.
- In games of more than one player, π^* depends on what the other player does.

Game Theory

In a single-agent RL problem:

- Other players can be part of the environment.
- Game is reduced to an MDP.
- Best line of play is following the optimal policy for this MDP.

Game theory, for 2+ players:

- 2-Player (N-Player) games involve interactions between 2 or more individuals who make decisions in their own interests.
- Game outcomes provide **payoffs** or **rewards** to individuals based on what actions they choose and what actions others choose.
- **Payoff** is the profit gained through a game by an agent. A rational player's objective is to choose a strategy that maximizes this payoff.

Game Theory

Definition (Game)

A game in normal form may be defined as a triplet $\langle N, S, \pi \rangle$:

- N is a finite set of players
- S is a finite set of pure strategies
- π is the combined payoff function

- There is a set of players $N = 1, \dots, n$.
- Each agent i has a set of strategies S_i (*pure strategies*).
- Payoffs: $\pi_1, \dots, \pi_n : S \rightarrow \mathbb{R}$, where $S = S_1 \times \dots \times S_n$.
- The payoff to a particular agent depends not only on his action but the actions of the players that interact with it.
- Payoffs reflect an agent's preference over the possible outcomes of an interaction.
- Agents are assumed to be *rational* (i.e. they try to maximize their, and only theirs, expected payoff).

Example: Prisoner's Dilemma (PD)

- Two individuals are being held in a prison in separate and isolated cells.
- They have been told that:
 - If one defects (from their partner in the crime) by confessing, he/she will go free while the other will get 3 years in prison.
 - If they both cooperate (with their partner in crime) by not confessing, they will both be sent to prison for 1 year.
 - If they both defect, by confessing, they will each get 2 years.

Each player has two strategies:

- C – Cooperate (Remain silent)
- D – Defect (Confess)

Payoff matrix:

	C	D
C	$[-1, -1]$	$[-3, 0]$
D	$[0, -3]$	$[-2, -2]$

PD forms the foundations for many inquiries into the likelihood and persistence of cooperative behaviours in different fields: economics, sociology, politics and evolutionary ecology.

Q? What's the best strategy?

Example: Prisoner's Dilemma (PD)

- Mutual cooperation pays reward ($R = -1$).
- Mutual defection pays punishment ($P = -2$).
- Otherwise, defector gets the temptation payoff ($T = 0$) while cooperator gets heavily penalized with “sucker’s” payoff ($S = -3$).
- In PD, always $T > R > P > S$.

	C	D
C	$[-1, -1]$	$[-3, 0]$
D	$[0, -3]$	$[-2, -2]$

- This payoff matrix makes defection the *preferred* strategy: even if mutual cooperation has a higher payoff!
- **Dilemma:** Rational players get P instead of the highest payoff for both players, R !

Pure versus Mixed Strategies

A **pure strategy** provides a complete definition (i.e. is deterministic) of how a player will play a game. Let s be a strategy profile, composed of multiple strategies: $s = (s_1, \dots, s_n)$, each s_i being a *pure* strategy.

Definition (Mixed Strategy)

A mixed strategy for player i is a probability distribution over his set s of pure strategies. Since for each agent the set s is finite, we can represent any mixed strategy x_i for player i as a vector:

$$x = (x_1, \dots, x_n), \text{ where } \forall i, x_i \geq 0 \text{ and } \sum_{i=1}^n x_i = 1$$

- An agent may choose any pure strategy or pick one of those at random.
- In PD, examples of different strategies are:
 - Pure strategy: always Cooperate.
 - Pure strategy: always Defect.
 - Mixed strategy:
 - 50% Choose pure strategy: *always Cooperate*.
 - 50% Choose pure strategy: *always Defect*.

Best Response and Nash Equilibrium

Assuming all the other players fix their strategies (s_{-i}).

- Given the opponent is behaving in a certain way, what's the optimal policy/strategy I should have?
- The best response $s_i^*(s_{-i})$ is the optimal policy against those fixed strategies.

Definition (Best Response)

s_i is a **Best Response** to s_{-i} if and only if:

$$\pi_{s_i, s_{-i}} \geq \pi_{s', s_{-i}} , \forall s' \in S$$

Where $\pi_{s_i, s_{-i}}$ is the payoff of playing with strategy s_i against fixed opponent strategies s_{-i} .

Definition (Nash Equilibrium)

The strategy profile $s = (s_1, \dots, s_n)$ is a **Nash Equilibrium** if and only if, for each player i , their strategy s_i is a best response to s_{-i} .

Nash Equilibrium

- In Nash Equilibrium, no player can do better by unilaterally switching to a different strategy.
- In PD, the strategy (D,D) is a Nash Equilibrium. Why?

	C	D
C	$-1, -1$	$-3, 0$
D	$0, -3$	$-2, -2$

Theorem (Nash, 1950)

Every finite game has a Nash equilibrium in mixed strategies.

Best Response, Nash Equilibrium and Self-play RL

Best response is a solution to the single-agent RL problem:

- Other players become part of the environment.
- Game is reduced to an MDP.
- Best response is the optimal policy for this MDP.

Nash equilibrium is a fixed-point of self-play RL:

- Experience is generated by playing games between agents.
- Each agent learns best response to other players.
- One player's policy determines another player's environment.
- All players are adapting to each other.

Game Theory and Adversarial Games

Two-Player Zero-Sum Games

Two-Player Zero-Sum Games

We'll focus now on **Two-Player Zero-Sum Games** with Perfect Information. A Two-Player Zero Sum Game has equal and opposite rewards for both players:

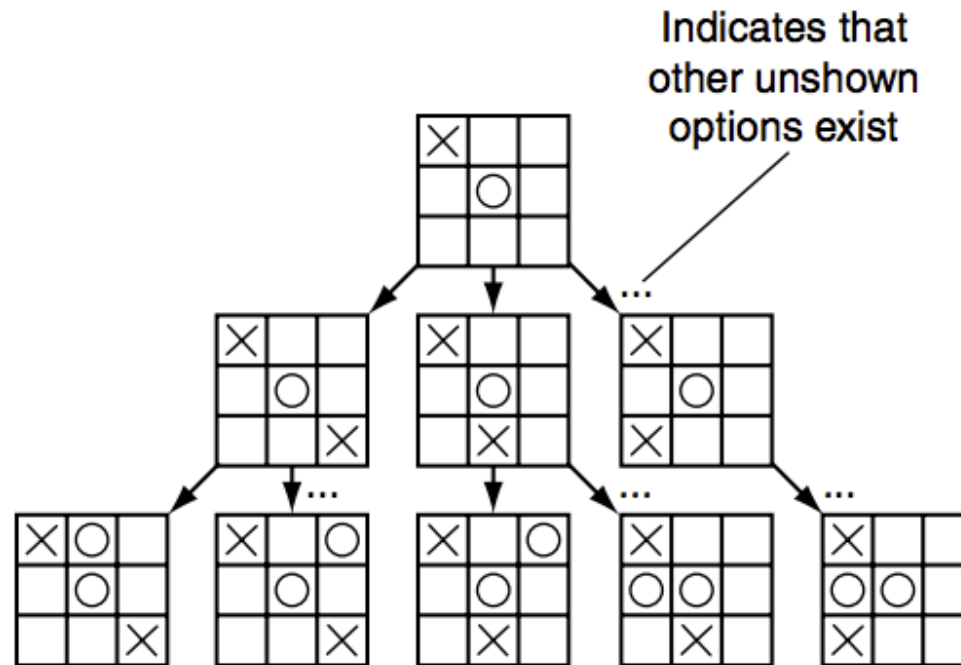
$$R_1 + R_2 = 0$$

		Blue		
		A	B	C
Red	1	-30 30	10 -10	-20 20
	2	10 -10	-20 20	20 -20

The Game Tree

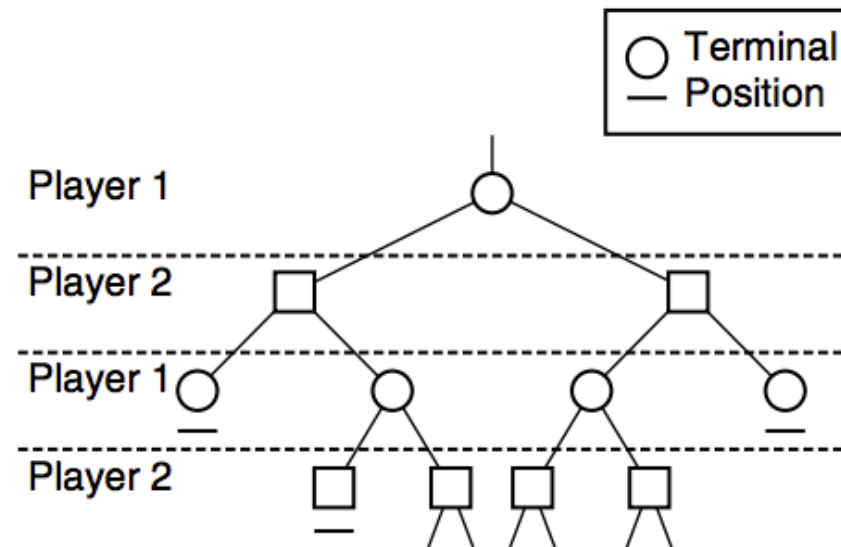
Any turn-based game can be represented as a game tree.

- Each node represents a board position.
- The root of the tree is the current game state.
- Each branch is a possible move.
- A move leads from one board position to another.
- Each player gets to move at alternating levels of the tree.



The Game Tree

- **Terminal states/positions:** nodes without possible moves (end of the game). For each terminal state, a game score is given to each player, typically:
 - +1 to the winner.
 - -1 to the loser.
 - 0 in case of a draw.
- **Branching factor:** number of possible moves from a given state. Indicator of how difficult the game is.
- **Depth:** maximum number of turns until a terminal state is found.

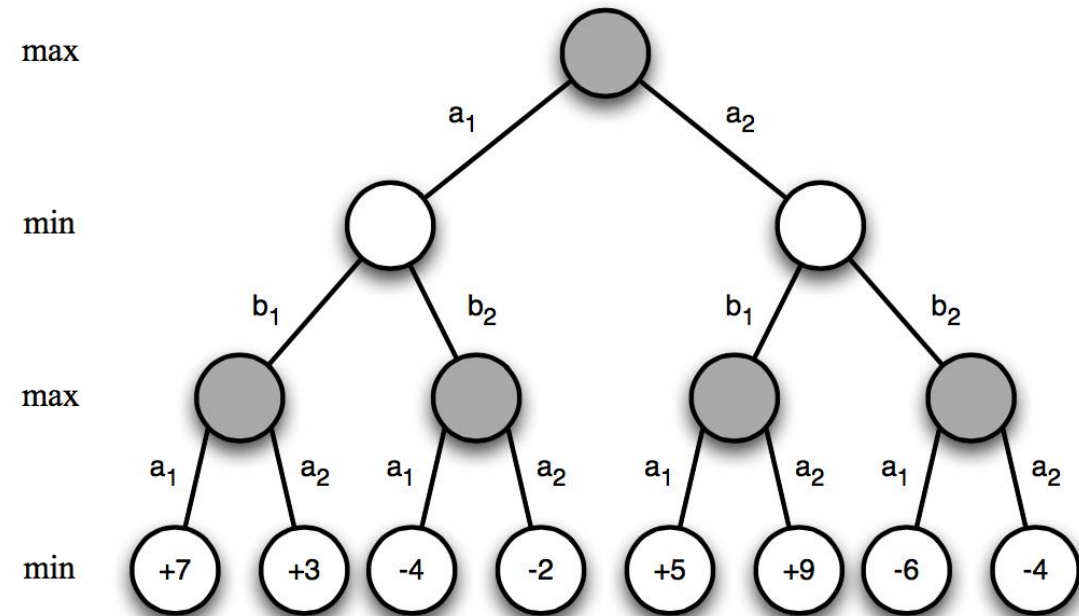


Game Theory and Adversarial Games

Minimax and Alpha-Beta Search

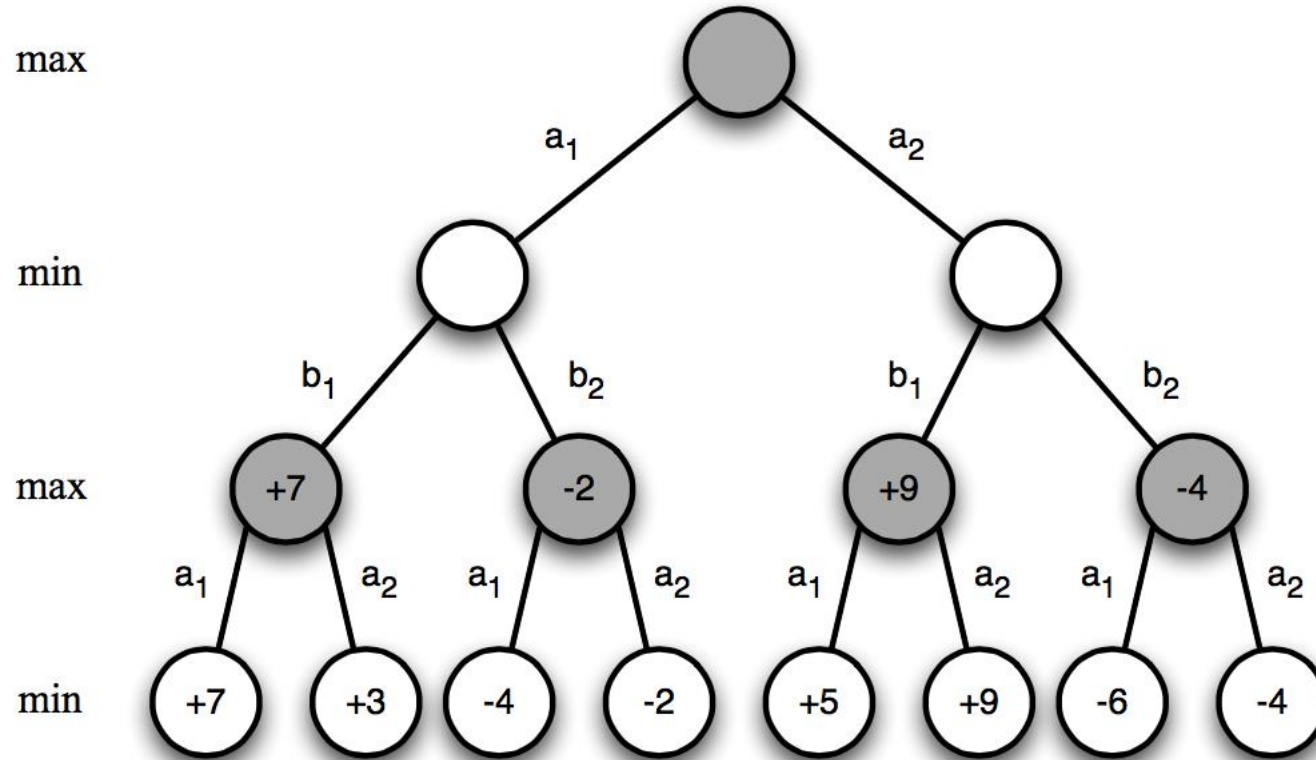
Minimax

- By performing self-play, we define two players:
 - **MAX** player:
 - “Owns” the tree.
 - Attempts to maximize.
 - **MIN** player:
 - Opponent.
 - Attempts to minimize.

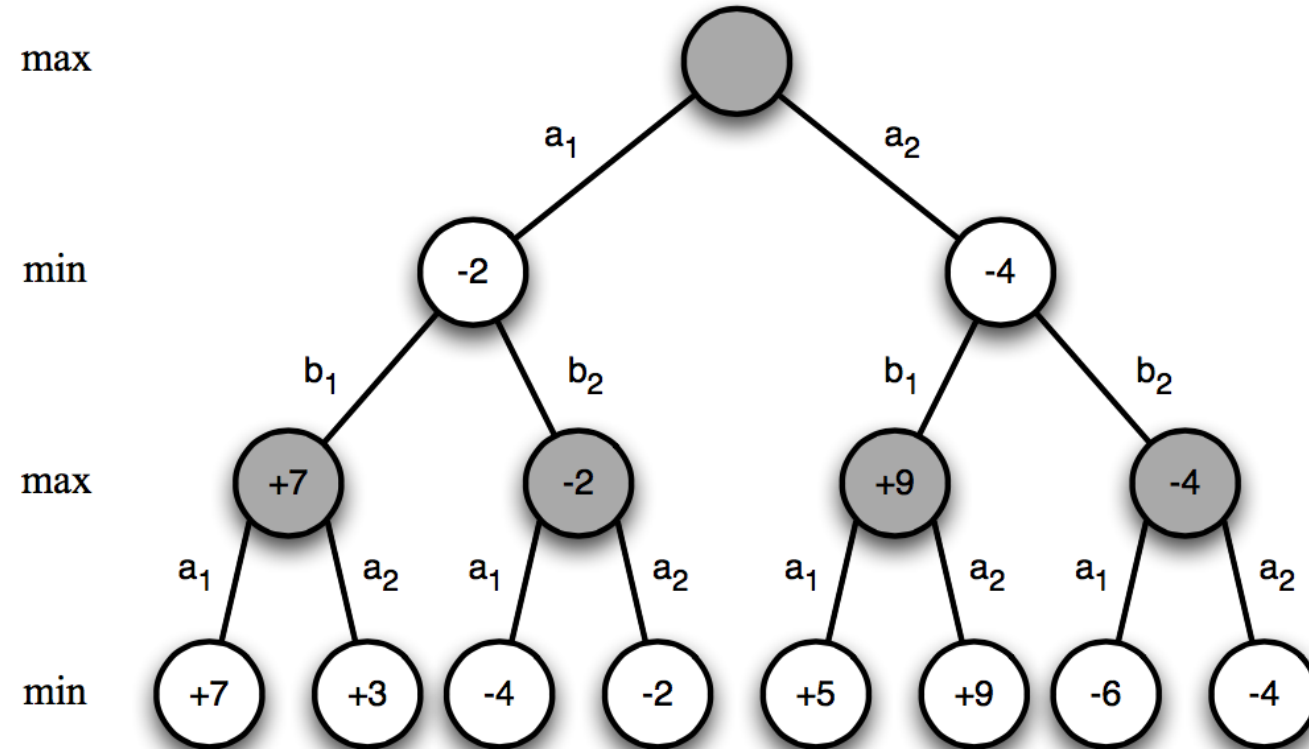


- Values at the terminal state are the state-values seen from the perspective of the top player (MAX).

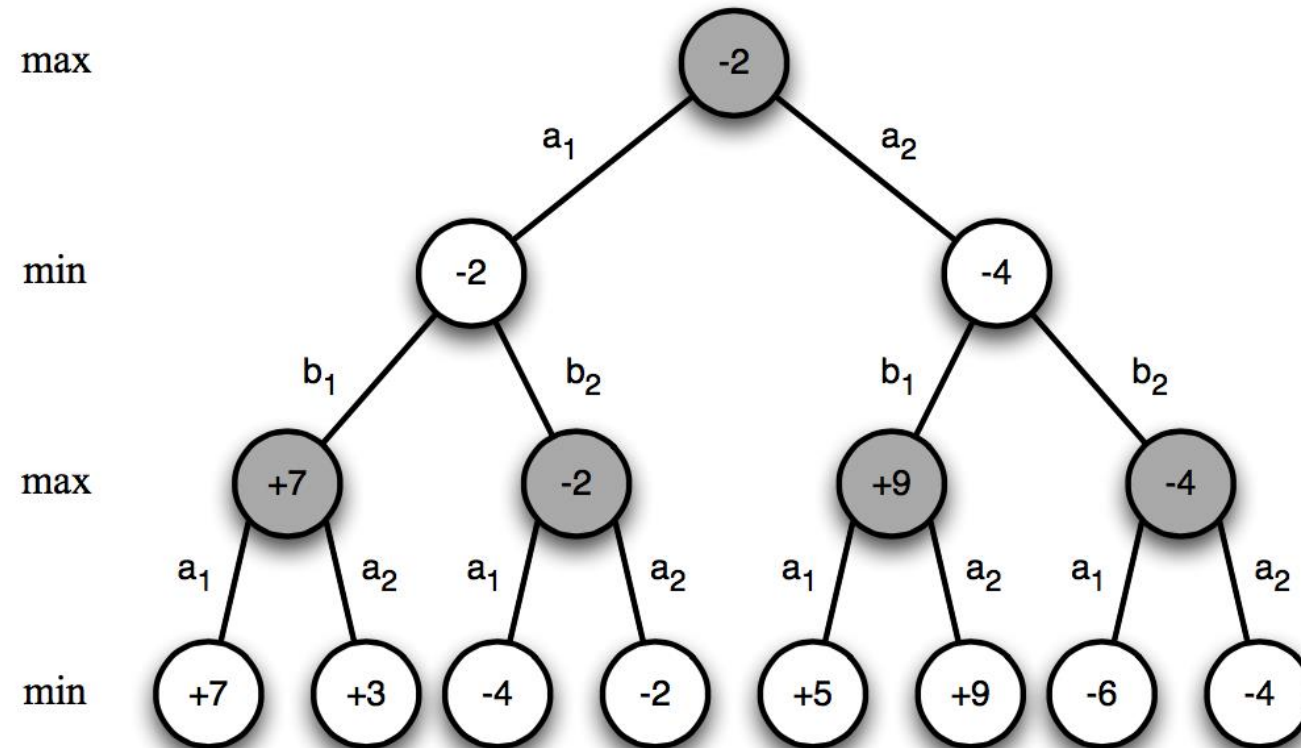
Minimax Example



Minimax Example



Minimax Example



Q? Is this the optimal policy?

Minimax Policy

Is this the optimal policy? **YES**.

A value function defines the expected total reward given two policies (π_1 and π_2) as:

$$v_{\pi}(s) = \mathbb{E}[G_t | S_t = s]$$

A minimax value function maximizes **MAX's** expected return while minimizing **MIN's** expected return:

$$v_*(s) = \max_{\pi_1} \min_{\pi_2} v_{\pi}(s)$$

Not only that. There is a **unique** minimax value function (in two-player zero-sum games), and the policy that achieves it ($\pi = \langle \pi_1, \pi_2 \rangle$) is a **Nash Equilibrium** (proof in respective theorems).

Q? Where's the catch?

Approximating Value Functions

The tree grows exponentially! It is impractical to build the whole minimax tree in larger (real!) games.

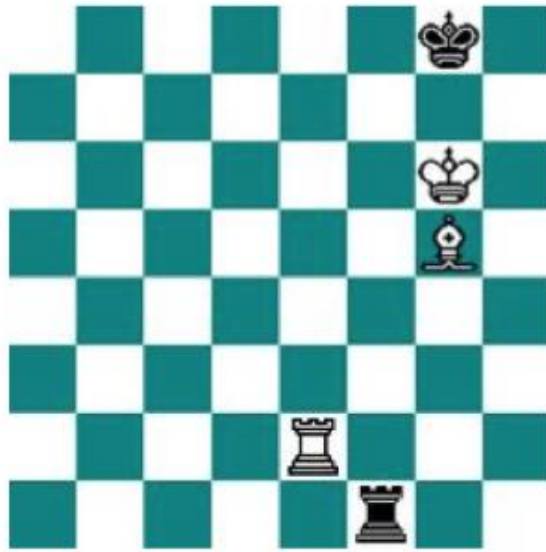
Solution 1: Value function approximator ($v(s, w) \approx v_\pi(s)$).


- Determine a set of n features (x_0, \dots, x_n) from the board. Examples:
 - Presence/number of pieces in the board.
 - Presence of pieces in certain positions.
 - Patterns within the board (N-Tuples).
- Give weights (manually, from expert, learnt) to these features.
- Use a function approximator to evaluate non-terminal states:

$$v(s, w) = \sum_{i=0}^n x_i(s) \times w_i$$

Value Function Approximator (Examples)

Binary-Linear function approximator in Chess:

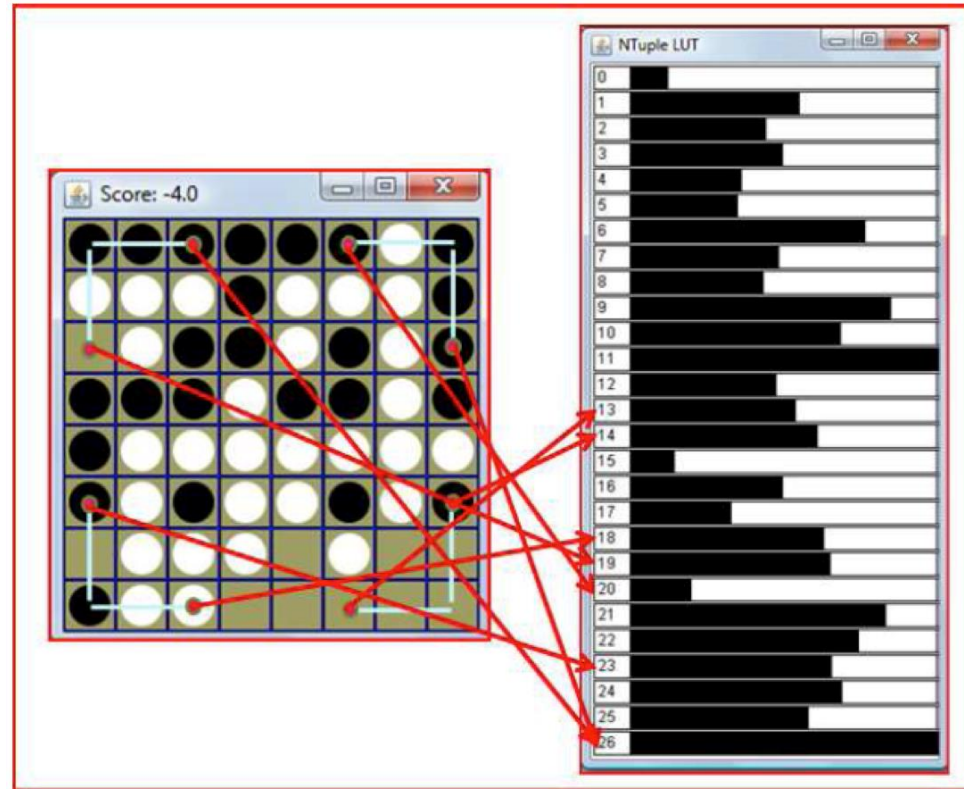


$$v(s, \mathbf{w}) = \mathbf{x}(s) \cdot \mathbf{w} = \begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} +5 \\ +3 \\ +1 \\ -5 \\ -3 \\ -1 \\ \vdots \end{bmatrix}$$


$$v(s, w) = \sum_{i=0}^n x_i(s) \times w_i = 5 \times 1 + 3 \times 1 + 1 \times 0 + (-5) \times 1 + (-3) \times 0 = 3$$

Value Function Approximator (Examples)

N-tuples in Othello:

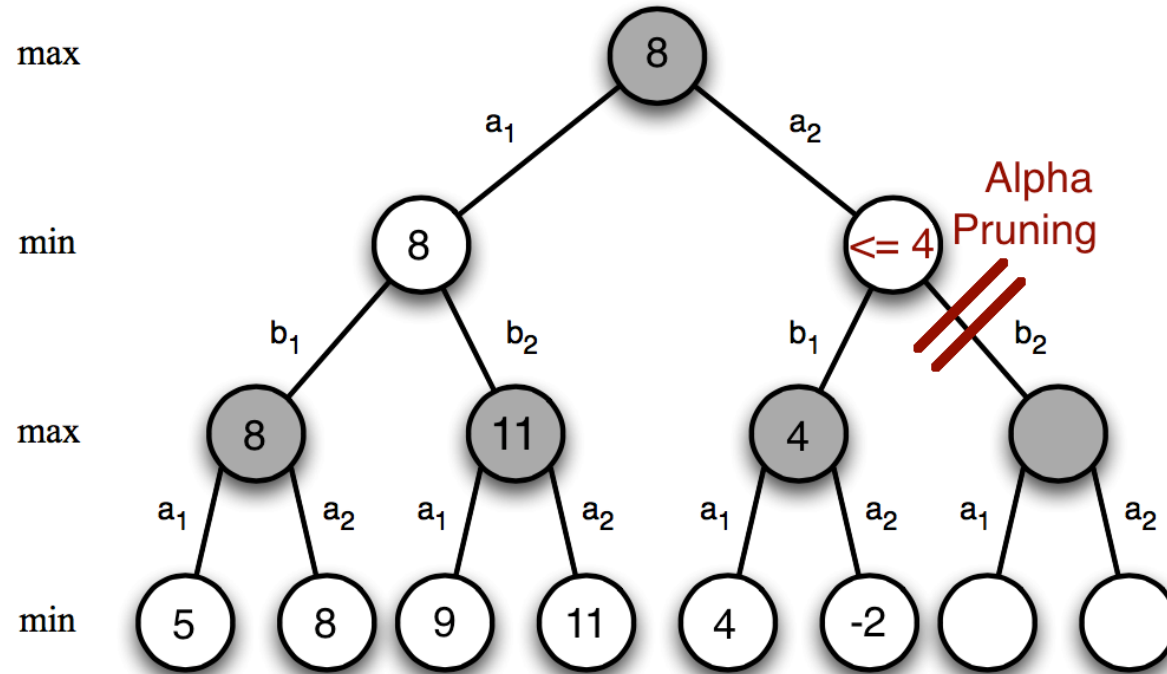


Simon M. Lucas, Learning to Play Othello with N-Tuple Systems, Australian Journal of Intelligent Information Processing (2008), volume 4, pages: 1 - 20.

Alpha-Beta Search

Solution 2: Pruning the Minimax Tree: Alpha-Beta Search.

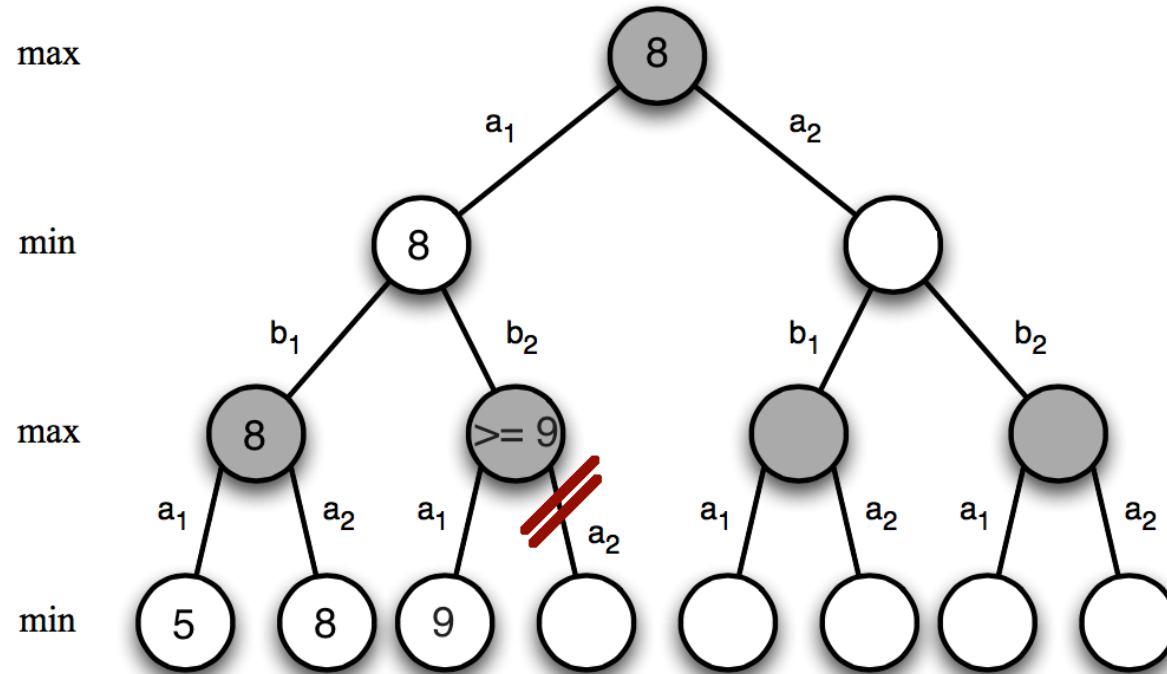
- During search, it becomes clear that some branches are suboptimal.
- Alpha pruning: A **upper** limit on the achievable score.
 - **MIN** player is never going to choose something higher than 4.
 - **MAX** player will always prefer 8 to something ≤ 4 .



Alpha-Beta Search

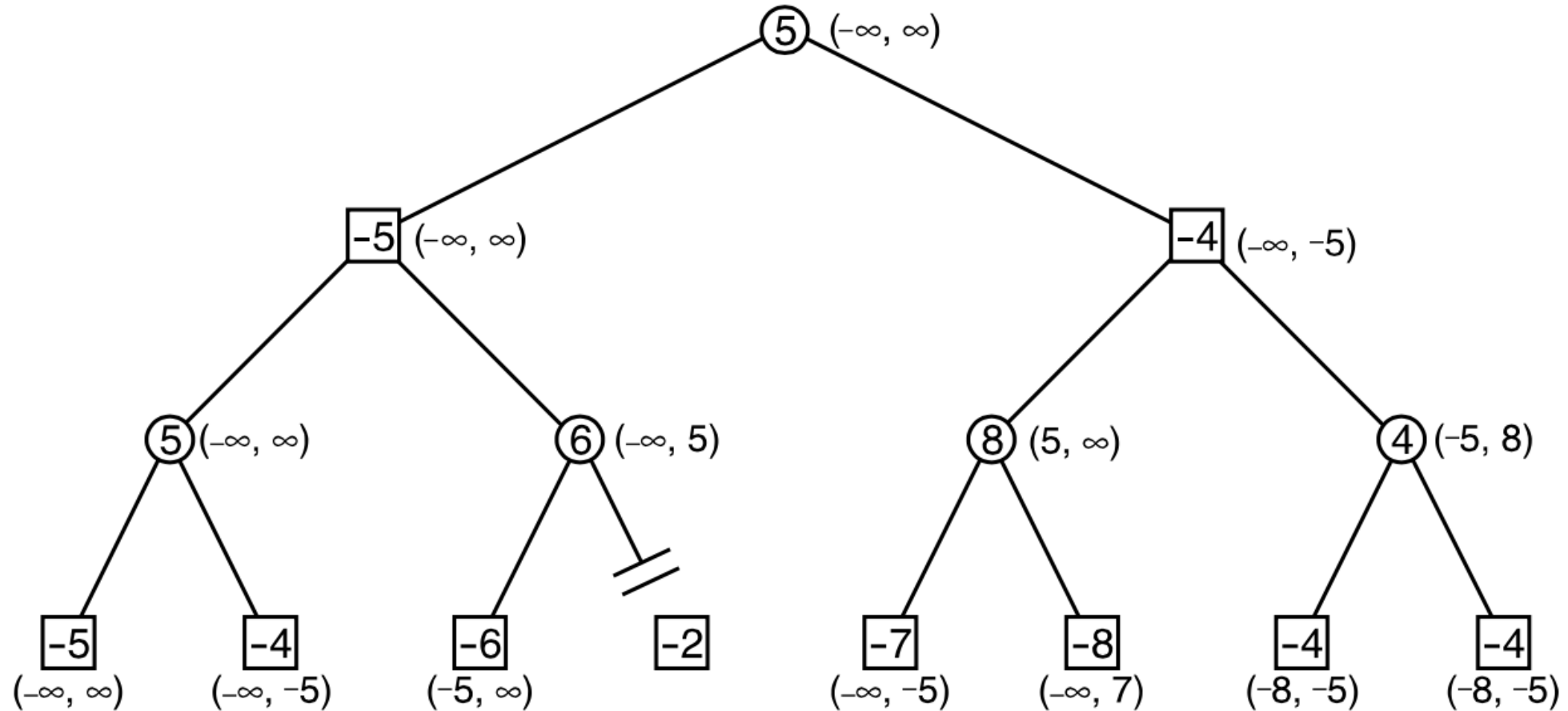
Solution 2: Pruning the Minimax Tree: Alpha-Beta Search.

- During search, it becomes clear that some branches are suboptimal.
- Beta pruning: An **lower** limit on the achievable score.
 - **MAX** player is never going to play for a value lower than 9.
 - **MIN** player will always prefer 8 than a value ≥ 9 .



Alpha-Beta Search

Alpha-Beta Search combines both limits, keeping track of the α and β values as the search progresses.



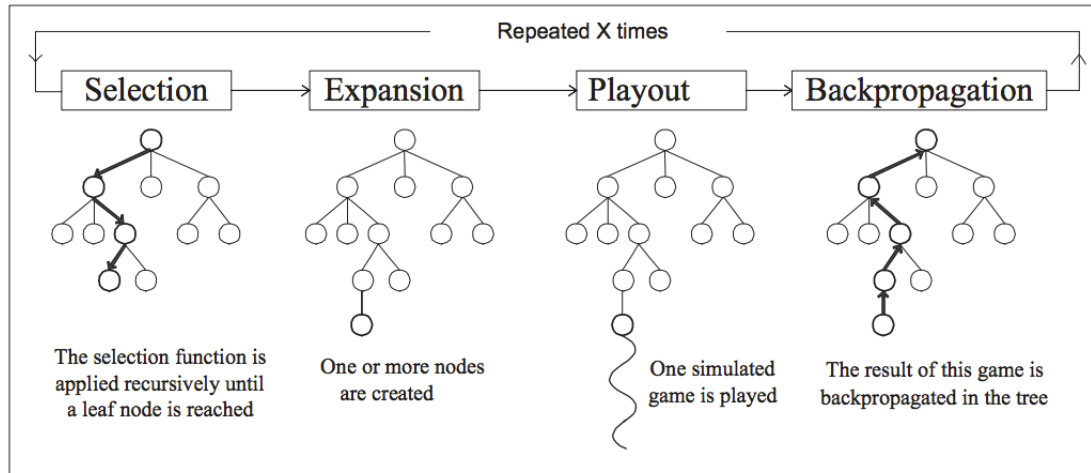
Deep Blue (1997)



- Binary-linear value function, with 8000 handcrafted features.
- Search:
 - High-performance parallel Alpha-Beta search.
 - 480 processors, 200 millions positions per second.
 - Look ahead 16-40 plies.
- Deep Blue 4–2 Gary Kasparov

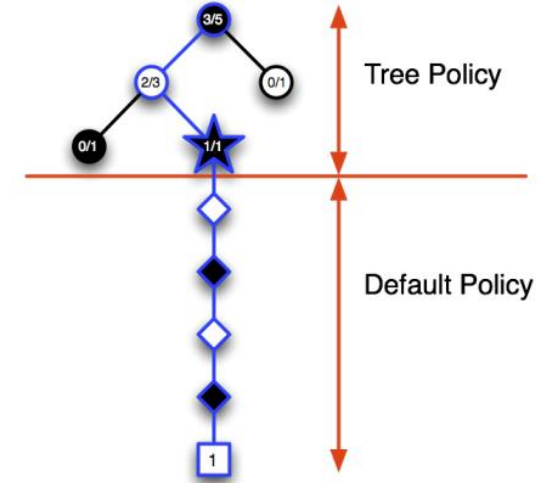
MCTS in 2 player games

Monte Carlo Tree Search was a breakthrough in Go... a 2-player game.



G. Chaslot, M. Winands, H. van den Herik, J. Uiterwijk, and B. Bouzy, "Progressive strategies for Monte-Carlo tree search", *New Mathematics and Natural Computation*, vol. 4, no. 3, pp. 343–357, 2008.

S. Gelly and D. Silver, "Monte-Carlo tree search and rapid action value estimation in computer Games", *Artif. Intell.*, vol. 175, no. 11, pp. 1856–1875, Jul. 2011.



UCT, Monte Carlo, Temporal Difference Learning, etc: same approach (alternating players on each level of the tree) for 2+ player games.