ADCAIJ: Advances in Distributed Computing and Articial Intelligence Journal
Regular Issue, Vol. 6 N. 4 (2017), 5-14
eISSN: 2255-2863
DOI: http://dx.doi.org/10.14201/ADCAIJ201764514

Using trust degree for agents in order to assign spots in a Smart Parking

Lucas Fernando Souza de Castro[a], Gleifer Vaz Alves[a], and André Pinz Borges[a]

[a]Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná - Campus Ponta Grossa, Brazil, Av. Monteiro Lobato Km-4
lcastro@alunos.utfpr.edu.br, gleifer@utfpr.edu.br, apborges@utfpr.edu.br

KEYWORD

Smart Parking; Multiagent system; JaCaMo; Trust models

ABSTRACT

The process of searching for a parking spot could be a problem. There are computing solutions being developed to optimize this problem. One of these solutions is using multiagent systems (MAS). In this paper a MAS is developed in order to allocate spots in a smart parking using the framework JaCaMo. This MAS comprises of two types of agents: manager and drivers. The manager is responsible to administrate the parking spots which will be assigned for drivers according to a corresponding degree of trust. The trust degree is a value which shows the commitment of the driver before the manager. In order to verify the effectiveness of the MAS, several simulations were conducted in empirical scenarios. Experiments shows that the trust degree impacts in the parking spot allocation process.

1. Introduction

There is a large demand for technologies in order to facilitate the human life due to a necessity of man in do tasks using fewer resources and time. However, most of the time these technologies are not able to comply with demand, thus it is necessary to optimize them since they are working in a finite resource world. Cities, place of business, and houses are not only a social place but also a social system between people and electronic devices (Caragliu *et al.*, 2011).

Intelligent cities (or smart cities) aims to provide objects which become easier the life of population through technology use to minimize costs and use of resources. There a re still sub-categories in these cities that could be optimized as well, such as the economy, mobility, environment, people, living and governance (Caragliu *et al.*, 2011). In a smart parking, there is the so-called smart parking which uses a place which uses technology to automatize and improve the daily tasks at the parking (e.g., allocate a spot, parking payment, etc.) (Revathi and Dhulipala, 2012). Cities like San Fransisco in the United States (SFPark, 2015) and others in the North America (Parkingedge, 2013) have been developing computational systems to automate the process of allocating and pricing a spot according to its usage. Thus, it is possible that drivers check both: parking availability and price even before they get off home.

There are computational approaches trying to bring solutions for smart parking problems by means of multiagent systems in different languages and platforms such as the Jade language (Di Napoli *et al.*, 2014). The use of multiagent systems (MAS) is highlighted in smart parking context due to the dynamic scenario of a parking

lot. In these places, there are a huge number of variables which must be controlled at the same time (e.g. drivers, spots, payments, gates, and sensors). In addition to the spot allocation process, stands out the negociation of spots due to their limited quantity. Yet, in Naples (Italy**)** there is an example of MAS applied to the smart parking scenario. (Di Napoli *et al.*, 2014) presents a MAS to negotiate and pricing parking spots. Solutions using multiagent systems have been highlighted due to social factor, where the agents could share or even negotiate to get a resource, like a better parking spot (Di Nocera *et al.*, 2014).

Multiagent systems are composed of autonomous agents with a social level inserted in a dynamic environment. The agents have goals to accomplish and they are inserted in dynamic environment (Wooldridge, 2009). When implementing a MAS system one may has different sort of models, like the BDI (Belief, Desire, and Intention) to abstract this human reasoning. The framework JaCaMo provides a multiagent system development through the BDI model. The name JaCaMo is due to the three main components that compose the framework: (i) Jason language is based on AgentSpeak(L), and it is used to program the agents; (ii) Framework Cartago provides an artifact programming (artifacts work in the agents environment); (iii) Moise tool acts in the social organization of the agents, it bounds what the agents can and cannot do.

This paper aims to model and develop a multiagent system able to allocate spots in a smart parking through a manager agent to control these spots. The driver is the agent who uses the spots. Moreover, every single driver agent owns a trust degree value (only used in the smart parking). The trust degree value represents how committed an agent is with the smart parking, in a way that the driver will be rewarded as long as the driver uses the smart parking according to the manager agent. There are two kinds of agents in this MAS: drivers and a manager. The driver is the agent who uses the smart parking to get a spot and the manager is the agent who administers the smart parking. This work is part of a project called MAPS (MultiAgent Parking System), which aims to develop technologies and an architecture to a smart parking (de Castro *et al.*, 2016).

The main goal of this paper is to analyze how the trust degree value impacts in the allocation of spot process since there are different drivers agents with distinct trust degree values but desiring the same goal: get a spot in the smart parking.

The remainder of this paper is organized as follows. In Section 2, it is described how the MAS was designed and developed. In Section 3, it is showed the results and the simulations that were conducted in different scenarios. Finally, Section 4 presents the conclusions.

2. Multiagent system model

The main goal of our MAS is to assign parking spots to driver agents. The agent responsible to allocate the spots is the manager. However, the process of allocating a spot is composed of sub-processes running to guarantee that a spot will be allocated correctly. The figure 1 shows a use case diagram demonstrating the actions that both agents can play in the multiagent system.

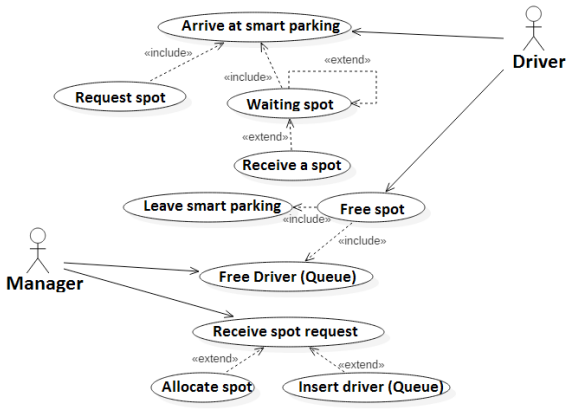

Figure 1: Use case diagram - MAS

2.1 Agent requirements

The first step in developing the MAS was list the requirements provided by the agents to make a robust and flexible system to the MAPS.

Manager Agent

1. **Receives a spot request:** The manager can at any time receive requests from the drivers in order to obtain a parking spot. According with the amount of requests and the smart parking capacity, the manager should allocate the parking spots.
2. **Decides which agent will receive a spot:** If the smart parking is full or even almost full, the manager uses the trust degree value to decide which driver will receive the requested spot. Hence, a driver with a greater degree trust value than another driver will firtsly receive a spot. There is a formula able to calculate the trust degree value, it is showed bellow:

$$newDegreeTrust(X) = oldDegreeTrust(X) + \alpha$$
$$\alpha = 1 \quad \textit{when the driver uses the parking assigned by the manager}$$
$$\alpha = 0 \quad \textit{otherwise}$$
$$newDegreeTrust: \textit{ new trust degree value}$$
$$oldDegreeTrust: \textit{ previous trust degree value}$$
$$X: \textit{ variable which represents the trust degree value}$$

It could generate a problem to a driver with a low trust degree value, so to solve it the manager checks first the waiting time of the drivers (60 seconds) and then check the trust degree value.

1. **The agent is aware and controls the smart parking:** The manager controls the smart parking because is through him that a driver can request, get, use and leave a spot. Besides, the manager has a complete knowledge of all parking spots (their status, location, and which driver is using);
2. **Receives a notice about a driver who is leaving:** When a driver agent leaves a spot, the agent has to alert the manager that the spot is free to use. After that, the manager decides which agent (based on the trust degree value) will receive that spot (in case if there is a waiting queue);
3. **Controls the waiting queue:** If the smart parking is full, the new driver agents that request a spot will be inserted into a priority queue. It is sorted by the trust degree value (descending) and according to with the waiting time (There is a priority to drivers who wait more than 60 seconds).

Driver Agent

1. **Requests a spot to the manager agent:** When a driver agent arrives at the smart parking, he will send a request to get a spot.
2. **Receives a spot and park in it:** After receiving a spot, the driver agent must drive to the spot and park in it;
3. **Waits for a spot:** If the smart parking is full while the agent requests a spot, the agent should wait in the waiting queue until the manager notifies to get the spot.
4. **Owns beliefs and characteristics as a real driver:** In order to create an agent driver representing a real driver, the drivers (agent) running in the MAS have the following beliefs: (a) time to get the smart parking; (b) approx. time to stay at smart parking; (c) spot in use; (d) trust degree value;
5. **Owns a trust degree value before the manager agent:** As described before, a driver agent is able to have a trust degree value before the manager. This value comprises how the driver uses the smart parking.

The figure 2 shows a basic structure of the smart parking, its features and how the trust degree value impacts to get a spot. The figure 2 illustrates two drivers trying to get a spot. The driver agent in green has a greater trust degree (900) than the red one (400), so the driver agent in green has the priority to get a spot.

2.2. MAS Development through JaCaMo

JaCaMo is a framework based on BDI model and comprises the agent, artifact, and the normative programming The development stage is divided into three steps. The first is the programming of agents and their interactions (Jason); the second one is the development of the artifacts (Cartago) and their interactions with the agents and the environment[1]. Finally, the simulation is the last step and it is shown next at Section 3. The figure 3 shows a general diagram to demonstrate the main architecture of the MAS designed with the Prometheus tool (Prometheus, 2015).

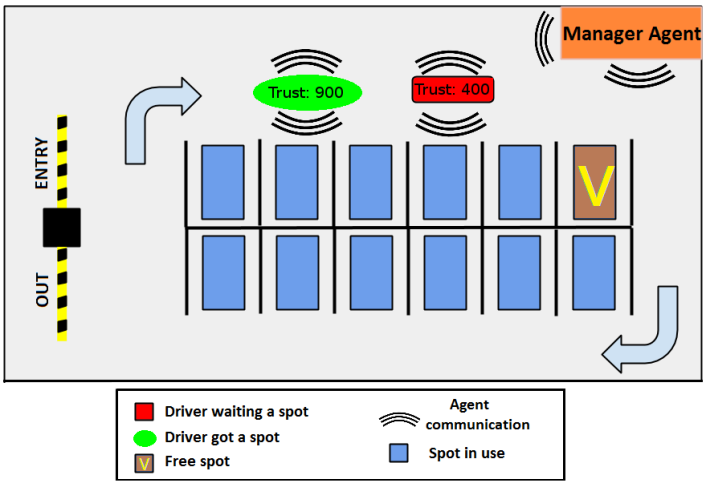

Figure 2: Basic structure adopted - Smart parking - Modified from (Gonçalves and Alves, 2015)

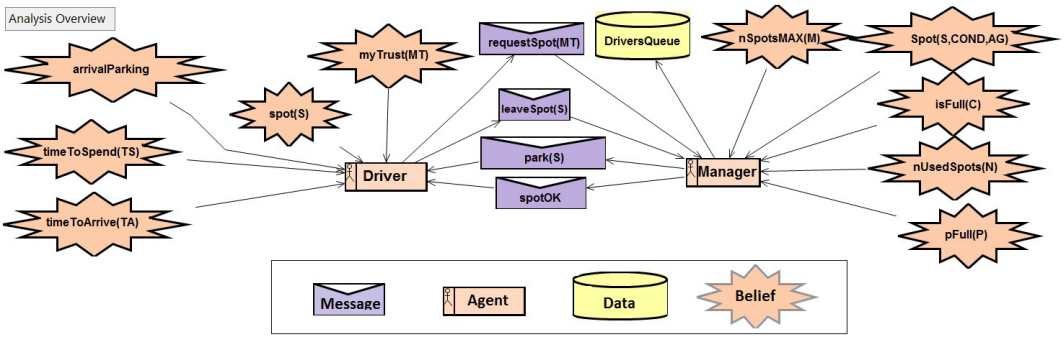

Figure 3: General Diagram - MAS Architecture

Step 1: Agent programming

The agent programming in the JaCaMo framework is done by the Jason, which is a language based on AgentSpeak(L) and used to programm the agents. The BDI views a belief is an information about the agent and its environment. The desires are the goals that the agent has or wish to accomplish. Intentions are how the agent achieves these goals, in Jason language, the intentions are denoted as plans (Bordini *et al.*, 2007).

The JaCaMo has a language called JCM able to set some information of the agents (initial beliefs and its goals), environment artifacts declaration and others parameters related to the MAS. In the code 1 is showed a

1 The present work does not use the Moise tool, however, an extension has been developed to use the Moise in the MAS.

brief JCM code describing two driver agents (*d1* and *d2*). Besides the initial beliefs, there are beliefs that are acquired acquired in running time. One example about an acquired belief is the spot that a driver receives *after* a request.

```
mas mAPS{
agent manager
agent d1: driver.asl {
beliefs: myTrust(100)
        timeToSpend(10000)
        timeToArrive(17514)}

agent d2: driver.asl {
beliefs: myTrust(450)
        timeToSpend(3000)
        timeToArrive(25307)}
```

Code 1: JCM File with two agent drivers

The code 1 shows only the initial beliefs of the driver agents. However, the figure 4 shows the acquired beliefs and they are described bellow:

- **myTrust(*mt*):** Trust degree value as initial belief. The variable *mt* indicates the value of the trust degree between $0 - 999$;
- **timeToSpend(*ts*):** Initial belief shows how much time the driver will spend inside of the smart parking. The *ts* is the corresponding value to show this time (in milliseconds);
- **timeToArrive(*ta*):** Initial belief showing how much time it takes him to get at the smart parking. The *ta* is the variable that indicates the value of the time to arrive at the smart parking.
- **spot(*s*):** Acquired belief when the manager agent notifies the driver about the requested spot. The *s* demonstrates the *number* of the spot;
- **arrivalParking:** Acquired belief after the driver agent gets at the smart parking.

In addition the beliefs, the agents have goals and plans. The figure 4 shows a general overview through a Prometheus diagram. This diagram illustrates the acquired beliefs, goals, plans, and the messages that the agent exchanges during the MAS execution.

The JCM file does not have the initial beliefs of the manager, they are inserted in a specific file called MANAGER.ASL (see code 2). The figure 5 shows a general overview of the manager agent and its beliefs, goals, plans and exchanged messages with the driver agents.

In the code 2 is showed part of the MANAGER.ASL with the initial beliefs of the manager.

```
nSpotsMAX(4).
nUsedSpots(0).
isFull(false).
pUsage(0).
spot(0,0, "EMPTY").
spot(1,0, "EMPTY").
spot(2,0, "EMPTY").
spot(3,0, "EMPTY").
```

Code 2: Beliefs and goals - Manager Agent

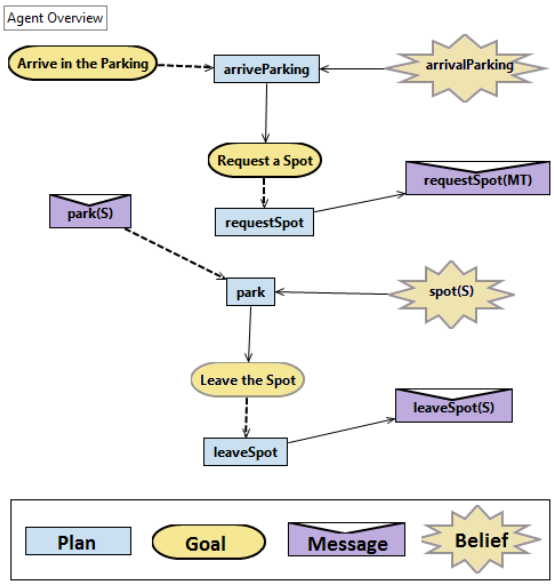

Figure 4: General overview - Driver Agent

- **nSpotsUsed(*n*)**: How many spots are used at the moment. The *n* comprises an integer value showing the used spots;
- **nSpotsMAX(*m*)**: The maximum spot capacity of the smart parking can manage. The *m* shows an integer value with the capacity;
- **isFull(*c*)**: Boolean condition to show if the smart parking is full or not. *c* is a boolean variable; The true value means that the smart parking is full, otherwise false means the smart parking is available;
- **pUsage(*p*)**: Percentage of use. The *p* is a value between 0 − 100 which represents the percentage;
- **spot(*s,cond,ag*)**: Belief which represents a spot before the manager. The *s* is the parking spot id; *cond* is condition (0 represents *free*, 1 represents *occupied*, and 2 represents *reserved*); *ag* shows which agent is using the spot;

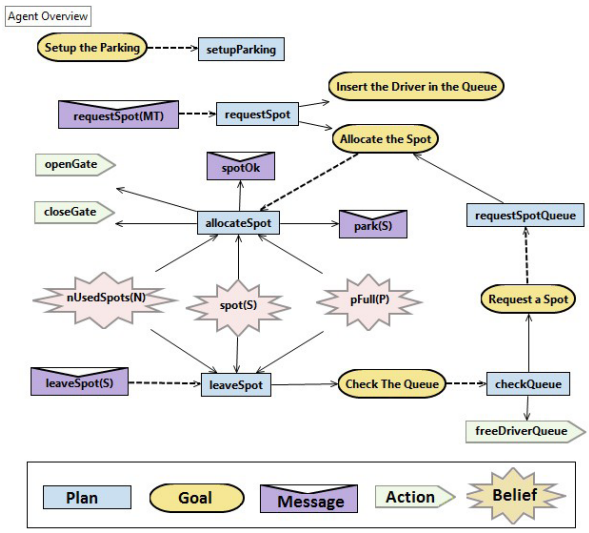

Figure 5: General overview - Manager Agent

Step 2: Artifacts programming

The artifacts in Cartago are able to provide functionalities to the agents running in the environment. In the smart parking, the environment is the parking. The developed MAS uses two artifacts: *Control* and *Gate*.

The waiting queue is managed by the artifact *Control* and it is sorted according to the waiting time (60 seconds) and the trust degree value (0 − 999). However, the waiting queue could sort only using the trust degree value. The SMA is designed to small and medium smart parking size (e.g. small = 1250 drivers / medium = 251500 drivers). Nevertheless, the simulations were conducted using two kinds of waiting queue: 1) priority queue based on waiting time and trust degree value; 2) priority queue based only on trust degree value.

3. Results

In order to simulate and test the MAS and how the trust degree value impacts in the allocation spot process, it is described bellow three scenarios with different configurations. Our results intend to show how the manager actually handles the waiting queue and the corresponding drivers requests. That is why we have chosen test scenarios with few parking spots (from 10 *to* 25). The three scenarios are composed of 500 driver agents with different beliefs. The corresponding JCM code has been generated by a script in order to obtain the code for 500 agents which has set random values to the beliefs. However, the values have to respect the range showed bellow:

Trust degree value: 0 -
999 Time to Spend: 0-
60000() Time to Arrive:
0 - 50000()

In Table 1, we show the three scenarios with the corresponding number of parking spots, method applied for the priority queue and average waiting time to receive a spot.

Table 1: Scenarios - Configurations

Scenario	Spots	Priority queue method	Avg. Waiting time (s)
1	10	Trust Degree	601
2	10	Waiting time | Trust Degree	650
3	25	Trust Degree	159

Scenario 1

In this scenario, there are 500 driver agents running for 10 spots. The priority queue is based *just* on the trust degree value. Then, a driver with a greater trust degree value will receive a spot first than a driver with a low trust degree value. The figure 6 shows a chart which demonstrates the impact of the trust degree value in the allocation spot process. Notice that the charts (figures 6-8) show two Y axis, the left axis is in charge of showing the waiting time and the right shows the trust degree value. Also, each chart has two interleaved charts: the red one represents the waiting time, and the blue shows the trust degree value.

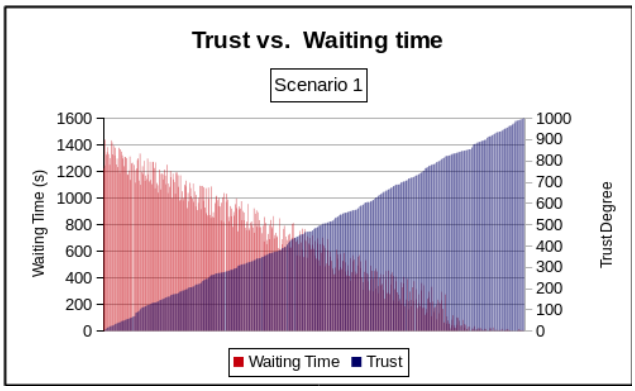

Figure 6: Scenario 1 - Trust degree value vs waiting time in queue

This scenario shows how the trust degree impacts in the allocating spot process. If a driver with a great trust degree value request a spot, almost imediatly (if available) he receives the spot. However, a driver with a low trust degree value has to wait a long time (> 800 seconds). Finally, a driver with a regular trust degree value (+/- 500) gets a spot with an average time (+/- 400 seconds).

Scenario 2

The scenario 2 is based on the scenario 1, however, not only the trust degree value is used, but also the waiting time, which demands that a given driver should not wait longer than 60 seconds in the queue. The figure 7 shows a chart illustrating how the priority queue and the trust degree value impacts.

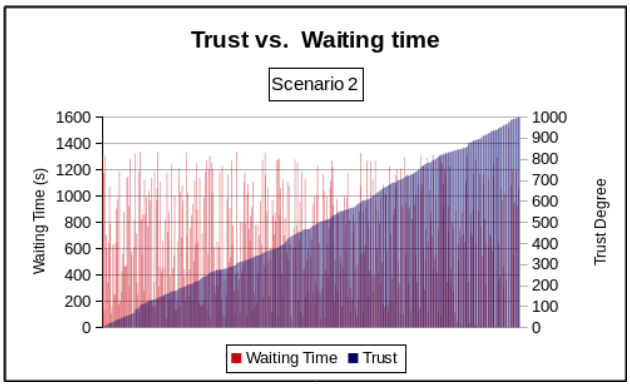

Figure 7: Scenario 2 - Trust degree value vs waiting time in queue (Time limited queue)

In this scenario, it is possible to see how the priority to the waiting time impacts. The waiting queue was used in order to allow a given driver (perhaps because he is a new driver in the MAS with a low trust degree value) not to wait too long in the queue. In this scenario the time was 60 seconds, however, it is variable and could be ajusted easily.

Scenario 3

The last scenario keeps the 500 agents but with more spots: 25, and only with trust degree value without a time limited queue. The figure 8 shows a chart illustrating how the trust degree value impacts.

As showed in the figure 8, the trust degree value impacts in the allocation process as the scenario 1. However, the waiting time average (159) was lower than the scenario 1 (601) (see table 1). It shows that a bigger

ammount of spots provide a low waiting time. Also in this scenario, a driver agent with a regular trust degree value (+/- 600) gets a spot with a low time (+/- 20 seconds).

The main goal of these scenarios is to check how the trust degree value impacts and its importance. Also, the scenarios showed that when there is a priority queue based also in the waiting time, the trust degree value will not make a significant impact. In order to summarize all the information from the three scenarios, the table 2 shows the difference among three drivers: agent1, agent2, and agent3, respectivatelly with a low, medium, and high trust degree value.

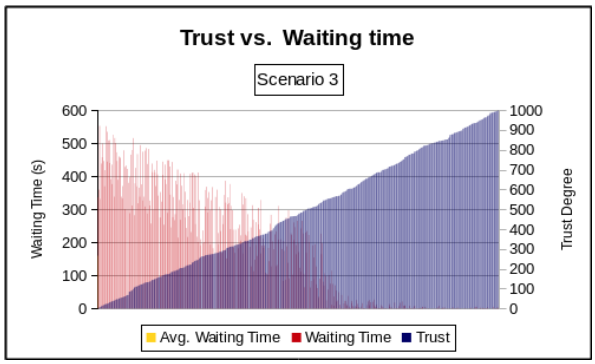

Figure 8: Scenario 3 - Trust degree value vs waiting time in queue

Table 2: Results - Scenarios 1, 2, and 3

Agent	Trust Degree Value	Scenario 1 Waiting time(s)	Scenario 2 Waiting time(s)	Scenario 3 Waiting time(s)	Average time(s)
agent1	132	1069	1182	287	846
agent2	562	546	229	36	270
agent3	739	136	1007	17	386

In table 2 we have three different agents with distinct trust degree values in all three scenarios and the average waiting time.

According with the table 2, we can check how the trust degree impacts in the average time, most of times a greater trust degree (e.g., agent3) value guarantees a spot faster (< 30) seconds. Also, it is possible to see the quantity of spots impact in the waiting time. Finally, according to the table 2, it is possible to check that the agent1 and the agent3 own differents values of trust degree values (agent1 - 132 and agent3 - 739). However, in the scenario 2 they have almost the same waiting time (agent1 - 1182 seconds and agent3 - 1007 seconds). It shows how the limited time queue impacts in the process of allocating parking spots.

3. Conclusion

The main goal of this paper is to design and develop a multiagent system to allocate parking spots based on trust degree values. Besides the trust degree value, there is the waiting time priority that also impacts in the allocation of spot process. The simulations were conducted to show if the trust degree value really impacts in the process of allocating a spot and they showed that a driver with a great trust degree value (scenarios 1 and 3) gets a parking spot first. Nevertheless, in the scenario 2, there was a priority based not only on the trust degree value but also in the waiting time and it showed that the waiting time impacts as well. Finally, a bigger quantity of spots decrease the waiting time by close of 200% (scenario 1 (10 spots) - 601 seconds *vs* scenario 3 (25 spots) - 159 seconds).

The main contributions of this paper are analyse how the trust degree value impacts in the process of allocating a parking spot, and also verify how the trust degree value impacts when there is a limited time queue. Without a limited time queue the drivers with a high trust degree values (750+) gets a spot lower than 100 seconds, and also drivers with a low trust degree value gets a spot greater than 300 seconds. Lastly, with a limited time queue the trust degree value does not have much impact due to the priority to the waiting time.

Our MAS developed by using JaCaMo framework has provided a robust and flexible system to forthcoming extensions of the MAPS project. Moreover, it is possible highlight some future works such as: (i) database artifact to store the trust degree values; (ii) artifact that provides a graphical interface with the general status of the MAS; (iii) developing social rules of the MAS through Moise from the JaCaMo framework that is responsible to implement the organizational layer of a MAS; (iv) expand the MAS in order to implement the Jason agents in an embedded plataform and with this our MAS could be deployed in a more real scenario with sensors and embedded devices, for example.

4. References

Bordini, R. H., Hübner, J. F., and Wooldridge, M., 2007. *Programming Multi-Agent Systems in AgentSpeak Using Jason (Wiley Series in Agent Technology)*. John Wiley & Sons. ISBN 0470029005.

Caragliu, A., Bo, C. D., and Nijkamp, P., 2011. Smart Cities in Europe. *Journal of Urban Technology*, 18(2): 65-82. doi:10.1080/10630732.2011.601117.

de Castro, L. F. S., Alves, G. V., and Borges, A. P., 2016. A proposal of an architecture for a Smart Parking based on intelligent agents and embedded systems. In *Anais do I Workshop de Pesquisas em Computação dos Campos Gerais - WPCCG*. ISSN 2526-1371.

Di Napoli, C., Di Nocera, D., and Rossi, S., 2014. Using Negotiation for Parking Selection in Smart Cities. In Demazeau, Y., Zambonelli, F., Corchado, J., and Bajo, J., editors, *Advances in Practical Applications of Heterogeneous Multi-Agent Systems. The PAAMS Collection*, volume 8473 of *Lecture Notes in Computer Science*, pages 331-334. Springer International Publishing. ISBN 978-3-319-07550-1. doi:10.1007/978-3-319-07551-8_31.

Di Nocera, D., Di Napoli, C., and Rossi, S., 2014. A Social-Aware Smart Parking Application. In *Proceedings of the 15th Workshop «From Objects to Agents»*, volume 1269.

Gonçalves, W. R. C. and Alves, G. V., 2015. Smart Parking: mecanismo de leilão de vagas de estacionamento usando reputação entre agentes. In *Anais do IX Workshop-Escola de Sistemas de Agentes, seus Ambientes e Aplicações - IX WESAAC*.

Parkingedge, 2013. Best Parking. Disponível em <http://www.bestparking.com>. Prometheus, 2015. The Prometheus Methodoloy.

Revathi, G. and Dhulipala, V., 2012. Smart parking systems and sensors: A survey. In *Computing, Communication and Applications (ICCCA), 2012 International Conference on*, pages 1-5. doi:10.1109/ICCCA.2012.6179195. SFPark, 2015. SFPark. Disponível em <http://www.sfpark.org>.

Wooldridge, M., 2009. *An Introduction to MultiAgent Systems*. Wiley Publishing, 2nd edition. ISBN 0470519460, 9780470519462.