



Introduction to Artificial Intelligence

Week 12



Monte Carlo Tree Search

Monte Carlo Tree Search

- In computer science, **Monte Carlo tree search (MCTS)** is a heuristic search algorithm for some kinds of decision processes, most notably those employed in game play
- A leading example of Monte Carlo tree search is recent computer Go programs, but it also has been used in other board games, as well as real-time video games and non-deterministic games such as poker





MCTS History

- In 2006, Rémi Coulom described the Monte Carlo method to game-tree search and coined the name Monte Carlo tree search
- The Monte Carlo method, which uses random sampling for deterministic problems which are difficult or impossible to solve using other approaches, dates back to the 1940s



Idea

- The focus of MCTS is on the analysis of the most promising moves, expanding the search tree based on random sampling of the search space
- The application of MCTS in games is based on many *playouts*. In each playout, the game is played out to the very end by selecting moves at random
- The final game result of each playout is then used to weight the nodes in the game tree so that better nodes are more likely to be chosen in future playouts



Monte Carlo Tree Search



- The most basic way to use playouts is to apply the same number of playouts after each legal move of the current player, then choosing the move which led to the most victories. The efficiency of this method — called *Pure Monte Carlo Game Search* — often increases with time as more playouts are assigned to the moves that have frequently resulted in the player's victory (in previous playouts)
- Full Monte Carlo tree search employs this principle recursively on many depths of the game tree. Each round of MCTS consists of four steps:
 1. Selection
 2. Expansion
 3. Simulation
 4. Backpropagation



Steps

1. Selection

- Start from root R and select successive child nodes down to a leaf node L . The section below says more about a way of choosing child nodes that lets the game tree expand towards most promising moves, which is the essence of MCTS

2. Expansion

- Unless L ends the game with a win/loss for either player, either create one or more child nodes or choose from them node C

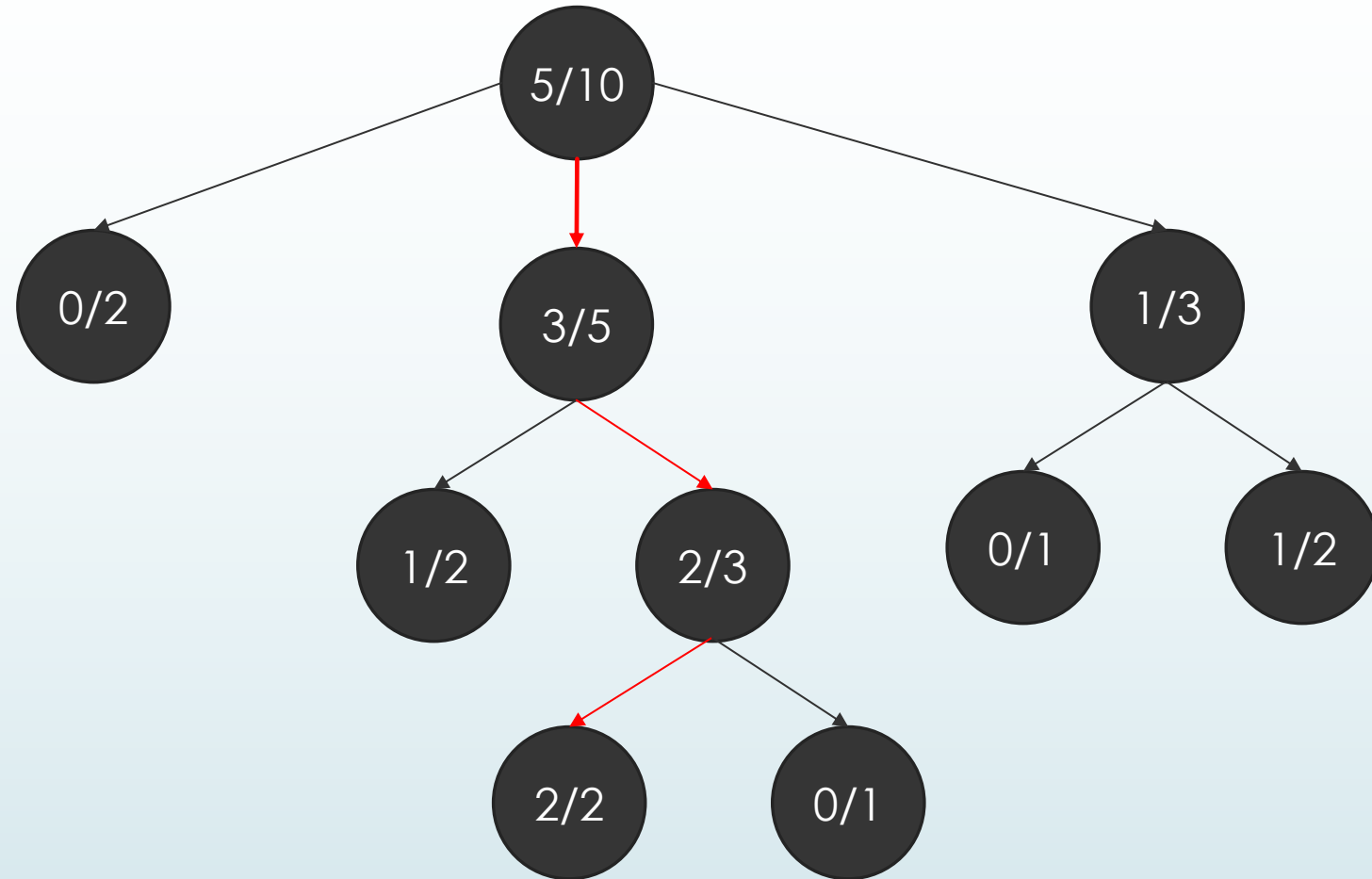
3. Simulation

- Play a random playout from node C . This step is sometimes also called playout or rollout

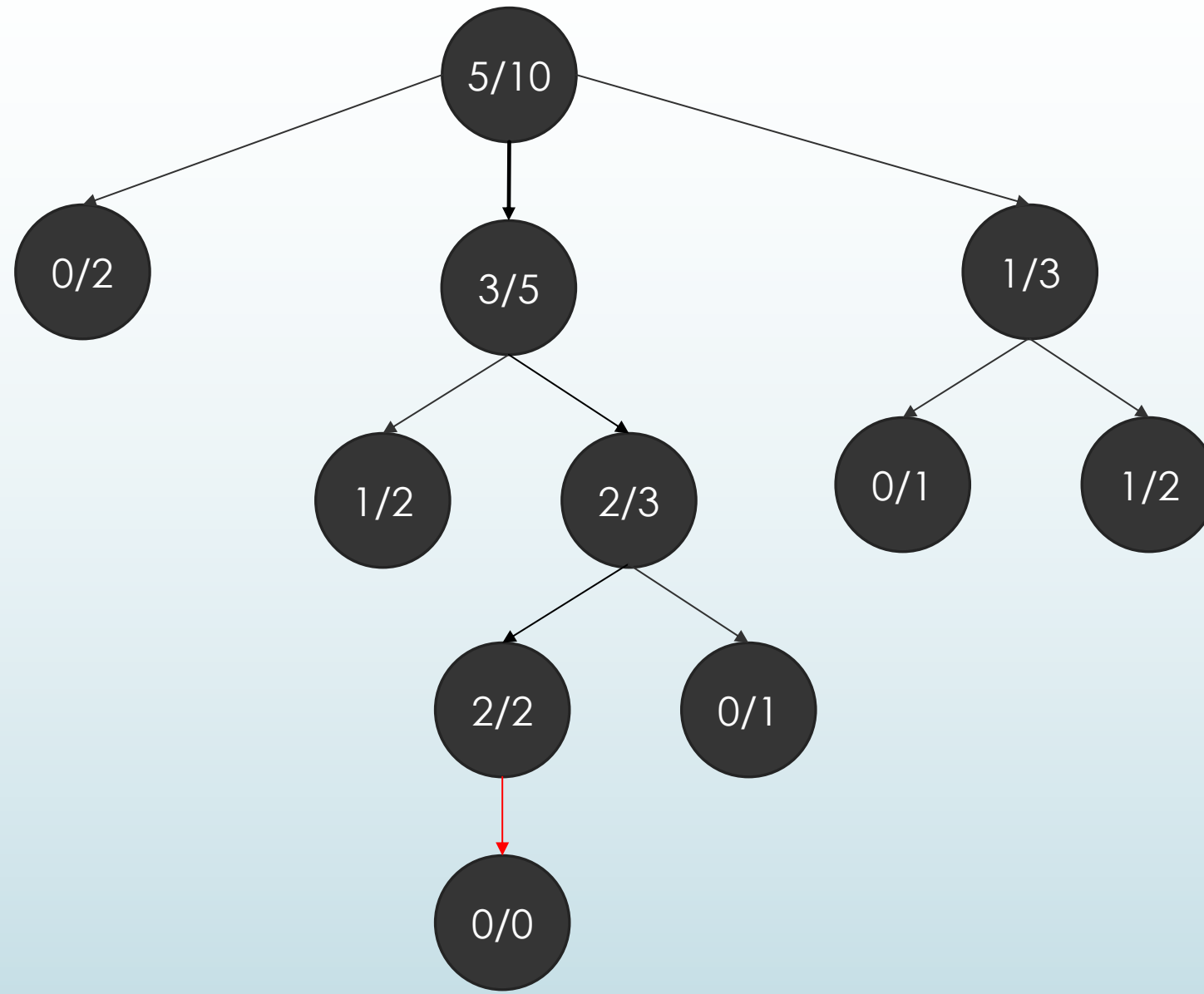
4. Backpropagation

- Use the result of the playout to update information in the nodes on the path from C to R

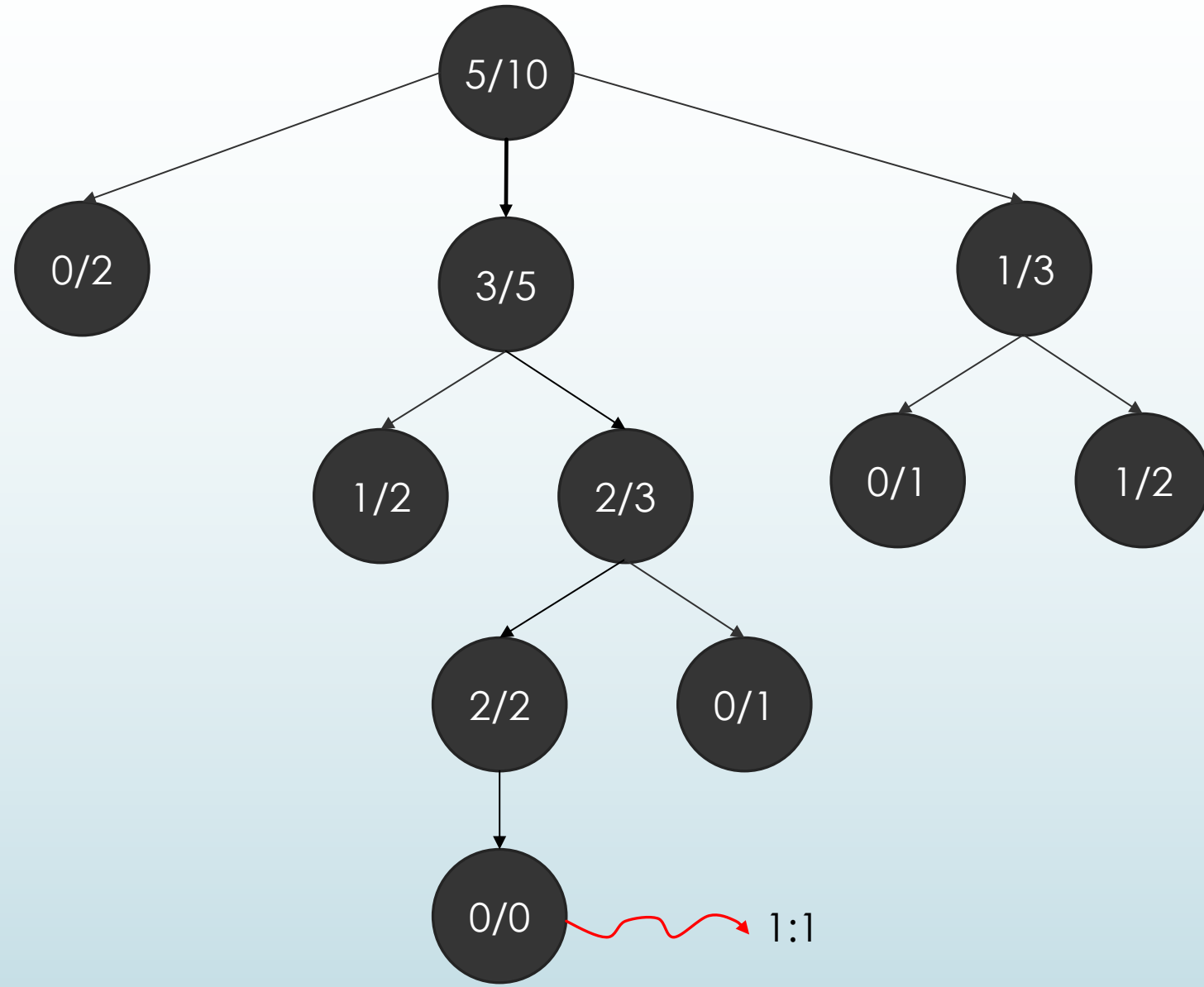
Selection



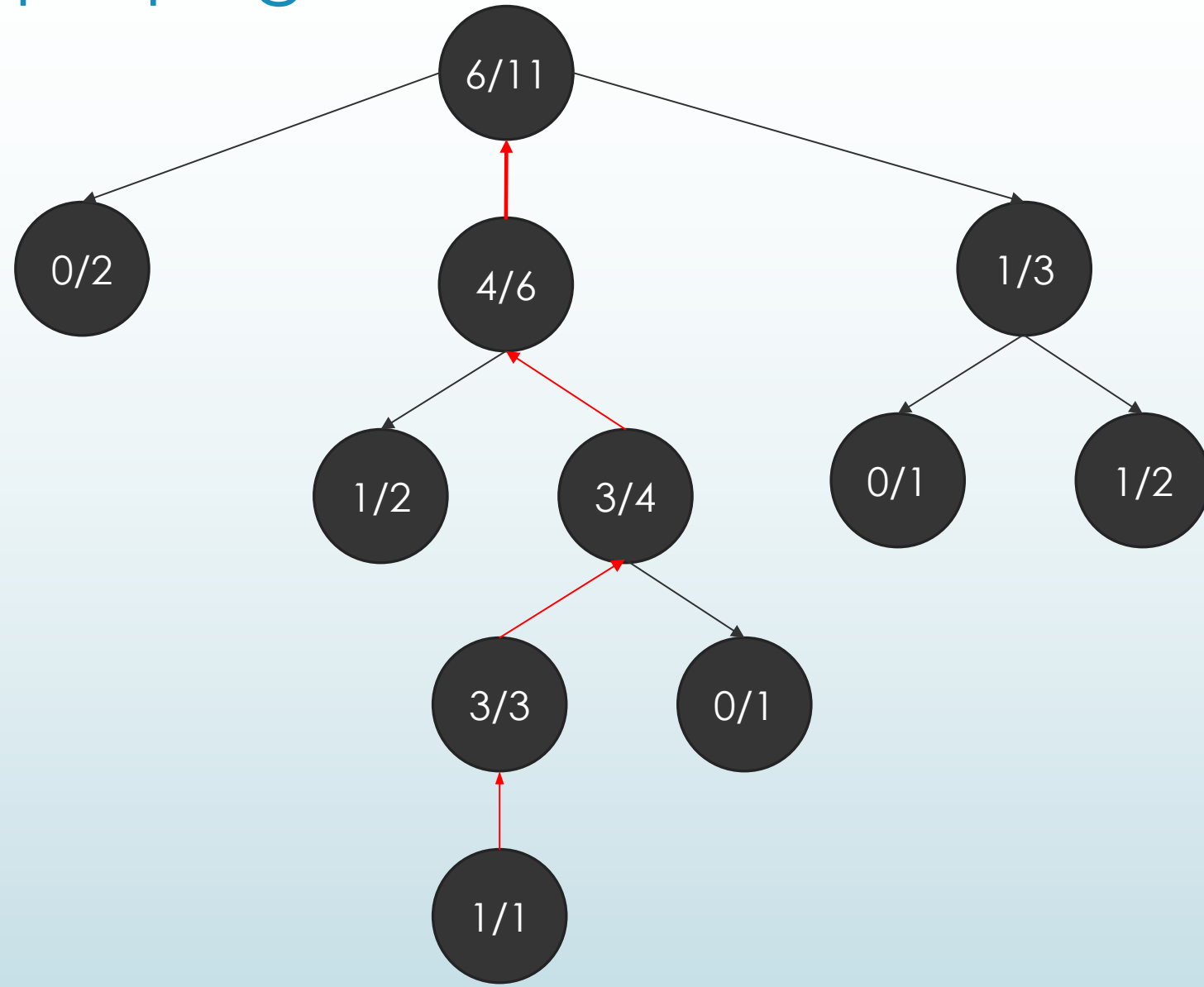
Expansion



Simulation




Backpropagation





What is the Solution?

- 
- Usually, the node with the highest denominator should be the solution
 - In other words, the node with the biggest number of simulations
 - The number of simulations should be big enough for statistical independence

Exploration vs Exploitation

- The main difficulty in selecting child nodes is maintaining some balance between the *exploitation* of deep variants after moves with high average win rate and the *exploration* of moves with few simulations
- It is recommended to select nodes with the highest value of the next formula called UCT (*Upper Confidence Bound 1 applied to trees*)

$$\frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$$

- Exploitation is the first term
- Exploration is the second term

Exploration vs Exploitation

$$\frac{w_i}{n_i} + c \sqrt{\frac{\ln N_i}{n_i}}$$

- w_i stands for the number of wins for the node considered after the i -th move
- n_i stands for the number of simulations for the node considered after the i -th move
- N_i stands for the total number of simulations after the i -th move run by the parent node of the one considered
- c is the exploration parameter — theoretically equal to $\sqrt{2}$; in practice usually chosen empirically



Advantages and Disadvantages

- Advantages

- No evaluation function that makes the algorithm ultimate for different tasks
- Achieves better results than classical algorithms in games with a high branching factor
- Shortens the search time for large tasks

- Disadvantages

- In certain positions, there may be moves that look superficially strong, but that actually lead to a loss via a subtle line of play



Parallelization

- Leaf parallelization
 - Parallel execution of many playouts from one leaf of the game tree
- Root parallelization
 - Building independent game trees in parallel and making the move based on the root-level branches of all these trees
- Tree parallelization
 - Parallel building of the same game tree, protecting data from simultaneous writes either with one, global mutex, with more mutexes, or with non-blocking synchronization



References



- [MCTS](#)
- [MCTS on Tic-Tac-Toe](#)