UNIVERSITY OF TRENTO - Italy
Department of Information Engineering
and Computer Science

AUTONOMOUS SOFTWARE AGENTS

Project Presentation

presented by

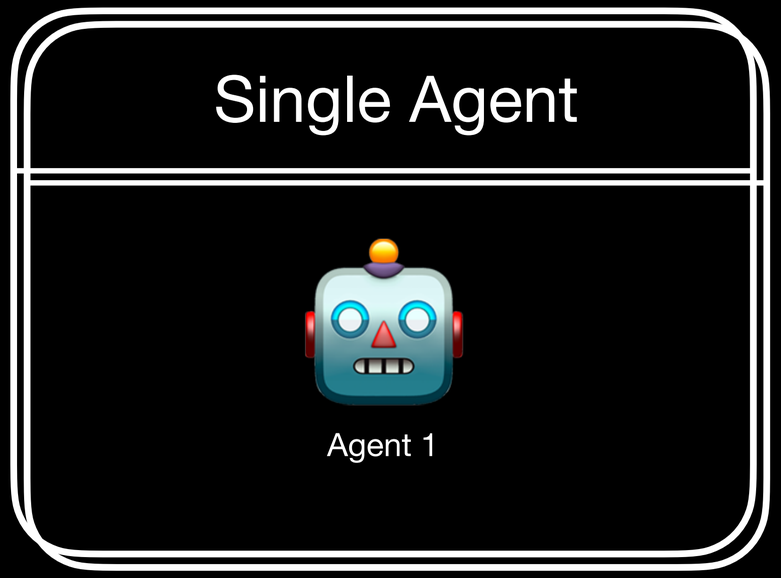
Descanta Bauchi

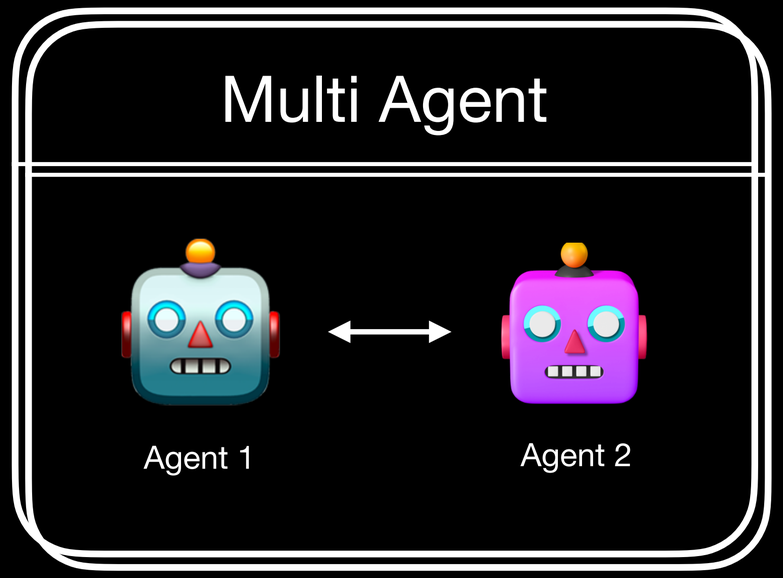
Fabio Missagia
256141
fabio.missagia@studenti.unitn.it

Alessandro Sartore
256152
alessandro.sartore-1@studenti.unitn.it

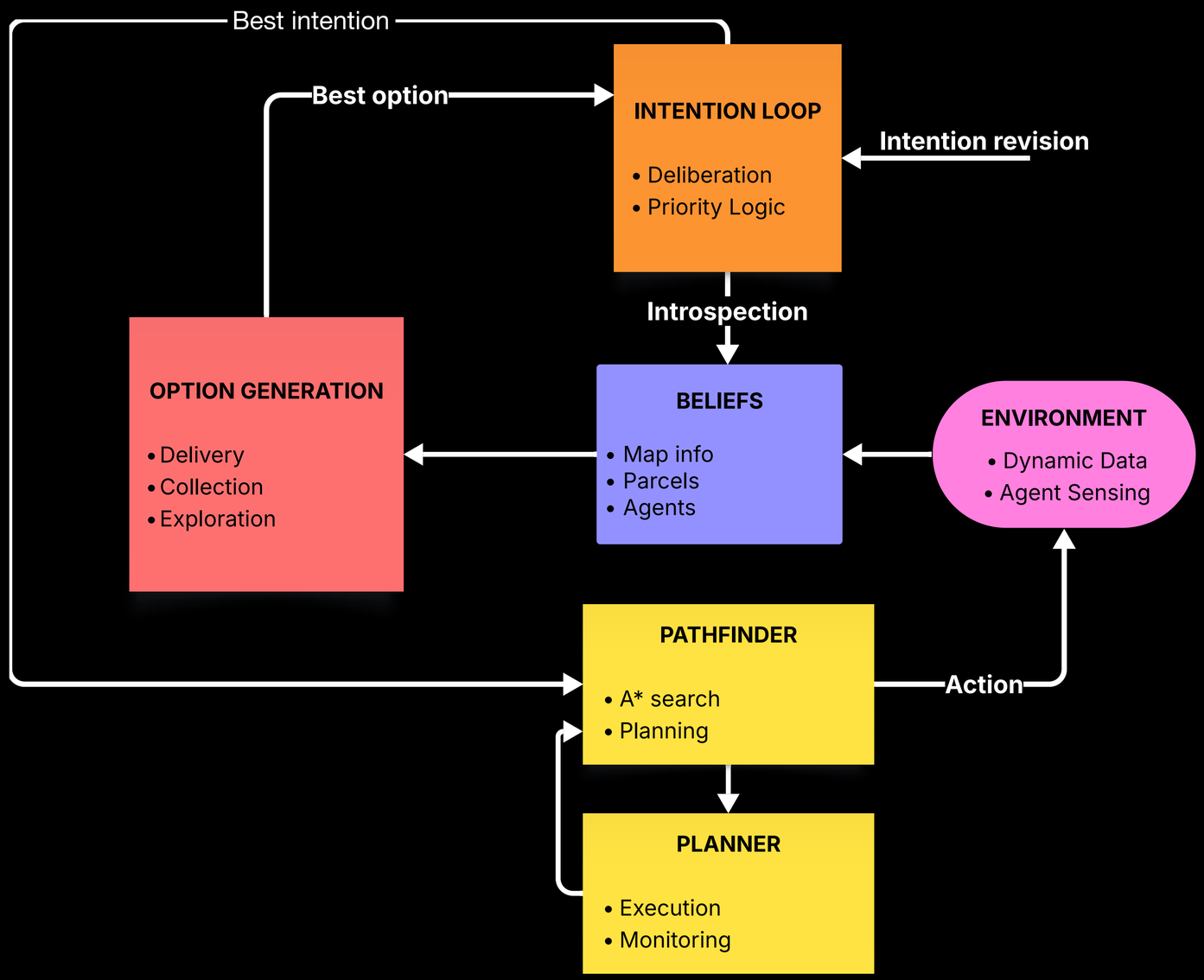
🎯 Objective

- The goal is the creation of **autonomous software agents** that can play the Deliveroo simulation game
- Our project is structured in **two** main parts:



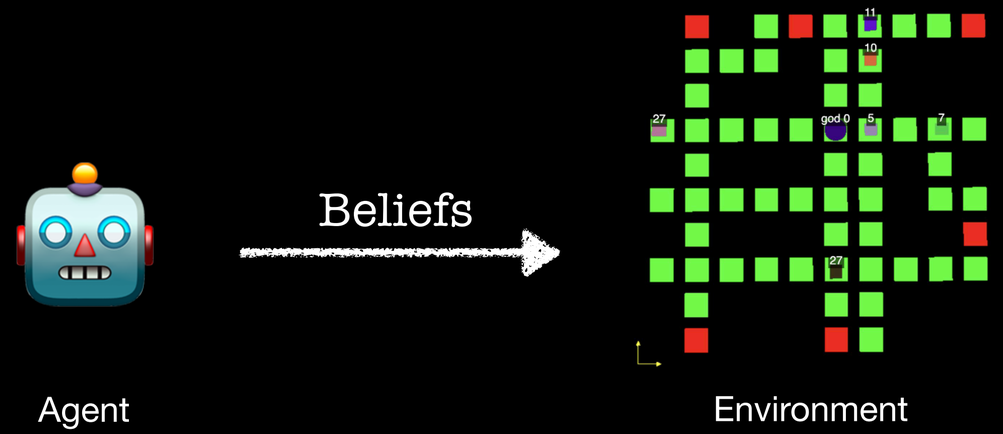


Belief-Desire-Intention



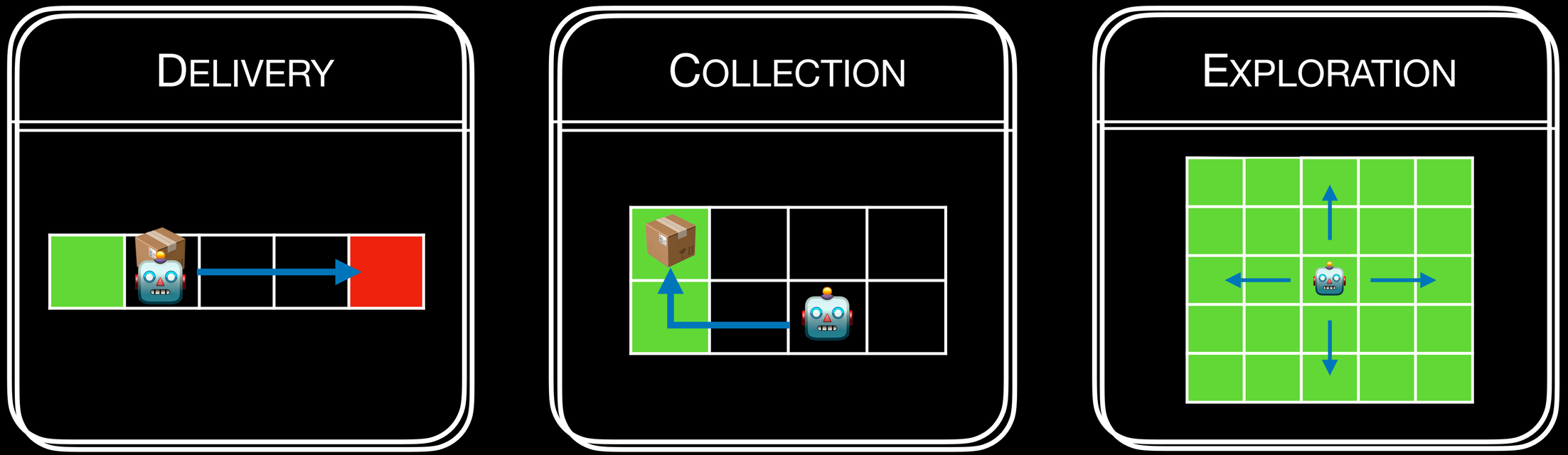
🧠 Beliefs

- Agents perceive different entities from the **environment**, and build and update **asynchronously** their **beliefs** based on:
  - **Map:** normal, delivery and spawning tiles
  - **Parcels:** nearby valuable parcels
  - **Agents:** nearby agents



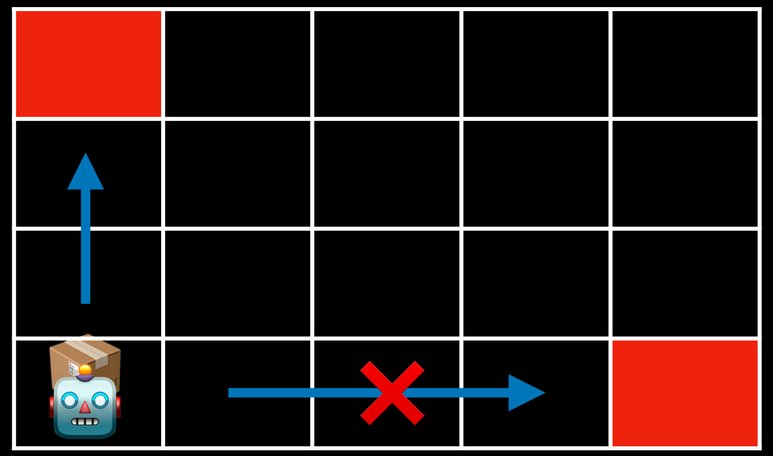
❓ Options Generation

- At each iteration, the agent evaluates the current world state and produces a set of candidate options:



❓ Options Generation: Delivery

- As soon as an agent is **carrying** a parcel, the **delivery** is prioritized.
- The delivery tile **closest to the agent** is the **best option**.

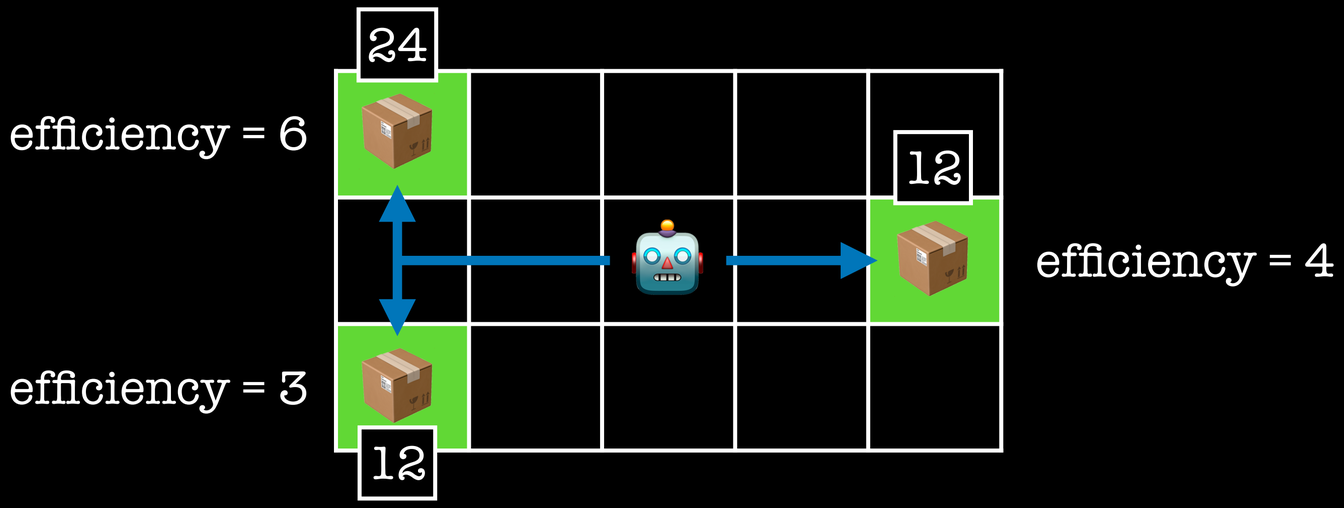


❓ Options Generation: Collection

- If **nearby parcels** are sensed while **exploring**, they get **evaluated**[1]:

$$\text{Efficiency} = \frac{\text{reward} \times \text{timeFactor}}{\text{distance} + 1}^{[2]}$$

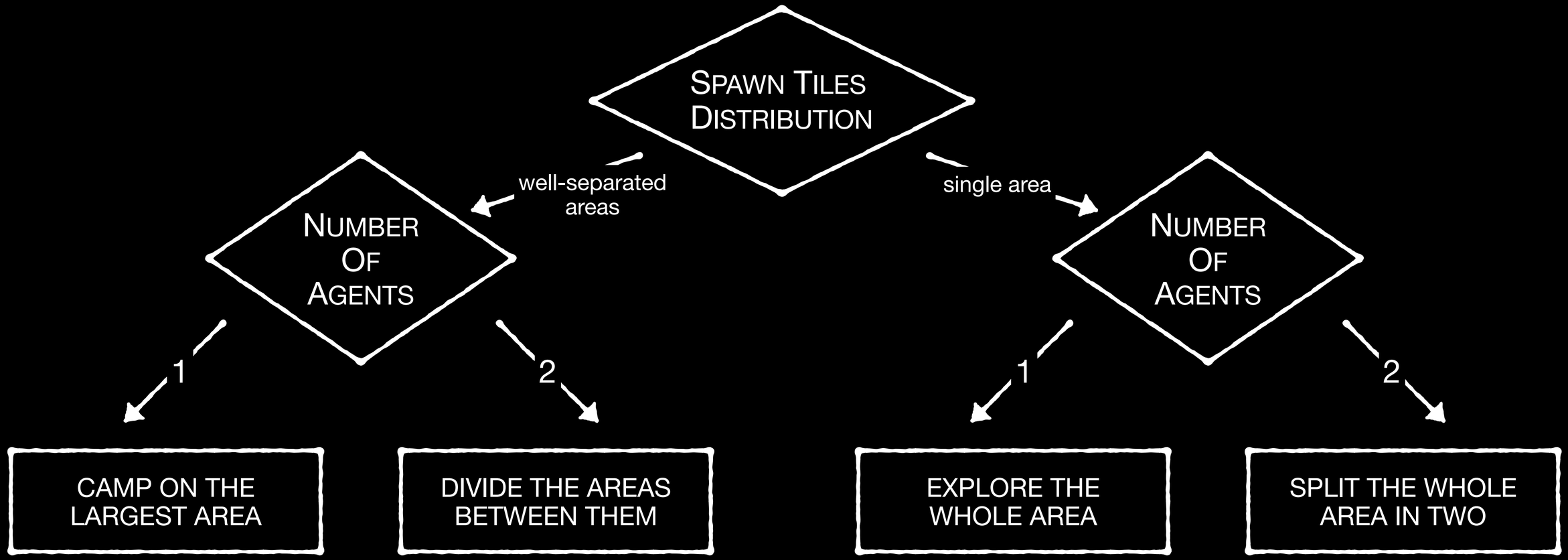
[2] $\text{timeFactor} = \frac{\text{currentReward}}{\text{originalReward}}$



[1] *Before computing the score, some thresholds are checked*

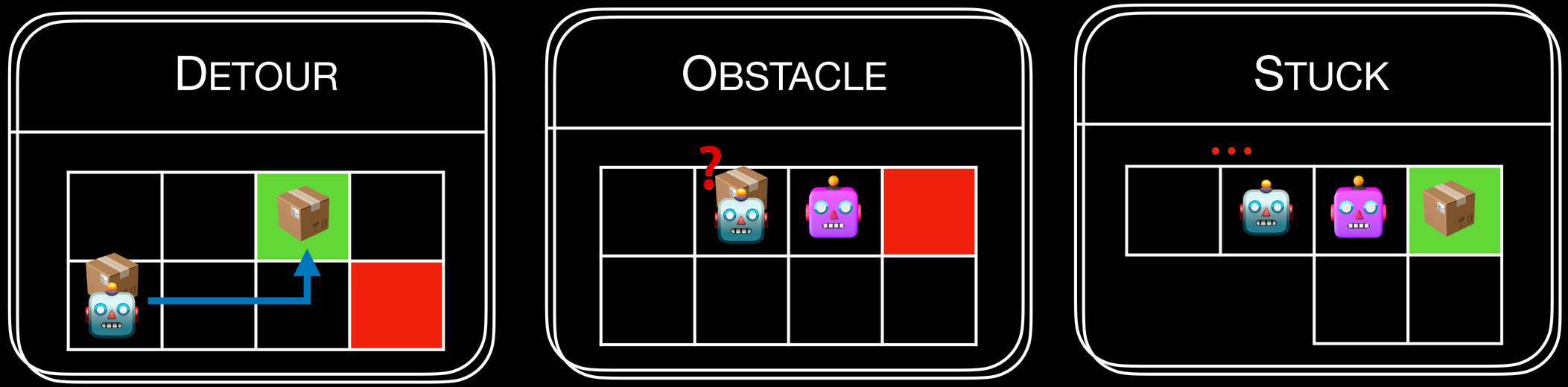
? Options Generation: Exploration

- Based on **tiles distribution**, adaptive **exploration strategies** are used:



♺ Intention Loop & Revision

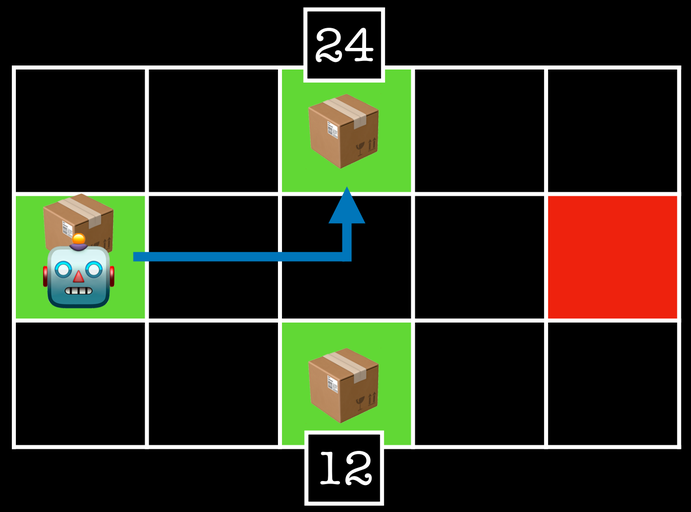
- The **intention loop** is responsible for continuously **processing** the agent's intentions.
- Agent continues to pursue the **current top intention**, until a new **best option** is selected (**Intention Revision**)

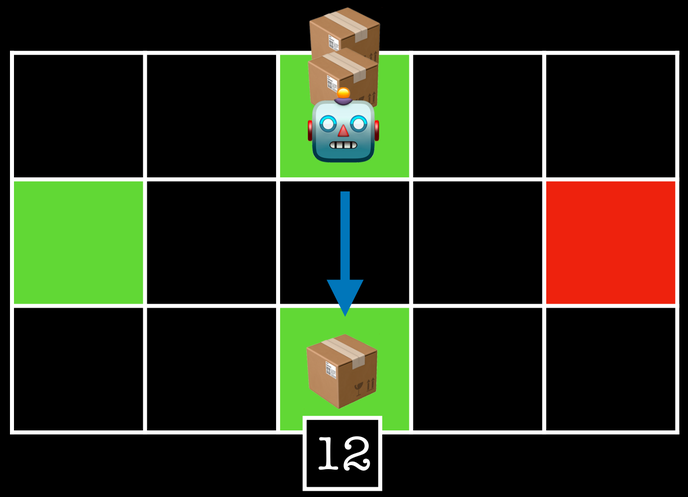


⚠️ Intention Revision: Detour

- What if the agent **discovers parcels** while delivering?
- A **detour-score** is used to evaluate detour-worthiness*:

$$\texttt{Score} = \frac{\texttt{reward}}{\texttt{addedSteps} + 1}$$

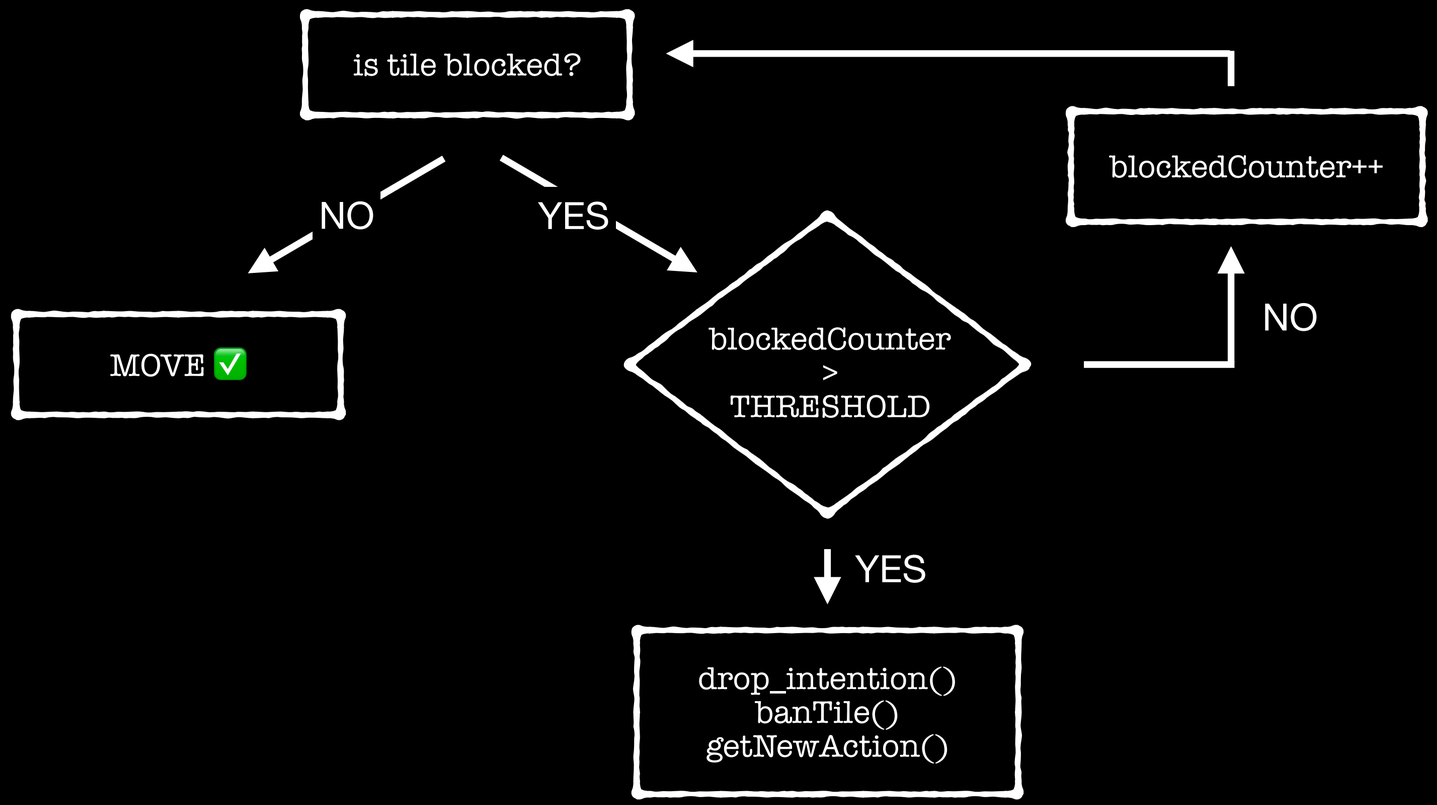


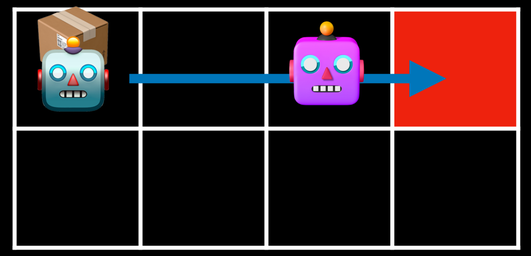


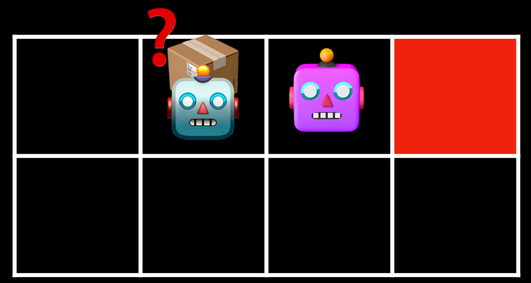
*Before computing the score, some thresholds are checked

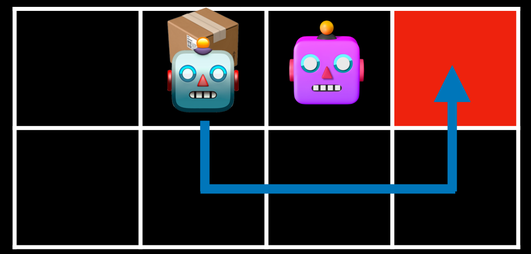
⚠️ Intention Revision: Obstacle

- What if the agent finds an **obstacle?**





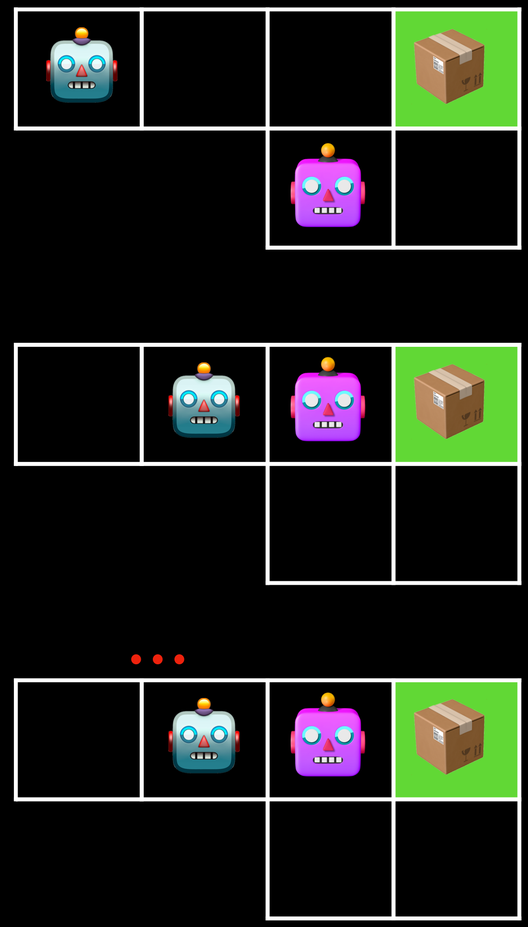




⚠️ Intention Revision: Stuck

- What if the agent is **stuck**?

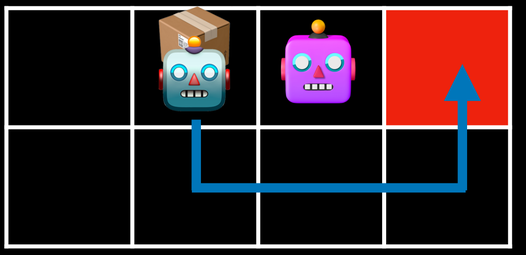
- We can’t do much…
  - **Drop** the current **intention**
  - **Try** to get a new **intention**
  - The **intention** will **fail**
  - **Repeat**



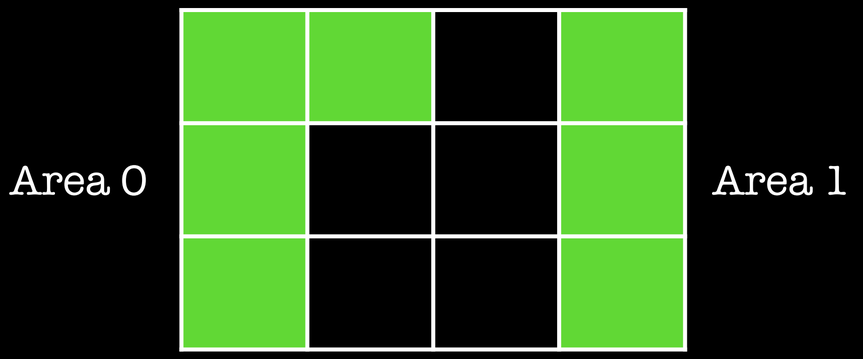
🔎 Planning

- Different algorithms are used:
  - **A*:** for **quick path finding**, faster than BFS
  - **BFS:** for **clustering spawn tiles areas**
  - **PDDL: efficient path planning** but **slower**
    - ***Domain:*** **fixed**
    - ***Problem:*** **dynamic** based on map updates

A* (Path finding)

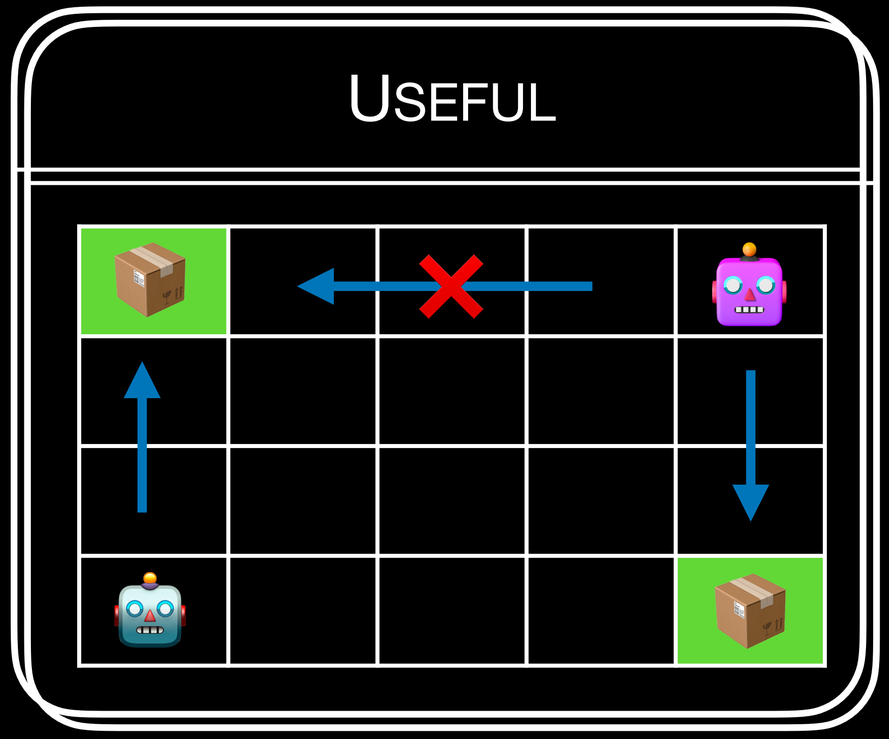


BFS (Area clustering)

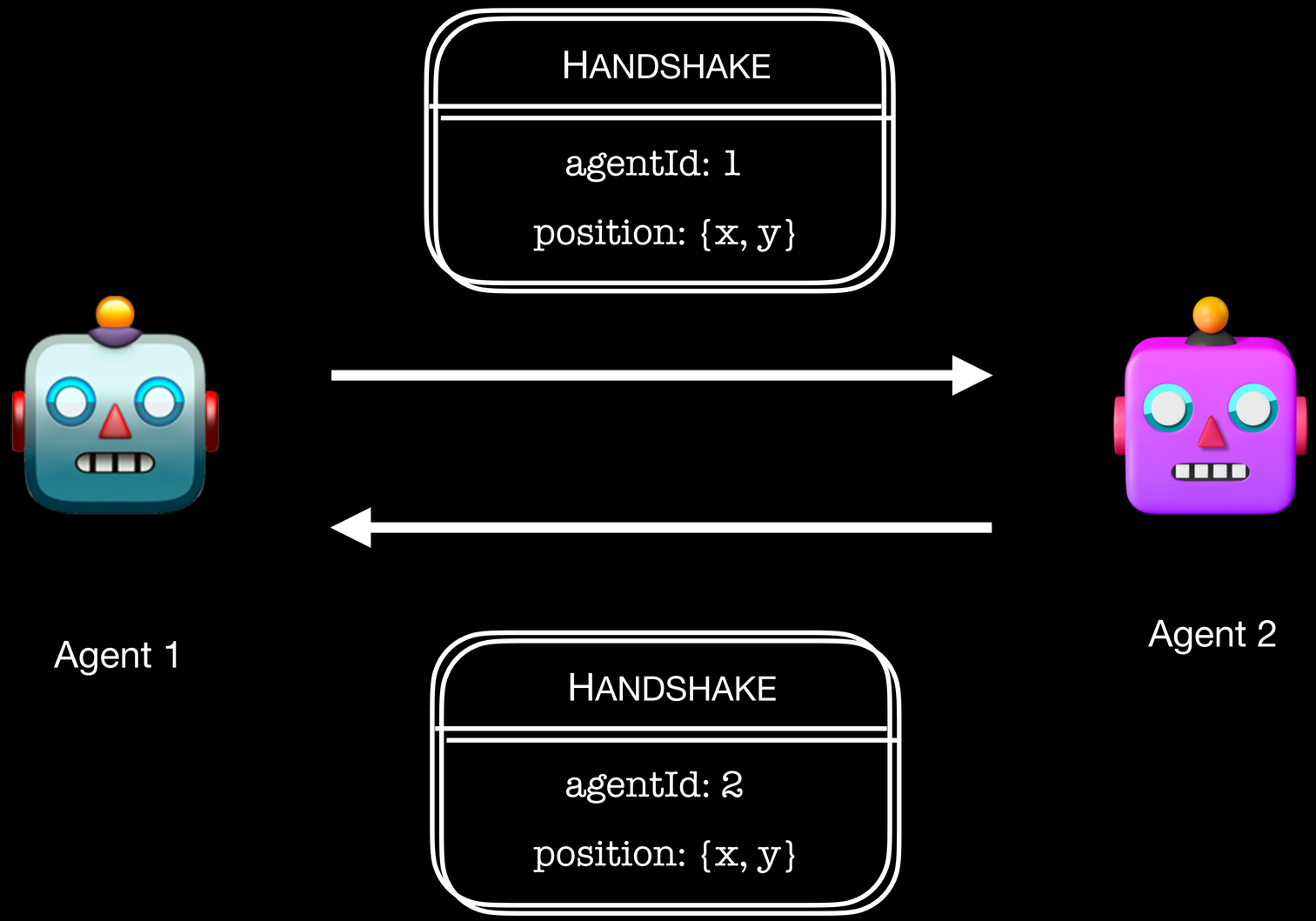


📩 Communication

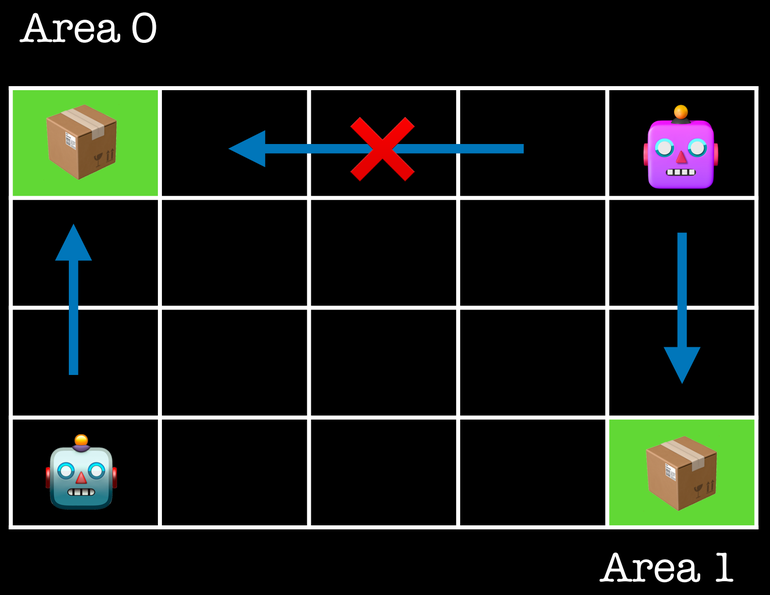
- There are cases where communication can be **useful** or even **essential.**

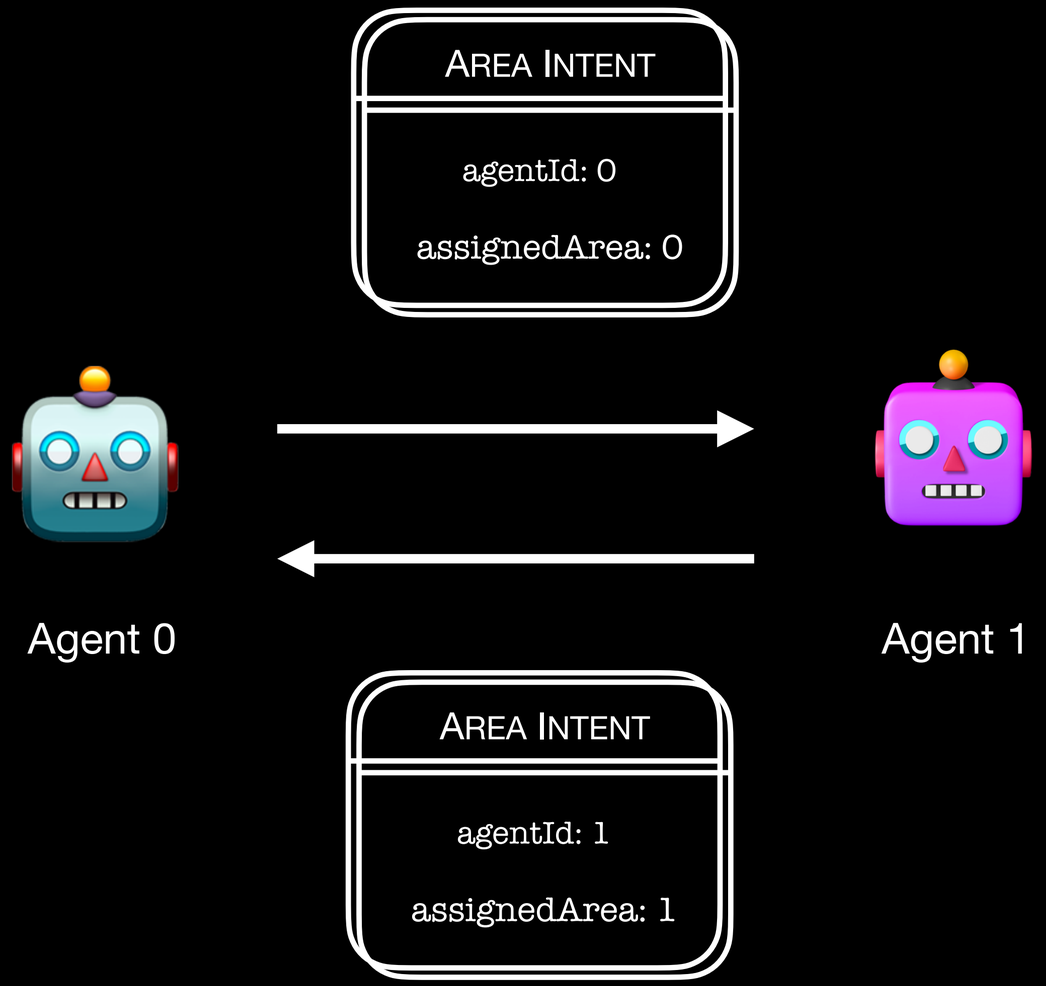


📩 Communication: Handshake

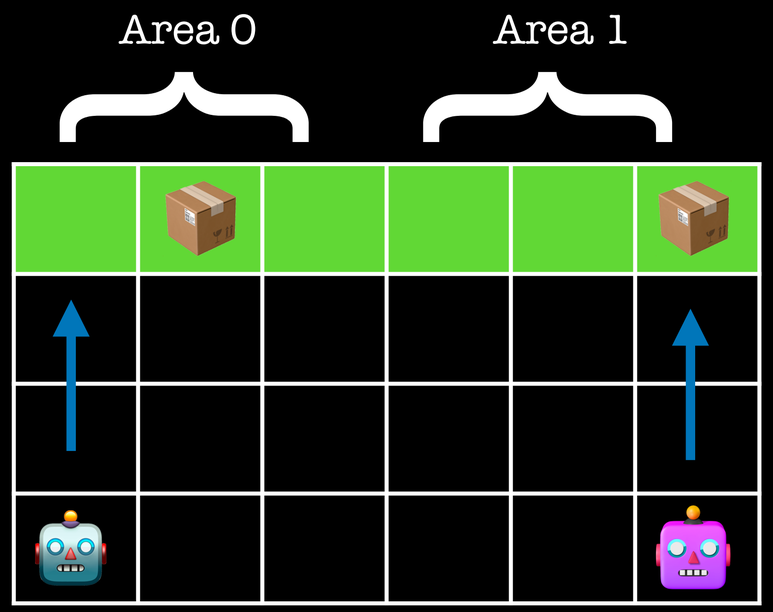


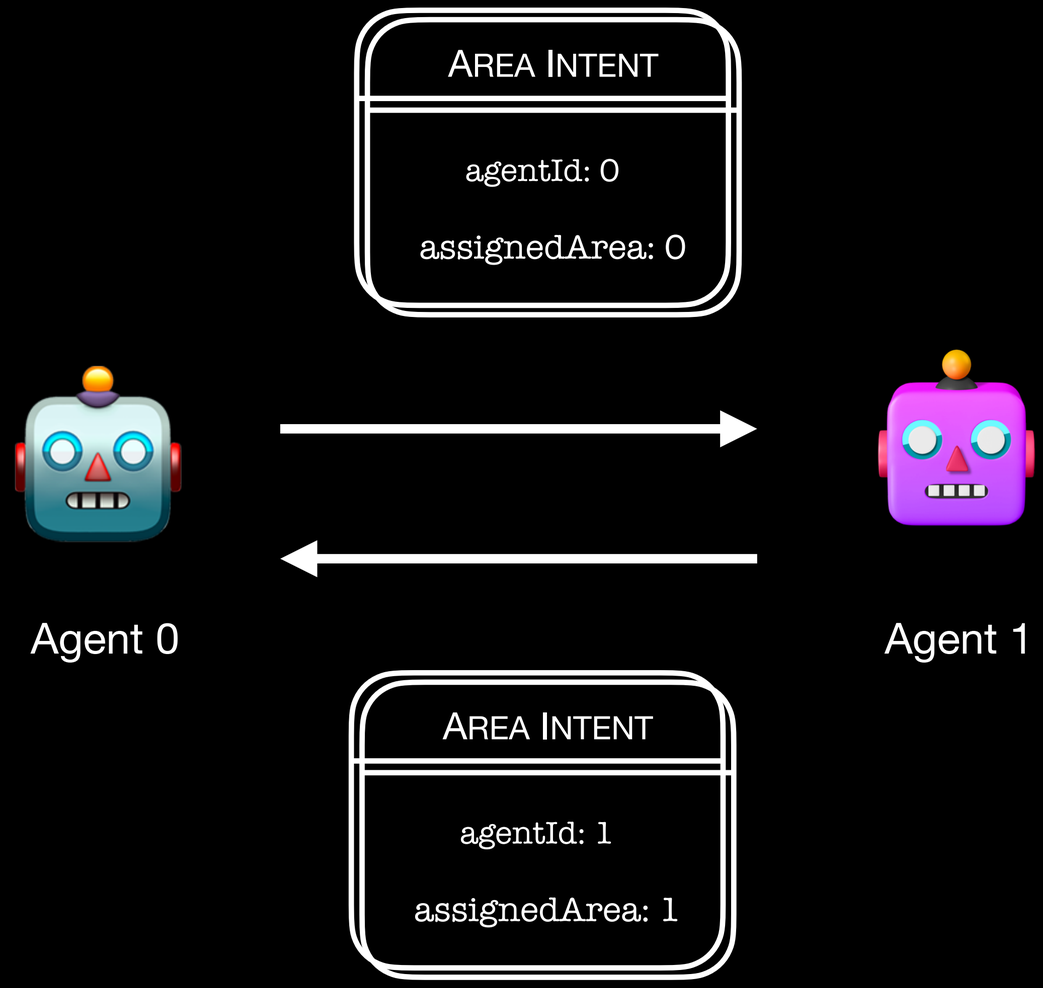
📩 Communication: Area Intent



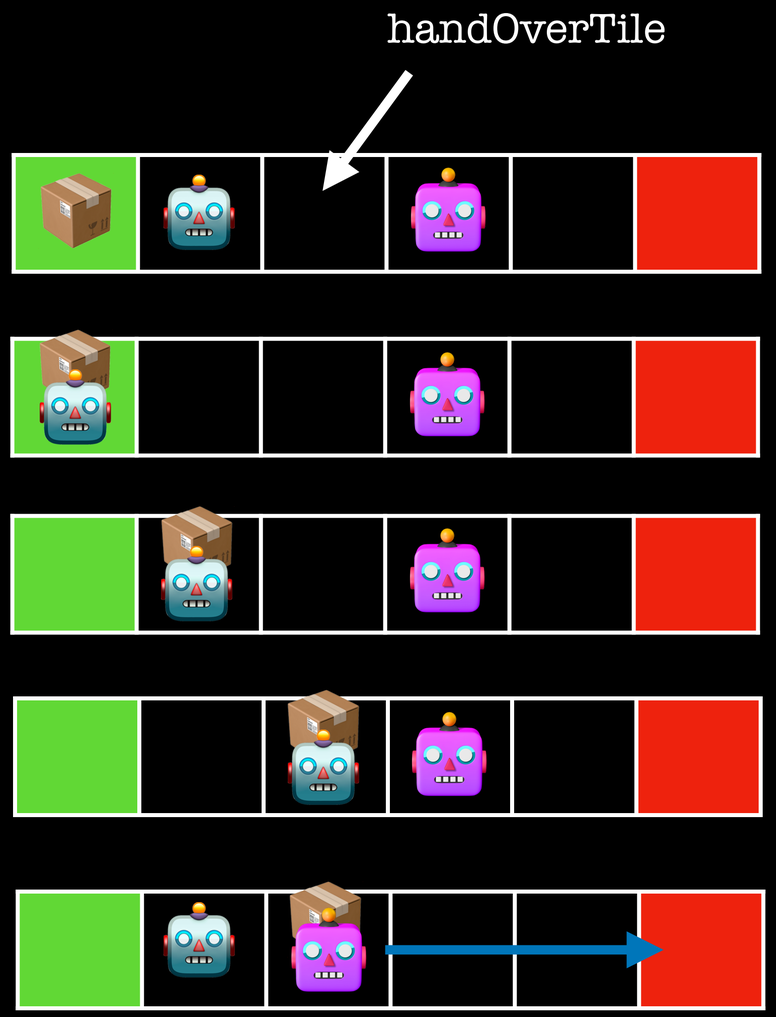


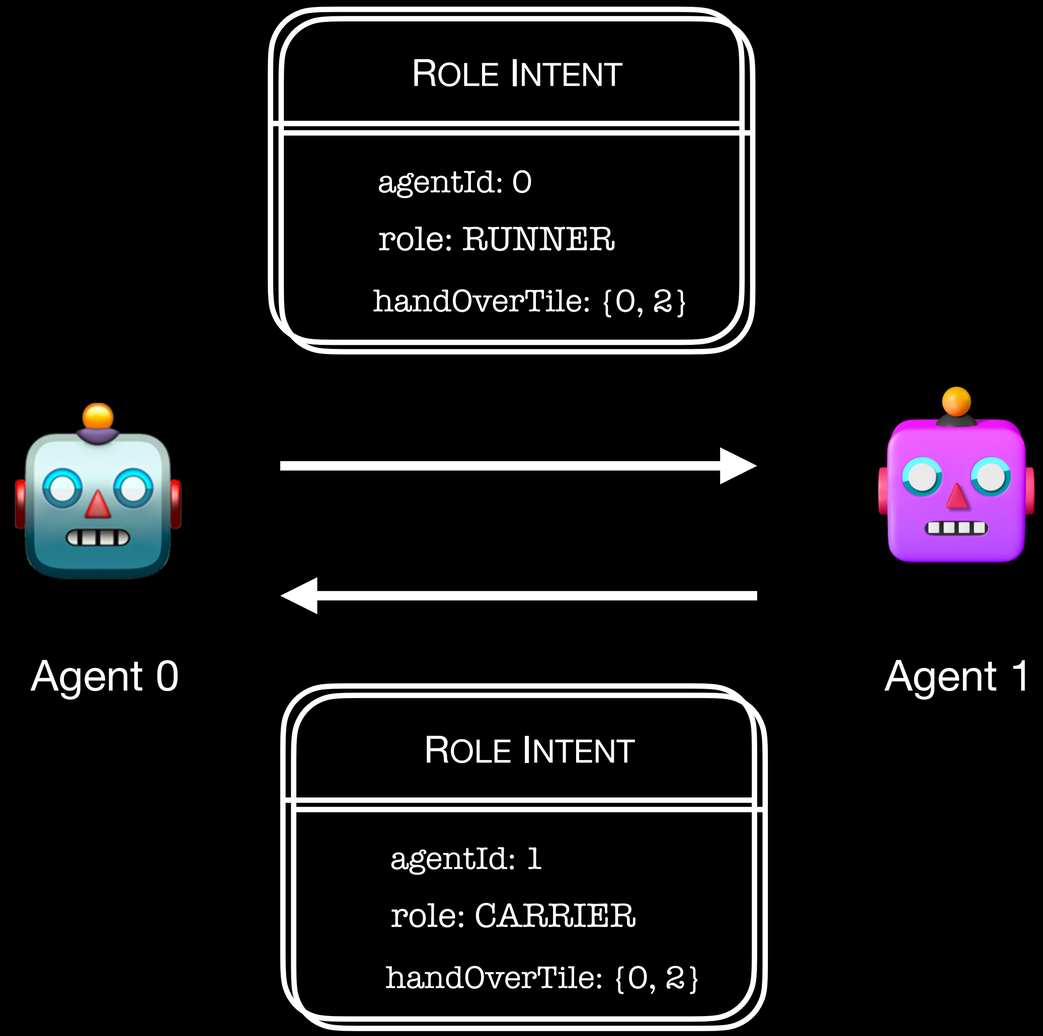
📩 Communication: Area Intent





📩 Communication: Runner/Carrier





🔮 Future work

- Several **improvements** could be done:
  - more **efficient PDDL integration** or **hybrid** planning methods
  - exploring only the **biggest** cluster is **not optimal**
  - **wait** a bit on the **spawn tile** before **delivering**
  - **improve** the **communication** protocol

UNIVERSITY OF TRENTO - Italy
Department of Information Engineering
and Computer Science

Thanks for the attention!

by

Descanta Bauchi

Fabio Missagia
256141
fabio.missagia@studenti.unitn.it

Alessandro Sartore
256152
alessandro.sartore-1@studenti.unitn.it

? FAQ

- Results

Level	A*	PDDL
25c1-1	1623	380
25c1-2	2471	1329
25c1-3	1569	1381
25c1-4	3502	2605
25c1-5	2963	2078
25c1-6	1391	1167
25c1-7	3513	2236
25c1-8	1487	842
25c1-9	9400	5282

Table 1: Single-Agent Results

Level	Total	A1	A2
25c2-1	3235-2736	1639-1540	1596-1196
25c2-2	1931-1094	1146-638	785-456
25c2-3	2519-2140	1645-1341	874-799
25c2-4	7235-1630	4190-724	3045-906
25c2-5	4786-2841	2593-1934	2193-907
25c2-6	4402-3103	2308-1845	2094-1258
25c2-7	11022-4340	8718-2959	2304-1381
25c2_hall	2070-1296	2070-1296	0-0

Table 2: Multi-Agent Results. `A* - PDDL`

❓ FAQ

- How exactly does your utility function work for package selection?

```
class ParcelSelector {
// other code...
    calculateParcelEfficiency(parcel) {
        if (!this. beliefs.myPosition) return -Infinity;
        const distance = this.beliefs.calculateDistance(parcel.x, parcel.y);
        const timeFactor = parcel.reward / originalReward; // Decay factor
        return (parcel.reward * timeFactor) / (distance + 1);
    }
}
```

❓ FAQ

- How do you manage coordination?

```
class CommunicationHandler {
    announcePresence(agentId, position) {
        // code...
    }
}
class AreaManager{
    divideArea(area, myIndex) {
        // code...
    }
}
```

❓ FAQ

- How do you implementate BDI in your code?

```
class Planner{
  async getAction() {
    // 1. UPDATE BELIEFS: Position tracking
    this.recordPosition(currentPos.x, currentPos.y);
    // 2. REACTIVE PICKUP: Immediate desire for nearby parcels
    const reactiveAction = await this.handleReactivePickup();
    if (reactiveAction) return reactiveAction;
    // 3. DELIBERATION: Choose between delivery, collection, exploration
    if (this.deliveryStrategy.shouldDeliver()) {
      return this.deliveryStrategy-getDeliveryAction();
    }
    // 4. INTENTION EXECUTION: Follow planned path or replan
    return this.handleParcelCollection(bestParcel, currentPos) ||
      this.handleExploration(currentPos);
  }
}
```