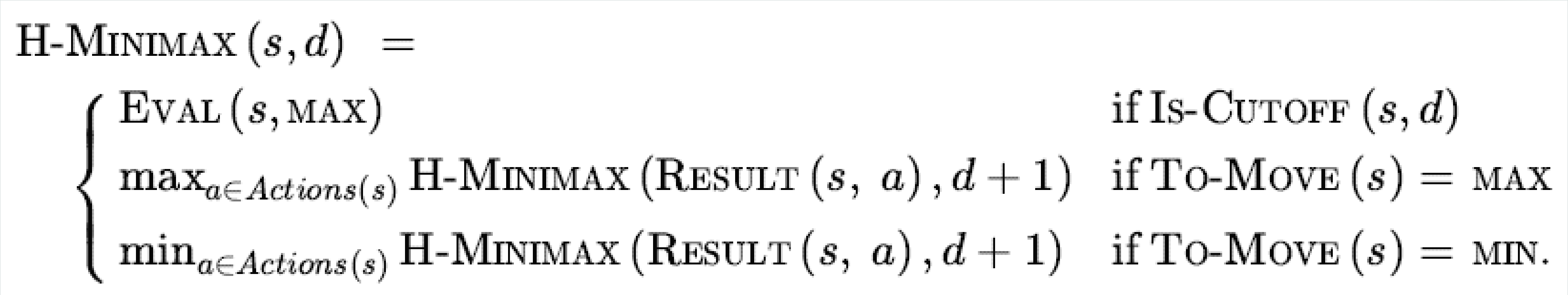
# Heuristic Alpha–Beta Tree Search

# Heuristic Alpha–Beta Tree Search

- H-MINIMAX(s, d) for the heuristic minimax value of state s at search depth d:



$$
\text{H-MINIMAX}(s,d) \;=\;
\begin{cases}
\text{EVAL}(s,\text{MAX}) & \text{if IS-CUTOFF}(s,d) \\
\max_{a\in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,\ a), d+1) & \text{if TO-MOVE}(s) = \text{MAX} \\
\min_{a\in Actions(s)} \text{H-MINIMAX}(\text{RESULT}(s,\ a), d+1) & \text{if TO-MOVE}(s) = \text{MIN}.
\end{cases}
$$

# Evaluation function

$$\text{UTILITY}(\text{loss}, p) \leq \text{EVAL}(s, p) \leq \text{UTILITY}(\text{win}, p)$$

- Good evaluation function?

1. Computation must not take too long!
2. The evaluation function should be strongly correlated with the actual chances of winning.

# Monte Carlo tree search (MCTS)

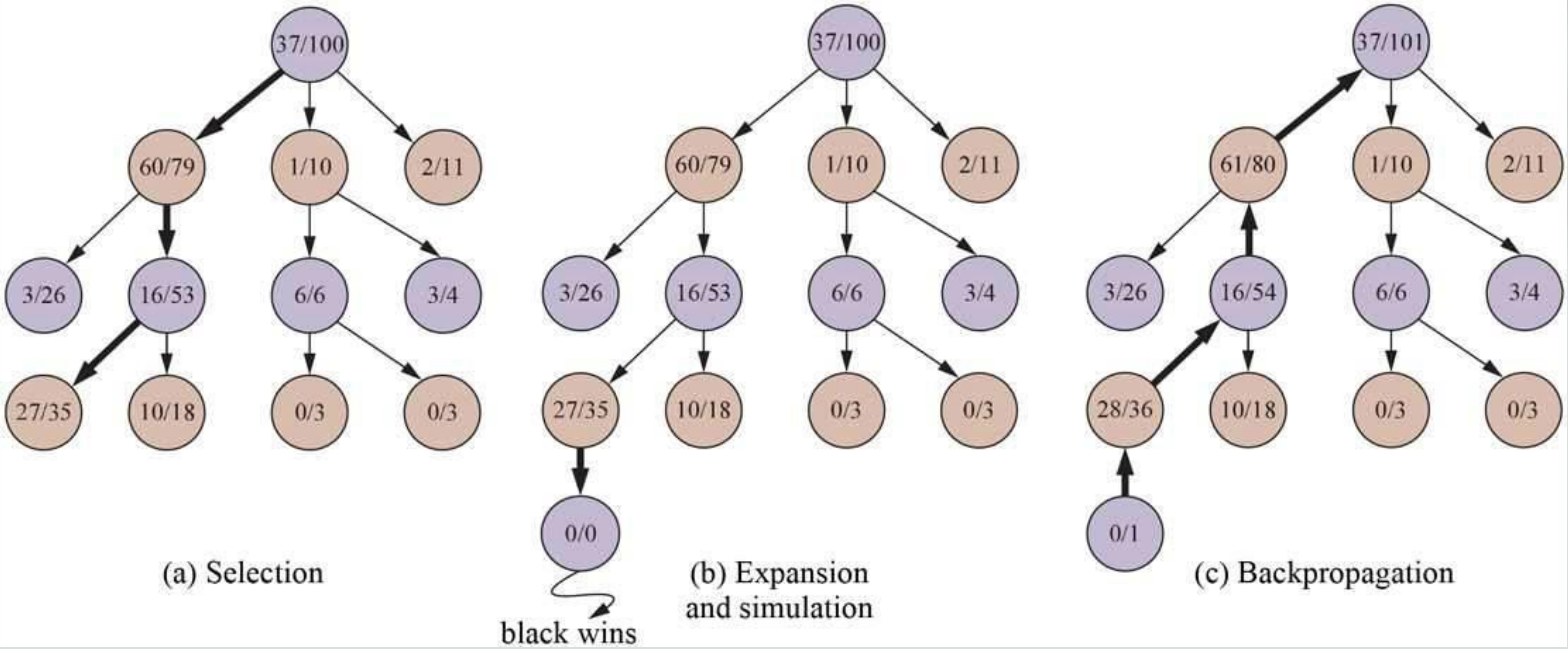
- The basic MCTS strategy does not use a heuristic evaluation function.
- The value of a state is estimated as the average utility over a number of simulations of complete games starting from the state.
- A simulation (also called a playout or rollout) chooses moves first for one player, than for the other, repeating until a terminal position is reached.
- Playout policy that biases the moves towards good ones.
- From what positions do we start the playouts, and how many playouts do we allocate to each position?
- Pure Monte Carlo search, is to do N simulations starting from the current state of the game, and track which of the possible moves from the current position has the highest win percentage.

# Monte Carlo tree search (MCTS)

- Selection policy
  - Exploration
  - Exploitation

- Selection
- Expansion
- Simulation
- Back-Propagatoin

# Monte Carlo tree search (MCTS)



(a) Selection

(b) Expansion and simulation

(c) Backpropagation

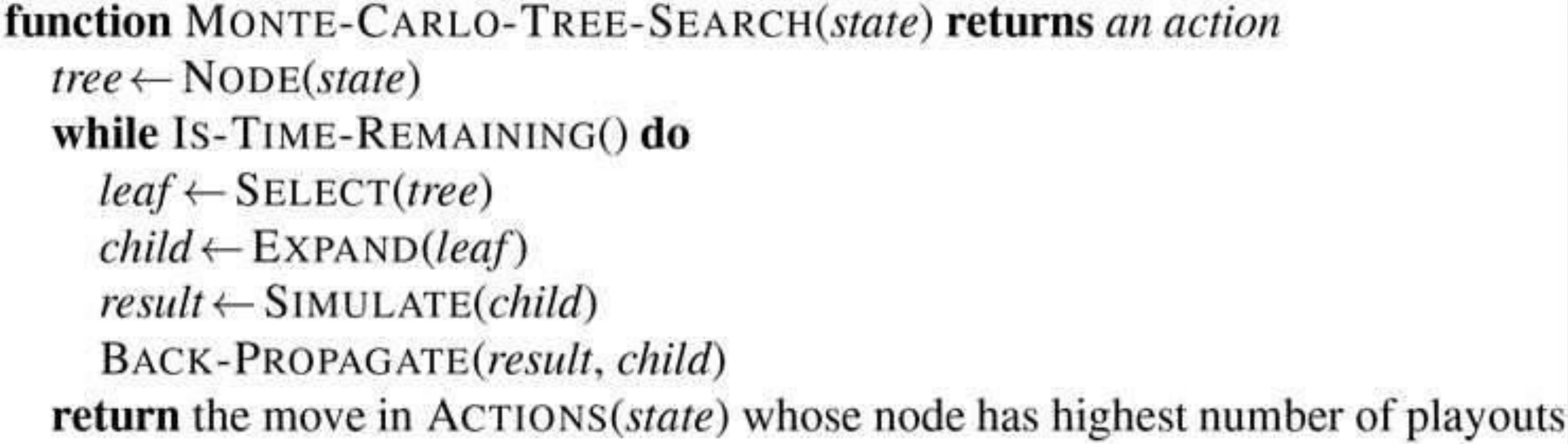
# Selection Policy

- Upper confidence bounds applied to trees

$$UCB1(n) = \frac{U(n)}{N(n)} + C \times \sqrt{\frac{\log N(\textsc{Parent}(n))}{N(n)}}$$

- U(n) is the total utility of all playouts that went through node n.
- N(n) is the number of playouts through node n.
- PARENT(n) is the parent node of n in the tree.
- C is a constant that balances exploitation and exploration.

# Monte Carlo tree search (MCTS)



```
function MONTE-CARLO-TREE-SEARCH(state) returns an action
    tree ← NODE(state)
    while IS-TIME-REMAINING() do
        leaf ← SELECT(tree)
        child ← EXPAND(leaf)
        result ← SIMULATE(child)
        BACK-PROPAGATE(result, child)
    return the move in ACTIONS(state) whose node has highest number of playouts
```