CS 4700: Foundations of Artificial Intelligence

ONLINE

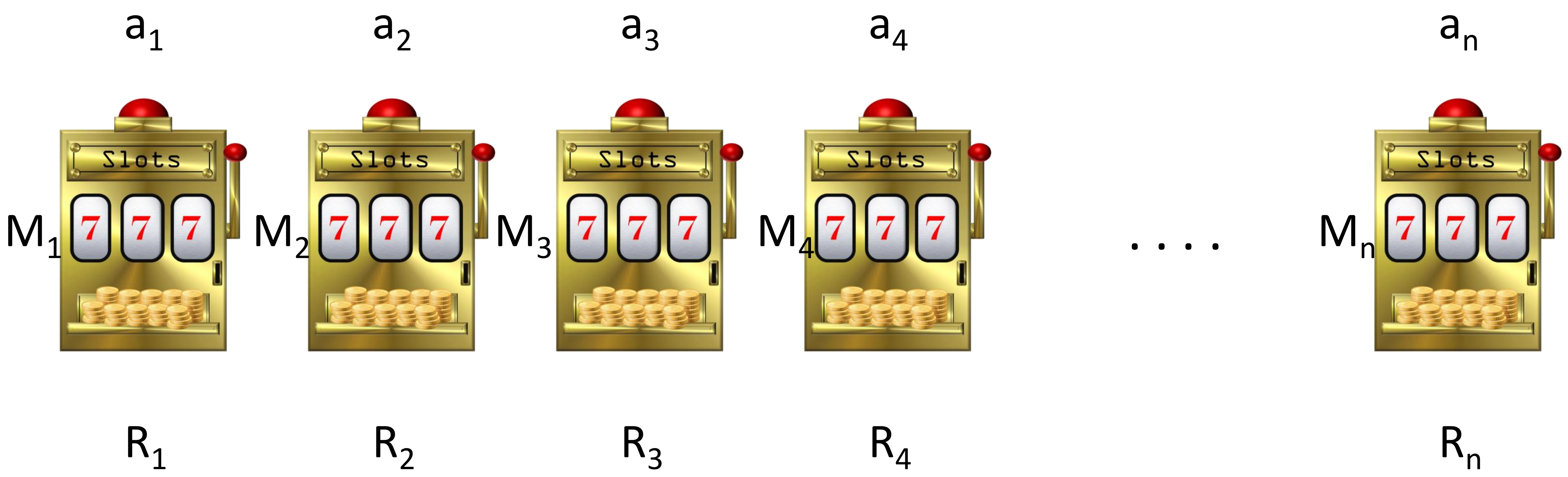
Spring 2020
Prof. Haym Hirsh

Lecture 24
April 10, 2020

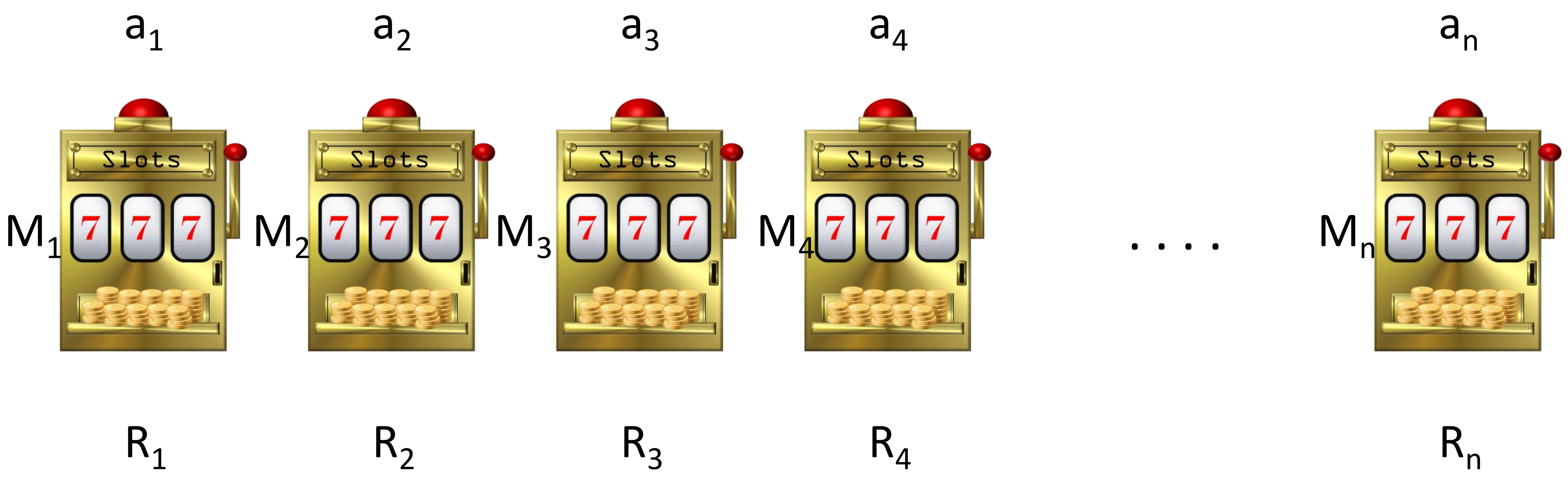
Backup plans:

If this Zoom meeting ends prematurely,
five-minute break, check Piazza

Multi-Armed Bandit



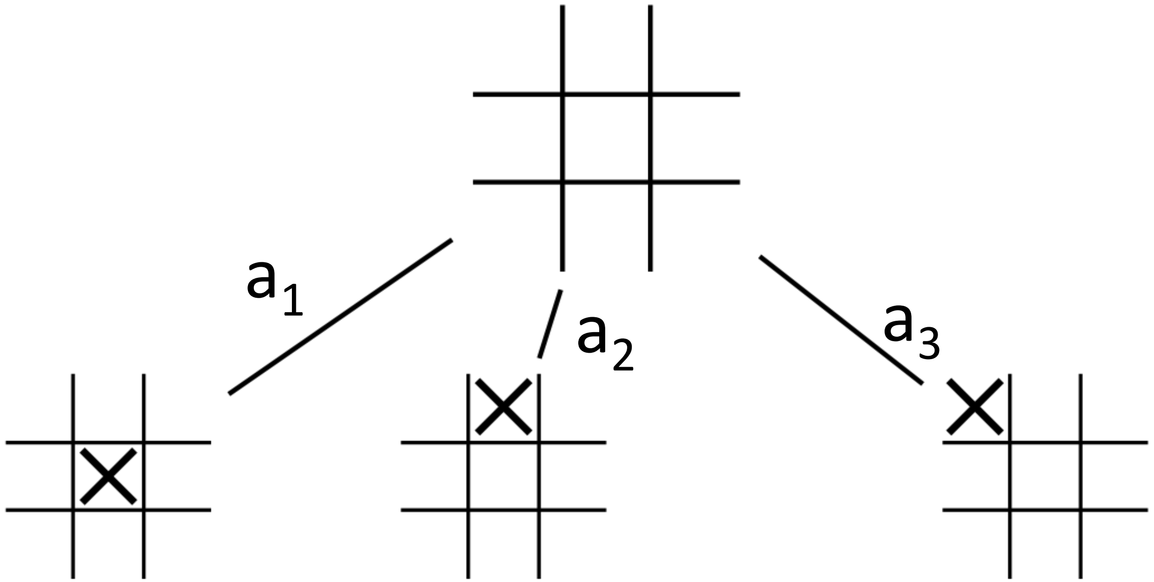
Multi-Armed Bandit for Game Tree Search



Key ideas of Monte Carlo Tree Search:

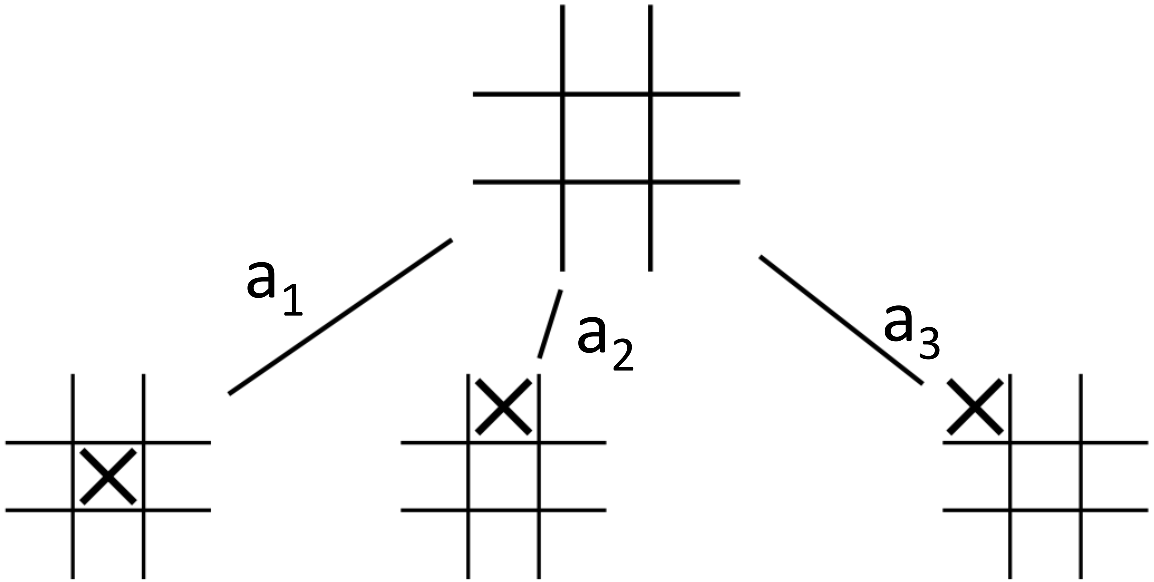
1. View move selection as a multi-armed bandit problem
2. Evaluate moves by simulating games

Multi-Armed Bandit for Game Tree Search



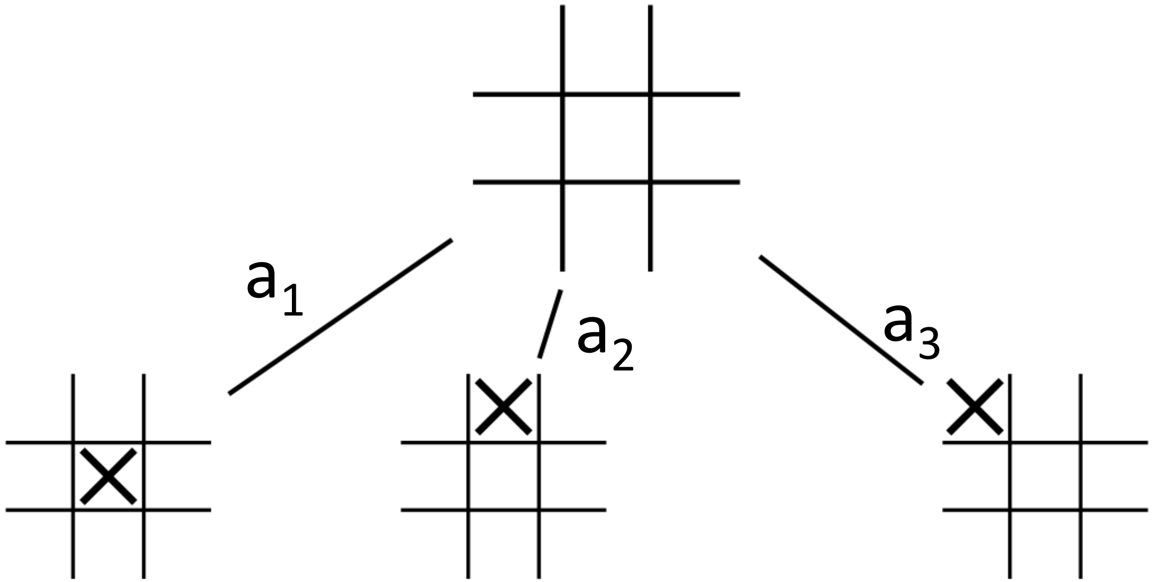
What move should I try?

Multi-Armed Bandit for Game Tree Search



Simulate games with each "arm"

Multi-Armed Bandit for Game Tree Search



What move should I try on each simulated game?

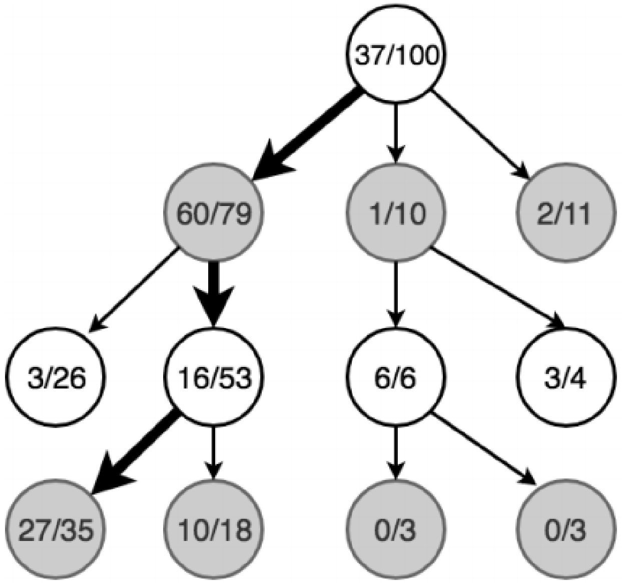
Monte-Carlo Tree Search (MCTS) Terms

- Leaf node:
  A state in the game tree that has successors for which no games have been simulated
  (has one or more “arms” that have never been pulled)
- Terminal node: End of game state
- Playout/rollout: Simulating a game from a leaf node to a terminal node

Three Steps in MCTS

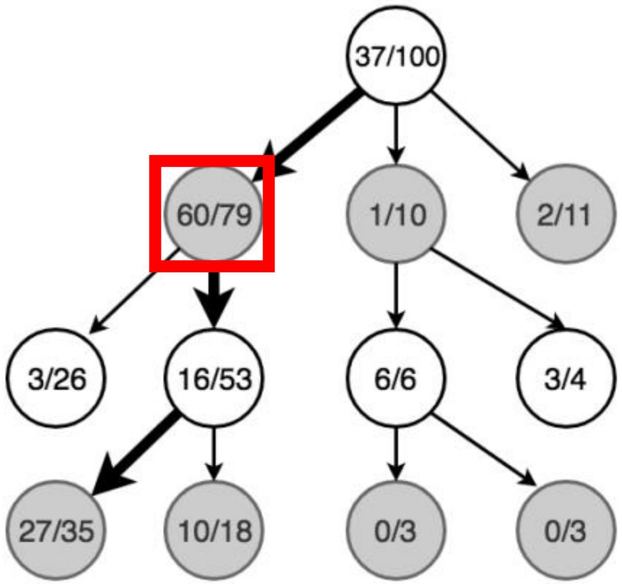
- Selection: Make move choices until a leaf node S is reached

Three Steps in MCTS



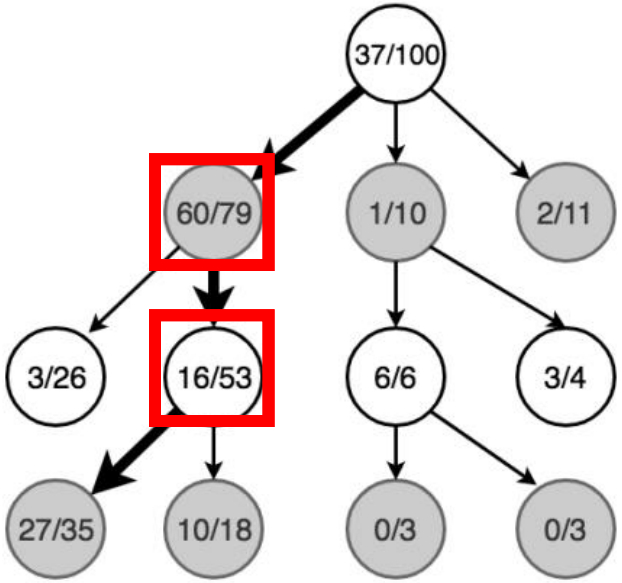
(a) Selection

Three Steps in MCTS



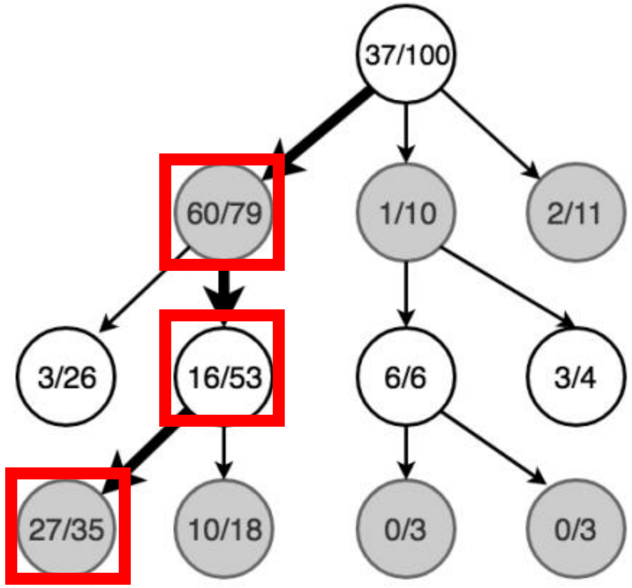
(a) Selection

Three Steps in MCTS



(a) Selection

Three Steps in MCTS



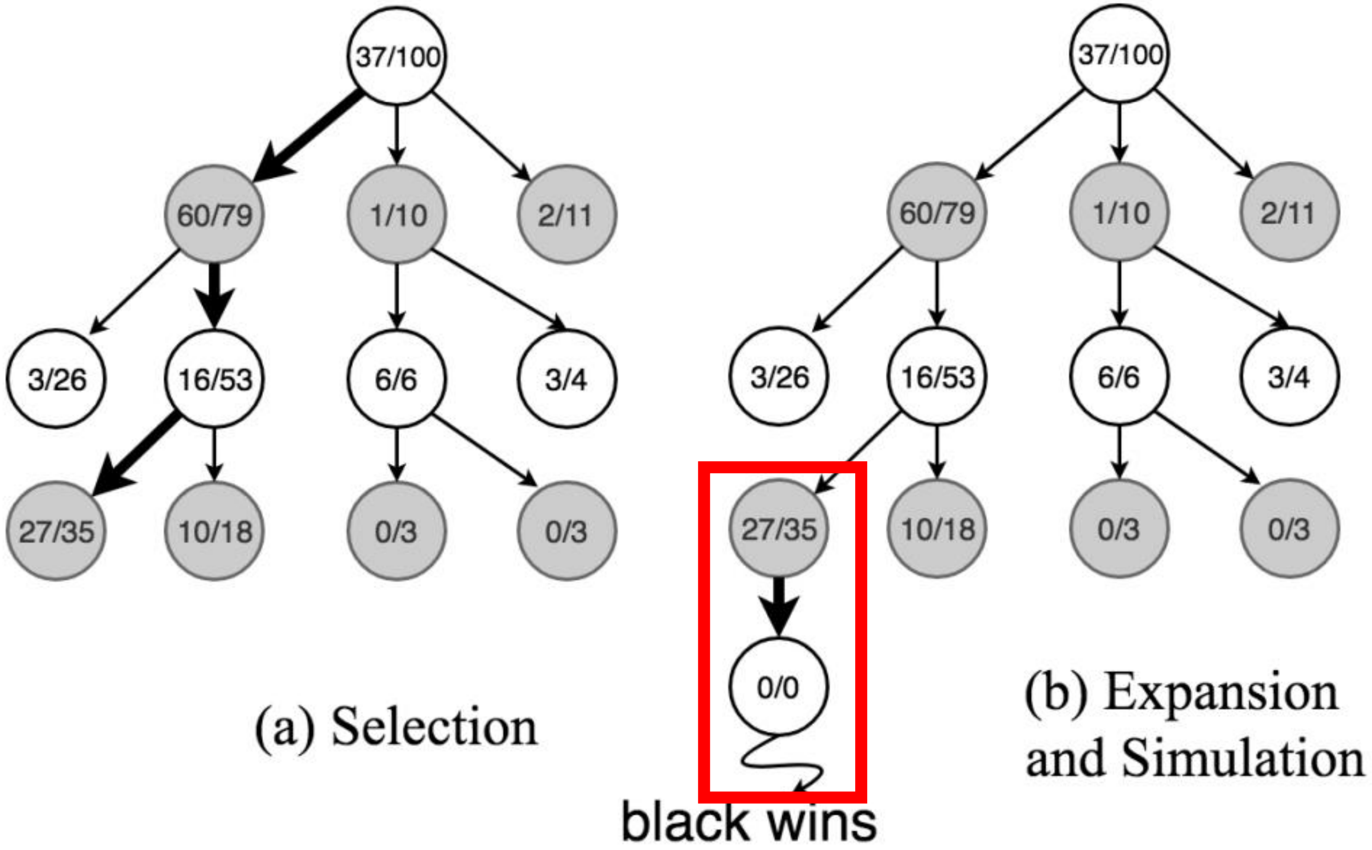
(a) Selection

Three Steps in MCTS

- Selection: Make move choices until a leaf node S is reached
- Expansion: Create a new successor state S’ for an untried action

  Simulation: Play a game until you reach a terminal node

Three Steps in MCTS

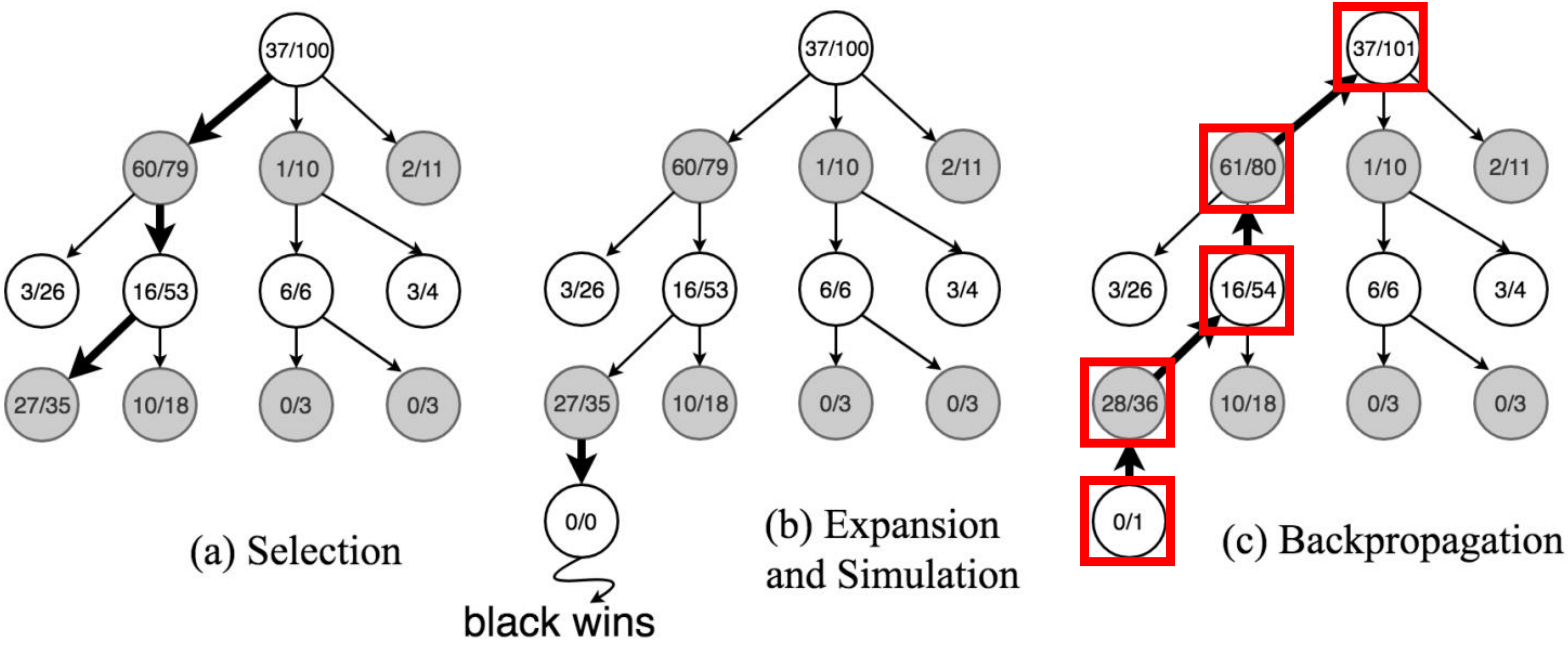


Three Steps in MCTS

- Selection: Make move choices until a leaf node S is reached
- Expansion: Create a new successor state S' for an untried action

  Simulation: Play a game until you reach a terminal node
- Backpropagation: Update game statistics for the path from S' up to the root

Three Steps in MCTS



MCTS(state):

```
while TIME-REMAINING() do
        leaf ← SELECT(tree)
        child ← EXPAND(leaf)
        result ← SIMULATE(child)
        BACKPROPAGATE(result, child)
return argmax_{a∈A} #playouts(apply(a,state))
```

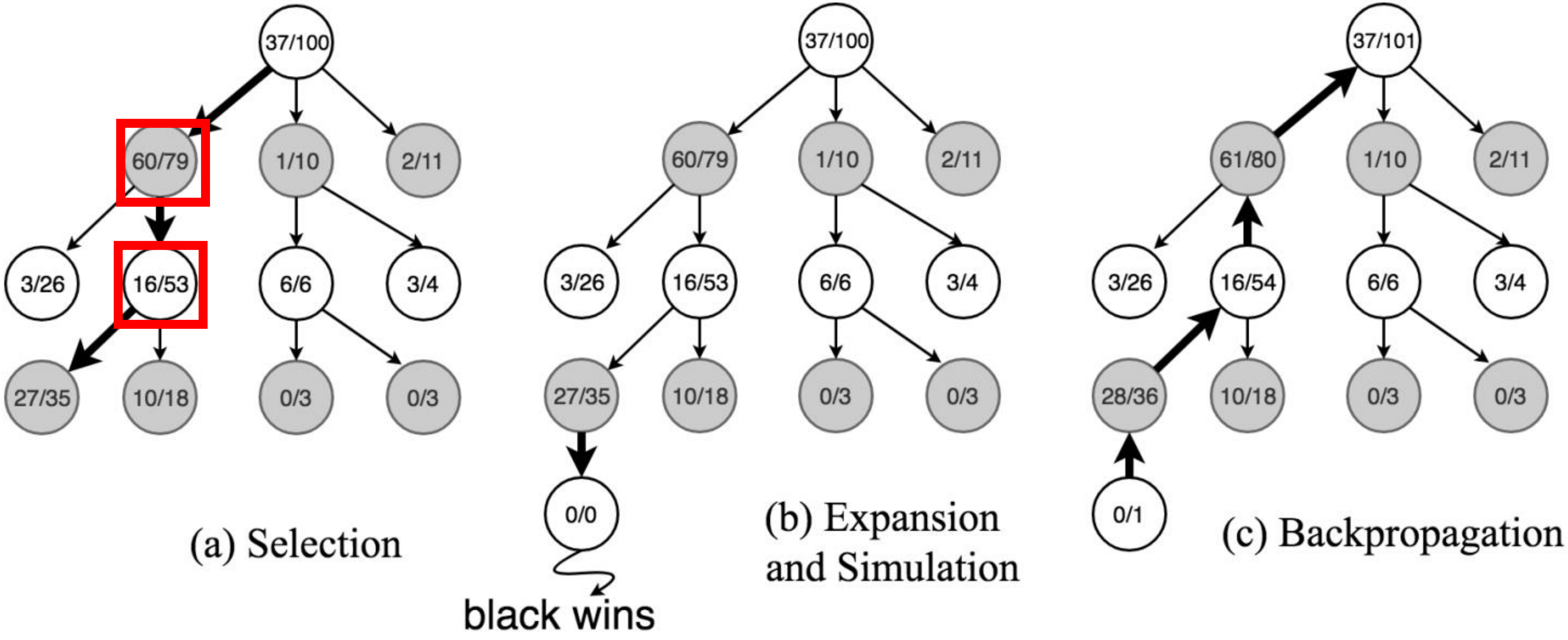
```
MCTS(state):
while TIME-REMAINING() do
        leaf ← SELECT(tree)
        child ← EXPAND(leaf)
        result ← SIMULATE(child)
        BACKPROPAGATE(result, child)
return argmax #playouts(apply(a,state))
        a∈A
```

$$\text{return } \underset{a \in A}{\operatorname{argmax}} \#\text{playouts}(\text{apply}(a,\text{state}))$$

Which move gives the game state with most playouts

Three Steps in MCTS

How do we pick moves?



Remember This? (UCB)

Algorithm:

Pull each arm once

For i ← 1 to n { Sum_i ← R(arm_i); $N_i \leftarrow 1$ }; $N \leftarrow n$ /* Initialization */

Loop Forever

$$\text{best} \leftarrow \underset{1 \le i \le n}{\operatorname{argmax}} \left[\frac{\mathrm{Sum}_i}{N_i} + c\sqrt{\frac{\ln N}{N_i}} \right]$$

pull arm a_{best} and get reward r

$\mathrm{Sum}_{best} \leftarrow \mathrm{Sum}_{best} + r$; $N_{best} \leftarrow N_{best} + 1$; $N \leftarrow N+1$

Picking a Move During Selection and Expansion (UCT – Upper Confidence bound applied to Trees)

Sum_i = # of wins

N_i = # of times i was tried

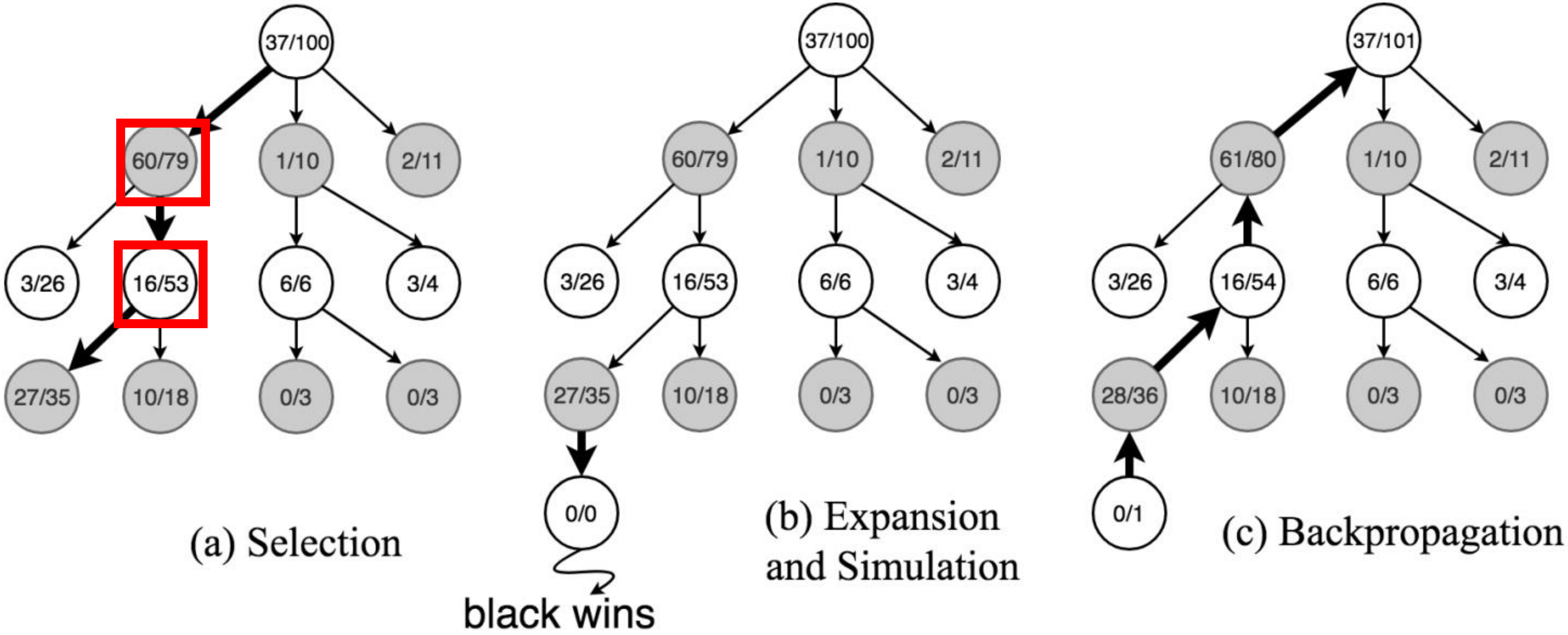
N = # of simulations thus far (N(parent(i))

$$\text{best} \leftarrow \underset{1 \le i \le n}{\operatorname{argmax}} \left[\frac{Sum_i}{N_i} + c\sqrt{\frac{\ln N}{N_i}} \right]$$

Lets you control how much exploration

Three Steps in MCTS

How do we pick moves?



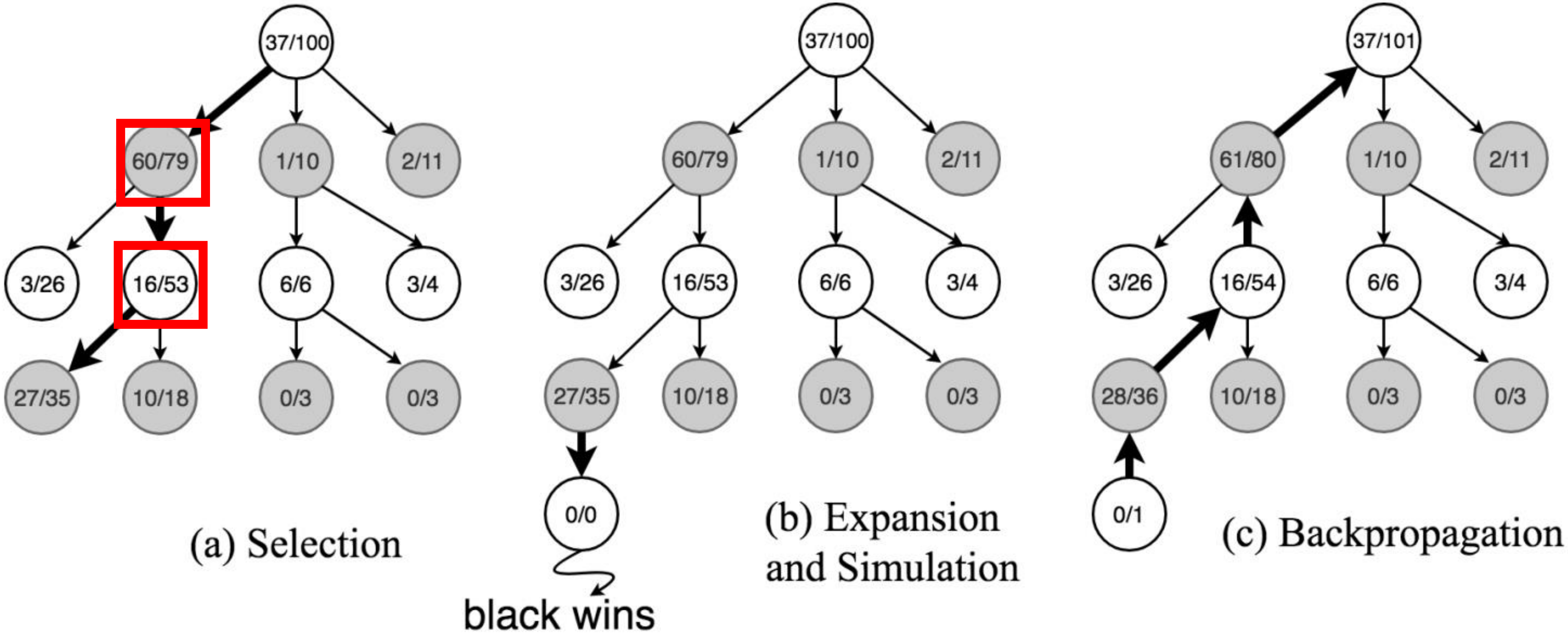
(a) Selection

(b) Expansion and Simulation

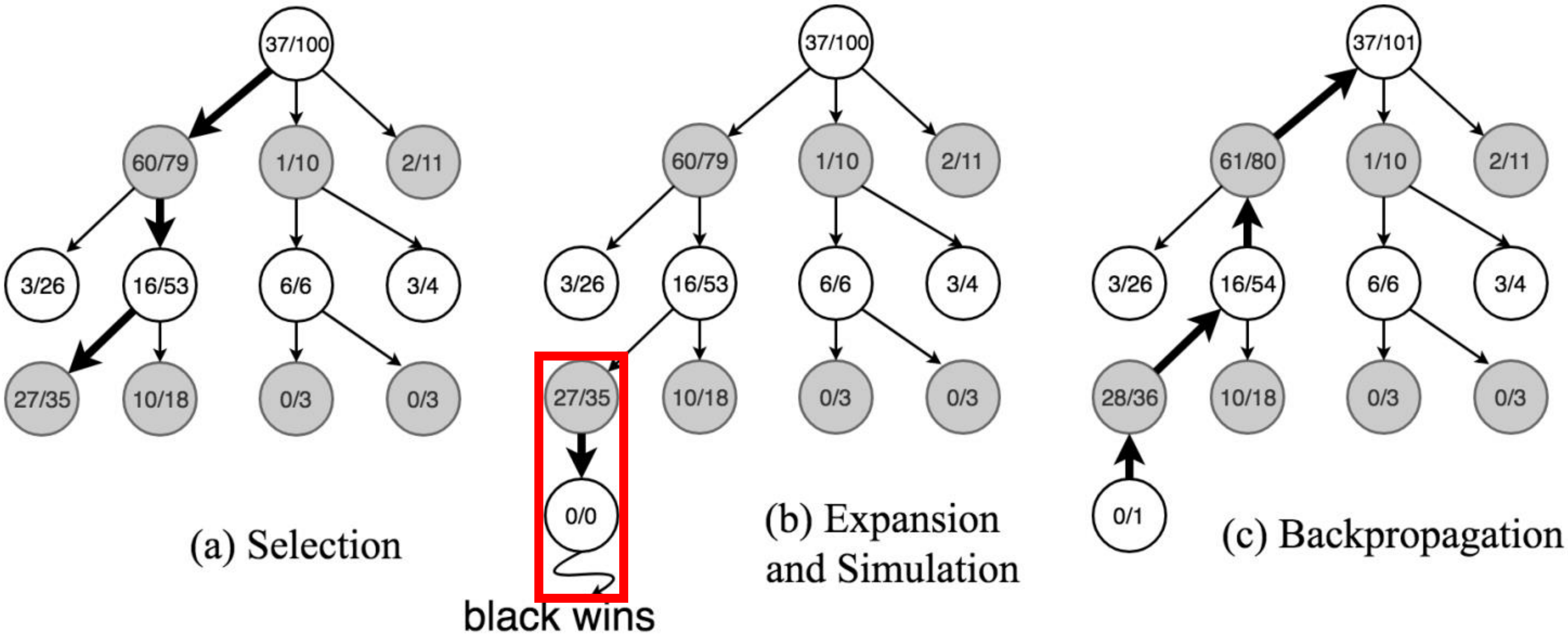
(c) Backpropagation

Three Steps in MCTS

How do we pick moves?
UCB heuristic



Three Steps in MCTS



How do we pick moves?

Picking a Move During Simulation

- Light playout: Pick uniformly at random
- Heavy playout: Make a biased selection
  - Simulation statistics
  - Game knowledge

Trade off: Slower run time vs missing a move

Benefits

- Doesn’t use an evaluation function!
- Time is linear in depth
- Handles large number of actions
- Let’s you make a move when a timer goes off (to manage time)

(“anytime algorithm”)