# Front Search and Encirclement for Detectives

## 1. Concept Overview

The **Front Search / Encirclement** strategy positions detectives along the expected frontier of Mr. X's escape routes. The algorithm maintains a probability distribution over unseen positions, projects likely motion sequences from the last reveal, and drives pawns toward the averaged centre of mass of the distribution while preserving coverage of high-probability exits.

**Belief model:**

Beliefs are updated by simulating all transport combinations Mr. X could have used since the last visible turn. Ticket counts and banned transports shrink the hypothesis space. Each node gets a probability weight proportional to the number of consistent trajectories reaching it.

**Operational phases:**

1. **Belief update:**
   - On each hidden turn, expand the frontier by applying all legal transport steps to each probable node.
   - Multiply probabilities by transport usage likelihood (taxi ¿ bus ¿ underground unless black tickets are known).
   - Normalize the map and discard nodes below a configurable threshold.
2. **Front determination:**
   - Compute the weighted centroid of the probability cloud.
   - Extract the top-$k$ most probable nodes to form the encirclement set.
   - Build Voronoi regions assigning each top node to the nearest detective to avoid overlapping pursuits.
3. **Movement planning:**
   - For each detective, generate shortest constrained paths toward its assigned region using ticket-aware A*.
   - Apply coordination heuristics: prevent detectives from entering the same node simultaneously and keep at least one detective guarding ferry hubs.
   - Recalculate if Mr. X reveals or if probability mass shifts significantly.
4. **Execution and feedback:**
   - Execute the first edge of each planned path.
   - Record blocked nodes to down-weight them in the next update.
   - Provide UI diagnostics: highlight top belief nodes and planned front.

## 2. Flowcharts

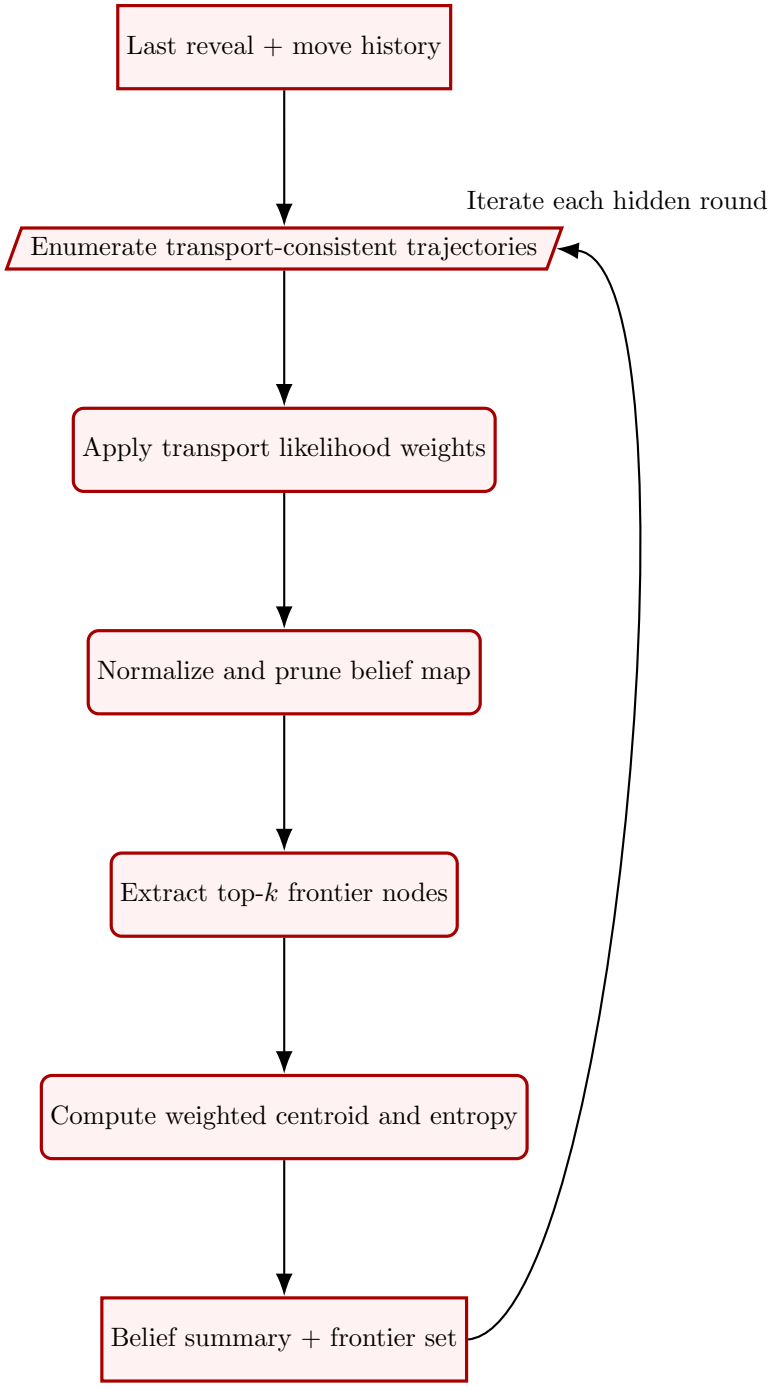


Figure 1: Belief update and frontier extraction pipeline

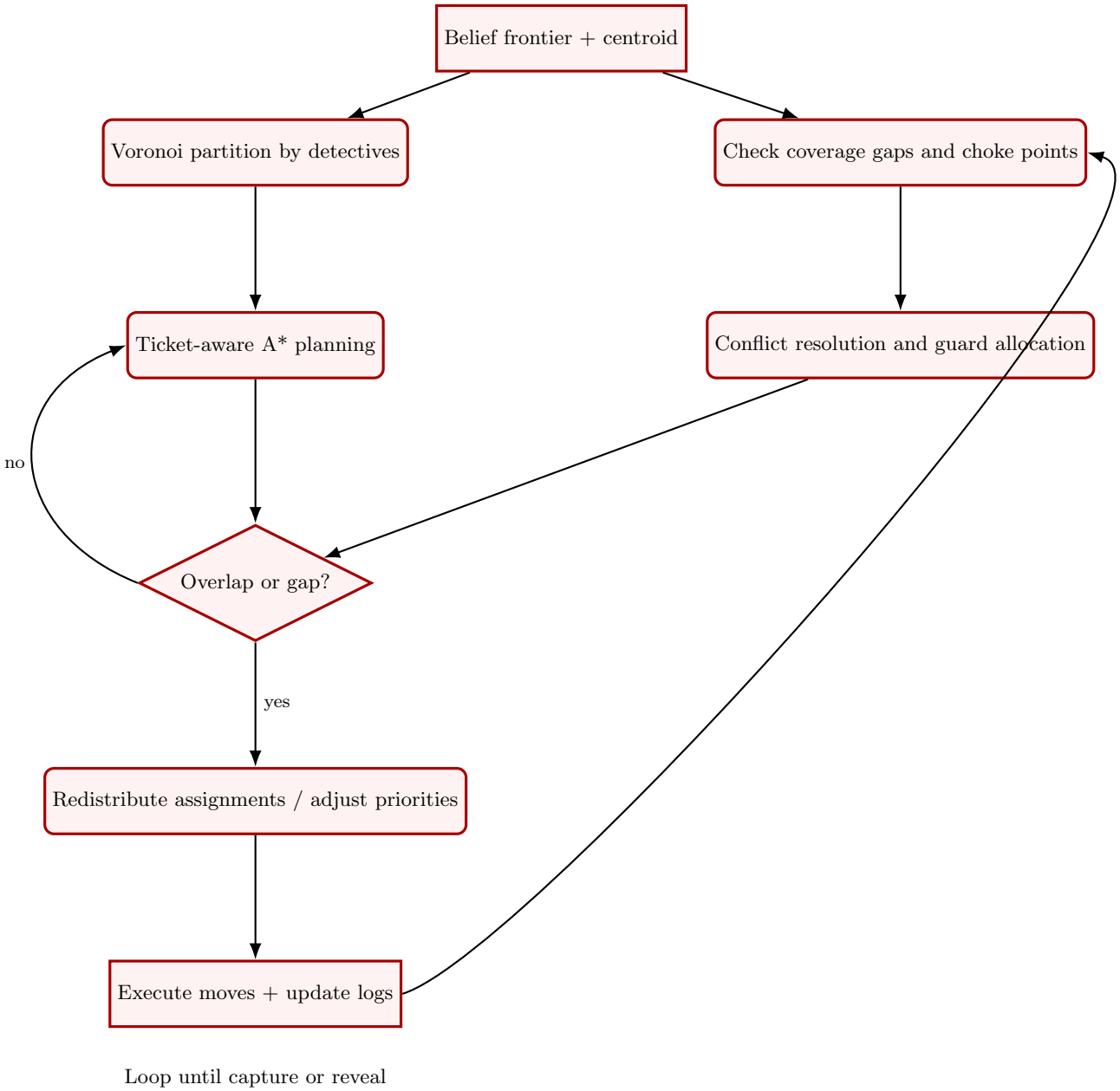


Figure 2: Detective coordination and encirclement decision flow

## 3. Pseudocode

**Algorithm 1:** Front search and encirclement strategy driven by probabilistic beliefs

**Input:** Current belief $belief$, detective tickets $tickets_D$, estimated Mr. X tickets $tickets_X$, transport graph $graph$, reveal schedule, heuristic weights

**Output:** Updated belief distribution and coordinated detective actions

**Function** `UpdateBelief(`*belief, tickets, moves*`):`
  *candidate* ← empty map;
  **foreach** $(node, p)$ *in belief* **do**
    **foreach** *transport in legalTransports(node, tickets)* **do**
      **foreach** *neighbor in neighborsByTransport(node, transport)* **do**
        **if** *transport not in moves* **then**
          continue;
        **end**
        $candidate[(neighbor, transport)] \leftarrow candidate[(neighbor, transport)] + p \cdot likelihood(transport)$;
      **end**
    **end**
  **end**
  $belief'$ ← aggregateByNode(*candidate*);
  normalize($belief'$);
  **return** pruneLowProbability($belief'$);
**Function** `SelectFront(`*belief, k*`):`
  *centroid* ← weightedCentroid(*belief*);
  $frontNodes$ ← topK(*belief*, $k$);
  **return** $(centroid, frontNodes)$;
**Function** `AssignDetectives(`*frontNodes, detectives*`):`
  *regions* ← voronoi(frontNodes, detectives);
  *assignments* ← empty map;
  **foreach** *detective in detectives* **do**
    *assignments*[*detective*] ← regionTarget(*regions*, detective);
  **end**
  **return** *assignments*;
**Function** `PlanMoves(`*state, assignments*`):`
  *plans* ← empty map;
  **foreach** *detective in assignments* **do**
    *target* ← *assignments*[*detective*];
    *path* ← ticketAwareAStar(state.graph, detective.position, target, detective.tickets);
    *plans*[*detective*] ← enforceNoOverlap(path, plans);
  **end**
  **return** *plans*;
Initialize *belief* ← uniform distribution over start candidates;
Initialize *movesSinceReveal* ← [];
**while** *Mr. X not captured* **do**
  **if** *Mr. X revealed* **then**
    *belief* ← deltaAt(revealedNode);
    *movesSinceReveal* ← [];
  **end**
  **else**
    append lastObservedTransport() to *movesSinceReveal*;
    *belief* ← `UpdateBelief(`*belief, mrXTicketsEstimate(), movesSinceReveal*`)`;
  **end**
  $(centroid, frontNodes)$ ← `SelectFront(`*belief, k*`)`;
  *assignments* ← `AssignDetectives(`*frontNodes, detectives*`)`;
  *plans* ← `PlanMoves(`*state, assignments*`)`;
  executeFirstSteps(*plans*);
  refreshBlockedNodes(*belief*, state.detectives);
**end**
Output captureTurn() when centroid overlaps with Mr. X position;