Monte Carlo Movement Algorithm for Detectives

1. Verbal Explanation

The **Monte Carlo Movement Algorithm** is designed to estimate the best possible move for each detective based on probabilistic simulations. Since Mr. X's position is unknown between reveals, the algorithm uses random sampling and repeated simulations to approximate which move leads to the highest probability of capturing Mr. X or reducing the average distance to him.

Core idea: Each possible first move of a detective is evaluated by running multiple randomized simulations (*rollouts*), where both detectives and Mr. X move randomly for a limited number of future turns. The outcomes of these simulations (captures, distances, etc.) are statistically analyzed to assign a score to each move.

Safety condition: Not applicable directly, but all simulations must use legal moves according to ticket and transport rules.

Core logic:

1. **Initialization:**
   - Start from the current positions of all detectives.
   - Obtain the last known position of Mr. X and the number of turns since he was revealed.
   - Generate all possible reachable positions for Mr. X after that many turns.
   - Define the number of simulations per move (*e.g.* 200) and the lookahead depth (*e.g.* 5 turns).
2. **Monte Carlo Simulation Loop:**
   - For each detective, for each of their possible first moves:
     (a) Perform N simulations (randomized playouts).
     (b) In each simulation:
       – Randomly choose a possible initial position of Mr. X.
       – Simulate 5 turns ahead:
         i. All detectives move randomly (obeying ticket rules).
         ii. Mr. X moves randomly as well.
         iii. If a detective reaches Mr. X's node, record a capture and end the simulation early.
       – Measure key outcomes:
         * Whether Mr. X was captured.
         * Average and minimal distances between detectives and Mr. X.
     (c) Aggregate results across simulations to compute average statistics.
3. **Evaluation Function:**
   - For each move, compute a score:

$$score(move) = w_1 \cdot P(\text{capture}) - w_2 \cdot \text{avgDistance}$$

   - The weights w_1 and w_2 determine the balance between capture probability and proximity pressure.
4. **Move Selection:**
   - Choose the move with the highest Monte Carlo score.
   - Execute it as the next detective move in the actual game.

2. Flowcharts

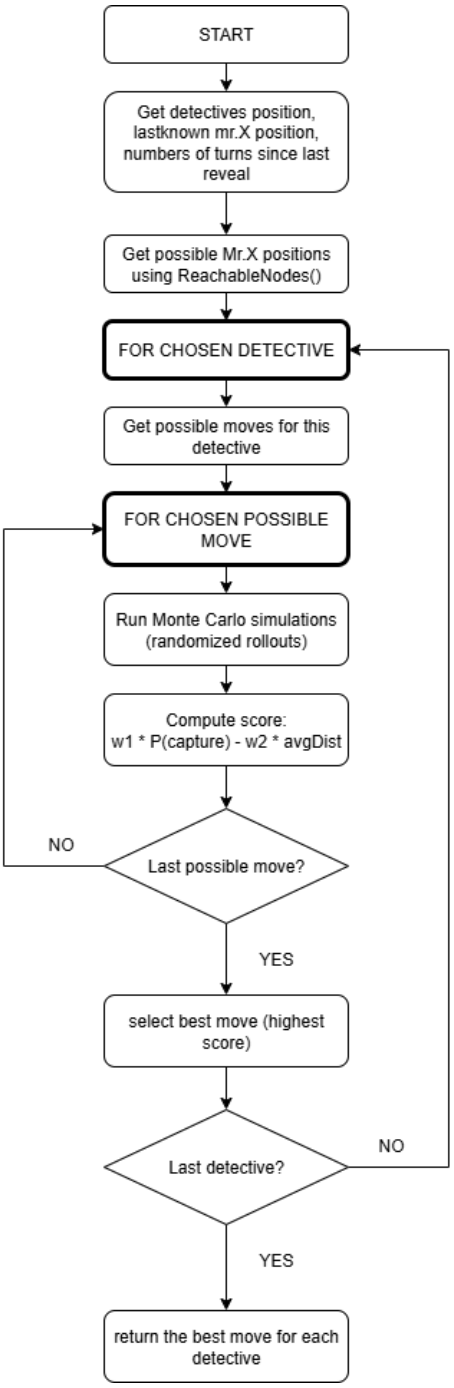

Figure 1: Overall Monte Carlo Simulation Flowchart for Detectives

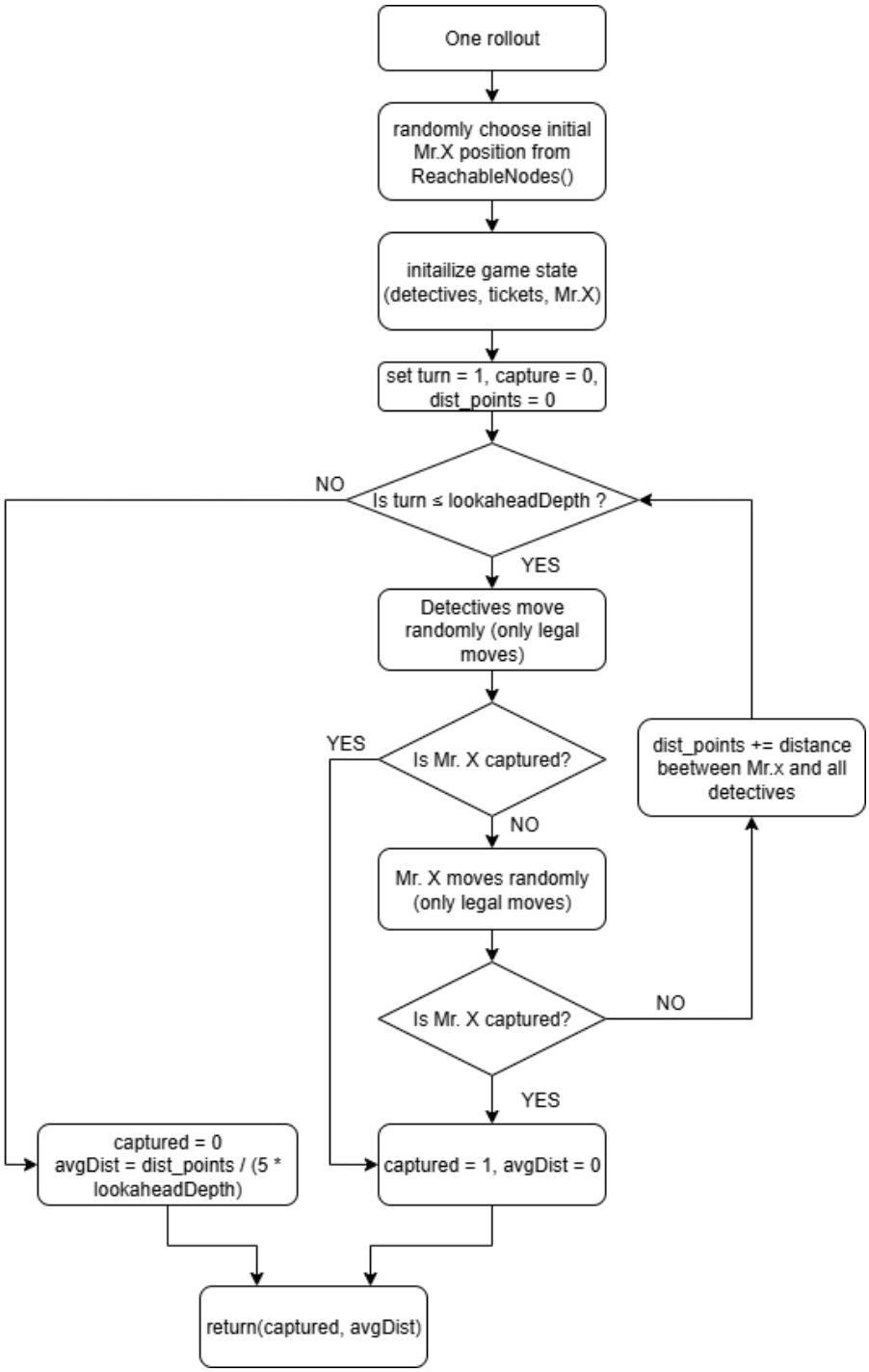

Figure 2: Detailed Flowchart of One Monte Carlo Rollout

3. Pseudocode

Algorithm 1: Monte Carlo Movement Algorithm for Detectives (aligned with rollout and main flow diagrams)

Input: $posDetectives$, $lastKnownPos$, $turnsSinceReveal$, $graph$, parameters N, $lookaheadDepth$, w_1, w_2
Output: Best move for each detective

Function `Rollout`($graph$, $lookaheadDepth$)**:**
  randomly choose initial $mrXPos$ from ReachableNodes();
  initialize game state: detectives, tickets, $mrXPos$;
  set $turn = 1$, $captured = 0$, $distPoints = 0$;
  while $turn \leq lookaheadDepth$ **do**
    Detectives move randomly (only legal moves);
    if *Mr. X is captured* **then**
      return $(captured = 1,\ avgDist = 0)$;
    end
    Mr. X moves randomly (only legal moves);
    if *Mr. X is captured* **then**
      return $(captured = 1,\ avgDist = 0)$;
    end
    $distPoints$ += distance$(Mr.X, Detectives)$;
    $turn$ += 1;
  end
  $captured = 0$;
  $avgDist = \frac{distPoints}{5 \times lookaheadDepth}$;
  return $(captured, avgDist)$;
Function `MonteCarloEvaluate`($initialMove$, $detectiveID$, $graph$, N, $lookaheadDepth$, w_1, w_2)**:**
  $results \leftarrow []$;
  for $i = 1$ **to** N **do**
    $(captured, avgDist) \leftarrow$ `Rollout`$(graph, lookaheadDepth)$;
    append $(captured, avgDist)$ to $results$;
  end
  $P_{capture} \leftarrow \frac{\text{count}(captured=1)}{N}$;
  $\overline{avgDist} \leftarrow \text{mean}(avgDist)$;
  $score \leftarrow w_1 \cdot P_{capture} - w_2 \cdot \overline{avgDist}$;
  return $score$;
foreach *detective in posDetectives* **do**
  $bestMove \leftarrow None$;
  $bestScore \leftarrow -\infty$;
  Get possible moves for this detective from $graph$;
  foreach *move in possible moves* **do**
    $score \leftarrow$ `MonteCarloEvaluate`$(move, detective, graph, N, lookaheadDepth, w_1, w_2)$;
    if $score > bestScore$ **then**
      $bestScore \leftarrow score$;
      $bestMove \leftarrow move$;
    end
  end
  assign $bestMove$ as chosen move for this detective;
end