Monte Carlo Tree Search

Monte Carlo Tree Search (MCTS) is a search algorithm that is used to find the best moves in a two-player game by **simulating** multiple games from different starting points and using the results to inform its next moves.

MCTS uses a **probabilistic approach**, it estimates the value of a move based on the average outcome of the simulations.

MCTS is **different:** Minimax determines the best move by evaluating all possible future game states and selecting the one with the highest score.

MCTS is **useful** in situations where there is uncertainty.

The algorithm finds the best move even when there is limited information about the game state.

MCTS **does not require any heuristics** about the game. It can be used in any two-player game with well-defined rules, regardless of the complexity or strategy involved.

MCTS can figure out how to master a task **without direct guidance** from programmers.

MCTS can be **enhanced by incorporating heuristics,** if they are available.

For example, information about the **best opening moves** in a game can be incorporated into the selection step of the algorithm, biasing the algorithm towards exploring promising moves first.

Incorporating heuristics can help **speed up the convergence** of the algorithm and improve its performance.

Heuristics are **not necessary** for the basic operation of MCTS.

The four main MCTS functions

1. **Selection:** MCTS selects a node from the current game state tree to explore based on the highest value of the Upper Confidence Bound for Trees (UCB1) formula.
2. **Expansion:** If the selected node has untried moves, MCTS adds one of these untried moves as a child node of the selected node.
3. **Simulation:** MCTS simulates random playouts from the current state until the end of the game and collects the results.
4. **Update:** MCTS updates the statistics of the selected node and its ancestors with the results of the simulation.

Upper Confidence Bounds Applied to Trees

$$UCB1(n) = \frac{U(n)}{N(n)} + C * \sqrt{\frac{logN(Parent(n))}{N(n)}}$$

Exploitation

Exploration

Exploitation vs. Exploration

Exploitation:

- Using information that is already known to make the best decision.
- Selecting moves that have been estimated to have the highest value, based on the statistics collected from previous simulations.
- The goal of exploitation is to make the best move possible with the information that is available.

Exploration:

- Searching for new information about the state of the game or the value of a particular move.
- Select moves that have not been tried before and simulate the game from these moves.
- The goal of exploration is to discover new information that will allow the algorithm to make better decisions in the future.

Balancing exploration and exploitation is crucial

Too much exploration: slow convergence and poor performance

Too much exploitation: missed opportunities and suboptimal decisions

The UCB1 formula provides a **trade-off** between exploration and exploitation.

The algorithm finds a balance that is appropriate for the particular game state.

The simulation step in Monte Carlo Tree Search (MCTS) involves **randomly playing out the game from the current state** until the end of the game:

a terminal state or a pre-defined depth limit.

The goal of the simulation is to **estimate the value of a particular move**, by randomly generating playouts and averaging the results.

The simulation is done with a **bias towards moves** that have been **successful** in previous simulations.

This bias can be introduced by incorporating **information from the UCB1** formula used in the selection step.

During the simulation, the algorithm keeps track of any relevant information.

After it is complete, the result of the playout

either a **win, loss, or draw**

is used to update the statistics of the selected node and its ancestors with this information.

The simulation step is a **crucial** part of MCTS

It allows the algorithm to explore the game state space and estimate the value of each move.

The more simulations are performed, **the more accurate** the estimated values become.

This allows the algorithm to make better decisions in future iterations.

Example of MCTS at work

1. Start with the current game state and create the root node of the game state tree.
2. Select a node using the UCB1 formula and move to it.
3. If the selected node has untried moves, select one and expand it by adding a child node to the game state tree.
4. Simulate random playouts from the newly expanded node until the end of the game and collect the results.
5. Update the statistics of the selected node and its ancestors with the results of the simulation.
6. Repeat steps 2 to 5 until a stopping criterion is met (e.g., a certain number of simulations have been run or a certain amount of time has passed).
7. Select the child node of the root node with the highest average score or highest number of visits as the best move.