Overview

Mr. X plays a hidden-information evasion game against several detectives. He knows all detectives' positions but does not know their future moves. A Monte Carlo approach estimates the long-term value of candidate actions by simulating many random futures and averaging outcomes.

Monte Carlo principle

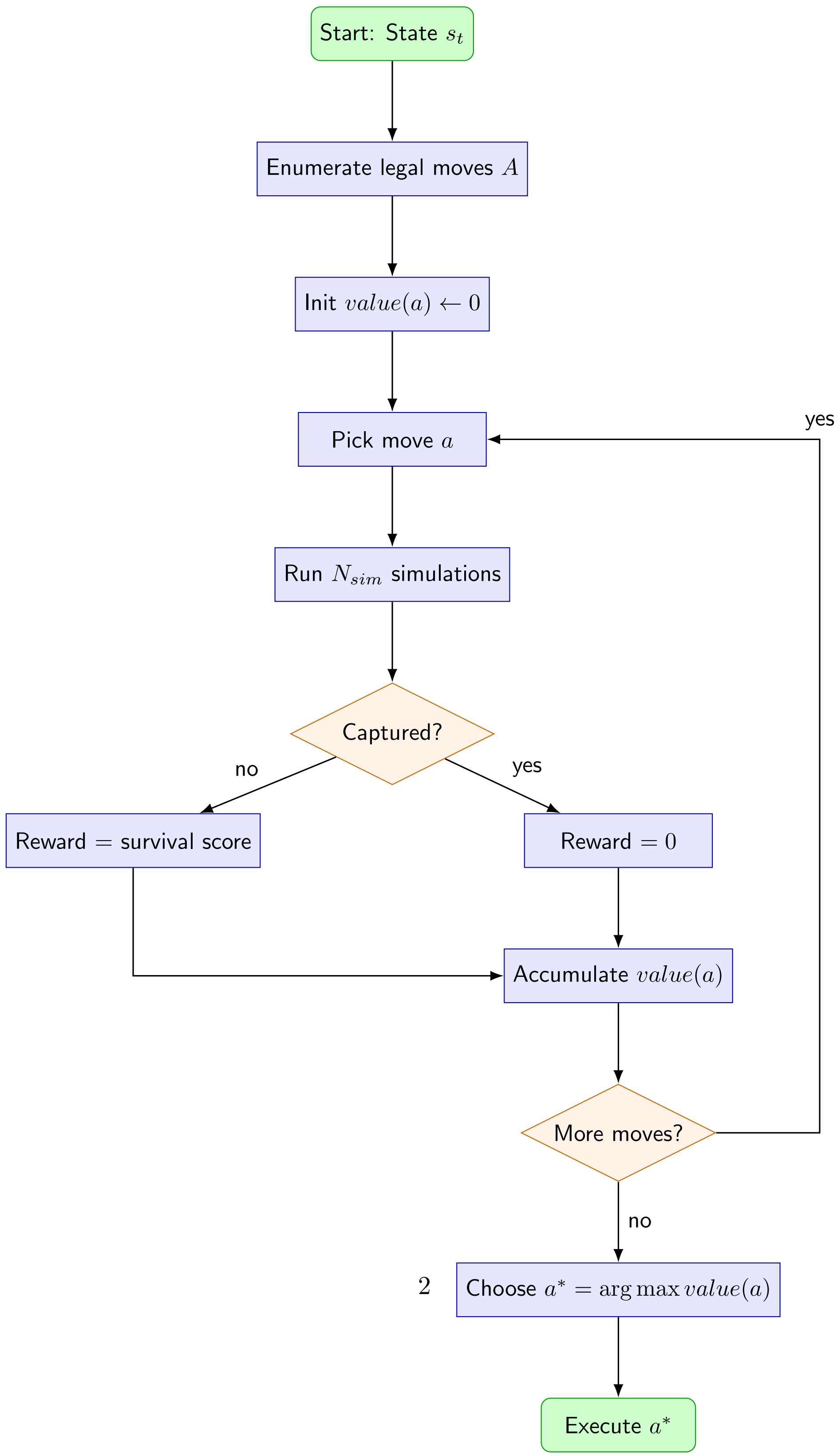
For each legal move a available to Mr. X at state s_t:

$$V(a) \approx \frac{1}{N} \sum_{i=1}^{N} R_i(a)$$

where $R_i(a)$ is the terminal reward (e.g., survival indicator or distance-based utility) obtained after a simulated sequence of moves in rollout i. The move with highest $V(a)$ is chosen:

$$a^* = \arg\max_a V(a)$$

Block diagram



Pseudocode

Algorithm 1 Monte Carlo decision for Mr. X

```
procedure MCDecision(s_t, N_sim, H)
    A ← legal moves for Mr. X from s_t
    for each a ∈ A do
        value(a) ← 0
        for i = 1 to N_sim do
            s ← SimulateStep(s_t, a)                          ▷ apply a
            for h = 1 to H do
                Mr. X move ← HeuristicX(s)
                Detectives move ← SampleDetectiveMoves(s)
                s ← ApplyMoves(s, Mr. X move, Detectives move)
                if IsCaptured(s) then
                    reward ← 0; break
                end if
            end for
            if not captured then reward ← EvaluateSurvival(s)
            end if
            value(a) ← value(a) + reward
        end for
        value(a) ← value(a)/N_sim
    end for
    return arg max_a value(a)
end procedure
```

Components

SampleDetectiveMoves: Generate random yet plausible detective actions. Each detective moves towards the last known or most probable position of Mr. X using shortest paths plus noise factor ϵ.

HeuristicX: Simple rule for Mr. X during rollouts (e.g., prefer moves increasing total distance to nearest detective or leading to high-degree nodes).

EvaluateSurvival: Returns 1 if Mr. X survives the horizon, or a continuous score such as average distance to nearest detective.

Monte Carlo Tree Search

A more powerful variant keeps statistics $(N(s,a), Q(s,a))$ over visited states and uses the UCT selection rule:

$$a^* = \arg\max_a \left[Q(s,a)/N(s,a) + C\sqrt{\tfrac{\ln N(s)}{N(s,a)}} \right]$$

This allows deeper reasoning without full enumeration of all futures. Each simulation consists of:

1. **Selection:** follow UCT until an unvisited state is found.
2. **Expansion:** add new node.
3. **Simulation:** rollout using stochastic detective policy.
4. **Backpropagation:** update Q and N.

Reward design examples

$$R = \begin{cases} 0 & \text{if captured during rollout,} \\ 1 & \text{if still free after } H \text{ steps,} \\ \lambda \cdot d_{\min} & \text{(optional distance-based bonus).} \end{cases}$$

Practical parameters

- N_{sim}: 50-500 simulations per move.
- Horizon H: 6-10 turns.
- Exploration constant C: 0.5 (for MCTS).
- Reward discount λ: 0.1-0.3 for distance bonus.

Algorithm 2 Monte Carlo Tree Search (MCTS) for Mr. X

Initialize root node with current game state s_t
for $i = 1$ to N_{sim} **do**
    $node \leftarrow$ root   ▷ Start traversal from current state

    while *node* is fully expanded **and** not terminal **do**
        $a \leftarrow \arg\max_a \left[\frac{Q(node,a)}{N(node,a)} + C\sqrt{\frac{\ln N(node)}{N(node,a)}} \right]$
        $node \leftarrow$ child reached by applying a
    end while

    if *node* not terminal **then**
        Expand *node* by generating all legal moves for Mr. X
        Choose one new child $node'$ at random
    else
        $node' \leftarrow node$
    end if

    $reward \leftarrow$ **Simulate**$(node')$ ▷ Rollout using stochastic detective behavior up to horizon H

    while $node'$ not null **do**
        $N(node') \leftarrow N(node') + 1$
        $Q(node') \leftarrow Q(node') + reward$
        $node' \leftarrow$ parent of $node'$
    end while
end for

Choose final move

$$a^* = \arg\max_a \frac{Q(root, a)}{N(root, a)}$$

Execute a^*