Rent-A-Car - Group 35

Eduardo Paiva
89441
eduardo.paiva@tecnico.ulisboa.pt

Pedro Diogo
89521
pedro.m.diogo@tecnico.ulisboa.pt

André Santos
91000
anselmo.santos@tecnico.ulisboa.pt

1. ABSTRACT

Nowadays, our society is heavily dependent on vehicles of transportation, with the private car being the most common one. However, this type of vehicles is very fuel consuming, which leads to extremely expensive trips and causes harm to the environment.

A lot of people are concerned to take their private cars to large cities, just as Lisbon, because usually there are no parking slots available or because they are quite expensive for a long parking time. To solve this issue our group has decided to create a rent-a-car approach with a few tweaks to it, starting with the inclusion of electric and autonomous cars. The former prevents the emission of greenhouse gases into the atmosphere. While the latter opens up a number of interesting opportunities, such as allowing the client to leave the car wherever he pleases, since the autonomous vehicle will easily find its way back into the parking lot.

To sum up, with this propose we expect to witness a cutback on the use of private cars in the large cities and an increase in the number of tourists due to this new way of exploring the city.

2. INTRODUCTION

2.1 Motivation/Problem Relevance

Our motivation for this project was mainly driven by the vastly amount of reports on the environmental problems caused by the use of private transportation [1].

Another issue that we detected was the lack of parking slots and how expensive they tend to be in large cities, such as Lisbon, and the huge amount of time wasted finding a place to park [3].

The solution that we propose to solve this problem will not only reduce the emission of harmful gases to the atmosphere, but it will also make sightseeing through the capital more appealing. With this approach, tourists and locals will be able to enjoy the city's landscapes without car noises and polluted air [2]. And to top it all off, everyone can choose their own route, instead of having to rely on any other type of public transport.

2.2 Problem Definition/Objectives of the problem

We plan to analyze how cars manage all the renting requests that the central receives and how they interact with each other to maximize the satisfied requests.

In order to select the best car to execute the client's request, for each renting request, a car will search if it is the most suitable car to attempt that request among the other cars. Therefore, and with the help of the Central, at a given timestamp a car can have information about all the other cars such that position of the cars, battery life, etc.

Therefore, the main goal of all the agents is to cooperate and to fulfill all the requests, minimizing the distances that the cars have to travel, in order to maximize their battery life.

2.3 Problem Requirements

The biggest challenge we have to overcome in this project is how to deal with the dynamic changes that can occur in the environment, such as the random client's spawning position. Another issue we will have to deal with is the complex implementation of the coordination mechanisms, for example deciding which agent accepts the client's request, that can improve our agents performance over time. The final requirement will be allowing our agents to freely navigate the map searching for clients, but always making sure that they do not run out of battery.

3. Multi-Agent System Architecture

To approach this multi-agent problem, we consider that there will be only one Central unit to keep tracking of all the renting requests received, like a Database. However, there will also exist a Workshop, several Parking Spots, and several Cars (the agents) with an incorporated battery.

3.1 Organization of the System

The following figure provides a representation of how the system is organized.

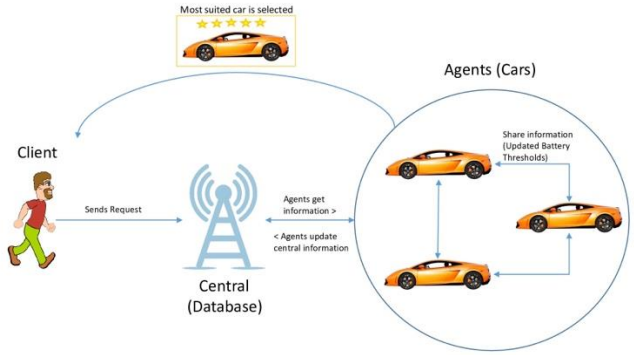

Figure 1

The Central, which works as a database to store information from all the environment and the agents, receives all the requests to rent a car and in each step, the cars "ask" the Central for renting requests. After receiving the requests, the cars communicate with each other and deliberate about their current position and the position of the clients to identify the closest car to the client's position and, if the car fulfills the request's requirements (enough battery to travel a certain distance) then it shall fulfill the request. Although this will not always be the case when the agent is in a risky approach. The agent will often take risks to learn the optimal battery threshold.

Thenceforth, it starts moving towards the client until it reaches him and from that point on, he has full control over the car. However, there is always the possibility that the nearest car does not meet the requirements, due to low battery or due to the fact that is unavailable at the moment of the request. In this case, the request should be forwarded to the next nearest car and repeat this process until a car that meets the requirements is found. In case there is no car available to fulfill the renting request, the client will be inserted into the waiting queue.

3.2 Properties of Agents and Environment

The agents in our system follow a reactive horizontal architecture. It is reactive because will react to the renting requests that are sent to the central and will also make reactive decisions/actions when having a destination goal

3.2.1 Environment Properties

The environment, shown in Figure 2, consists of a simplified map of Lisbon containing parking spots where the agents are supposed to charge their batteries while waiting for requests. The requests distance can vary between 10 and 50 km, meaning that each person needs to share in advance his estimation of the distance she wants to travel.

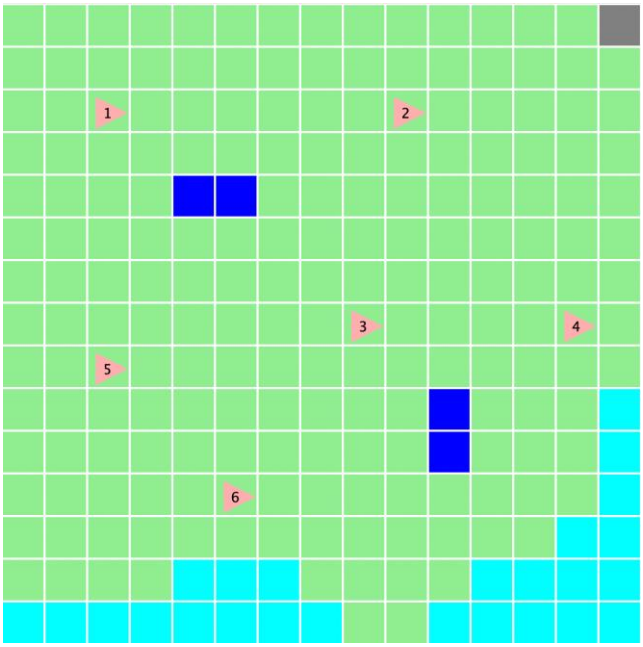

Figure 2. Environment: a simplified map of the metropolitan area of Lisbon.

The cars are represented as triangles and are numerated so that we can follow their trajectory and visualize their behavior individually. They also have a color code, being pink the one representing that the battery is above the threshold, yellow the one representing that the level is below the threshold, and red meaning that they will run out of battery in the next step.

The parking spots are represented as dark blue squares and the workshop is represented as a grey square on the top right corner.

The environment also has a Workshop, where a car goes once it runs out of battery. Once it recharges fully, it will return to action from here. There can only be one car at a time in the workshop, so, if one is already there, a car needs to wait for the other to leave in order to go to the workshop.

Several Parking Spots, distributed in the environment and always on inferior number than the cars. These are used by cars to recharge their batteries and there can only be a car per parking spot at a time.

3.2.2 Agents Properties

The agents are going to be:

- Autonomous: agents will know how to achieve their goals on their own;
- Adaptive: The cars will learn from experience how to better handle their battery life, by redefining a new threshold of the distance needed to recharge, every time a car attempts to fulfil a request.
- Rational: agents will try to minimize the battery spent in every trip;
- Reactive: If there is a request, the central will react to it by sending a car to the client's position.
- Cooperative: The cars will have to cooperate among them so that none of the cars' batteries reach zero. For example, if a car that just arrived to a parking spot needs to recharge but there are no slots available, then the car with the highest battery level will give up is parking slot, as long as its battery level is equal or superior to the threshold defined, otherwise, the car will have to wait until one of the others cars battery reaches the threshold value;
- Mobile: The autonomous cars travel/move through the city of Lisbon. On the other hand, the central, parking spots and workshop do not move;

3.3 Sensors and Actuators

3.3.1 Cars

The Cars will have the following sensors:

- What is my current location?
- Is my ahead location free?
- Is this my destination point?
- Do I need to recharge the battery?
- Did I run out of battery?
- What is the nearest available car parking?
- Did I travel the requested distance?

The Cars will have the following actuators:

- Move;
- Link to a client;
- Give control to the client;
- Drop client;
- Link to a car parking;
- Recharge battery;

3.4 Decentralized Architecture

We opted for a decentralized approach, as shown in Figure 1, instead of a centralized one, since we realized that having a centralized architecture with a central unit and making it control all the cars and requests would be way more complex to implement when comparing to a decentralized approach. Despite not being as efficient as a centralized approach, we felt that sticking to this choice would be an opportunity to address different contents that were taught throughout the Autonomous Agents and Multi-Agent Systems course, providing us the opportunity to explore concepts such as cooperation and negotiation in broader terms, since this approach is much richer from the point of view of learning and actually visualizing the results unfolding themselves.

The cars negotiate with each other, distributing the renting requests and eventually reaching a consensus. The chosen car can then proceed to conclude the renting request.

One of the advantages of this approach is that, if a lot of renting requests are issued at the same time they can be distributed among all cars in the same negotiation and every decision taken by the agens broadcasted to every other agent through the central. The allows the agents to react more effeciently to changes in the environment.

4. Agents Behavior

Regarding the agents' behavior, we chose two approaches to further investigate how the car selection can affect how the renting requests will be handled.

4.1 Conservative

With this approach, we want to avoid decisions that would produce a negative effect on the way the renting requests are dealt with.

Let's take the example where a car picks up a client and has a battery level of 70km to fulfill a request of 80km. In this scenario, is very likely that the agent will not be able to fulfill the client request and make him unsatisfied. Here, the decision is sub-optimal, and that's not what we intend.

So, in the Conservative behavior, cars do not learn, they will always fulfill requests that they are sure they can accomplish, they always play *safe*.

And what implications does this have on the behavior of the agents?

Well, to begin with, the most notorious impact should be the low number of unsatisfied clients.

On the other hand, we expect to obtain a high mean waiting time for the client's request to be satisfied.

4.2 Risky

With this approach, we want our agent to learn through experience, letting them accept requests even when their battery is below the threshold.

A car chooses between completing a request if it has enough battery to go to the client position and complete the distance requested (conservative) or it chooses to take a risk and learn.

The maximum distance made by a car from the moment it completed the request and drop the client until it reached a parking spot is always stored, so it can be used to calculate the reward's importance, as it will be explained later.

There are 4 situations where an agent learns:

- When runs out of battery before reaching a client
- When runs out of battery with a client
- When it reaches a parking spot, after completing the request
- When it runs out of battery before reaching the parking spot, after completing the request

Table 1 - Reward function

	Reward
1º situation	Minimum distance left to the client + Client's request distance + Default reward
2º situation	Distance left to complete the client's request distance + Default reward
3º situation	Battery wasted to reach the parking slot - the current threshold
4º situation	Battery wasted trying to reach the parking slot - the current threshold

The reward function has two possible situations: the car reaches a park or not. If the car reaches the park and the battery wasted is lower than the current threshold, then it means we need to reduce threshold's value.

To do this we apply the third case in the reward function table. It has to be said that the battery wasted only starts to count from the time the client was dropped to the time the car reached the park. The other decisive value is the importance, which is used to make sure the reward obtained for cars that travel long distances to reach the park are more important. This way, a car that wasted small amounts of battery, but was very close to the parking spot does not have a big influence in the threshold calculation.

The other possible situation is when the car does not reach the park. However, in this case there are a couple of minor situations that can occur. The first one being when the car fails to reach the clients position. The update reward used in this situation is that presented in table1 for the first situation. The default reward applied is the battery the car needs to take one more step after dropping the client. With this default reward, we seek to achieve the battery threshold that allows a car to fulfill a client's request most of the times, but always making sure that it has at least the possibility to recharge in the nearest park. The reason why this default reward is not higher is simply because the workshop allows the cars to be repaired fairly quickly. Altough, we noticed that with a greater number of agents this would not be case.

The next minor situation occurs when the car fails while carrying the client. The reward used in this situation is the one given by the second situation in table 1.

The last situation is when the car successfully fulfills the clients request, but fails to reach the park and the battery wasted is greater than the current threshold. When this event occurs we want to increase the value of the threshold, so we use the fourth situation presented in table 1.

Knowing the reward, it is now possible to calculate the new battery threshold. This is done by adding to the existing one the new value learned plus the average between the other thresholds. This new threshold is then propagated to all other cars.

Initially the cars will learn a lot since the probability for them to do this is extremely high, but with time, this value will go down and so will the learning.

5. METRICS

In order to reach our goals and take conclusions, we decided to include as many metrics related to clients and cars as possible since these are the main focus of our project: satisfy the largest number of customer orders and not let cars ran out of battery.

5.1 Number of cars that battery ran out

We consider that a car fails after its battery is fully discharged. When this occurs, the car informs the central and it increments this variable. This metric is important so that we can keep track of whether the number of cars that are running out of battery is decreasing over time, as it is to be expected, or not.

This metric is used in the risky behavior approach.

5.2 Satisfied clients / Unsatisfied clients

Every time a client travels the entirety of the distance requested, we consider the client was satisfied. On the other hand, a client is considered unsatisfied when the car runs out of battery before he completes the requested distance.

This measure is useful when comparing risky behavior vs conservative behavior.

5.3 Learned battery threshold

This is a metric that only has relevance when cars are in learning mode(Risky). Since in this mode cars are trying to learn what is the best threshold that they need to have to reach the nearest car parking for charging, this metric keeps track of the threshold variation over time.

5.4 Mean waiting time for clients

Every time a client requests a car, the central recalculates the average time that the client had to wait from the moment of the request till the moment the car arrives.

This measure is useful when comparing risky behavior vs conservative behavior.

5.5 Number of times parking has been given up

Every once in a while, a car may not find an available parking slot, so it travels to the nearest park to wait for the other to give up his place. When this happens, the central keeps record of this event.

This measure is useful when comparing risky behavior vs conservative behavior.

6. Experiment Discussion

The experiments presented below were executed using a conservative approach with a threshold of 100, which is really close to the optimal threshold for this given environment. It was also used a risky approach with a initial threshold of 50, a learning rate of 0.3 and an initial epsilon of 0.9 for the e-greedy decisions. Both of these approaches were left to run for 50000 steps. Noteworthy, the epsilon used for the e-greedy decision in the risky approach never goes below 0.05, that is the reason why the battery threshold never stabilizes.

6.1 Learned battery threshold

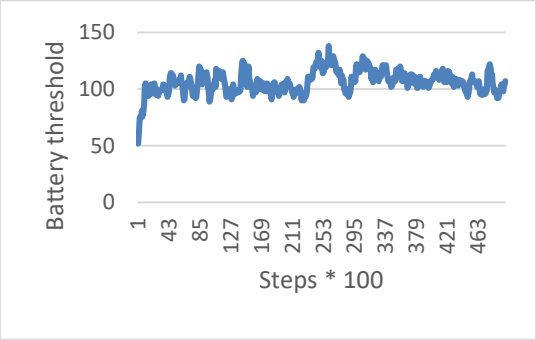

Figure 3. Learned battery threshold per 100 steps

As it is possible to observe in figure 3, the optimal battery threshold for this environment is around 100. We are also able to notice that the threshold changes rather quickly in the beginning and then starts to stabilize in a certain interval of values. This interval of values exists because the enviorenmnet is not static, the client's spawning position is always random.

6.2 Number of satisfied clients

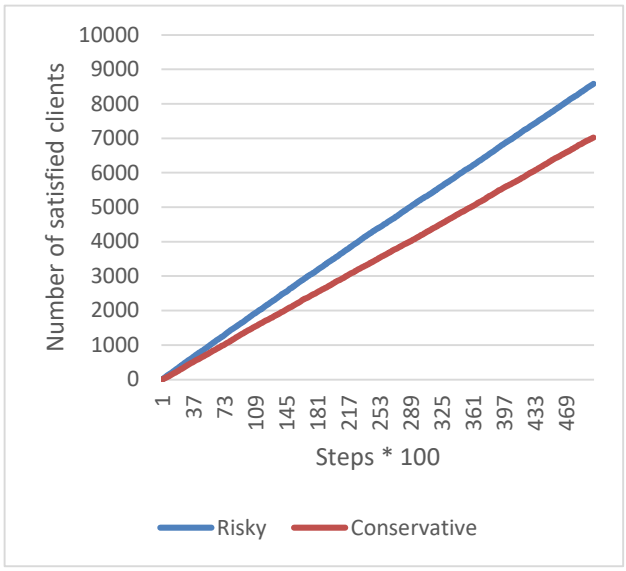

Figure 4. Number of statisfied clients per 100 steps

The number of satisfied clients is way higher in the risky approach than the conservative one, as it is possible to observe in figure 4. This occurs because the agents will take a risk and try to fulfill the client's request, even if it means that they can not reach a parking spot to recharge.

6.3 Number of unsatisfied clients

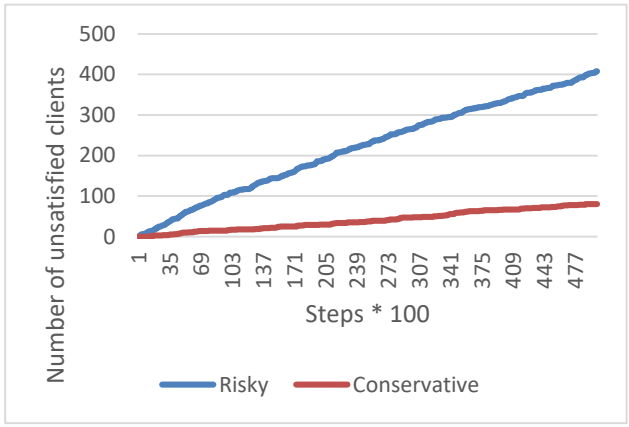

Figure 5. Number of unstatisfied clients per 100 steps

On the other hand, the risky approach has a higher number of unsatisfied clients, as we can see from figure 5. This happens because the agents are trying to learn the best threshold through experience, which results in them making mistakes, such as failing to complete the client's requested distance.
However, we predicted that the number of unsatisfied clients would start to decrease over time and this was not the case. We assume this was due to the fact that the client's spawning position is not constant, which led to an error while learning the optimal threshold.

6.4 Number of times a park has been given up

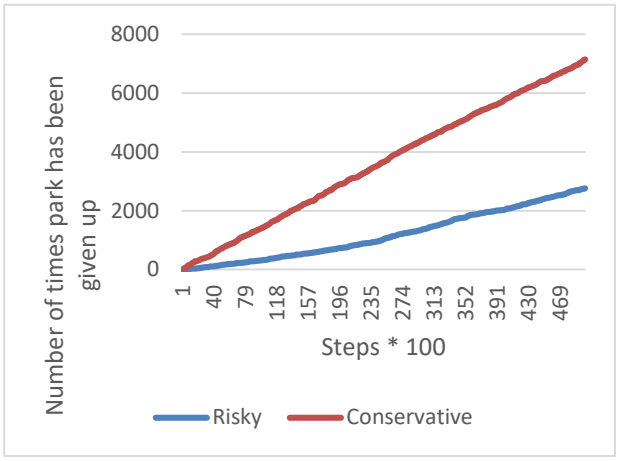

Figure 6. Number of times a park has been given up per 100 steps

In figure 6 it is possible to witness that the risky approach has more agents trying to learn and accepting client's requests and less agents trying to recharge, which implies less parks being given up. As it was to be expected.

6.5 Mean time for client's request to be fulfilled

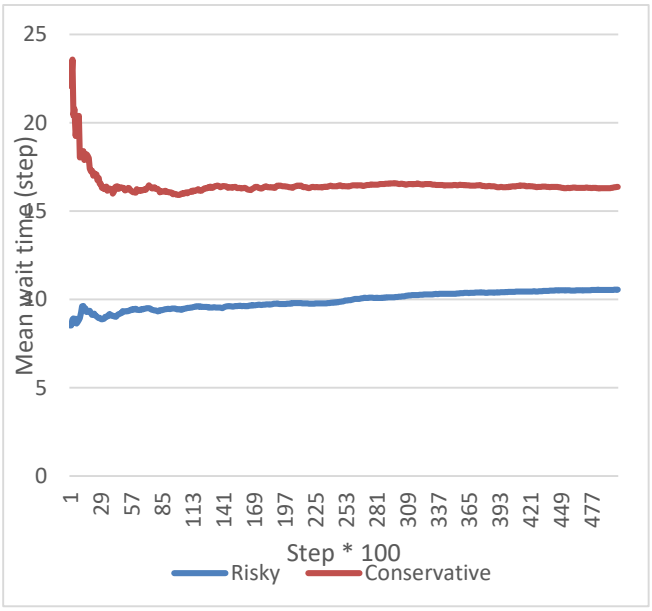

Figure 7. Mean waiting time per 100 steps

Final, we have the mean time a client as to wait to be picked up. We expected a huge difference between the values obtained from the two approaches, with the conservative having a higher value, which is clearly represented in figure 7. This occurs because in the conservative approach the agents spend more time trying to keep their battery above the threshold. Consequently, the mean waiting time for clients to be picked up will increase.

7 Conclusion

With this project we were able to identify the main differences between a risky and conservative approaches, as well as their respective advantages and drawbacks.

The risky approach allowed our agents to learn from experience and to adapt to a given enviorenment. Despite failing a lot in the early steps of the run, the agents were quickly able to learn from these mistakes and use the newly adquired information in their favor.

In spite of having a nearly optimal threshold, the conservative approach does not allow our agents to take any risks. This results in way less failed attempts, but it also means that it is almost impossible to achieve the goal set for this project, which is to fulfill the maximum number of clients' requests in the shortest amout of time.

To conclude, during the implementation of this project we were also able to witness the importance of implementing mechanisms of agents' coordination, such as having to decide which agent is the most suited to fulfill a certain client's request and also discussing which agent needs to take the parking slot to recharge.

8 References

[1] https://www.publico.pt/2020/02/05/local/opiniao/futuro-lisboa-nao-passa-carros-1902869

[2] https://www.cmjornal.pt/portugal/cidades/detalhe/estacionamento-vai-ser-mais-caro-em-lisboa

[3] https://eco.sapo.pt/2021/03/09/lisboa-e-5a-cidade-europeia-com-mais-carros-eletricos-e-melhor-qualidade-do-ar/