Final report - Multi-Agents Systems

Emma Machielse, Federico Tavella, Alessandro Tezza

April 1, 2018

1 Introduction

Public transport is an universal subject of travellers' complaints, targeted at the waiting and travel time. Attempts of transportation companies to reduce costs often have a deleterious effect on travelling time. A network of intelligent autonomous transportation units could optimize the desired result,lowering costs and travelling time. To illustrate this, we created a multi-agents system for an intelligent buses transportation mechanism in Amsterdam. The goal is to transfer passengers as efficiently as possible. To create this system, we implement several methods stated in multi-agents theory [1] like communication between buses, coordination and negotiation.

2 Framework

We build our system using a Beliefs-Intentions architecture, similar to the Beliefs-Desires-Intentions one [2]. Intentions are future directed and trigger certain actions [3]. In fact, every agent has a set of intentions - i.e., a set of tasks that it wants to achieve. Based on the state of the environment at a specific time, stored in the beliefs and some local variables, the agent chooses the best intention to accomplish. This framework facilitates flexibility in generating actions, because an agent can switch between intentions depending on the state of the environment.

The fleet of buses share the same interests: every bus wants to minimize the costs, the average traveling time of the passengers and the number of messages it sends. However, there are some cases in which the agents compete between each others to obtain a resource, as we describe later. The cooperation and competition behaviour are generally achieved through communication: different agents can send messages to each others in order to express facts or ideas.

3 Conceptual description and strategy

In this section, we explain the various agent-related concepts that we use, how we design and implement them, and along which strategies. Four types of costs shape our strategies: (i) the amount of money we spend; (ii) the amount of messages buses send; (iii) the amount of people waiting for a bus; (iv) the average amount of time a person spends waiting for a bus.

Costs arise by traveling and buying more buses. To reduce costs, the buses need to transport travelers efficiently. The more passengers they can transport in the shortest amount of time, the lower the costs are.

3.1 Transportation and Coordination

Buses travel according to a specific schedule. These different schedules describe the connections between various bus stops. In the following paragraphs, we focus on how we chose the different schedules and how the buses travel through them.

Different fixed schedules In order to reduce the problem into smaller subproblems, we split the map four different *areas*. These areas can be classified under the labels *"West", "North", "Center" and "East"*. Thus, each area defines a set of bus stops, and each bus is assigned to a specific area. Each set has one or more elements (i.e., bus stops) in common with at least another set. Splitting the bus stops into sets has two advantageous effects. For example, a bus focuses only in its specific area: thus, it has to complete a shorter distance before passing the same stop again, so the waiting time is reduced. Moreover, the time that a traveler spends in the bus will be shorter, which will reduce chances of buses being full and unable to pick up other travelers.

This solution occasionally requires transferring passengers from one area to another. This reduces the advantageous effects. To discuss about transportation of people between bus stops in different schedules, we define the concept of *joint position*. A joint position is a bus stop that is shared between different areas. For example, if the bus stop labeled as 4 is shared between Area 1 and Area 2, we say that *bus stop 4* is a joint position between Area 1 and Area 2. Stop 3 (`Centraal`) is a shared stop for all four areas. As a result, when a bus picks up a passenger that needs to change area, it will consider the nearest joint position with the desired area as the passenger destination: the agent brings the passenger there and it drops it. Then, an agent from the destination area picks up the passenger from the joint position and it brings the passenger to his real destination. In order to avoid useless movements of passenger, we define some conditions to pick up a passenger from a bus stop:

- the passenger destination is in the bus area, **or**
- the passenger destination is not in the bus area **and** the passenger is not already in a joint position with the destination area.

Direction-based pick up Within an area, we define two different schedules. The first one is a route through all the bus stops of the area, the second one is the reverse of the first one. Thus, we can have bus traveling in the same area but following two different directions. Consequently, we improve the traveling time by moving passengers along the direction that is faster in order to reach the destination. This adds a new condition to check whether a bus should pick a passenger to the ones listed in the previous paragraph: the passenger has to be picked up if and only if it is faster to move the passenger to the destination (or to the joint position) using the current bus schedule, compared to the reverse one. If this condition does not hold, the passenger is not picked up, and he/she

waits for another bus that is traveling with the opposite schedule. This increases transportation efficiency: since buses carry passengers for a shorter amount of time, they are able to carry more passengers.

The number of buses assigned to areas and schedules is distributed based on the needs of that specific schedule, as we explain later. In this way, schedule with higher need of buses are provided with new buses.

3.2 Roles

To ensure that the buses cooperate efficiently, we assign them specific roles. These roles differ in their ancillary responsibilities and so a hierarchy of buses was created.

Roles can also define the *behaviour* of a bus. The usage of roles makes it easy to change the behaviour of a specific group of buses.

In addition, the roles define the creation time of a bus. At the beginning of the simulation, there is only one bus, with ID 24. However, as soon as it is able to create new buses (i.e., after few ticks), this bus creates another seven initial buses: three local coordinators and four scouts, as we explain in the next paragraphs.

Global coordinator A major responsibility is buying new buses in case of shortage. If all buses would be able to do that, it would happen inefficiently and increase costs unnecessarily. Therefore we assign this responsibility only to the first bus created, the one with ID 24. This bus is on top of the hierarchy, and its role is defined as *global coordinator*.

The *global coordinator* adds buses based on shortage. Shortage is defined as the difference between the total number of people waiting and the total capacity of the bus fleet. It adds the cheapest bus type that is able to manage the amount of waiting people. Thus, when a shortage is experienced, the global coordinator creates new buses: as we explain later on, the newly created buses are going to be items of auctions in order to be assigned to a specific schedule.

Local coordinators One step down in the hierarchy, we have the *local coordinators*. Each of them is responsible for one of the four areas in Amsterdam. Note that the *global coordinator* is also a **local coordinator**. The *global coordinator* is assigned to the "Centre", whereas the buses with ID 25, 26 and 27 are responsible for "West", "East" and "North" respectively. The glocal coordinator creates all of them at the beginning of the simulation. The local coordinators keep track of their area by listing the buses assigned to their area, their fleet capacity and the number of people waiting. When a new bus is created, all the local coordinators compete between each other to assign the new bus in their area, as described in Section 3.5.

Scouts Less important than the local coordinators are the *scouts*. Note that the global coordinator and the local coordinators are also scouts. They are created at the beginning of the simulation: each schedule has one scout. Their aim is to have a bus that is always moving in the schedule. They do not have specific responsibility.

Simple Finally, at the bottom of the hierarchy we have the simple buses. Every bus is also a simple bus. However, the bus created after the beginning of the simulation are *exclusively* simple buses. This means that they do not have specific things to do, but they only have to pick, move and drop passengers.

3.3 Communication

As previously mentioned, to enable social interaction between agents with conflicting or agreeable goals, we implemented a form of communication. This communication happens through the exchange of messages. A bus sends a message with a specific goal, and the receiver performs actions based on the content.

Ontology To take into account the possibility of messages with different purpose, we define the following ontology for communication: `"message_type content"`, where `message_type` is the header of the message, or the performative [4]. This is a string that describes the goal of the message. `content` contains the effective content of the message. In this way, buses know how to use the content of the message based on the performative. The perfomatives of the messages can be related to: promotions, assigning buses to schedules, requesting/offering/accepting help, requesting a bid, bidding and reallocating a bus.

The reception of the messages goes as follows. At every tick, a bus checks its incoming messages. To check whether a message is new, its tick is compared to the latest tick in which the agent checked its inbox. This last tick is represented by a locally stored variable. The performative of the message implies how to interpret the content. Based on the content, the bus might changes its intentions. For example, when a local coordinator gets a message of type "*bid-request*", the new intention of performing a bid is added.

3.4 Group decisions

The division of the route into four areas can lead to certain scenarios that involve unnecessary costs. We take steps to deal with a situation, where one area has a shortage of buses whereas the other area has a remnant.

Whenever a bus is created and allocated to a new schedule, the coordinator sends a message to the corresponding local coordinator, in order to make it keep track of all buses that are working in that area. On the other hand, local coordinators compare the amount of people waiting in their area with the fleet capacity. Based on this comparison, they decide whether they need more buses. If this is the case, they collaborate with the other coordinators.

We implemented this collaboration by sending a message to the other local coordinators. The message has performative `help-request` and the content is a number, which represents the difference between the local fleet capacity and the amount of people waiting in that area. The other coordinators ask to their fleets if there are empty bus which can be transferred (i.e., performative `empty-check`), and if this is the case the empty buses send a message with performative `reallocable` and content their ID to the bus that requested help. Upon receiving this offer, the requester evaluates how many buses it needs to compensate the difference between the fleet capacity and the amount of people waiting, adding these buses to its fleet and sending them a message with

performative `reallocated` and with content corresponding to the local coordinator id. On the contrary, the ones that are not reallocated receive a message with performative `disengage`, which tells them that they do not need to change schedule. The other coordinator removes the bus from its bus fleet when it receives a message with performative `bye-bye`.

3.5 Negotiation

The local coordinators compete between each others in order to obtain the newly created buses.

When the global coordinator creates a new bus, it also sends a message `bid-request` to every local coordinator, that starts the auction for the new bus. Every local coordinator sends a message `bid` to the local coordinator specifying the need of the two schedules in its area. The need for a specific schedule is computed counting the number of passengers in the area that should be picked up by a bus following the schedule (remember that, based on the schedule, the direction of the bus changes and passengers are picked up based on the traveling direction). After receiving the bids from the local coordinators, the global coordinator chooses the schedule for which the need is higher, and it specifies to the newly created bus that it should drive using that schedule, sending it a message of type `schedule` containing the ID of the schedule.

3.6 Stopping buses

We allow a bus to stop when it matches some conditions. In this way, we can reduce the traveling costs whenever it is possible. In particular, a bus can stop when:

- the bus is empty **and**
- there are no passengers waiting in its schedule; here, we consider the number of passengers waiting in the bus stops composing the schedule of the bus that have to be picked up (considering the direction of the bus).

4 Code

For clarity and readability, we divided the code into different files: (i) the file `agent.nls` implements the decision process of one agent: the management of beliefs and intentions; (ii) the file `init.nls` is used to initialize all the local variables of the agents; (iii) the file `execute-intentions.nls` implements the actions related to every intention; (iv) the file `messages.nls` implements some functions to interact with a message, for example the retrieval of the content of the message; (v) the file `utils.nls` implements all the utility functions used by the other files.

5 Tuning

The agent logic consists of a set of parameters that need to be tuned for performance purposes. We try different combinations of these parameters in the

training set and in the two test sets, in order to find the best parameter setting. In particular, these parameters are:

- **Stop**: whether we allow the bus to stop or not, as explained in Section 3.6. The tuning showed better results when we do not allow the buses to stop;
- **Initial bus type**: the type of the buses created at the beginning of the simulation (i.e., the scout buses). We set this parameter to 3, consequently the firstly created buses are red;
- **Threshold for creating a new bus**: as we described in Section 3.2, the global coordinator creates a new bus based on the shortage - i.e., the difference between the number of people waiting and the fleet capacity. The shortage is compared with this threshold: when the shortage is higher, new buses are created. We set this parameter to 0, thus as soon as we have a shortage, we need to create new buses;
- **Threshold for the bus types**: when the shortage is higher than the previous threshold, new buses are created. The type of the newly created buses is determined by how the size of the shortage. Thus, we have three different thresholds for the three bus types. We set the threshold for creating a red bus to 60, the threshold for creating a yellow bus to 12 and the threshold for creating a green bus to 0. This means that we create the cheapest bus that can "handle" the shortage situation.

6 Performance

6.1 Improvements

To establish the performance of the final code, we made a comparison in Table 6.1, using the first day data as a baseline. In the baseline code, that is our solution developed for the first assignment, buses only travel around all bus stops, pick up passengers and drop off them. In the final version, we implemented all the improvements explained in Section 3.

Measurement	Baseline	Final Code
Avg travel time	150	72.53
Buses' expenses	1039682	711251
Messages sent	0	285
Final amount waiting	232	255
Avg travel time remaining	100	78.12
Final avg travel time	156	75.48

Table 1: Performance results

Table 6.1 shows that there is a significant decrease in the average traveling times and costs. Obviously, the number of messages increase because during the first version no message was sent. However, the number of people waiting at the end of the simulation slightly increases in the newer version.

This result highlights the advantages of an intelligent coordination and competition between buses compared to a system of buses that does not exploit these concepts.

6.2 Performance on test data

For this project, we have three different data set, corresponding to three different days. We used day 1 to improve our solution, while day 4 and 5 as tests for our proposal. In this section, we show the results of our solution in the three different data sets.

The final value of the different metrics for the dataset of Day 1 are shown in the right column of Table 6.1, whereas the plots for the expenses, the messages, the number of people waiting and the travelling time are shown in Figure 1, 2, 3 and 4 respectively.

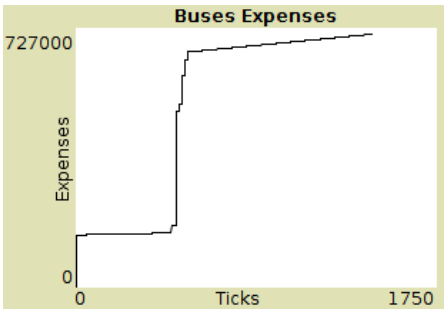

Figure 1: Expenses of the buses in Day 1.

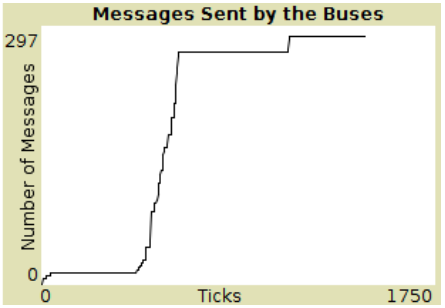

Figure 2: Number of exchanged messages in Day 1.

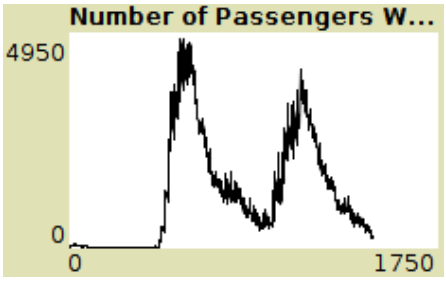

Figure 3: Final number of passengers waiting in Day 1.

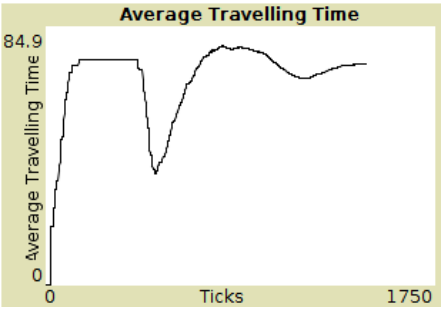

Figure 4: Average traveling time in Day 1.

Using the dataset of Day 1, we can see that there are two peeks in which the number of people suddenly appear in the map, and in which the fleet have to deal with this situation. It is important to note that during the first peek, our solution creates a number of buses that is probably enough to deal with the second peek (we can see a sudden increase in the bus expenses when there is the first peek, followed by a constant increase of the expenses for the rest of the simulation).

We are now going to focus on the results using the test sets, namely Day 4 and Day 5. In Table 6.2, the metrics value on the two test sets are shown.

Measurement	Test Data set 2	Test Data set 3
Avg travelling time	92.36	86.72
Buses' expenses	2393732	2346943.5
Messages sent	950	992
Final amount waiting	2603	2513
Avg travel time remaining	182.97	206.91
Final avg travel time	101.42	95.97

Table 2: Performance results

Our solution experienced a lot more difficulties in managing the situations of the test set. While the average travelling time is slighlty increased, compared to the one on the training set, we can notice a strong increase in the expenses and in the final amount of passengers waiting.

The plots using Day 4 for the expenses, the messages, the number of people waiting and the travelling time are shown in Figure 5, 6, 7 and 8 respectively.

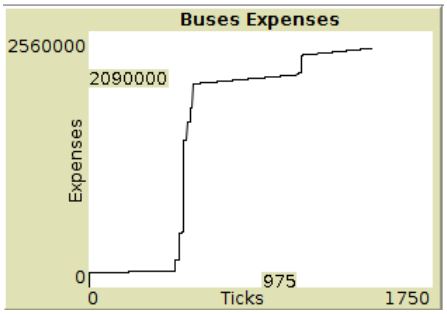

Figure 5: Expenses of the buses in Day 4.

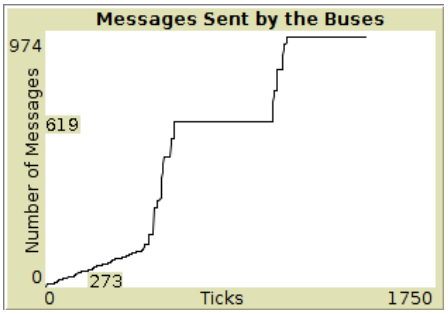

Figure 6: Number of exchanged messages in Day 4.

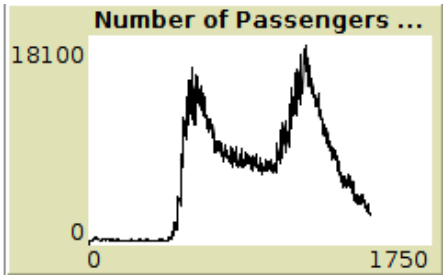

Figure 7: Final number of passengers waiting in Day 4.

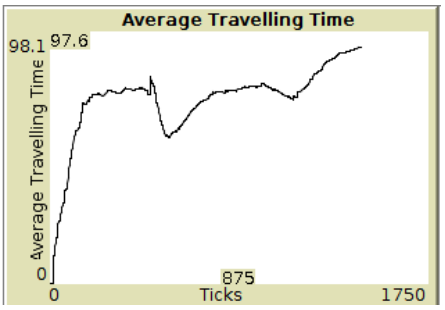

Figure 8: Average travelling time in Day 4.

The plots using Day 5 for the expenses, the messages, the number of people waiting and the travelling time are shown in Figure 9, 10, 11 and 12 respectively.

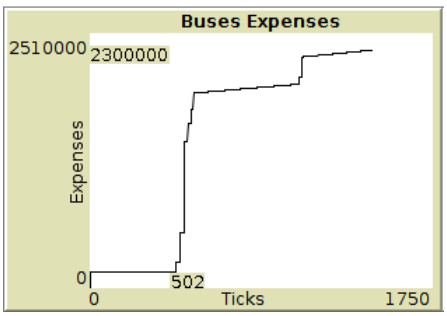

Figure 9: Expenses of the buses in Day 5.

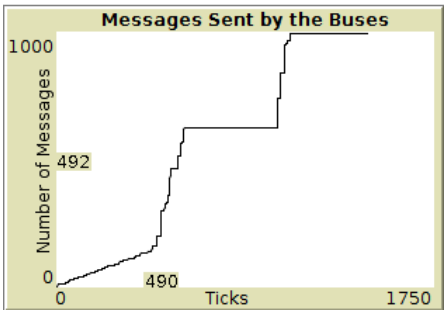

Figure 10: Number of exchanged messages in Day 5.

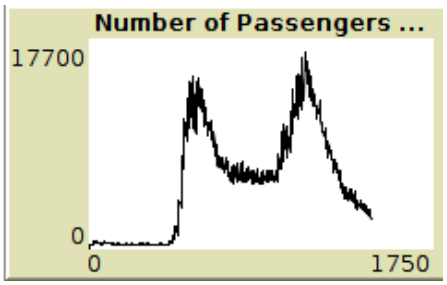

Figure 11: Final number of passengers waiting in Day 5.

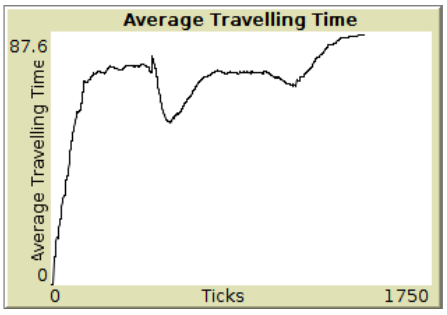

Figure 12: Average travelling time in Day 5.

7 Discussion and conclusion

In this project, we implemented a multi-agents system composed out of a set of buses that interact between each others in order to move passengers around the city of Amsterdam.

Each bus behaviour is managed by its knowledge of the environment (belief), its goals (intentions) and its responsibility within the fleet (role). Each bus travels in a specific area of the map, that is associated with a set of bus stops. Furthemore, within the same area buses can move along two different directions, based on the assigned schedule. This helps to achieve an intelligent and efficient management of passengers.

When the waiting passengers increase, new buses are created. The local coordinators of each area negotiate between each others to obtain the new buses for their area. However, in danger situations, local coordinators from different areas can help each others by exchanging bus from area to area.

This solution showed improvements in the average travelling time, while generally it is difficult to manage the number of passengers waiting at the end of the simulation and to further reduce the expenses. Probably, a solution that is not based on fixed schedule but that moves the buses based on the position of the passengers could led to improved results. However, such solution could be difficult to extend and to interpret, whereas a schedule-based solution is particularly easy to improve with new ideas and new behaviours.

As future works, we are planning to perform a detailed study of the behaviour of our system, in order to understand which are the situations where our solution is not working adequately. This study could highlight problems in the conceptual model that we design, and could help to greatly increase the performance of our solution.

References

[1] M. Wooldridge, *An introduction to multiagent systems.* John Wiley & Sons, 2009.

[2] P. Caillou, B. Gaudou, A. Grignard, C. Q. Truong, and P. Taillandier, "A simple-to-use bdi architecture for agent-based modeling and simulation," in *Advances in Social Simulation 2015*, pp. 15–28, Springer, 2017.

[3] Y. Shoham and K. Leyton-Brown, *Multiagent systems: Algorithmic, game-theoretic, and logical foundations.* Cambridge University Press, 2008.

[4] P. D. O'Brien and R. C. Nicol, "Fipa—towards a standard for software agents," *BT Technology Journal*, vol. 16, no. 3, pp. 51–59, 1998.