Autonomous Software Agents - Report

Michele Minniti, Martina Scheffler

Student IDs: 247168, 249609

michele.minniti@studenti.unitn.it

martina.scheffler@studenti.unitn.it

https://github.com/Sminnitex/ASAProject

***Abstract*—*This paper introduces our project for the course of Autonomous Software Agents. The objective was to build an agent capable of moving in the environment of and participating in a game called Deliveroo.js. In this game, the aim is to deliver parcels in order to make points competing against other agents. We built two autonomous agents capable of tackling this challenge alone or in cooperation. Using a PDDL path planner, they can move on the environment and perform deliveries. When they coordinate, they can use more sophisticated strategies exchanging information about the environment and cooperating to deliver a package together, or even split and try to deliver as many packages in parallel as possible. We are going to explain our thought process and comment on some results going in the detail of our work.*

keywords*—*Autonomous Software Agents, Multi-Agent System, PDDL, BDI Agents

1. Introduction: Deliveroo.js

Deliveroo.js is the parcel delivering game used to benchmark and test our autonomous agents. The game consist of three basic elements: agents, parcels and tiles. Each element can be described as follows:

- **Agent:** the program we developed in order to interact with the environment, its purpose is to deliver parcels to specific tiles.
- **Parcels:** packages to deliver in order to earn points. Each package has a timer that scales the amount of points the agent gains upon delivery. When the timer reaches 0, the parcel disappears.
- **Tiles:** walkable elements for our agent, grouped in variables maps. Tiles can be of 3 types:
  1. **Spawning tiles:** where parcels can spawn.
  2. **Non-spawning tiles:** only useful for movement.
  3. **Delivery tiles:** where we have to deliver parcels in order to earn points.

The goal is to create an agent capable of making as many points as possible, therefore requiring it to have a good decision making strategy that changes with the environment.

2. Single Agent

Our analysis begins considering the case of creating a single agent capable of performing as well as possible in the Deliveroo environment. We will describe the architecture of the agent, and then give the result of some tests we performed on benchmark maps.

2.1. Sensing

In order for the agent to adapt to changes in the environment, like other agents' movements or the spawning and disappearing of parcels, it needs to be able to sense the environment and create a Belief Set about it. For the case of a single agent, sensing is done on:

1. the tiles of the map;
2. our own agent;
3. the other agents;
4. the parcels;
5. the map configuration.

2.1.1. Tiles

The `onTile` functionality can be used to create a map of tiles from the environment map. It returns the x- and y-position and whether the tile is a delivery tile or not. Therefore we decided to create an overall map and another map only for delivery tiles. The second one is used to quickly find the nearest delivery tile without having to go through every tile.

2.1.2. Own Agent

Further, the `onYou` function returns the `id`, `name`, `position` and `score` of our own agent. It is used for path planning to determine the paths' starting point.

2.1.3. Other Agents

Similarly to the own agent, information about the other agents can be obtained via `onAgentsSensing`, returning the same parameters than `onYou`, which will be saved to a map to track the other agents' positions.

2.1.4. Parcels

Most importantly, our agent can sense parcels using `onParcelsSensing` in a radius around it limited by the agent observation distance specified by the map. The parcels have the parameters `id`, `position`, `carriedBy` and `reward`. `carriedBy` here is the ID of the agent that is holding the parcel, null/undefined if nobody has it. The parcels visible to the agent are added to a map with their IDs as keys.

2.1.5. Configuration

The configuration that can be obtained from the server via `onConfig` includes parameters like the agent observation distance, the time movements take and information about the parcels. In our case, the most important parameters are the parcel decaying interval (`PARCEL_DECADING_INTERVAL`) which specifies how often the value of a parcel decreases and the parcels' average reward (`PARCEL_REWARD_AVG`).

2.2. BDI Loop

The BDI loop, which is triggered by the sensing, works generating the possible desires to pursue and then choosing the best one among them. We chose to consider three basic desires which are:

- **Move:** it represents the desire to explore, it works choosing a random tile, weighted for it's usefulness on the map,

Predicate	Input	Description
`me`	`agent`	specify the name of our agent
`at`	(`agent` or `parcel`) and `tile`	assign an object to a specific location
`delivery`	`tile`	assign to the variable delivery to the delivery tiles in the map
`holding`	`agent` and `parcel`	to know if our agent is holding a parcel
`left`, `right`, `up`, `down`	`tile` and `tile`	build a map to do path planning

Table 1. *PDDL Predicates*

Action	Parameters	Precondition	Effect
`Movement`	`agent` and `tile`	Location of agent and destination	Our agent moves
`Deliver`	`agent`, `parcel` and `tile`	Agent in the delivery tile	Deliver and delete parcel
`Pick up`	`agent`, `parcel` and `tile`	Agent and parcel in the same tile	Agent holds the parcel
`Drop off`	`agent`, `parcel` and `tile`	Agent holds a parcel	Agent drops the parcel

Table 2. *PDDL Actions*

and pursuing the desire to go there. If during the exploration we sense a parcel, we stop the exploration and pursue the desire to collect the parcel. The move desire is chosen when we don't have any other desire to pursue, or when we get stuck in trying to pick up a package or delivering something.
- **Go Pick-up:** we activate this desire when we sense a parcel, the objective is of course to collect it. The decision-making is based on distance so we give priority to packages near to us, so our agent will collect the closest packages and then try to deliver them.
- **Deliver:** we activate this desire when we are carrying a parcel and nearby there is a delivery tile. We keep track of delivery tiles in our sensing, in order to measure the distance from our agent and understand when it's convenient to deliver the parcels collected.

Once, all desired options are generated, the best one is chosen and added to the intention queue in order to be executed. The functionality of the agent loop is shown in Figure 1 as a flowchart.

2.3. Belief Revision

During the generation of options in the BDI Loop, Belief Revision is done for the parcels in three ways. Firstly, if a visible parcel's reward drops below one, it is deleted from the map as it is no longer a viable option to be picked up. Furthermore, old non-visible parcels will be deleted after a time `PARCEL_REWARD_AVG * PARCEL_DECADING_INTERVAL`, when they are likely to have disappeared from the map. Finally, visible parcels that are picked up by an agent other than ours are also deleted, as they can not be picked up anymore.

2.4. Intention Revision

In addition to the belief revision, which is done during the intention generation, Intention Revision is done to check if a selected intention is still valid once the agent wants to execute it. We specify these different cases:

- **Pick Up**: If someone (including ourselves) has already picked up the parcel that we want to go pick up, it is no longer a valid intention. The same goes for the parcel having timed out. Additionally, a parcel can also be blacklisted (in the case of two agents), this will be explained later.
- **Delivery**: If we are not / no longer holding a parcel (possibly because it timed out and disappeared along the way), we do not want to go to the delivery tile anymore and can discard this intention.
- **Move / Explore**: If during our exploration of the map we find a parcel, we want to change course and discard this intention.

The full flow chart of the intention revision is shown in Figure 2.

2.5. PDDL Path Planner

The domain file of our PDDL path planner defines only the three basic elements we introduced in our introduction: agents, parcels and tiles. We then specified some predicates in order to use them as goal conditions, pre-conditions or effect of actions. The objective is to maintain the path planner as simple as possible, but also sufficiently complete to mannage the environment. The PDDL predicates used for this are shwon in Table 1.

Similarly, the actions are kept simple as well and include only basic tasks like moving in all directions, picking up, delivering or dropping a parcel. These simple actions give us the possibility to interact with the environment as much as we need to complete our tasks. The actions including preconditions and effects are shown in more detail in Table 2.

We generate the PDDL problem file directly inside the code using the APIs. Depending on the case, the goal could be to reach for a parcel or a delivery, or then again to just move and explore or, as we will see in the next part of the paper, to coordinate with the other agent. An example of a generated problem file can be found in our GitHub Repository.

2.6. Results

Now we will examine the result we obtained by running our single agent for 5 minutes on the maps used for the first challenge and comment the results. The collected points for the runs are shown in Table 3.

As it can be seen from the table, the agent works well overall and manages to make a decent amount of points on all maps. However, there are some cases that - while not necessarily reflected in the points - are more challenging for the agent than

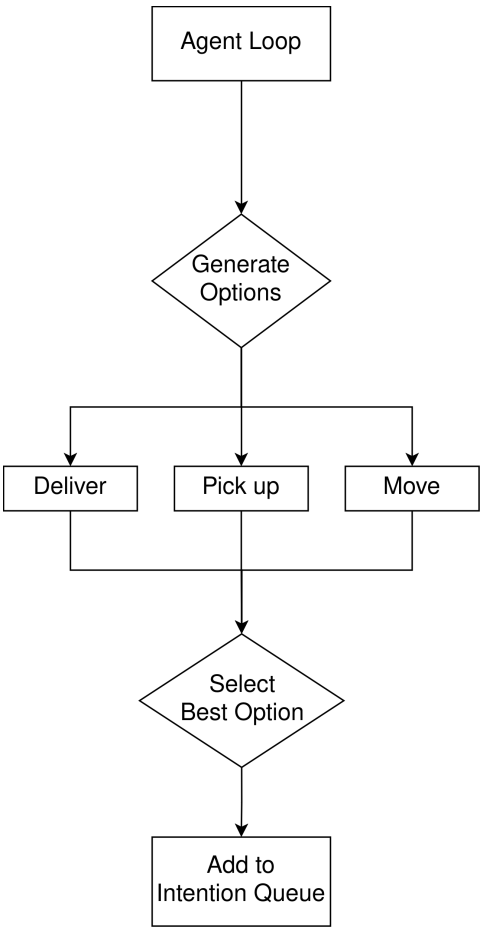


Figure 1. *Flowchart of the Agent Loop*

Map	Points
24c1_1	390
24c1_2	236
24c1_3	506
24c1_4	73
24c1_5	1.804
24c1_6	142
24c1_7	158
24c1_8	210
24c1_9	155

Table 3. *Runs of 5min on Maps of Challenge* 1

others. Two examples for this are shown in Figure 3. The first, Figure 3a, has spawning tiles only on the outermost edges of the map. This poses a challenge for our agent's exploration strategy which favours movement towards the center of the map. Once the agent has however found a parcel, it is likely to see other parcels spawning while it moves away to deliver, making it return to the area later. While it is still possible to generate a number of points on this map, as shown in Table 3, it is somewhat up to chance and results may vary greatly.
The second map, Figure 3b, creates a similar but slightly different problem. All spawning tiles are grouped together in an area in the top left corner, again making it difficult for the agent to find parcels in the first place. Additionally, once it has found them, new parcels appear as it is picking up old ones, making it stay in the area and "hoard" parcels instead of

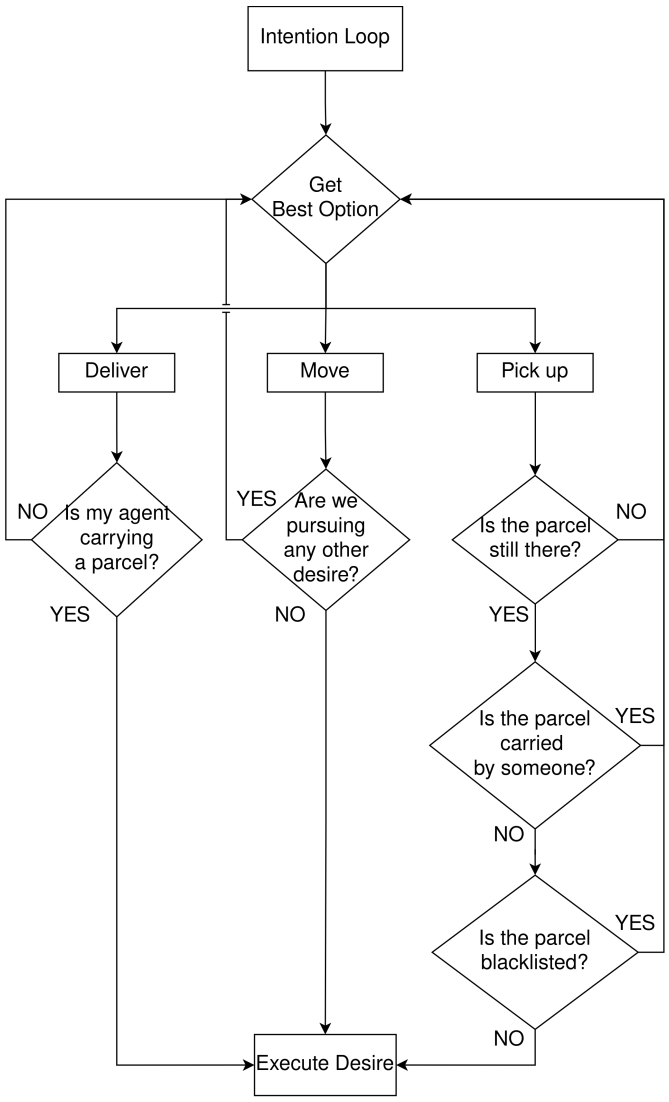


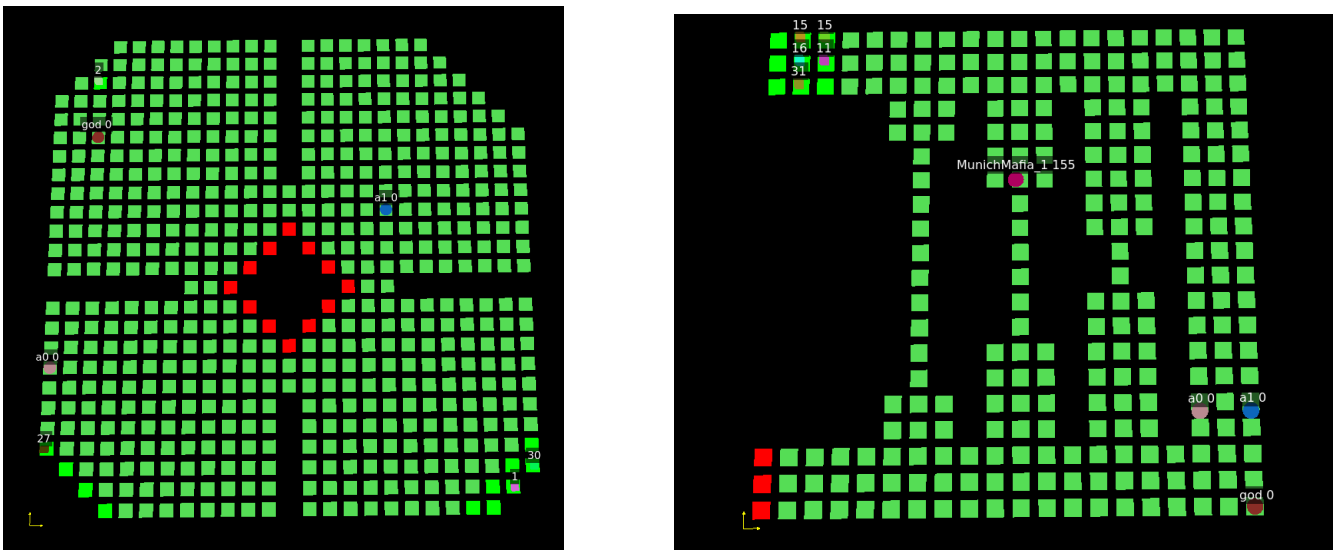
Figure 2. *Flowchart of the Intention Revision*

moving away to deliver them. This will lead to parcels timing out and disappearing while the agent is still picking up others. Furthermore, all delivery tiles are in the lower left corner of the map, a great distance away from the spawning tiles and the way to them leads across paths that are only one tile broad, making it easy for the agent to be blocked by others. Again, it is possible to make a lot of points on this map, but the score is highly dependent on the timing of the spawning parcels and the movement of other agents on the narrow paths.

In the test of the map shown in Figure 4, our agent performed incredibly well. The reason for this is the large amount of points that every package gave us, and the fact that the corridors do not pose a problem for our path planner if there are no other agents around to block them.

3. Two agents

After testing the behaviour of a single agent, we implemented a second agent to leverage the benefits of a multi-agent system: communication, reliability and distribution. Multiple agents can pick up different parcels at the same time, or work together to pick up and deliver a single parcel.



(a) *Map 4 from Challenge 1*

(b) *Map 9 from Challenge 1*

Figure 3. *Most Difficult Maps from the First Challenge*

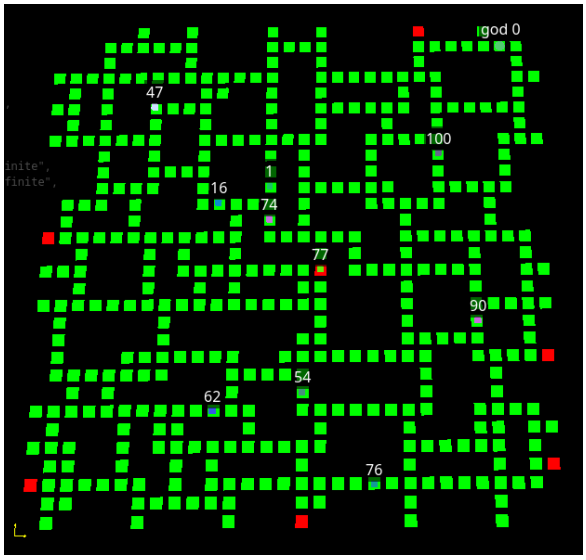


Figure 4. *Best Map from the First Challenge: Map 5 From Challenge 1*

3.1. Communication

In order for them to work together, a communication system is implemented using the `onMsg` functionality and the methods to `shout`, `say`, `ask` and `reply`. In the beginning, the agents broadcast (`shout`) until the other one has received a message and tells (`say`) them to stop broadcasting. In this way, they exchange their IDs and know where to send targeted messages from now on. Afterwards, they continuously exchange parts of their Belief Set, e.g. the parcels' positions and the other agents' positions. Further, they use a `ask-reply` structure to (a) find out where the other agent is going and (b) ask if the other agent can go to a certain position that is unreachable for the one that is asking.

3.2. Coordination

In order to avoid conflicts, the first agent has a higher priority than the second. This means, that if the second agent wants to go pick up a parcel, he asks the first agent where it is going and if the destination is equal, it revises its intention.

Additionally, both agents can ask the other one for help delivering a parcel. If they can not reach a delivery tile, they ask the other agent whether it is able to. If it replies yes, the agent drops the parcel and blacklists it for itself so it will not pick it up again.

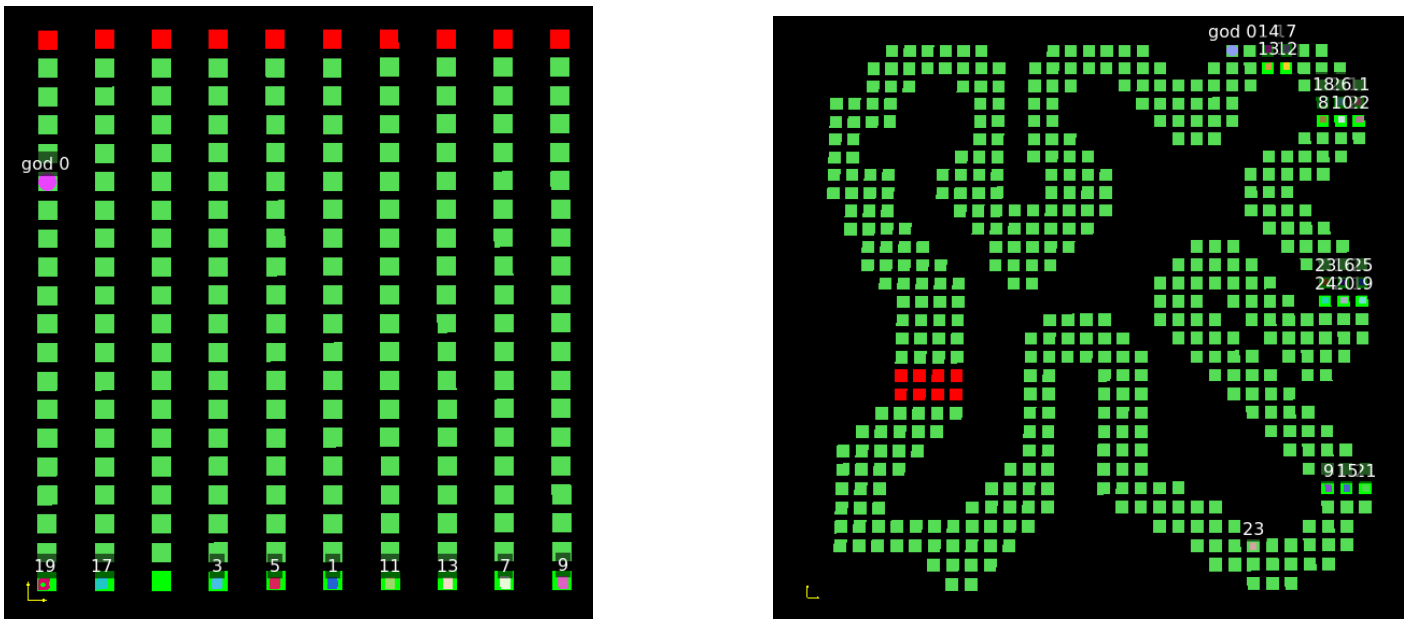
3.3. Results

And now we will take a look on how both agents work in the maps of challenge 2, you can see the results we obtained in Table 4. As before, we will examine the result we obtained by running the agents for 5 minutes on the maps used for the second challenge and comment the results.

Map	Points	Agent 1	Agent 2
24c2_1	490	290	200
24c2_2	124	124	0
24c2_3	1.544	774	770
24c2_4	476	395	81
24c2_5	684	292	392
24c2_6	338	201	137
24c2_7	799	217	582
24c2_8	0	0	0
24c2_9	646	233	413

Table 4. *Runs of 5min on Maps of Challenge 2*

As we can see from the results, the two agents managed to obtain some solid results in almost every map of the second challenge. The map shown in Figure 5a however posed a serious challenge to our agents which were not capable of delivering any parcel at all. The problem with this map was the fact that while the agents were able to pick up the parcel on their lane, they continuously tried to reach the parcels in the neighbouring lanes instead of delivering the one they were holding. While this could be counteracted by checking in advance whether parcels we want to pick up are actually reachable or not, this feature does not exist in the code of our agents, making it impossible for them to succeed on this



(a) *Map 8 from Challenge 2*

(b) *Map 2 from Challenge 2*

Figure 5. *Most Difficult Maps from the Second Challenge*

map. will The second map which posed difficulties was the one shown in Figure 5b. Here, only the first agent managed to make points at all and not a lot of them, while the second agent finished with zero points. The reason for this was similar to the one for Figure 3b in the first challenge: Since the parcel spawning tiles are closer to each other than the delivery tiles and new parcels spawn at high frequency, the agents were "hoarding" as many parcels as possible and never going to deliver. The few points, the first agent made were more due to luck, than any specific strategy.

4. Conclusion

To sum up, it can be said that our agents performed well in most situations, and the implementation of the PDDL path planner and the communication system were successfully implemented. The intention revision system - even if it is not yet particularly sophisticated - is still effective and gave us significant problems only in the scenario shown in Figure 5a. Unfortunately, for some edge cases the system still needs work, but aside from that the planning and BDI Loop worked as intended.

5. Future Work

One aspect to improve in our work could definitely be a better intention revision system that gives us the possibility to ignore parcels that are impossible to reach. A more sophisticated communication system that could enhance our strategies with multiple agents is definitely something to look for as well as a logic that forces the agents to stop collecting and go for a delivery if they already have a certain number of parcels.