Self-driving agents adapting to a city

Agente autónomos e Sistemas Multi-Agente

José Cruz
IST
Mealhada, Portugal
jose.pedro.cruz@tecnico.ulisboa.pt

Pedro Monteiro
IST
Viseu, Portugal
pedro.n.monteiro@tecnico.ulisboa.pt

Rui Melo
IST
Coimbra, Portugal
rui.melo@tecnico.ulisboa.pt

ABSTRACT

Nowadays, ubers and taxis are an essential part of any major city, since they provide easy and quick transportation. With the rise of autonomous vehicles and being impossible for a company to design software specifically for each city it provides, machine learning algorithms are the best solution to ensure a good service.

This project aims to create agents that have the objective of learning how to provide this kind of service using machine learning to allow the agents to create a model that would represent the best place to wait for potential clients which would result in a better service for a client and less cost for the company providing that service.

Naturally, it would be incredibly inefficient for a single agent to learn a city by itself so communication and cooperation with agents from the same company is necessary to guarantee a rapid adaptation to the city.

Our solution, although it's adapted for AI agents, could also be helpful to human drivers who can apply our methods to create models of the city they work in and achieve higher performances and, consequently, accomplish better profits. We plan to have an impact not only in the future but in the present as well.

KEYWORDS

Multi-agent system, Machine learning, self-driving, commercial driving, reinforcement learning.

1 Introduction

In a high functioning city with multiple interest points (ex: commercial center, railway stations, etc.) transportation services are needed. These different points have different influx of people, so knowing where people will be and when is key to provide an optimal service while reducing costs like fuel and tires.
Nowadays, millions of commercial drivers face this problem every day but, in a future, where transportation will be mainly made by self-driving vehicles, those will also have to adapt to different cities so solving it will not only improve the present but create good roots for what is to come.

Our objective is to create a number of self-driving agents that would work for the same company and cooperate within that same company to provide clients with rides from one location to another. Since each agent belongs to a company and cooperates with its co-workers, many other companies can emerge to compete between them.

The goal of an agent is to pick up the passenger at its location and deliver it at a location set by that same client without making it wait too long, while minimizing traveling costs that in real life would result in less expenses like gas and energy.. With this goal in mind, the agents of the same company must communicate between each other so that every agent learns at the same rate and has the maximum information available when making a decision.

In our project, we use reinforcement learning to study the agents and find how well they can adapt to our environment and provide a better service to the clients.

We'll study 4 cases involving learning and reactive agents that will ultimately legitimize our results.

2 Approach

The environment consists of a 2-D city with 7 different interest points with those being a hospital, a restaurant, a school, a shopping center, a set of apartments, a court and a market, represented by letters from A to G, respectively. These points are connected through a simple one-way road.
The city is placed over a grid to simplify the movement process and overall city creation.

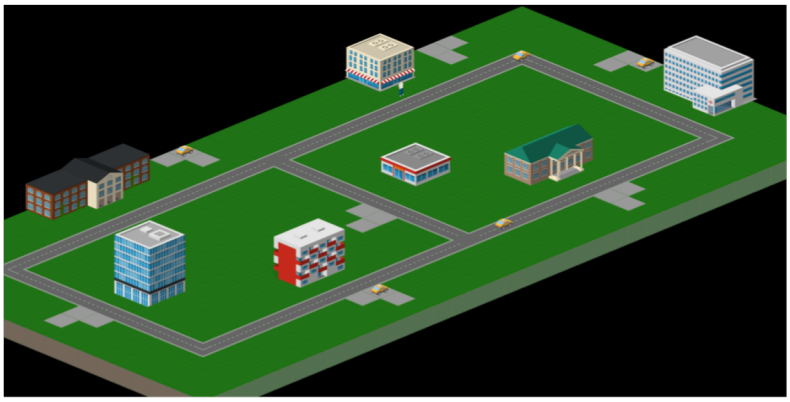

Figure 1 - Environment

Each interest point has a parking lot with 3 parking slots to allow the agents to wait there while parked. It would not be good if the agents were driving around until they found clients as they would in a real context lose gas. So, after leaving a client, each agent will either stay at that interest point parking lot or drive to another interest point and wait in its parking lot.

Since different interest points will have a different probability of generating clients than others, the agents should be capable of relocating themselves to minimize costs and waiting times.

The overall rate for spawning a client is the same, but each interest point has a different probability of generating, in fact, a client. This phenomenon is intended as we tried to simulate a real city.

The spawning probability for each interesting point is as follows: 80% for the hospital, 70% for the restaurant, 20% for the school, 10% for the shopping center, 10% for the set of apartments, 60% for the courthouse and 50% for the market.

The agents have access to a GPS class that, like in the real world, returns the path from one place to another. To be noted that, even though the GPS returns a path to be followed, it is the agent who decides where to go at each interest point.

To summarize the agent's actions can be as followed:

- Stop at an interest point.
- Drop a client.
- Access a parking lot and wait for the next client.
- Move from one interest point to another.
- Pick-up client.
- Stop to avoid crashing.
- Wait at a parking lot.
- Choose a parking lot.

Four agents' architectures were implemented, and each one can be selected through the initial menu. In all the four architectures, the agent who will be selected to pick-up the client will be the nearest one. Also, the action that will interfere with the result is the last one, "**Choose a parking lot**" as that is the objective we aim to optimize in this project.

To further understand and evaluate our solution, a trip is only assigned to an agent when it is parked. For that reason, agents positioning in the city is even more important as it wouldn't be enough to just drive to the furthest parking lot waiting for a trip to be assigned at any point it passes through.

3 Agent Architectures

Before talking about the decision-making process, the rest of the agents sensors and actuators. The agents when assigned to a trip leaves the parking lot and travels to the client's location, picks them, then delivers the client to its destination and drops them, then chooses a new parking lot taking in consideration its architecture and drives to it and the process repeats itself indefinitely.

Also, the agent cannot refuse or ignore trips, so it always performs the supra mentioned process.

At any time, the agent may face an obstacle, another agent that is picking or dropping a client, that's why agents have a sensor that before moving forward detects if there is another car stopped, if so, the agent stops and waits for the car in front of him to start moving. The same applies in the crossroad where the external road has priority so an agent that is trying to turn left there must first verify if a car is already on the crossroad.

Given all this, we tested 4 different decision-making processes but, no matter what process the agent has this reactive behavior when it comes to actions that are not "**Choose a parking lot**", with "**Stop to avoid crashing**" being prioritized.

3.1 Decision Making Process

We test 3 different decision-making processes.
The first one is completely random. After dropping a passenger, the agent chooses the next parking lot randomly, so it does not take in consideration passed trips. When arriving at the parking lot if it is full, then the agents make another random decision to decide the substitute spot to park.

The second alternative is always selecting the nearest parking lot, again not taking in consideration past events. If that parking lot is full the agent selects the second nearest and repeats the process until it parks.

Finally, the last approach is having a learning agent. Using reinforcement learning (described in the next section) the agents try to learn the environment and find by itself the best parking lot to park to better serve the clients and to minimize costs like gas.

The agent, when choosing the next parking lot will have an 85% chance to choose the best parking spot taking in consideration the learning and 15% a random one. This probability is to allow the agent to continue to explore new options and learn the city in its entirety without an option being missed. This approach also allows the agent to adapt if the probabilities change mid-way through the testing. This last hypothesis was not tested.

Even though there are only 3 different decision learning processes we tested 4 cases.

The last one also uses the learning approach but when an agent concludes a trip it communicates the trip to the other agents, so they can all learn at the same pace and achieve convergence faster.

Recapping we've tested agents completely Random, Purely Reactive which we can predict the behavior as they always select the nearest parking lot and Hybrid Agents that can learn the environment and react to adversities that may come in the road.

3.2 Reinforcement Learning

Reinforcement Learning is the training of machine learning algorithms models to find the best action to perform under a certain situation/state.

The world is modeled as discrete and finite and can be mapped as a Markov chain. Every action performed in a certain state is associated with a reward or cost (in our project we penalized agents for doing a certain action so from this point further we will consider only the cost). The aim of reinforcement learning is to minimize this cost.

In our project we decided to use the Q-learning Algorithm. This algorithm updates the agent's Q-function using the following formula.

$$Q(s,a) <- Q(s,a) + \beta(cost + \gamma * q - Q(s,a))$$

with:

$s = nearest\ park\ after\ concluding\ a\ trip$
$a = chosen\ park\ to\ go\ after\ the\ trip$
$q = minimum\ Q\ value\ of\ the\ chosen\ parking\ lot$
$\gamma = discount\ rate = 0.9$
$\beta = learning\ rate = 0.3$

Adapting to our scenario, each agent has a Q value matrix that is 7x7, each line represents the state and each column an action. For example, Q (A, B) represents that after the trip the nearest parking lot is A and the chosen to wait for a passenger is B.

The cost is calculated by adding the length of the path to the chosen parking lot and the length of the path to pick the client. Minimizing this cost is ideal in our approach because it would represent an equilibrium between the distance traveled by the cars, which results in less expenditure, and the time Clients wait for a ride, which is representative of a good service provided by the agents.

4 Comparative Analysis

The first metric used to evaluate the different approaches was the distance (measured using the grid) in order to get the passenger. Regarding this metric, the best approach expected was when the agents were communicating and learning. In fact, this happened, also it was expected that the communication approach was slightly faster than what did indeed happen.
It was expected that, with the agents learning progressively, they would stabilize in a value that should be lower than choosing a place at random (having the worst performance between all the approaches). The Purely Reactive architecture even though it does not have a learning ability did not have that much worse results as the best approach.

Although, this test displays a counter-intuitive result, after various iterations, the Communication Architecture starts getting worse than the Learning by Itself one, this is because the agents start to learn that is better to travel a greater distance in the path to the passenger if it means a smaller distance in the path to the parking lot. As seen in Figure 3, the difference between the learning by itself and the communication architectures is way bigger than the one shown in Figure 2. This results in a better performance overall displayed in Figure 4.

More iterations would cause the Learning by Itself agents, that are slower in learning, to realize the same thing, and the lines would be of the same value and not different.

The values presented in the chart presented in Figure 2 would vary even more if the amount of passengers at each moment decreased, their overall spawning probabilities went down or a different number of cars (hypothesis tested ahead). This way, if the agents did not recognize a semi-optimal place to wait for customers, they would have to travel more than they did in this scenario.

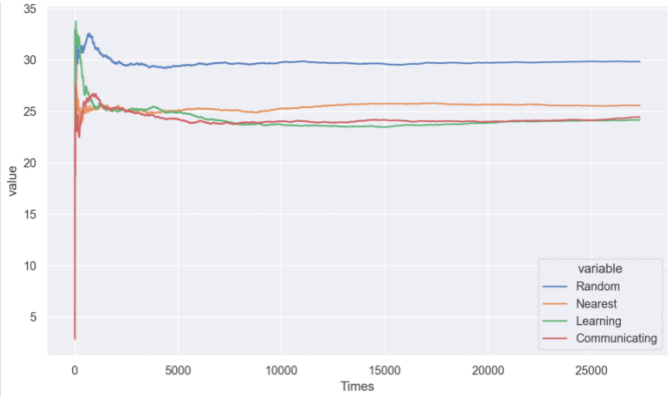

Figure 2 - Distance Agents travel to get the passenger

The second metric used to evaluate the different approaches was the distance each Agent travels to a parking lot after delivering a customer to its destination.

Obviously, as seen in Figure 3, the expected lower value is the approach where agents would travel to the nearest parking lot, whereas the higher value is when a parking lot is chosen at random. The approach where agents would travel to the nearest parking lot implies that, after delivering a client to the proper destination, they would park at that destination's parking space. This is a simpler way to try to minimize costs, even though in the great scheme of things, it is not ideal.

The next best approaches are Communication and Learning, with Communication outperforming the Learning by 16% after 25000 iterations. To be noted that, even though the Learning approach can convergence to numbers identical to the Communication in terms of performing eventually, having multiple agents communicating in an environment can lead to achieve better results faster (less iterations).

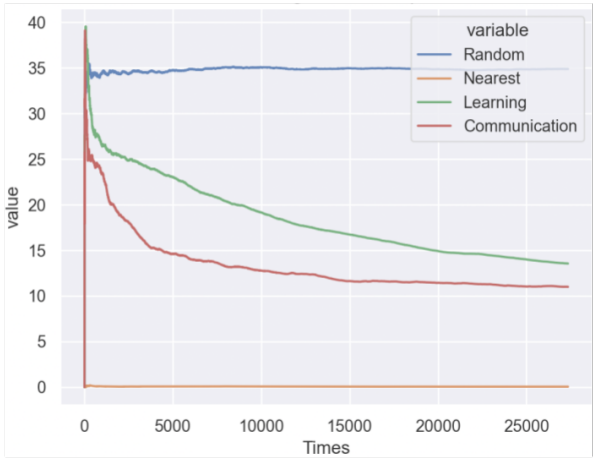

Figure 3 - Distance Agents travel to park

The third metric is regarding the quality of service, more specifically, the amount of time a client has to wait for an agent to arrive at its location, Figure 4.

The worst scenario is regarding the Random approach. Having the agents positioned themselves randomly across the city to pick-up clients is, as expected, the worst approach, taking the largest amount of time to reach a customer.

In our environment, the best performing approach regarding this metric is the Nearest approach. Currently, we have 5 agents for 7 interest points, meaning that, even with all the agents spread randomly, they might attend their clients with a good performance. In a larger environment or with a lower number of agents, both the Learning and Communicating approaches would outperform the Nearest one, since spreading randomly would not provide a consistent faster service.

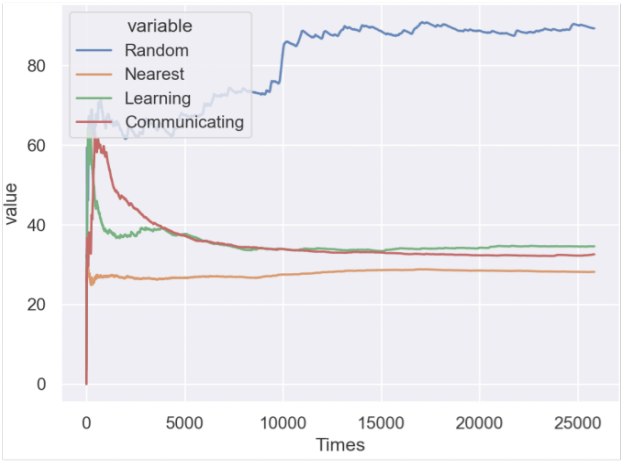

Figure 4 - Time that Clients wait for a ride

All things considered, with 7 interest points and 5 agents operating in the city, we can see that having a Random Policy is, as expected, not ideal in any scenario and the agents should be able to have a defined policy associated.

Regarding the Nearest approach, in a small environment, it might be useful, since an agent would not get high costs related to fuel or extra kms. It does not minimize costs in any way, but it still has fast customer service. This approach will turn out to not be beneficial once the agents are presented in a larger environment.

Both the Learning and Communication approaches have similar performances but allowing agents to communicate between each other provides faster recognition of the policy and strategy they will follow in the near future (in that same environment).

As explained, the Nearest approach in this specific environment with this exact number of cars is the best architecture, so it would be a good thing for the Learning agents to converge to a Q-value function that would be very similar to the Nearest One. This is exactly what happens, as seen in Table 1, the policy learnt by the Q-learning Algorithm is very similar to the Nearest Approach, which is a very good indicator that the Learning Approach was successful. The policies only differ in one action and more iterations that unfortunately we did not have the time to perform, could result in the perfect match between the two.

Nearest Park	A	B	C	D	E	F	G
Chosen Park by ML Agents	A	B	C	E	E	F	G

Table 1 - Learning Agents Policy

In order to verify the hypothesis mentioned priorly and further verify conclusions, the environment was tested by having 7 agents instead of 5.

As a hypothesis, it was suggested that the performance regarding the time that a client has to wait for a ride was directly dependent on the environment, more specifically, to the amount of agents present and the spawning rates. It can be further concluded, by Figure 5 and Figure 6 that when presented with a bigger number of clients, the performance attached to the Nearest approach lowered.

Once again, the Communication Agents may seem to be worse than the single learning ones, but the agents are simply learning that it is preferable to have a huge diminution in the distance to the parking lot in exchange for a small increase in Clients waiting time. With more iterations, the green line would have the same behavior.

It is notable that with a bigger number of agents the Purely Reactive agent has a worse overall performance than the Hybrid Ones, if we take in consideration not only these 2 charts but the distance to pick up the client as well that is better in the Hybrid Agents by about 22%.

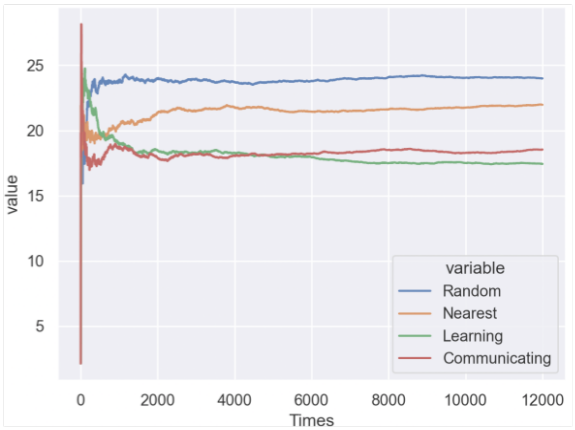

Figure 5 - Time that Clients wait for a ride with 7 agents

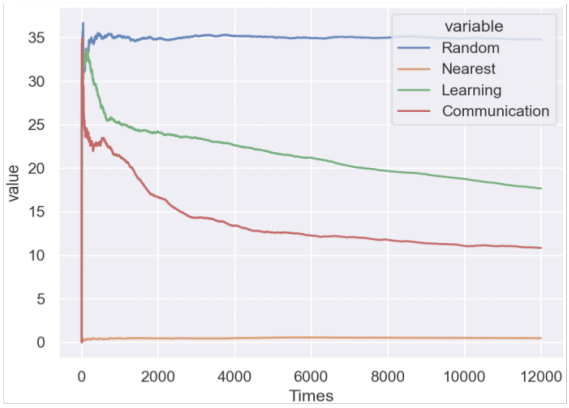

Figure 6 - Distance travelled by agents to park with 7 agents

To further evaluate the impact of having more cars we decided to test again but this time with 10 agents. Not only that but with a higher number of agents it is also interesting to evaluate if a higher number of parking spots would make a difference.

Figure 7 and 8 show that with more agents not only the time a client waits for a trip reduces but also is very similar to the distance a car travels to catch the client. This is because as there is a higher number of cars almost every time a client requests a trip there is one agent available and instantly proceeds to pick the client.

Furthermore, with 10 agents the number of parking spots is not a determinant factor in the results.

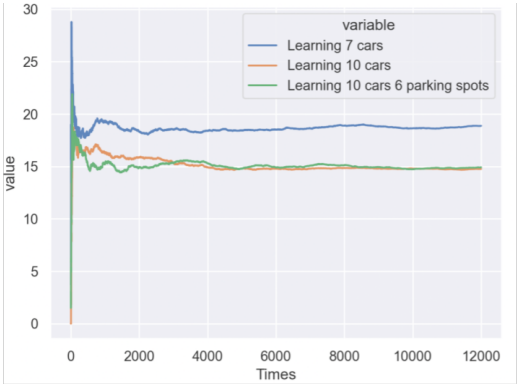

Figure 7 - Time Comparison between different number of cars and parking spots

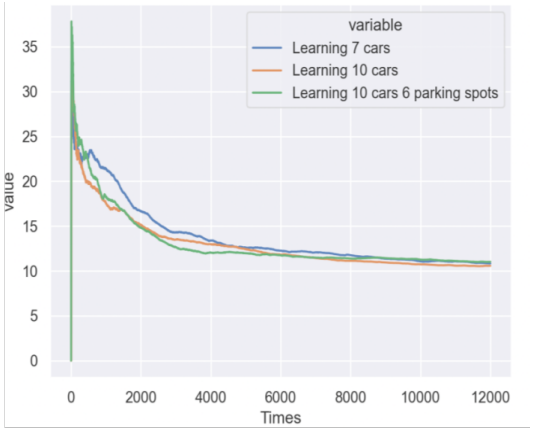

Figure 8 - Distance to pick the passenger Comparison between different number of cars and parking spots

Finally, we wanted to test if the agents were capable of learning the interest points with highest probabilities of spawning a client.

For that a change in the reward was made where the only factor was the distance it took to pick the client with absolutely no regard for the costs of traveling.
Remember that the interest point with highest probability was point A, the hospital and the second one was point F, the courthouse.

This test was made with 10 agents and 6 parking spots available at each location. Table 2 shows the difference between the policies of the previous test (normal reward 10 agents 6 parking spots) and the agents with the new reward.

The results are clear, and the agents were able to identify the interest points with the highest probability of having a client. Although it may seem that agents in some situations would prefer F or B this view is not entirely correct because as all agents choose A to park the parking lot would immediately be filled and cars had to go to park somewhere else. Note that the second favorite spot to park in D and E was A in both, confirming that agents did learn the highest probability.

Nearest Park	A	B	C	D	E	F	G
Chosen Park by Agents with normal reward	A	B	C	D	F	F	G
Chosen Park by Agents with the new reward	A	A	A	F	B	A	A

Table 2 - Policies for Agents with normal reward and the new reward.

5 Final Considerations

In conclusion, we believe that our project was a success. We were able to implement the proposed environment and agents with different architectures that would have different behaviors.

Focusing only on the Hybrid Agents that were capable of learning the environment, we successfully implemented those, as shown in the previous section. They started with high values but by interacting with the environment were able to outperform themselves until convergence.

We tested various possibilities and changed some variables to precisely address the problem described and evaluate our solution.

Even though we believe we did a good work, some other factors could be tested such as changes in probabilities midway through the simulations (For example a restaurant constantly open and closing like in real life).

Our solution was proven to be adequate to our problem and can give insights to future implementations of self-driving agents in our normal day life.