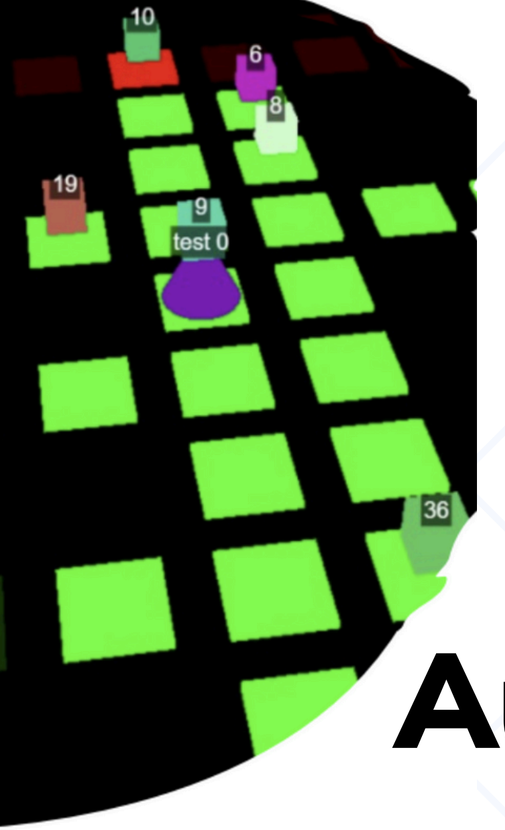
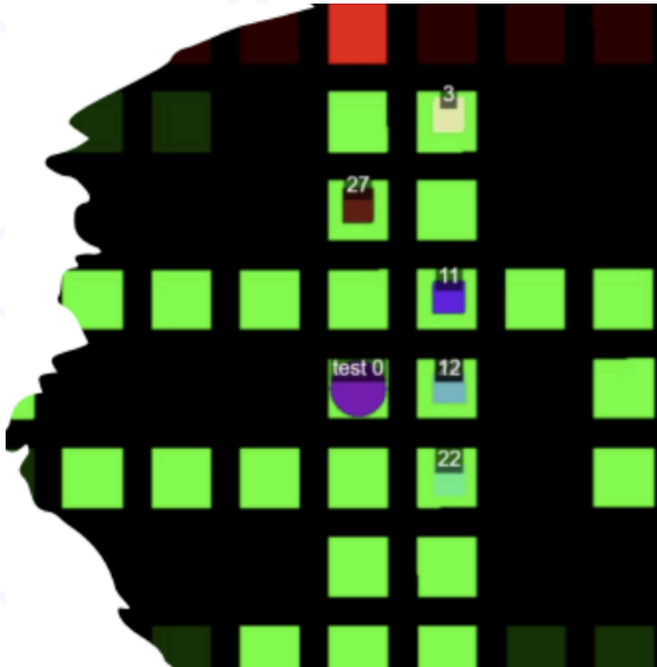


University of Trento

Autonomous Software Agents

Murtas Cristian & Wang Marco

Team Baozi



Overview

01

Introduction

The scope of the project was to develop autonomous agent leveraging the **BDI architecture** that is able to play the **Deliveroo** game, furthermore we integrated an **external planner (PDDL)** to deal with the generation of plans.

The overall structure is divided into three main components: **beliefs**, **intentions**, and the **planner**.

02

Beliefs

01 Information that changes frequently

02 Constant data about the map

03 Hyper parameters

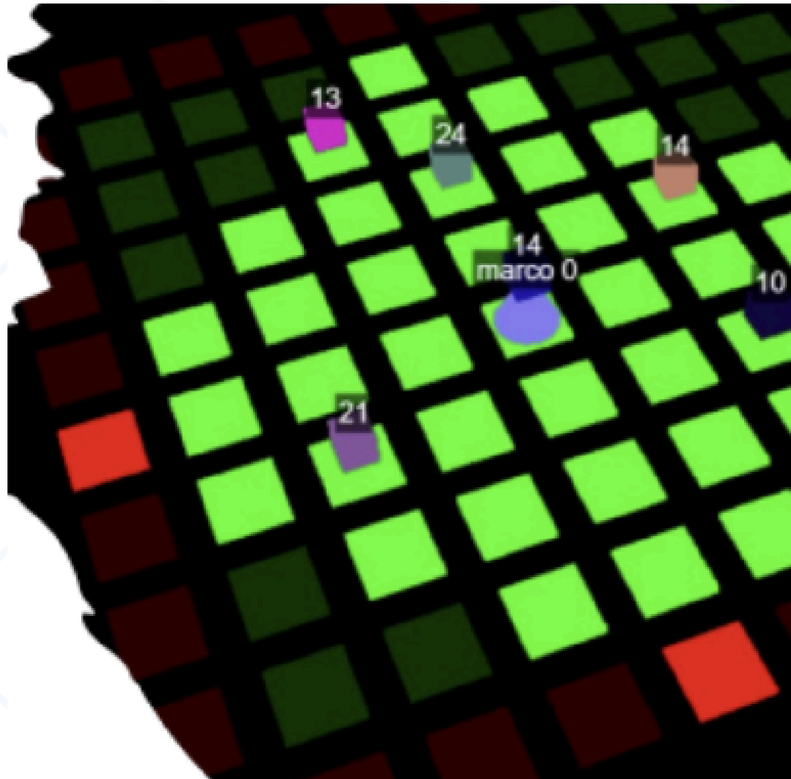
04 Communication buffer used for coordination

03

Intentions

Put down

- if the agent is carrying a certain number of parcels
- if the loss of the parcel that i’m carrying is greater than the reward of the parcel
- if the there are no near parcels



03 Intentions

Pick up

A : pick up the nearest parcel

B: pick up the parcel that have the highest value when reaching it

C: pick up the parcel that have the highest absolute value

Crowdness	Variance	Decay	Criteria
0	0	0	A
0	0	1	B
0	1	0	C
0	1	1	B
1	0	0	A
1	0	1	A
1	1	0	A
1	1	1	A

03

Intentions

Target move

- Decision Strategy
- Heat map Implementation

Revision

- All the intentions are subject to revision

04

Planners

Online

- Domain definition
- Problem generation

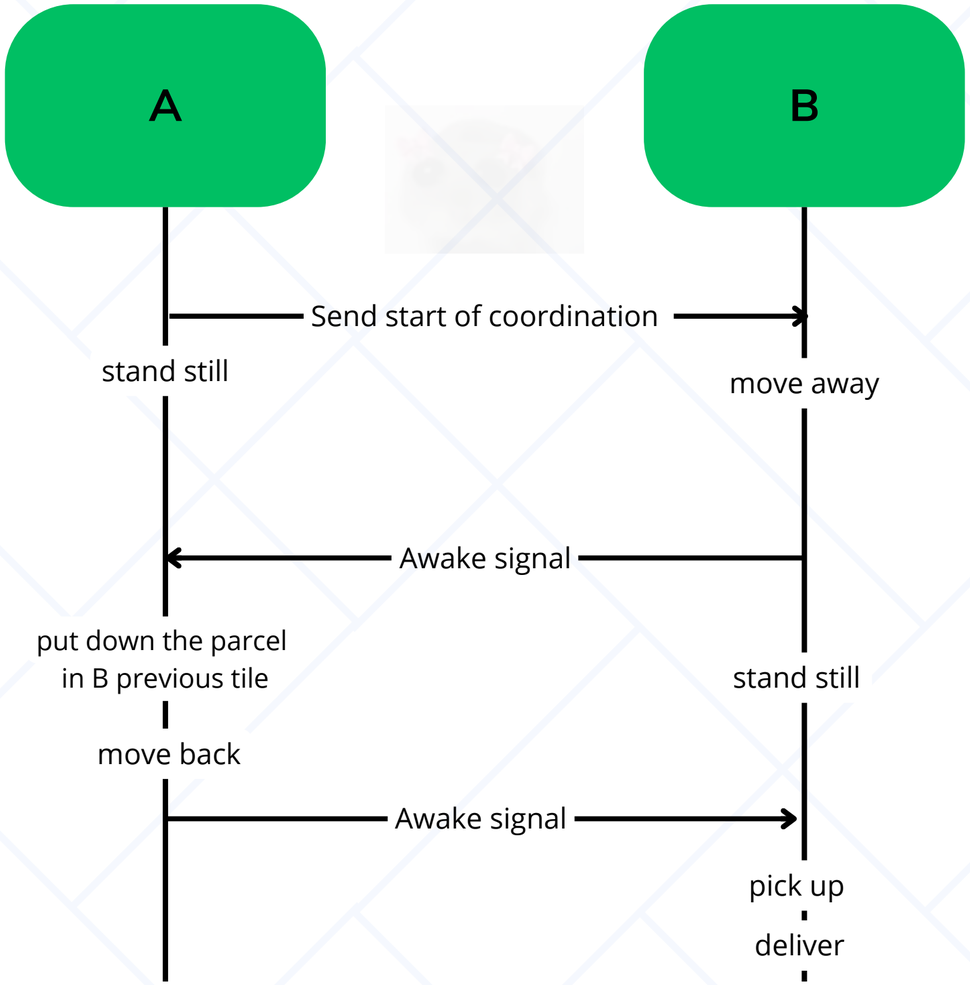
Offline

- A* Algorithm

```
(define (problem grid-navigation)
    (:domain default)

    ;; Define the objects: tiles, agent, and parcels
    (:objects
        tile1 tile2 tile3 tile4 tile5 tile6 tile7 tile8 tile9  ;; tiles
        agent1                                                  ;; agent
        parcel1                                                 ;; parcel
    )

    ;; Define the initial state
    (:init
        (tile tile1) (tile tile2) (tile tile3) (tile tile4) (tile tile5)
        (tile tile6) (tile tile7) (tile tile8) (tile tile9)
        (agent agent1)
        (parcel parcel1)
        (me agent1)
        (at agent1 tile1)
        (at parcel1 tile5)

        ;; Define tile adjacency
        (right tile1 tile2) (right tile2 tile3)
        (right tile4 tile5) (right tile5 tile6)
        (right tile7 tile8) (right tile8 tile9)
        (left tile2 tile1) (left tile3 tile2)
        (left tile5 tile4) (left tile6 tile5)
        (left tile8 tile7) (left tile9 tile8)
        (up tile4 tile1) (up tile5 tile2) (up tile6 tile3)
        (up tile7 tile4) (up tile8 tile5) (up tile9 tile6)
        (down tile1 tile4) (down tile2 tile5) (down tile3 tile6)
        (down tile4 tile7) (down tile5 tile8) (down tile6 tile9)
    )

    ;; Define the goal state
    (:goal
        (and
            (at agent1 tile5)       ;; agent1 should reach tile5
            (carrying agent1 parcel1) ;; agent1 should be carrying parcel1
        )
    )
)
```

05 Multi-agent & Communication

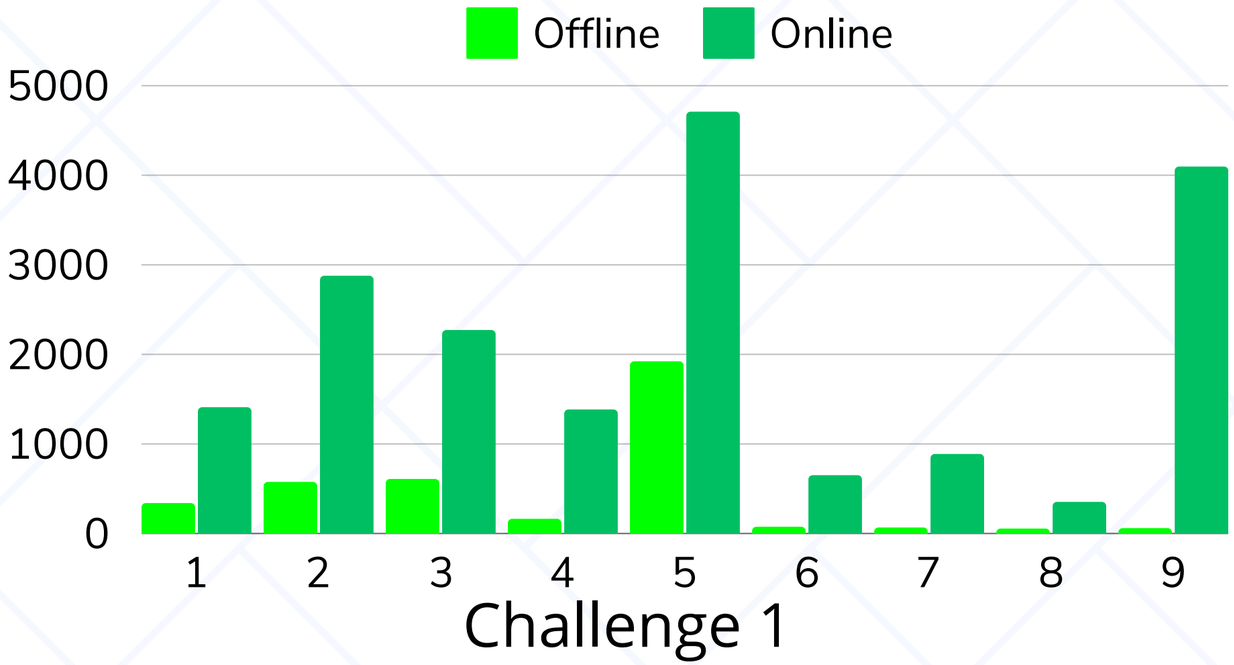
- Communication Primitives
- Handshake Protocol
- Agent Role
- Coordination Mechanism



06

Tests & Results

- Tested scenarios
- Perfomance comparison



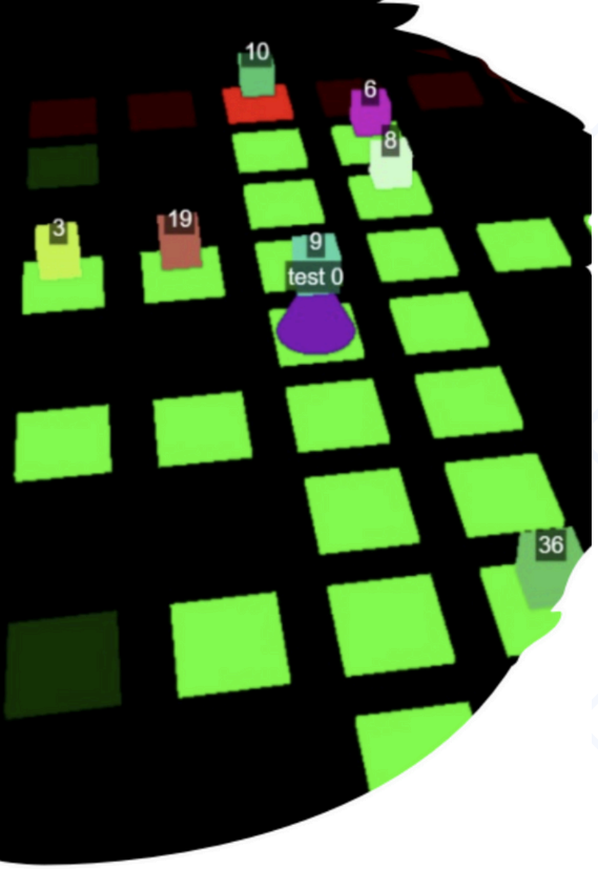
07

Conclusion

01 Modular architecture: we can substitute the planner without influencing the BDI architecture

02 Constant revision, replanning and noise help us to reduce the probability of getting stuck and having conflicts

03 Coordination is role-independent, whoever is found to be in a certain situation can lead the coordination



University of Trento

Thank You

Murtas Cristian & Wang Marco

Team Baozi