ADCAIJ: Advances in Distributed Computing and Artificial Intelligence Journal
Regular Issue, Vol. 10 N. 3 (2021), 209-225
eISSN: 2255-2863
DOI: https://doi.org/10.14201/ADCAIJ2021103209225

An Agent-Based Simulation to Explore Communication in a System to Control Urban Traffic with Smart Traffic Lights

Robson Teixeira[a], Roberta Sousa[a], Enyo Gonçalves[a], and Marcos de Oliveira[a]

[a] Universidade Federal do Ceará (UFC), Av. José de Freitas Queiroz, 5003, Cedro, Quixadá, Ceará, Brazil, 63902-580

robertasousa714@gmail.com, robson.teixeira2000@alu.ufc.br, enyo@ufc.br, marcos.oliveira@ufc.br,

KEYWORDS

Agent-Based Simulation; Multiagent Coordination; Urban Traffic Management;

ABSTRACT

Population growth increases the number of cars and makes the transport infrastructure increasingly saturated. The control of traffic lights by intelligent software is a promising way to solve the problem caused by this situation. This article addresses this problem that occurs in urban traffic. We propose an agent-based simulation of an urban traffic control system. The solution is offered as intelligent traffic lights as agents to alleviate traffic congestion at a given location. Each agent controls a crossing and maintains communication with agents from other corners. Thus, they can have greater control of a larger area and identify patterns that can help them to solve congestion problems. The results of our simulated experiments point to the improvement of the urban traffic when using the proposed Multi-agent System, in comparison with an approach that uses crossing agents without communication and others that implements static traffic lights.

1. Introduction

The increase in population and cars impacts the saturation of the transport infrastructure, and as a consequence, congestion spreads through traffic around the world (Junior et al., 2015). This problem has been the subject of research to improvement for a long time. Several solutions are applied and found, and even so, the research community continues to address the issue as a plan for significant works. Economic, social, and environmental disorders are caused every day by chaos in traffic (Bull et al., 2003); problems such as air and noise pollution and the drivers' stress with traffic jams are some of them (Bull et al., 2003). Because of this, studies that seek to optimize traffic, like this one, make it possible for society to reduce the negative impacts caused.

Works such as (Junior et al., 2015) propose Multi-agent Systems as part of the solution to the problem of traffic congestion. This has been happening because Multi-agent Systems (MAS) are composed of several agents that interact with each other, performing actions to achieve their individual and shared objectives, such as managing traffic lights. The knowledge and operational capacity of agents are limited, and, therefore, they often need to interact to achieve their goals (Lupasc et al., 2005). Simulation-based MAS are used in a wide range of applications, such as games (Marín-Lora et al., 2019), energy optimization (González-Briones et al., 2018) and contamination with pathological agents, which can be found in a simulation in the NetLogo[1] software as an example.

According to (Fernández-Isabel et al., 2020), a multi-agent simulation can also collaborate in the context of the transport infrastructure problem, as it alleviates the difficulties in the development and configuration of a given system, in the controlled environments conducive to testing components. Note the use of MAS in research related to complex urban traffic problems and in (Wu et al., 2020) the authors' report, the traffic light control problem seems to be the easiest to solve. In addition, it will indicate two main problems to be solved in traffic light control currently (1) communication between traffic lights and (2) lack of consideration of network limitations. With this, this work aims at the problem of control of traffic lights seeking to optimize the problem (1) through communication between agents.

Some MAS are highlighted in the analysis (Fuelber and Frozza, 2017), indicating an interesting path for research in the area. MAS can represent participants' behavior in a traffic system, enabling the understanding and specification of these participants, reproducing their behavior and decision making. In the area of coordination and adaptive traffic control, MAS is able to promote the effective use of the installed infrastructure.

This paper presents a MAS based on simulation using SUMO[2], TraSMAPI[3], Jade[4], and OpenStreetMap Project[5] to coordinate urban traffic. The structure of the six sections of this paper is presented next: The sectiorefsection-2 offers the background of the methods and tools used to develop our simulations and related work. In section 3, we explain in detail our simulations and results. Section 4 presents our conclusions and future work.

2. Background

2.1 Urban Traffic

With the growth of urban areas worldwide, there is a direct implication of increased urban traffic demand. Understand the necessities and problems of urban traffic is necessary to create plans for management and improvements for daily traffic (Chow et al., 2014).

[1] https://ccl.northwestern.edu/netlogo/
[2] https://www.eclipse.org/sumo/
[3] https://github.com/STEMS-group/trasmapi/wiki
[4] https://jade.tilab.com/
[5] https://www.openstreetmap.org

(Chow et al., 2014) Explain that a significant number of factors are connected to the traffic jam problem besides the excess of traffic volume and that there are two types of jams: the recurring, referring to traffic jams that occur regularly as traffic in excess, traffic light control, etc.; and the not frequent, that is caused by incidents of traffic, like accidents, police checkpoints, construction works, etc.

(Liu et al., 2017) indicate that efficient urban traffic management solutions must explore strategies to bring emergent technologies to mitigate urban traffic jams. There are at least three primary questions that need to be addressed:

1. Real-time perception of traffic conditions, which includes identifying real-time traffic conditions through detectors that capture vehicle speed, direction, location, etc.
2. Low latency communication and massive data storage, concerning the data produced by the sensors in a real environment, are produced in an enormous quantity.
3. Traffic forecast and capacity of an answer in real-time, meaning that the sensors created data helping the traffic management and traffic control systems must have a capacity of response in real-time to answer to events and take decisions.

This work approaches two main questions exposed in (Liu et al., 2017): the first and third ones. The first because our solution includes the perception of traffic conditions through our detectors in the simulated environment. The third, because our solution is a MAS in which agents can answer real-time events in the simulated environment, being able to make decisions to a better solution. The second question does not fit in this work because it is a simulation with low data quantity compared with a real environment.

In addition, our main proposal is to add communication between agents in the system proposed (Junior et al., 2015). This is justified by sharing important information that agents will pass among themselves, in this case, the number of vehicles on a given avenue, which can avoid congestion caused by the careless release of vehicles by a given agent without information from neighboring avenues. We will explain the proposed algorithm in more detail. Still, we believe that our proposal prevents vehicles on routes with large quantities of vehicles from being harmed in situations where they have a similar average travel time.

2.2 MAS & simulation

A tool is needed to simulate and manipulate the environment to create an intelligent traffic system using multiagent systems. In this section, the tools used in our simulations are described. We have used the Simulation of Urban Mobility (SUMO) for the simulation, OpenStreetMap for defining the environment, and TraSMAPI for manipulating SUMO elements.

2.2.1 SUMO -Simulation of Urban Mobility

SUMO is a free and open-source tool that allows you to perform traffic simulations. It is possible to model traffic systems, define roads and their directions, traffic lights, and different vehicle types, among other structures. SUMO is present in several application areas, such as traffic light evaluation with modern algorithms, choice of routes, and emission calculations. Included in SUMO, various support tools automate the main tasks of creating, executing, and evaluating traffic simulations, such as network imports, route calculations, visualization, and emissions calculations (Lopez et al., 2018).

The entire simulation environment is defined in XML files. You can choose to create a fictional environment (recommended for testing) or use a real map that gives results closer to reality for the simulation. Once the environment is defined, it is necessary to model it with vehicles, routes that each one will use, and all the essential variables in that environment, such as changing the simulation elements' properties, for example, phases of a traffic light (Behrisch et al., 2011). The Figure 1shows how a city like Quixadá can be represented in SUMO.

2.2.2 OpenStreetMap Project

As explained earlier, one way to define a simulation environment in SUMO when the desired results need to be closer to reality is using maps. For this, some projects provide maps, or part of them, to be downloaded, modified, and used. OpenStreetMap is a free and open-source project that offers a world map and allows you to download specific areas of that map (Bennett, 2010), as the area shown in Figure 2. This tool is ideal for obtaining information that can be used in urban simulations that require data closer to reality. The Java OpenStreetMap Editor tool's availability allows the editing of elements from a downloaded map (Haklay and Weber, 2008), enabling the use of structures effectively used in a simulation.

For the downloaded map to be used in a SUMO simulation, it needs to be converted into netConvert, a tool provided by SUMO that allows you to convert the map into an XML file that the simulator can use.

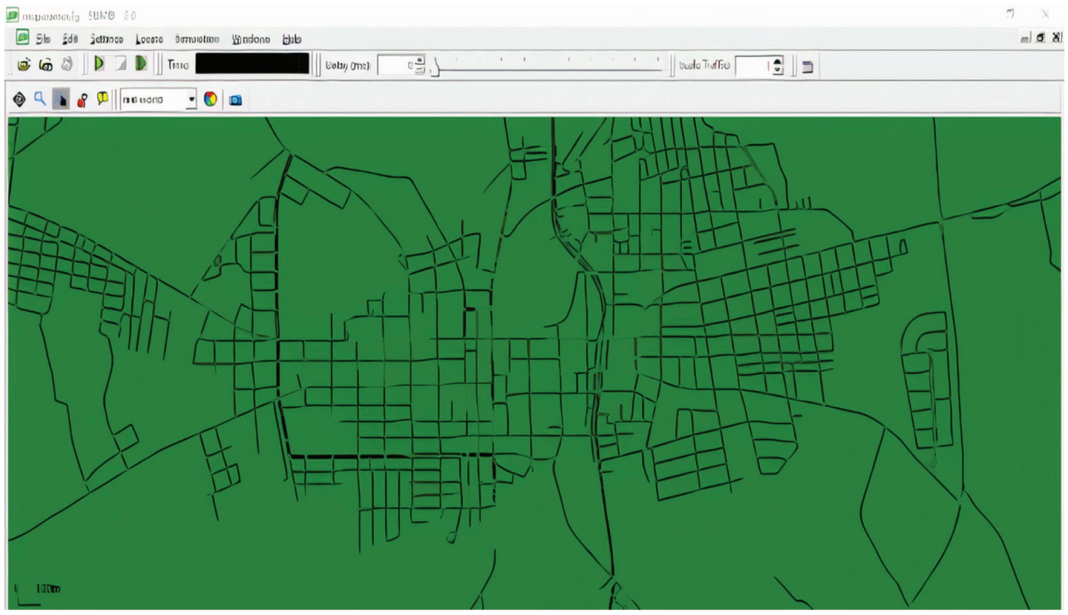

Figure 1: Traffic structure of the city of Quixadá simulated using the SUMO tool.

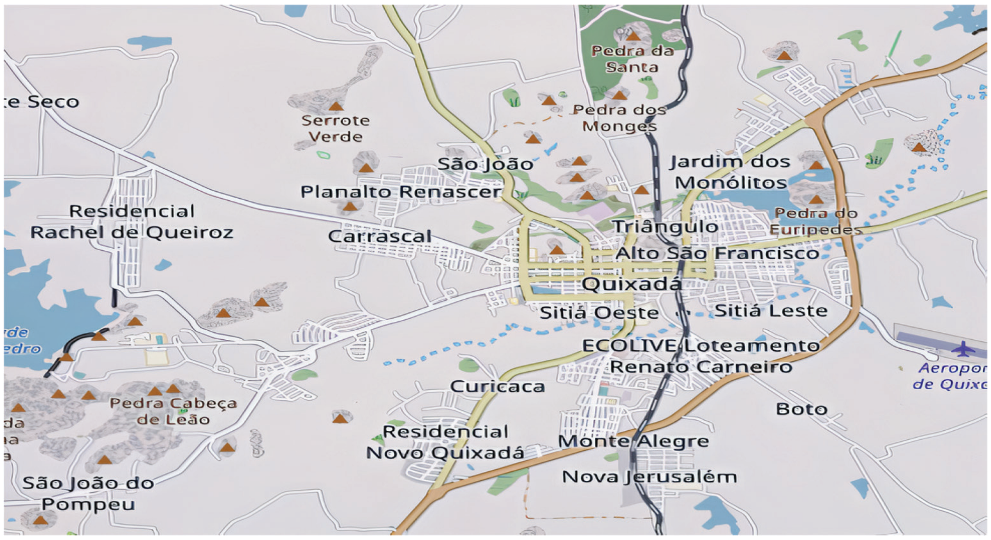

Figure 2: Open map area of Java OpenStreetMap Editor

2.2.3 TraSMAPI

It is necessary to use a tool to program agents' behavior and communicate with the simulator to assign this behavior to an environment element and use multiple agents in SUMO. One tool that makes this possible is TraSMAPI[6] (Traffic Simulation Manager Application Programming Interface). TraSMAPI is an API developed for the Java language that allows the implementation of agents and controls SUMO remotely. So, we can use it to implement the intelligent behavior of traffic lights, obtaining all the traffic lights in the environment, and assigning the behavior of these agents to these traffic lights.

The TraSMAPI architecture, detailed in (Timóteo et al., 2010), is organized in three modules:

- Communication module: Provide interaction based on queries to gain knowledge on the environment state and to change the state of the environment.
- Statistics module: A passive module that does not interfere with the simulation and records all the information regarding the simulation environment.
- The Multi-Agent System Framework: Allow the creation of agents following a common interface. Also has a moderator that controls the flow of information between the agents and the simulation environment.

[6] https://github.com/STEMS-group/trasmapi/wiki

2.3 Related Work

2.3.1 Simulation of Adaptive Urban Traffic Control Using a Multiagent System and Based on Real Data

As previously said, the work of (Junior et al., 2015) was our main related work. The problem of saturation of urban traffic that has been growing with the increase in population is addressed. They present a solution using MAS. They compare the results obtained from simulations from a model that uses real data from static traffic lights with traffic light programming, traffic lights with a fixed schedule, without any form of optimization. The work uses a model called agentBased2, an algorithm developed by the authors to transform traffic lights into agents.

To provide greater data credibility, the authors chose to use real data from Porto Alegre in the Rio Grande do Sul - Brazil, obtained from an entity in that area, the Public Transport and Circulation Company (EPTC). Data from six intersections were provided, as well as their respective traffic light schedules. Five simulations were performed in each of the models, Static and AgentBased2, simulating one day (24 hours). Based on the information obtained with the EPTC, the data analysis was provided through the simple average of the five simulations in each model.

The solution proposed by the work was creating intelligent traffic lights, contributing to a reduction in travel time, a better flow of traffic, and a decrease in congestion. The simulation agents are AgentBased2; they are agents with traffic detectors to observe the environment. Each sensor is added automatically in the lanes that give access to the intelligent traffic light in question. Through the detectors, the agents can observe the traffic concerning congestion, make decisions about it, and adjust the traffic light intervals. The type of detector chosen by the authors was E3, a detector placed at the ends of the section to be observed and capable of acquiring various data at each time interval, which is placed at the beginning and end of the road to be monitored, with them it is possible to know the exact moment when cars enter and leave the road.

This is important for calculating the time that a given car takes to complete the path monitored by the detectors; this information assesses the performance of the road. Besides, each sensor has the direction classification, East / West (LO), or North / South (NS), so that traffic lights identify the directions controlled by the detectors. The algorithm used by MASrt traffic lights is detailed in Section 3.

From the traffic assessment, through the "AgentBased2" traffic light, this particular agent performs the adjustments it deems necessary at a specific intersection. After the traffic light cycle is completed, this evaluation takes place (t_{decide}); this cycle is performed by completing the sequence of the green, yellow, red, and yellow again phases of a traffic light. When executing the process, the data stored by the detectors are read. They are returned to the traffic light grouped by the directions (LO or NS) formatted in an index ($i_{efiency}$) representing the traffic light's efficiency in clearing the passage for each of the vehicles. In directions (LO or NS), this index will define the next minimum time required to cross the detector by the vehicles that have passed it; the shorter, the further the traffic is congested towards the sensor in question, causing the green light of that direction increase. After reading the detectors' data, the lowest indexes of each direction (LO and NS) are assumed as the general index of the direction in question ($g_{LO}$and g_{NS}).

Each direction has a list (l_{LO} and l_{NS}) of size (t_{mem}) that stores the last three indexes calculated for each traffic direction, as long as the lists are not filled with the three indexes. No change in the traffic light is allowed after the traffic lights are permitted to carry out evaluations regarding traffic. For that to happen, each list's values are averaged, generating the average index of each direction (m_{LO} and m_{NS}).

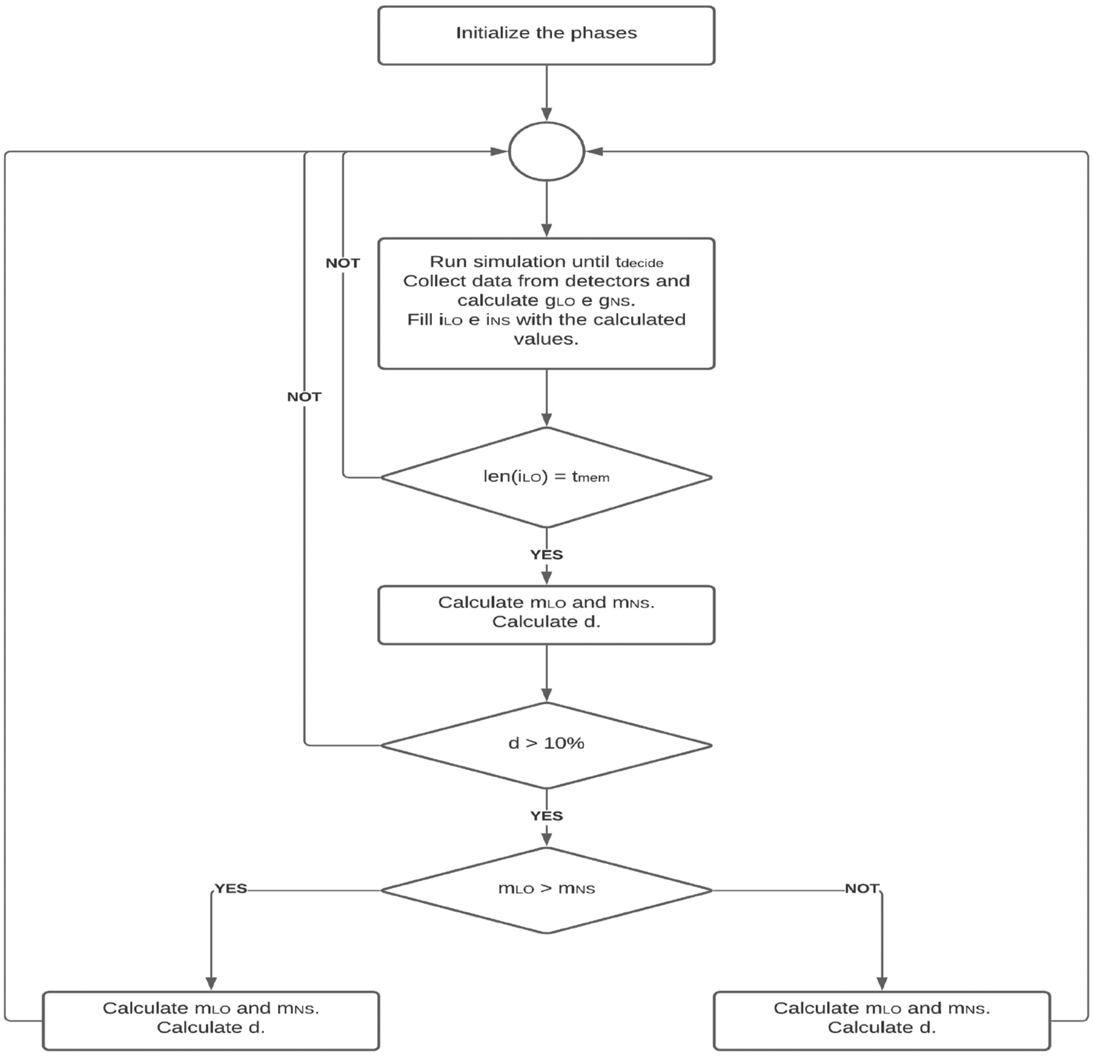

Figure 3: Algorithm for the control of intelligent traffic lights used in the work of Junior, Frozza and Molz (2015).

After calculating the average indices for each direction (m_{LO} and m_{NS}), the absolute difference between the two indices (d) is calculated. If this difference is less than or equal to 0.1(10%), no attempt is made to change the traffic light plan. The reason for that is if the difference is less than or equal to 0.1, the traffic distribution is balanced at that intersection and, therefore, there should be no changes; otherwise (>0.1), the traffic light plan must be adjusted. This adjustment consists of increasing the green light time in the direction with the worst traffic performance by 1second and increasing the red light in 1way in the opposite direction. The adjustment has restrictions that are detailed in the article in question. All simulations performed as an experiment took place using the same tools presented in this section previously. With the results obtained, the authors showed a greater efficiency of the model based on agents to the model used in a real environment (Static).

2.3.2 JADE, TraSMAPI, and SUMO: A Tool-Chain for Simulating Traffic Light Control

The work of (Azevedo et al., 2016) shows a MAS simulation for traffic light control. The authors aim to use a chain of tools to accomplish their goal to address the daily traffic jam problem. The chain of tools used for the system implementation is: JADE (Java Agent Development Framework) (Bellifemine et al., 2005), for the control of MAS, SUMO, shown in the section 2.2.1, for the traffic simulation, and the TraSMAPI, shown in section 2.2.3, for the communication between JADE and SUMO. Besides that, they have used Q-Learning (Watkins and Dayan, 1992) for the agents' learning method and implemented the communication protocol related to the traffic light methods to recover the value and change of state of it in the TraCI (Wegener et al., 2008).

The model that the agents organize themselves is decentralized, like ours. This model can hardly obtain synchronization, unlike a centralized model, but the spread to the whole system does not occur in case of failure. In the system, the agents communicate through a request for rewards and answers to rewards. In the first case, it is implemented using the FIPA performative QUERY REF with the content "reward", and in the last, it is used the FIPA performative INFORM REF with the reward in the content of the message, as shown in Figure 4 (O'Brien and Nicol, 1998).

Unlike our communication model, it (Azevedo et al., 2016) allows the agents to request rewards to their neighbor agents and so on in the simulation execution. In our communication model, each agent sends messages for all the other agents that belong to the simulation, which allows for more globally distributed information in the environment, and the agents use them to make decisions.

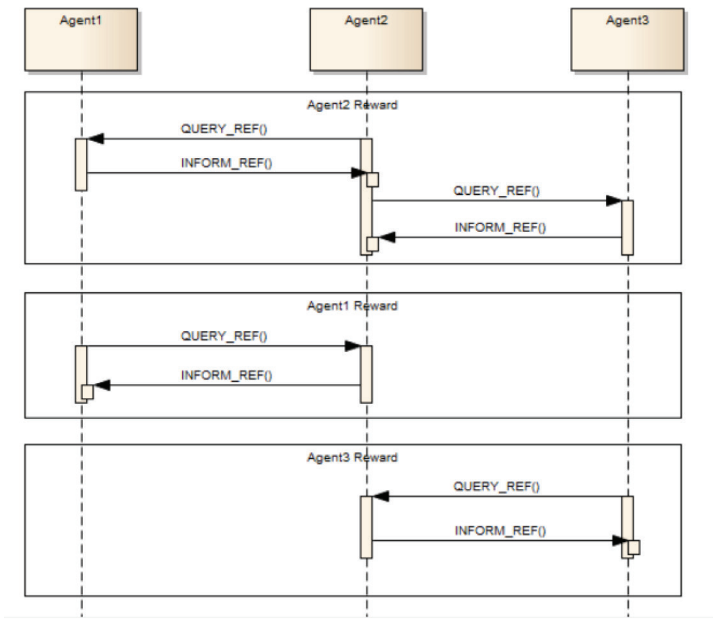

Figure 4: Example of interaction among agents from (Azevedo et al., 2016).

Based on this, our objective is to propose to add communication between the agents of the system proposed by Junior, Frozza and Molz (2015), where the agents would exchange messages with all the other agents present, these messages would share information necessary to influence the agents in their decisions. And we will compare the results of the simulations performed on each model. In the following section, we will present the algorithm with the proposed addition and the results of the simulations. We will also present the simulation results of an agentless model, in what we call the Static model; below, we will also talk a little more about it.

The table 1 present a resume about the concepts detailed in this section, where is possible see the main features of each one.

Table 1: Comparation between JADE, TrasMAPI and SUMO

Concept	Programming language used	Is open source	Objective
JADE	Java	yes	Software framework for the development of intelligent agents
TrasMAPI	Java	yes	Provide real-time interaction with Traffic Simulators, collect metrics and statistics, and offer an integrated framework to develop Multi-Agent Systems.
SUMO	Java	yes	Traffic simulator

3. Results and discussion

As previously stated, we carried out 3 types of simulations, Static (S1) -with static traffic light programming and without agents; Agents Without Communication (S2) -simulation based on the algorithm of Junior, Frozza and Molz (2015), and Agents With Communication (S3) -simulation proposed by us carried out with the algorithm of Figure 10, in this proposal the agents communicate with each other.

A central area of the city of Quixadá, CE -Brazil, was chosen on OpenStreetMap as the location for our simulations. Altogether 30 simulations were carried out, in which 10 were of type S1, 10 of type S2, and 10 of type S3. In each simulation, we use the same random value for the number of vehicles and routes in the scenario. The number of agents is equal to the number of traffic lights on the OpenStreetMap map for simulations S2 and S3. For the 10 simulations performed for each type, 10 different route configurations were used. These values are shown in Table 2. The S1 simulation has no agents; therefore, only the simulation data for the number of vehicles and the number of routes in the Table 2 coincide.

Each route is covered by a new car introduced in the scenario during the simulation, and once a vehicle finishes its course, it is removed from the scene. Each simulation below will present its values for the average travel time of the vehicles (Duration), the average time that the vehicles were stationary (Stopped), and the average movement time of the vehicles (Movement).

Table 2: Simulation data S1, S2 and S3.

Number of Vehicles	Number of Agents	Number of Routes
5000	17	5000

3.1 Simulation S1: Static

In the static simulations of the S1, we used the traffic lights already existing in OpenStreetMap in the scenario. Traffic light programming is used in real traffic to calculate the green, yellow, and red traffic lights. For this reason, we used no agent in this simulation, but only a standard traffic light schedule in which agents do not need to make traffic light time decisions, as these decisions are made. Each city has its own schedule according to its traffic flows, allowing for a whole week schedule based on peak data, days of the week, etc.

The plan is divided into phases, in which each phase will have a state and its defined duration. The state consists of a set of characters (G, g, y, Y, r, R), indicating which state will appear at each traffic light during a given phase, each character representing a state. Improvements to traffic lights consist of changing the stages, changing the duration, and adding new stages.

The Figure 5 presents the code related to the simulation without intelligent agents. Initially, the TraSMAPI API is instantiated (line 1). Then an object is created with the type of simulator to be used (SUMO) and its configuration parameters (lines 6 -10). Finally, this object is passed to the TraSMAPI API(line 12), and the simulation is started(lines 14-18).

This performed simulation is important to compare the method currently used with the method proposed by this work. The Figure 6 shows the averages of time for each agent‘s state.

```
1  TraSMAPI api = new TraSMAPI();
2
3  Sumo sumo = new Sumo("guisim");
4  List<String> params = new ArrayList<String>();
5
6  params.add("--device.emissions.probability=1.0");
7  params.add("-c=network/MAS.sumocfg");
8  params.add("--tripinfo-output=output.xml");
9  sumo.addParameters(params);
10 sumo.addConnections("127.0.0.1", 8813);
11
12 api.addSimulator(sumo);
13
14 api.launch();
15
16 api.connect();
17
18 api.start();
```

Figure 5: Code for simulation without agents

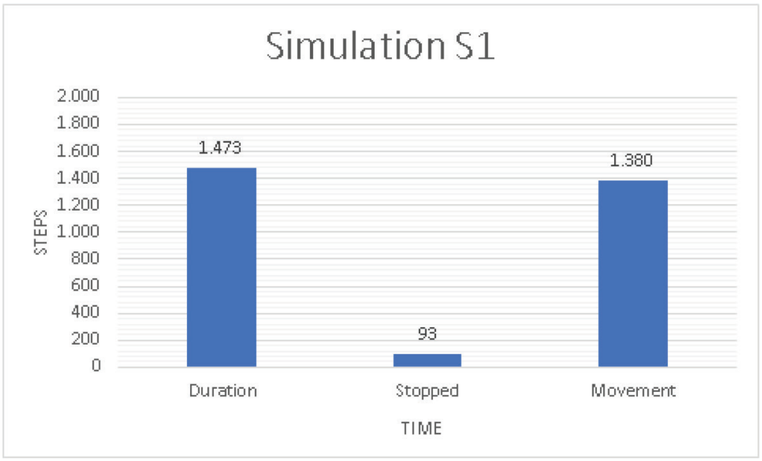

Figure 6: Time averages obtained in the S1 type simulation

3.2 Simulation S2: Agents without Communication

In the S2 simulations, the algorithm of Junior, Frozza and Molz (2015), presented previously in Figure 3, was used. The graph in the Figure 9 shows the results obtained with this simulation

As shown in the Figure 7, the code responsible for running the simulations is similar to the one in the Figure 5. With the addition of the AgentsManager (lines 18 and 19), a class will create an agent for each traffic light.

The agents generated by the Agents Manager have the structure shown in the Figure 8. In this figure, it is possible to see that the agent has a cyclical behavior(lines 16 - 22). This behavior is responsible for checking the sensors and changing the traffic light plan.

```
1  TraSMAPI api = new TraSMAPI();
2
3  Sumo sumo = new Sumo("guisim");
4  List<String> params = new ArrayList<String>();
5
6  params.add("--device.emissions.probability=1.0");
7  params.add("-c=network/MAS.sumocfg");
8  params.add("--tripinfo-output=output.xml");
9  sumo.addParameters(params);
10 sumo.addConnections("127.0.0.1", 8813);
11
12 api.addSimulator(sumo);
13
14 api.launch();
15
16 api.connect();
17
18 AgentsManager manager = new AgentsManager(sumo, mainContainer);
19 manager.startupAgents(mainContainer);
20
21 api.start();
```

Figure 7: Code for simulation with intelligent agents

```
public class TrafficLightAgent extends Agent {

  private String name;
  private double[] iLO, iNS;
      double gLO;
  double gNS, mLO = 0, mNS = 0;
  int position, quantity;
  Sumo sumo;

  public TrafficLightAgent(Sumo sumo, String name) throws Exception {...}

  protected void setup() {...}

  protected void takeDown() {...}

  private class changeTrafficLigthStateBehaviour extends CyclicBehaviour {
    public changeTrafficLigthStateBehaviour(Agent a) {...}

    public void action() {...}

    public SumoTrafficLightProgram createProgram(SumoTrafficLight tl ,int type) {...}
  }
}
```

Figure 8: Code for class of agents

Comparing with the results of the S1 model, it is already possible to see the decrease in the average travel time, as the authors Junior, Frozza and Molz (2015) had already presented in their work.

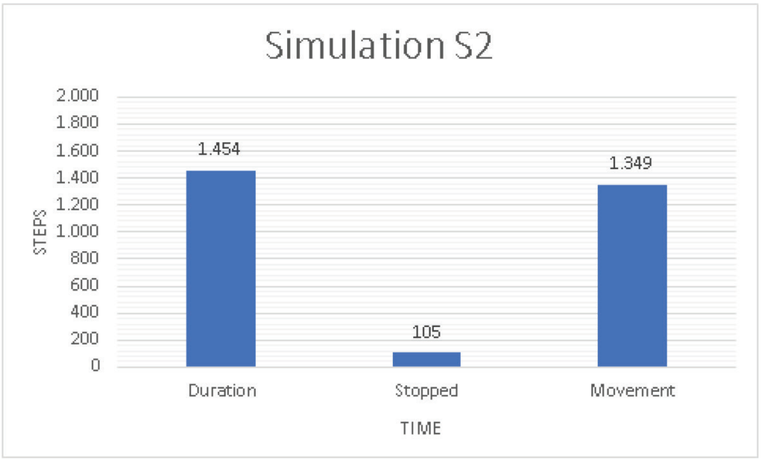

Figure 9: Time averages obtained in the S2 type simulation

3.3 Simulation S3: Agents with Communication

The algorithm used in our simulations applying agents with communication is similar to the algorithm used in (Junior et al., 2015), represented in Figure 3. However, we propose the additional step of checking the number of vehicles stopped on a stretch controlled by a traffic light.

This step consists of comparing the average of each direction, and if the difference between them is less than 0.1, that is $d < 10\%$, each agent will send the others the number of cars present on the road that it controls. Thus, the traffic light plan of the one with the largest number of vehicles is changed.

We can see this change in the Figure 11. The agent's structure is the same as shown in the Figure 8, changing the code of its behavior for sending messages.

If $d < 10\%$, agents will communicate to check the number of vehicles on each avenue with agents and make the change based on this value. Each intersection in our simulation is controlled by a traffic light with an E3 sensor. This sensor detects traffic from the beginning of the street to the traffic lights. It obtains traffic information on this part of the road. This sensor receives the number of cars arriving and leaving the street and the average speed of the cars. This data is saved in an XML file, which is updated every 90 steps of the simulation. The E3 sensor requires the specification of the sensor input, shown in Figure 12 by the green lines, and the sensor output, shown in Figure 12 by the red lines. Each entry and exit has information about the street and position based on the sensor's distance, and it is to one of the road ends.

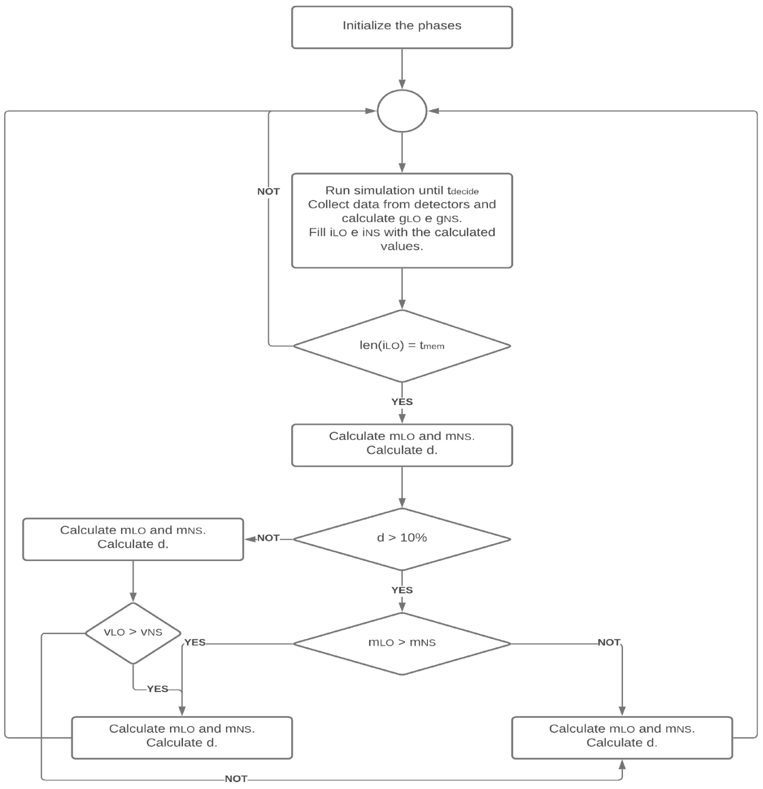

Figure 10: Our proposed algorithm to control traffic lights, based on (Junior et al., 2015)

```
if(Math.abs(mLO - mNS) > 0.1) {
  if(mLO > mNS)
    tl.setProgram(createProgram(tl, 2));
  else
    tl.setProgram(createProgram(tl, 1));
}else{
  ACLMessage msg = new ACLMessage(ACLMessage.INFORM);
```

Figure 11: Code for simulation with agents and messages

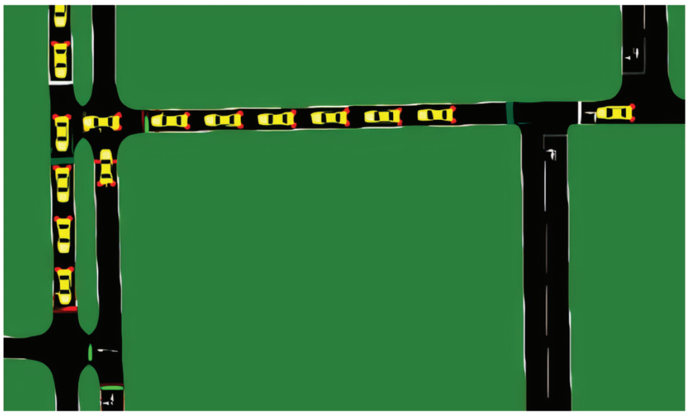

Figure 12: Part of the street supported by the sensor

This additional step aims to avoid that if one of the directions is in a lot of traffic, but in all directions the average speed of the vehicles is similar, the vehicles of the street with a lot of traffic will be harmed, having to wait a long time. The improvement of the traffic light plan consists of increasing the traffic light time in 1, in a certain direction with the worst index, and decreasing the traffic light time in 1, in another direction. Since the green light time cannot exceed twice the initial time and the red light time cannot be less than 8 seconds. This is done to avoid accidents and prevent vehicles from wasting too much time waiting for the light to turn green.

The Figure 13 shows a diagram with the actions taken by our agents, where each agent monitors the road, changes the traffic light, and sends messages.

In the Figure 14 is the graph of analysis of the S3 simulation. Simulations of type S3 obtained the shortest average travel time among the three types of simulation. The objective of analyzing the three simulations performed is to compare the decrease in the average travel time in the models. In our comparison, we see that the S3 model is the one that obtains the shortest average travel time of the vehicles, reducing the travel time by approximately 3% in comparison with the S1 simulation. This fact points to the agents' communication as a factor that causes that traffic happens in the best way, with a reduction in total time pointing to the least number of stops and allowing cars more time at a constant speed.

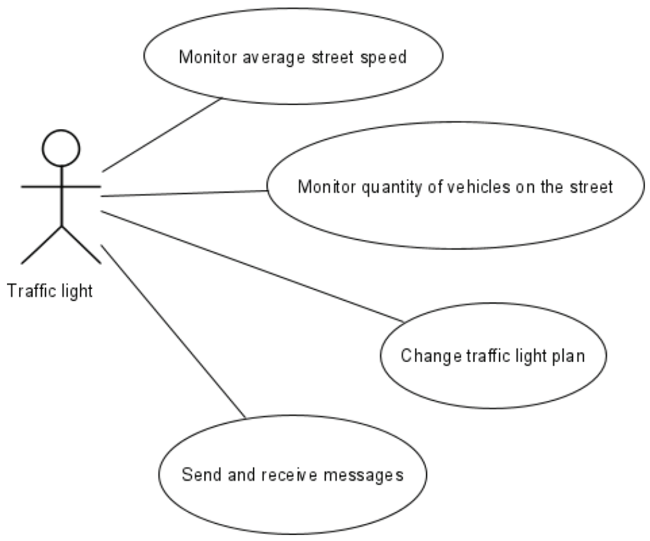
Monitor average street speed
Monitor quantity of vehicles on the street
Traffic light
Change traffic light plan
Send and receive messages

Figure 13: Diagram of Our algorithm to control traffic lights

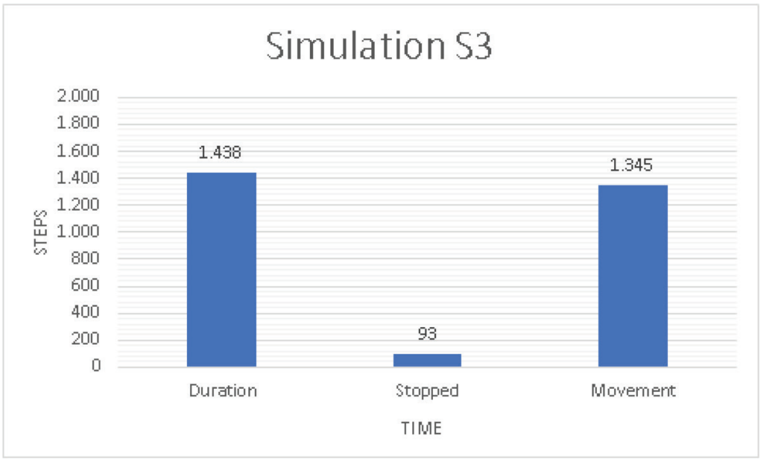

Figure 14: Time averages obtained in the S3 type simulation

4. Future work

This article presents a multi-agent system for simulating traffic control in the center of a Brazilian city. Our simulation uses data about your intersections to better compose data about traffic flow from a given location. Thus, after constructing intelligent traffic lights, we simulated traffic using the SUMO simulation tool to compare our work with that of the authors mentioned above. We analyzed the impact of communication between agents concerning the other two scenarios without communication. From the results, it was possible to compare the average duration times of each simulation. We found that our simulation with communication between agents reached a shorter duration compared to the other simulations.

This was made possible by adding communication between agents, making them share important information for their decision-making. Unlike the work of (Junior et al., 2015), the agents were unaware of the situation at other intersections nearby, which could cause congestion elsewhere due to the lack of information. This optimization that our experiments with agents with communication brought are significant given our current traffic. In a real scenario, agents could reduce travel time even further, making drivers have a better quality of travel in traffic. In addition to the fact that our results may encourage even more work to optimize traffic, expanding to machine learning areas could make agents learn to deal with different situations, such as road accidents.

With that, we intend to carry out the simulations with real data for future work as in (Junior et al., 2015). For this, it would be necessary to partner with an institution or body responsible for transportation and circulation. Completing this phase would adapt our simulations to larger scenarios and bring us more accurate and realistic results for a given location, enriching our contribution to the community.

5. References

Azevedo, T., De Araújo, P. J., Rossetti, R. J., and Rocha, A. P. C., 2016. JADE, TraSMAPI and SUMO: A tool-chain for simulating traffic light control. *arXiv preprint arXiv:1601.08154*.

Behrisch, M., Bieker, L., Erdmann, J., and Krajzewicz, D., 2011. SUMO–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind.

Bellifemine, F., Bergenti, F., Caire, G., and Poggi, A., 2005. JADE—a java agent development framework. In *Multi-agent programming*, pages 125–147. Springer.

Bennett, J., 2010. *OpenStreetMap*. Packt Publishing Ltd.

Bull, A., CEPAL, N. et al., 2003. *Traffic Congestion: The Problem and how to Deal with it*. ECLAC.

Chow, A. H., Santacreu, A., Tsapakis, I., Tanasaranond, G., and Cheng, T., 2014. Empirical assessment of urban traffic congestion. *Journal of advanced transportation*, 48(8):1000–1016.

Fernández-Isabel, A., Fuentes-Fernández, R., and Martín de Diego, I., 2020. Modeling multi-agent systems to simulate sensor-based Smart Roads. *Simulation Modelling Practice and Theory*, 99:101994. ISSN 1569-190X. doi:10.1016/j.simpat.2019.101994.

Fuelber, D. F. and Frozza, R., 2017. Gerenciamento de tráfego urbano empregando sistemas multiagentes: análise qualitativa de trabalhos relacionados. *Enegep*.

González-Briones, A., De La Prieta, F., Mohamad, M. S., Omatu, S., and Corchado, J. M., 2018. Multi-Agent Systems Applications in Energy Optimization Problems: A State-of-the-Art Review. *Energies*, 11(8). ISSN 1996-1073. doi:10.3390/en11081928.

Haklay, M. and Weber, P., 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive computing*, 7(4):12–18.

Junior, G. D., Frozza, R., and Molz, R. F., 2015. Simulação de controle adaptativo de tráfego urbano por meio de sistema multiagentes e com base em dados reais. *Revista Brasileira de Computação Aplicada*, 7(3):65–81. doi:10.5335/rbca.2015.4697.

Liu, J., Wan, J., Jia, D., Zeng, B., Li, D., Hsu, C.-H., and Chen, H., 2017. High-efficiency urban traffic management in context-aware computing and 5G communication. *IEEE Communications Magazine*, 55(1):34–40.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and WieBner, E., 2018. Microscopic traffic simulation using sumo. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2575–2582. IEEE.

Lupasc, A. et al., 2005. How to improve artificial intelligence through web. *Acta Universitatis Danubius. Œconomica*, 1(1):92–104.

Marín-Lora, C., Chover, M., Sotoca, J., and García, L., 2019. A game engine to make games as multi-agent systems. *Advances in Engineering Software*, 140. doi:10.1016/j.advengsoft.2019.102732.

O’Brien, P. D. and Nicol, R. C., 1998. FIPA—towards a standard for software agents. *BT Technology Journal*, 16(3):51–59.

Timóteo, I. J., Araújo, M. R., Rossetti, R. J., and Oliveira, E. C., 2010. TraSMAPI: An API oriented towards Multi-Agent Systems real-time interaction with multiple Traffic Simulators. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1183–1188. IEEE.

Watkins, C. J. and Dayan, P., 1992. Q-learning. *Machine learning*, 8(3-4):279–292.

Wegener, A., Piórkowski, M., Raya, M., Hellbrück, H., Fischer, S., and Hubaux, J.-P., 2008. TraCI: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163.

Wu, Q., Wu, J., Shen, J., Yong, B., and Zhou, Q., 2020. An Edge Based Multi-Agent Auto Communication Method for Traffic Light Control. *Sensors*, 20(15). ISSN 1424-8220. doi:10.3390/s20154291.