Multi-Agent Systems
A Multi Agent System for Autonomous Public Transport in Amsterdam
Final Report

Edwin Lima
11146222

Erwin Diepgrond
10514546

Mathijs Mul
6267939

Attached code

`agents.nls`, `Astar.nls`, `Utility functions.nls`

Abstract

In this work we study the application of different Multi Agent System (MAS) concepts, having as a use case the implementation of a MAS for public transport in the city of Amsterdam. The system has been implemented using the NetLogo platform as a simulation environment where buses (agents) travel the city picking up and dropping passengers at different bus stops. Buses are practical reasoning agents that interact with each other through message exchange. They work towards a common goal (benevolent agents), namely transporting as many passengers as possible in as little time as possible. They form coalitions and make group decisions, but also compete with each other by means of negotiation procedures.

1. INTRODUCTION

Recent technological advances have made it possible to start seriously considering the possibility of self-driving vehicles that move autonomously in cities and on the roads (Merzouki et al. 2013, Qureshi and Abdullah 2013, Victor et al. 2017). In particular, public transport companies are interested in the new developments, for instance Uber, which hires the services of individuals to offer taxi services all over the world, and which is busy developing self-driving cars which would do the job without the necessity of outsourcing the service to individuals or having to bargain prices during peak hours in order to attract more drivers to busy areas (Alley 2016).

But still, the coordination of driver-less transportation fleets brings together tremendous challenges, not only to self-driving car producers, but also to city planners who are interested in optimizing the transportation of thousands of people in the most efficient manner: timely, with a minimum cost, in an environmental friendly manner, non-stop, etc.

On the basis of many simplifications and assumptions, the current project aims to model an autonomous-buses public transportation system for the city of Amsterdam. As asked in the assignment, the buses have as ultimate goal to move people from one commute station (bus stop) to another in the most efficient manner. We use the efficiency concept as an abstraction which we have translated into utility. At all times the buses need to drive passengers, taking into consideration that the cost must remain as low as possible while the benefit of successful passenger logistics must be as high as possible. Thus, they try to optimize their utility.

The buses have been implemented as practical reasoning agents, which get information from the environment, reason about this information and make decisions based upon that. No centralized decisions are being made, but decisions are taken collectively through cooperation or voting, or individually, which means that each bus has a certain freedom to go around the transportation network without asking other buses about it. The actions that the buses can take are limited and they are based on the information that they get from the environment. Together, the agents perform communication, cooperation, coordination and negotiation in order to decide on their actions.

In the coming sections we will elaborate more deeply on the presented implementation of the MAS for Autonomous Public Transport in Amsterdam, the different concepts that we have used and how they translate into the NetLogo prototype system.

2. PROBLEM, ENVIRONMENT AND AGENT CLASSIFICATION

The problem that the transportation system is intended to solve is to determine the optimal set of routes for a fleet of buses to traverse in order to pick up and drop off a time-dependent set of customers. This is an example of a so-called 'Vehicle Routing Problem' (VRP). More particularly, as all the buses in our system have a fixed capacity that cannot be exceeded, the problem can be classified as a 'Capacitated Vehicle Routing Problem' (CVRP). Passengers are time-dependent in the sense that the number of waiting passengers change per time and per stop. There are no time windows, which specify between which moments t_1, t_2 a passenger must be dropped off.

Obviously, the aim is to drop off every passenger as soon as possible in order to minimize the average travel time.

Additionally, cost spent on leasing and driving of buses must be minimized. The number of messages that buses exchange in order to coordinate their actions must also be as low as possible.

The environment in which the buses operate is the distributed NetLogo simulation of the Amsterdam bus infrastructure. We can characterize this environment based on the different features distinguished in Chapter 2 of Wooldridge 2009. It is discrete, because only a fixed and finite number of actions and percepts are possible. It is dynamic, because not only the actions of the buses determine the next state of affairs, but also the developing passenger logistics. It is deterministic, because actions have a single, guaranteed effect.[1] It is also accessible, because agents always have correct and up-to-date information about the environment, in terms of the passengers waiting at specific stops. They have no access to each other's local variables (unless communicated), but internal states of the agents are not regarded as part of the environment.

As the reasoning performed by the buses is supposed to focus on their actions rather than on propositions or beliefs, they must be regarded as practical reasoning agents. They make their decisions in an autonomous and flexible manner, the latter meaning that they are reactive (responding to the environment), pro-active (capable of taking initiative) and social (able to communicate). They are benevolent in the sense that they all share the same goal, namely transporting as many passengers as possible in as little time as possible for as little cost as possible and with as few messages as possible. However, they all have some self-interested aspect as well because of the element of negotiation that is incorporated in their architecture, but this feature is subordinated to their cooperative benevolence.

3. BELIEF-DESIRE-INTENTION MODEL

The presented system uses an abstract version of the Belief-Desire-Intention (BDI) model to govern the agents' decision procedures. The BDI model is described in Chapter 4 of Wooldridge 2009 and Bratman 1987. Essentially, a BDI architecture is a design that uses the mentalistic concepts of 'belief', 'desire' and 'intention' to explain how and why the software does what it does. Schematically, the model can be visualized as in Figure 1.

Beliefs are statements about the environment that an agent considers true, based on sensory evidence or reasoning. Desires are overarching goals that an agent has and continues to have throughout the entire run. Intentions then are the result of the combination of specific beliefs and specific desires, which generally are supposed to filter the desires down to a specific goal that the agents explicitly wish to achieve at a given point in time.

[1]We noted that the NetLogo environment has a non-deterministic aspect because of the way in which it lists the elements of an iteration. This suggests a non-deterministic component, but as it is an unintended feature of the framework it should not affect the general characterization of the environment.

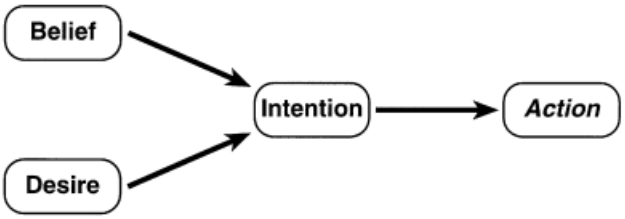

Figure 1: The BDI Model, after Bratman 1987

In our system, a utility function is used to express the benefit for a bus of moving to a particular stop. The specifics of this function are discussed in the Section 5. To understand the application of BDI in our implementation, it is enough to know that the buses are able to compute the estimated utility of going to a particular bus stop based on the information they can gather about the environment (distance, number of passengers waiting, number of passengers on board etc.). The computed utility estimates assigned to different bus stops and the resulting itineraries can be regarded as the belief basis of the buses: this information captures what they believe is true about their environment.

Obviously, buses always aim to maximize their individual utility. Hence, this can be seen as the desire of the buses. This is something that does not change, because obtaining a maximal utility is what the buses always want to achieve.

This desire, in combination with the beliefs given by the utility function, results in the intention of moving to the bus stop that yields the highest utility. Utility maximization can thus be seen as the mechanism that filters out one particular intention from the set of available options. This is highly time-dependent, as passengers and other buses continuously change their location. Hence, the agents can be seen as rather cautious agents, because they have to reconsider their intentions as they go along, and cannot blindly adhere to intentions specified in the past.

The intention of moving to a specific bus stop leads to the action of physically driving there. This is how the BDI model works in our application. Although the implementation does not explicitly use beliefs, desires and intentions as local variables, it is clear how the principle of utility maximization fits the model, and how it serves as an intuitive basis for the transportation system.

4. A* (A-STAR)

The underlying mechanism for the bus behavior is programmed in an A* variant which lies closer to the original Dijkstra algorithm for path-finding. The implemented A* searches the node space in both space and time without the usage of a goal, making it effectively a breadth first search algorithm for finding the optimal utility. Three different

utility wrapper functions are provided to the algorithm to determine the best path. The first utility function wrapper is one for arriving at a node for the first time from the current node where A* was initiated. The second function is another version of arrival at a node but that is applied always, and the last utility wrapper is the utility for the transit between nodes, or simply said: the utility for leaving a specific node. All wrappers are provided with the node in question and the time of arrival. The last wrapper, for leaving a node, is also provided with the next destination.

The algorithm takes only one argument, which is the maximum number of unique nodes visited, to improve performance. The algorithm starts by expanding the current node to all possible next states and selecting the highest utility next state. Thereafter it will again expand and select but this time for the highest state selected. This repeats until a desirable outcome has been found. In our algorithm, this desirable outcome state is the highest utility of all five node long paths.

The performance of this algorithm, due to it being a breadth first search, is $\mathcal{O}(x^n)$, where x is the highest number of possible state transitions for a node. This x includes waiting at a node, or going from node T to node T with a penalty, leaving node T for a new node Y. To improve the performance of the algorithm and to prevent long hanging times of the application NetLogo, a few optimizations have been made outside of the utility functions. The first and simplest optimization is the maximum number of iterations. When this number is exceeded, the algorithm stops and picks the best of the currently calculated routes. The second optimization is to remove all duplicate paths after each option generation step. The third optimization is to limit the maximum number of visited nodes. In our application this limit is set to 5, which should include most nodes in the map in the consideration of the algorithm. The final optimization is to remove all paths with utilities below a threshold.

The algorithm's final outcome is controlled by the three utility wrapper functions. If we, for example, give a specific node a high utility for every visit, all buses will keep visiting that node given that no other utility surpasses that specific node's utility. The used utility functions are discussed in the next section.

5. UTILITY

In the context of the MAS transportation system, agents' preferences are captured through a utility function which takes different factors into consideration.

A bus is restricted to a certain number of general actions as follows:

- Observe
- Travel
- Pick up passengers
- Drop off passengers
- Communicate
- Add new vehicles

Particularly the actions traveling, picking up and dropping off passengers are utility-driven. A bus will decide whether to visit a node according to the utility that such visit yields. Likewise, conflicts between buses will be solved competitively, using the utility function as deciding factor (see Section 9). The route to follows (travel action) is also defined by the utility function (see preceding section), as a certain route is chosen based on the utility that it yields. So, the utility function plays a crucial role in the implementation of the MAS transportation system.

We use the following utility function definition:

$$f(p,n,t,w) = visit(n,t,w) + exclusive\_visit(n,t,w) + leave\_node(p,n,t,w),$$

where p stands for the current location of a bus, n denotes the node to which the utility applies, t the expected time of arrival or departure and w the given information about the environment such as node distance, waiting passengers and bus content. Note that there are two utility functions for visiting a node, namely the exclusive and the normal visit utility function. The exclusive visit function only applies to the first visit to this node whereas the other function always applies. This prevents the bus from assigning utility to a node already visited whereas this utility does not apply anymore because the passengers were already delivered.

$$visit(n,t,w) = within\_focus\_area(n,t,w)$$

The 'visit utility' function, that is always applied on visiting a node, gives utility if the visited node is within the focus area. This prevents the clustering of buses by introducing a different utility for each bus based on its focus area. See Section 7 for a description of the application of focus areas.

$$exclusive\_visit(n,t,w) = passengers\_waiting(n,t,w) + passengers\_leaving(n,t,w) + central\_station(n,t,w)$$

The 'exclusive node visit utility' function gives utility the first time in a single route, which is always regenerated within 5 nodes. Important utility-based functions for a node visit take the following into account: the number of passengers waiting at the given node, the number of passengers that will leave the bus because this is their station and the number of passengers that are picked up and dropped off at Central Station, which serves as the main transit point.

$$leave\_node(p,n,t,w) = travel\_cost(p,n,w)$$

Leaving a node costs the travel cost between current node p and next node n.

Whenever an itinerary is determined, this is communicated (broadcast) to all other buses.

If a bus loses a negotiation procedure because he has the same itinerary planned as another bus, but lower associated utility, then the utility of the path is automatically set to 0

Utility component	*Calculation*	*Note*
$within\_focus\_area$	-5 + [5 OR 0]	5 if bus is within focus area
$travel\_cost$	$distance(A, B) * patch\_cost * 0.1$	Cost of going from A to B with bus-specific cost
$passengers\_waiting$	$(P(n)^{1.3}) * ((c - P(b))/c)$	$P(x)$ denotes the passengers waiting at x; n is a node, b a bus; c is bus capacity
$passengers\_leaving$	$destination(P(b), n)^{2.5}$	For each passenger in bus b whose destination is n, +1 utility
$central\_station$	$0.5 * destination(P(b), f) + destination(P(n), f)$	+ 0.5 for each passenger in bus b whose destination is not in focus area f; +1 for each passenger at node n (Central) whose destination is within focus area f

Table 1: Details of the utility components

to force the bus to compute a new itinerary. See Section 9 for the details of this process.

Details of the different utility components are given in Table 1. The different constants and coefficients have been determined on the basis experiments with different runs of the system.

6. COMMUNICATION

All the communication is handled through the inbox of the buses. Each bus is responsible for handling its own inbox and taking actions accordingly. For each received message a bus will take a certain action according to its current internal state and the information that it gets from the environment.

The implemented communication protocol is restricted and ad hoc, but inspired by the FIPA Agent Communication Language, a standardized agent communication language described in Chapter 7 of Wooldridge 2009. Utterances are interpreted as speech acts, which are meant to achieve a certain goal. Hence, they correspond directly to certain actions. The utterances corresponding to these actions can be decomposed into a performative and a set of parameters. In Table 2 we list the different speech acts of our system.

The exact meaning of the different messages will be discussed in the related subsequent sections. The examples show the representations of sample messages in the NetLogo syntax, where the first parameter (24 for all example cases) indicates the bus ID of the receiver, and the remainder is the message content.

Using Searle's classification of different kinds of speech acts, we recognize that most of the ones included in our protocol are representatives: they inform one or more other buses of some changes in the state of affairs. Accepting to help another bus can be regarded as a commissive: a bus promises to do something. Asking for help can be considered a directive: a bus requests something from others.

The communication protocol of Table 2 clarifies most aspects of the ontology underlying the communication between buses. As described in Chapter 2 of Wooldridge 2009, an ontology is the collection of concepts, hierarchies, relations, attributes and individuals that an agent communication languages is based on. The basic concepts of our ontology correspond with the speech acts listed in Table 2:

- assignment of focus areas
- route declaration
- asking for help
- accepting to help
- initialization of a voting procedure
- voting
- negotiation
- informing buses who have lost a bargain

The semantics of these concepts have not been formalized, because for the current purpose there is no such need. It is enough that they function as the shared vocabulary that every bus recognizes, and to which every bus knows how to formulate an adequate response, be it in the form of a replied message or a performed action.

The presented ontology is a typical example of an application ontology: the defined vocabulary is useful to the specific context addressed here, and does not aim to be more general in its applicability.

speech act	*performative*	*parameters*	*example*
Assign focus area	$\emptyset$	receiver, focus area	24 “north”
Declare route	$\emptyset$	receiver, route	24 [3 20 12]
Ask for help	‘help needed’	receiver, focus area	24 “help needed in north”
Accept to help	‘accept’	receiver	24 “accept”
Initialize voting	‘vote for help’	receiver, focus area	24 “vote for help in north”
Vote	‘vote’	receiver, vote	24 “vote 1”
Negotiate between buses with same itinerary	‘same path’	receiver, own utility of negotiated itinerary	24 “same path, utility 8.75”
Inform loser bargain	‘lost’	receiver	24 “you lost the bargain”

Table 2: Communication protocol

7. COORDINATION

As the presented system comprises a number of agents that mostly aim to cooperate in order to achieve the common goal of optimal passenger logistics with minimal costs and messaging, a vital aspect of the implementation concerns coordination. In Chapter 8 of Wooldridge 2009, the coordination problem is defined as ‘managing inter-dependencies between the activities of agents’. In our case, this boils down to the ways in which different buses intend to follow different routes without some centralized control unit telling them to do so.

Several mechanisms take care of the coordination in our system, but in this section we describe the basic structure that underlies all this functionality. This is the division of the entire infrastructure into the following five, fixed regions:

- *North*: Centraal - Hendrikkade - Hasseltweg - Buikslotermeer - Hasseltweg - Floradorp - Hasseltweg - Hendrikkade - Centraal
- *East*: Centraal - Waterlooplein - Muiderpoortstation - Science Park - UvA - Science Park - Amstel - Wibautstraat - Weesperplein - Waterlooplein - Centraal
- *South-1*: Centraal - Dam - Leidseplein - Museumplein - Zuid - RAI - Amstel - Wibautstraat - Weesperplein - Waterlooplein
- *South-2*: Centraal - Dam - Leidseplein - Surinameplein - Haarlemmermeerstation - Amstel - VU - Zuid - Museumplein - Leidseplein - Dam
- *West*: Centraal - Dam - Evertsenstraat - Surinameplein - Lelylaan - Sloterdijk - Centraal

Together, these regions span the entire city. At the beginning of the simulation, five buses are created, so that one is assigned to each of the separate regions. Apart from the first five buses, vehicles are not strictly limited to these areas, but they are strongly encouraged to stay within their boundaries through the utility function, which assigns a penalty if buses move outside of their assigned focus area. Within these regions, the buses have no fixed itineraries, but determine their routes dynamically through the utility function described in Section 5.

In Section 8, we describe exactly how buses are added or reassigned to a particular focus area. The most important thing for the overall coordination is that each focus area always has at least one bus assigned to it, so as to impose a certain division of labor on the bus fleet. We found that enforcing such preferred regions improves the performance of the system, and prevents buses from wandering around when they would do better by specifically catering to the passengers in one region. In fact, in earlier versions of the presented solution, the five regions listed above corresponded with fixed bus lines, all of which were traversed by buses in the same order. With a utility function, such a fixed order is no longer necessary and is replaced by autonomous decision-making of the individual buses.

Dividing the infrastructure up into a number of sub-infrastructures can be regarded as a decomposition of the problem that the buses aim to solve collectively. Hence, the system performs a kind of Cooperative Distributed Problem Solving (CDPS). We cite Durfee 2012 about CDPS:

> CDPS studies how a loosely coupled network of problem solvers can work together to solve problems that are beyond their individual capabilities. Each problem solving node in the network is capable of sophisticated problem solving and can work independently, but the problems faced by the nodes cannot be completed without cooperation.

In our case, the individual problem solvers are the buses. Indeed, they are loosely coupled by the transportation network, capable of sophisticated problem solving and independent functioning, but the total problem can never be solved by a single bus alone. This is why the buses must cooperate.

Although buses would come up with different tasks for themselves without the focus areas as well, these regions

provide the system with an extra mechanism to guarantee that the problem-solving happens in a distributed fashion. The distribution is now also imposed on a very concrete and explicit spatial level, representing the inherently spatial aspect of the vehicle routing problem.

Of the different ways in which a CDPS system can be designed, the method implemented here could best be classified as Partial Global Planning (PGP). This kind of planning is described in Chapter 8 of Wooldridge 2009. It is based on the principle that agents exchange information to coordinate their activities, without a central controlling unit. The following three iterated stages take place in any PGP process:

1. Agents make local plans based on their own goals and local information.
2. Information about local plans is communicated and exchanged.
3. If necessary, local plans are adjusted in order for coordination to improve.

Step 1 of this procedure was described in Sections 3 and 5: based on their information about factors such as passengers, distances and travelling cost, agents decide on an itinerary that maximizes their personal utility, which is a local notion. Their itinerary is then sent as a message to all other buses (broadcast). This is step 2 of the PGP routine. If another bus happens to have the same itinerary planned, then a negotiation procedure takes place as described in section 9. Based on the outcome of this negotiation, one of the involved buses will have to adjust its plans, thus performing step 3 of the PGP scheme.

Another aspect of the solution that must be described in this section is the actual strategy for picking up and dropping off passengers. For this purpose, a very simple method was implemented. Buses treat Central Station as the main stopover station. When buses reach any stop except Central Station, they pick up as many passengers as their capacity allows. They drop off passengers at their final destination, or at Central Station if their destination is not in the focus area of the bus. At Central Station, passengers are only picked up if their destination is in the focus area of the bus. Hence, not only the movements of the buses, but also the coordination of the actual passenger logistics is based on the distribution of the problem with respect to different focus areas.

Effectively, the division of the bus fleet into different focus areas can also be regarded as a way of constructing coalitions: groups of buses cluster together in order to address a more or less demarcated fragment of the overall problem. This is where the benevolent nature of the agents in the system becomes very clear: the more buses join in their coalition, the more likely they are to achieve a good performance in their respective areas. As described in Chapter 13 of Wooldridge 2009, coalitions can arise in contexts where agents have reason to trust each other and mutually profit from cooperation, which is exactly the kind of scenario we are dealing with here.

8. MAKING GROUP DECISIONS AND VOTING

Buses do not only make decision based on their individual utility estimates, they are also capable of making genuine group decisions. They do this by means of a voting procedure. Such a voting procedure can be initialized by buses that are 'in trouble'. This is to say that a bus is running into difficulties while attempting to achieve its intentions. To be precise, we consider this to be the case in two situations:

(a) The number of passengers on board of the bus exceeds 70% of its capacity.

(b) The number of passengers waiting in the focus area of the bus exceeds the remaining space on board.

In these two situations, a bus obviously would not be able to achieve a good performance on its own, because it would soon have to leave passengers behind due to the limited number of seats on board. So if there were only one bus, these situations would signal the need to add a new bus. However, because we are dealing with a multi-agent system here, this is typically not a decision that buses should make on their own. Most of the times, buses will not be on their own in a focus area, and so the decision whether a bus must be added or reassigned to their focus area should be made collectively rather than individually. This is to prevent cases where buses are added due to the situation on board of one bus, whereas there may be other buses right around the corner with plenty of remaining space.

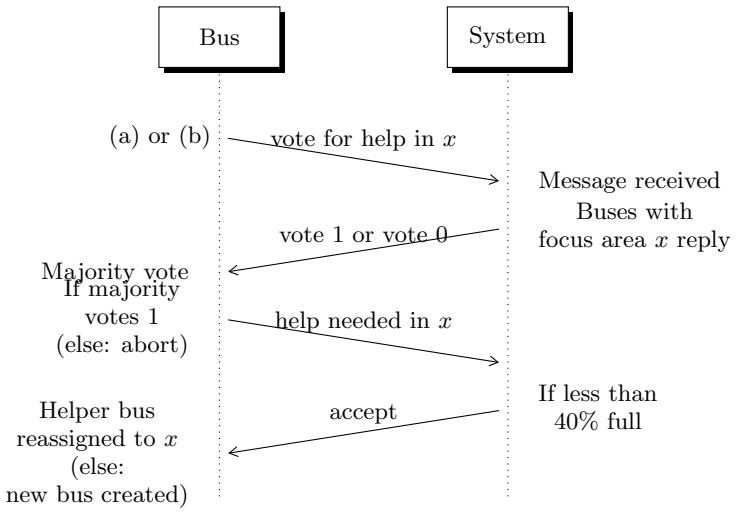

Figure 2: Voting protocol

Hence, if a bus encounters one of the two situations (a) or (b), a voting procedure will be started among the buses in its focus area. See Figure 2 for a schematic overview of the

voting procedure and Table 2 for details of the communicative actions that are involved. The message 'vote for help in x', for x the focus area of the bus concerned, is broadcast to all other buses. Buses in the same focus area will then check the number of passengers they have on board themselves. If this number exceeds 70% of their capacity, then they will be considered to agree with the bus in need of help, because they themselves have almost exhausted their capacity. Hence, these buses will vote 'yes', c.q. vote 1. Otherwise, the buses will vote 0.

Formally we have a set of two possible outcomes $\Omega = \{w_1 = 1, w_2 = 0\}$, where 1 stands for 'start help procedure' and 0 for 'start no help procedure'. We use a plurality vote, which in this case is effectively a simple majority election as there are only two outcomes (see Wooldridge 2009, Chapter 12). If the majority is in favor of starting a help procedure, then we proceed as such, otherwise it means that other buses in the focus area have enough capacity to carry more passengers. Considering that we only have two outcomes we do not to worry about the anomalies associated with plurality voting.

The help procedure works as follows. First a message is sent to all other buses in which the bus asks for help: 'help needed in x' for x the focus area of the bus. If there happens to be a bus somewhere else in the city with less than 40 % of its capacity used, then this bus will reply with an 'accept' message, and it will be reassigned to the focus area of the bus in need of help. If there is no bus that can afford to change its focus area in this way, then a new medium-size bus has to be added. This bus is then immediately assigned to the focus area where help was considered necessary.

The help procedure can be used only once per 250 ticks per bus, to prevent explosion of the number of vehicles. This can be considered as a refractory period. Evidently, during this period, buses do still participate in possible voting procedures initiated by other buses in their region.

9. NEGOTIATION

Although the buses are essentially benevolent agents, there is an element of competition between them as well. This is because negotiation procedures take place between them when their planned itineraries collide. More specifically, if a bus has made its utility calculations and on this basis plans to drive a certain itinerary, then it broadcasts this route to all other buses. If another bus already has the same itinerary planned, then this is considered a collision that has to be avoided through a bargaining procedure.

As we aim to make the problem-solving as distributed as possible, it is considered undesirable for two different buses to do exactly the same thing. Hence, collisions must be avoided. The most intuitive way of dealing with this problem is by again appealing to the utility estimates that all buses already have at their disposal. However, in this case we are not dealing with a decision made by a bus between different itineraries, but with the decision which one of two buses will be granted the 'right' to drive one specific itinerary. So instead of maximizing utility with respect to bus stops, the idea is now to maximize utility with respect to buses.

In practice, this has been implemented as follows (see Figure 3). When a bus, say B_1, broadcasts its itinerary, say [3 20 12], and another bus, say B_2, detects that it already has the same itinerary, then a collision is about to take place. B_2 then replies to the broadcast made by B_1 with a message informing B_1 about the collision. In this message, B_2 also sends its own utility estimate of driving itinerary [3 20 12]. This message could be, for instance, 'same path, utility 8.75' (see Table 2).

Now the actual negotiation takes place: buses B_1 and B_2 compare their respective utilities of traversing itinerary [3 20 12]. If B_1 has a higher utility, then B_1 sets [3 20 12] as its new itinerary and sends a message to B_2 informing the bus about its loss: 'you lost the bargain'. Each bus has its bargain status as a local variable, which by default is set to 'none'. However, as B_2 just lost a bargain, its bargain status is now set to 'lost bargain'. If, on the contrary, B_2 has a higher utility than B_1, then B_1 has to change its own bargain status to 'lost bargain'.

Whenever a bus has its bargain status set to 'lost bargain', the utility of its current path is forced to be 0, irrespective of any other factors. This compels the bus to recalculate its path, and come up with a new suggestion. This new planned itinerary is then broadcast again.

The described negotiation protocol is rather simple: bargaining takes place on a one-to-one basis. No many-to-one or many-to-many negotiations have been implemented.

In fact, the outcomes of negotiation procedures in the current implementation do not directly change the behavior of the buses. Instead, they manipulate this behavior through the utility function: by forcing the utility of the current path to be 0 for buses that just lost a bargain, such 'loser' buses are indirectly motivated to reconsider their intended itinerary.

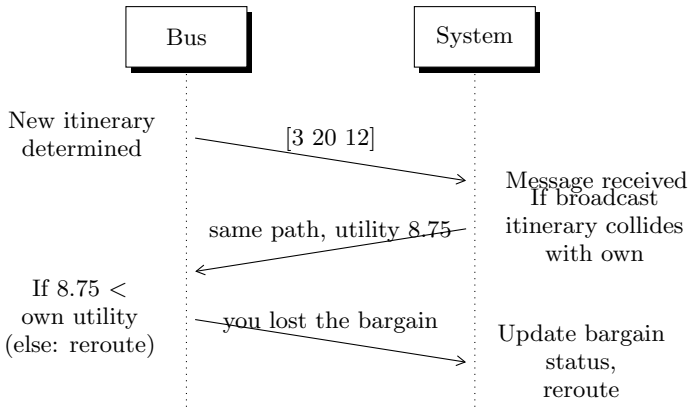

Figure 3: Negotiation protocol

10. RESULTS

Here we include the graphs generated by NetLogo after a complete run of the system. At the end of the simulation, some 35,000 messages have been sent, 600,000 money units have been spent, and the average travel time is approximately 350 minutes.

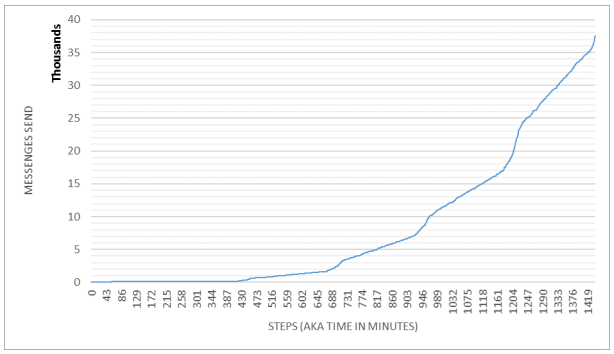

Figure 4: The number of messages sent

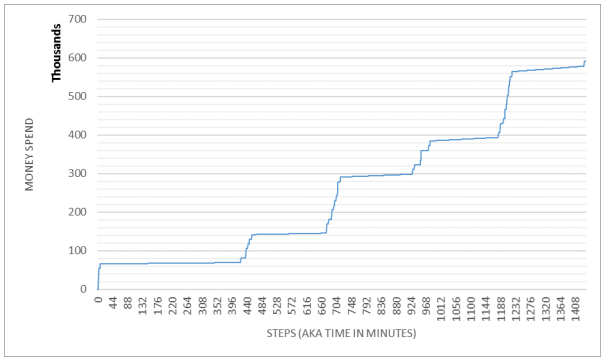

Figure 5: The amount of money spent

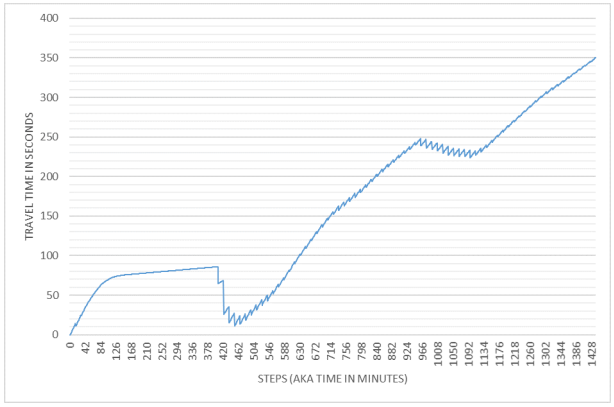

Figure 6: The average travel time

11. DISCUSSION

A variety of Multi Agent concepts were incorporated in the presented solution in order to exploit as many of the advantages that a MAS may have. However, the results of our implementation have indicated that, clearly, this is not enough if the aim is to achieve a good performance in terms of average travel time, amount of money spent and number of messages sent.

Although the aim of the project was not to optimize with respect to this performance, the results do invite a discussion about possible improvements that could be made in order to obtain better statistics. We will mention a number of such options here. Some of them were considered in earlier stages of the project, but not implemented because other suggestions were prioritized.

First, the BDI model underlying our solution may have to be reconsidered, not as much in terms of its general structure as in terms of the concrete ways in which we decided to shape the beliefs, desires and intentions. Especially, the use of our utility function must be critically assessed. In its current form, this function does not take everything into account that should be regarded as relevant information. For example, when dealing with waiting passengers, it does not pay any attention to the amount of time people have been waiting, and it does not consider for how long people have been on board. More generally, we could consider to abolish the node-dependent utilities altogether, and instead work with notions such as goals.

Also, the system of focus areas underlying much of the coordination that goes on in our system, may not be the optimal solution. In fact, these focus areas could be seen as an implicit global constraint. If we wish to design a truly distributed system with agents who make all their decisions on a completely autonomous basis, then perhaps it is better to dispense with this subdivision into regions. Buses would then optimize their planned itineraries with respect to the entire infrastructure instead of being associated with a restricted part of it. This would also open the way to changing the current stopover method. Quite primitively, all stopovers now take place at Central Station. With more dynamic itineraries, this could be made more efficient.

With respect to the CDPS that goes on, different methods for coordination could be investigated. As described in Section 7, currently this happens through a form of Partial Global Planning. Instead, it is possible for the buses to engage in mutual modelling, forming beliefs about the intentions and itineraries of the other buses on which they can base their own decisions. Another alternative is the use of social norms: generally, buses could be prevented from performing certain actions in a certain state of affairs. E.g., if bus B_1 is planning to pick up passengers at some bus stop, bus B_2 could decide not to do so because a social norm tells the bus to refrain from picking up passengers 'reserved' for another bus. A third alternative is the adoption of joint intentions, which buses share because they are collectively committed to some goal. This goal could be quite specific, e.g. delivering a restricted set of passengers. Buses could pursue it together, only giving it up if it is satisfied or no

longer considered possible.

Instead of enhancing cooperation, we could also further exploit the competitive aspects of the system. This could be done by intensifying the negotiation procedure. Currently, this routine is very restricted and only takes place when buses have the same itinerary planned. In a more complex, dynamic solution, we could envision bidding for clusters of passengers, depending on their current and planned location. The negotiation could then involve more than two buses, and be many-to-many or many-to-one instead of one-to-one.

Aspects lacking in the current system are argumentation and use of game theoretic concepts. Argumentation was considered, but we did not find any immediate use for this in the context of the bus infrastructure. However, in a more advanced system it would be possible for buses to convince each other of driving certain itineraries on the grounds of communicated arguments. These arguments might be based on different sources (different utility estimates, path collisions, fixed beliefs, agreements), but they would all be logical rather than emotional or visceral.

Many other options were considered. For instance, the voting procedure could have been designed according to a different model than the basic plurality vote. Also, different kinds of coalitions could have been realized. In addition to, or instead of, buses working together in focus areas, they could have been clustered according to the final destination of their passengers. Moreover, all buses (except for the first one, which is small by default), are currently medium-sized. A more sophisticated solution could experiment with different bus sizes. Buses currently keep driving, whereas sometimes it may be better to let them pause for a while in order to reduce the travel costs. A more advanced system of agreements between buses is possible. And a probabilistic approach to passengers statistics may be attempted, in order to intelligently design dynamic bus itineraries on the basis of observed patterns in the past. Machine learning techniques could be applied to estimate a prior distribution over time and locations. However, this would require more data than the available statistics for one day.

This discussion is not exhaustive, because many other alternative suggestions can be made in order to improve the presented Multi Agent System.

12. CONCLUSION

The Multi Agent System for autonomous public transport in Amsterdam has been implemented with all the expected functionality as per the requirements of the assignment. The presented system has proven to have the required characteristics that make it a good candidate for a MAS-based solution:

- Control, data, expertise are distributed
- Centralized control is impossible or impractical
- Processing nodes have competing viewpoints or objectives

The system offered many challenges, but in particular two aspects turned out to be the hardest ones:

1. The coordination of the system
2. The limitations of the NetLogo environment

The first aspect gradually became more explicit when functionality was added to the system. More functionality demanded more control in order to avoid that functionality A could conflict with functionality B. For example, a voting system where all the agents make a collective decision versus a competition functionality where the agents decide independently of each other which route to follow.

Additionally, the NetLogo platform itself is quite limiting in terms of its programming language, rather unintuitive data structures and slow performance as the complexity of the system increases.

In the Discussion it was pointed out that machine learning techniques could have been applied, but the platform would have obliged us to use another environment in order to learn parameters and be able to use existent algorithms, which NetLogo does not offer.

Last but not least, the messaging system via inbox was not exactly what one could call an ideal messaging solution. In a real life situation, ideally agents should be able to communicate with each other interactively using a clearly defined ontology. Imposing such an ontology on the messaging system, as we attempted, is very cumbersome in the Netlogo platform.

A more sophisticated solution, which takes into consideration a broader functionality would also need a platform that offers a richer feature set, especially if we consider that technology such as self-driving buses is no longer a far future but an imminent future.

Nevertheless, the platform allowed us to implement the expected functionality and to put in practice some important MAS concepts and techniques. The present work could hopefully serve as a basis for future work on the implementation of a Multi-Agent System for autonomous transportation.

References

Alley, Jonathan K (2016). "The Impact of Uber Technologies on the New York City Transportation Industry". In:

Barbucha, Dariusz and Piotr Jedrzejowicz (2008). "Multi-agent platform for solving the dynamic vehicle routing problem". In: *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*. IEEE, pp. 517–522.

Bratman, M. E. (1987). *Intention, Plans, and Practical Reason*. CSLI Publications.

Durfee, Edmund H (2012). *Coordination of distributed problem solvers*. Vol. 55. Springer Science & Business Media.

Larsen, Allan and Oli BG Madsen (2000). "The dynamic vehicle routing problem". PhD thesis. Technical University of DenmarkDanmarks Tekniske Universitet, Department of TransportInstitut for Transport, Logistics & ITSLogistik & ITS.

Merzouki, Rochdi et al. (2013). "Intelligent transportation systems". In: *Intelligent Mechatronic Systems*. Springer, pp. 769–867.

Qureshi, Kashif Naseer and Abdul Hanan Abdullah (2013). "A survey on intelligent transportation systems". In: *Middle-East Journal of Scientific Research* 15.5, pp. 629–642.

Savelsbergh, Martin WP and Marc Sol (1995). "The general pickup and delivery problem". In: *Transportation science* 29.1, pp. 17–29.

Toth, Paolo and Daniele Vigo (2014). *Vehicle routing: problems, methods, and applications*. SIAM.

Victor, Trent et al. (2017). "When Autonomous Vehicles Are Introduced on a Larger Scale in the Road Transport System: The Drive Me Project". In: *Automated Driving*. Springer, pp. 541–546.

Wooldridge, Michael (2009). *An introduction to multiagent systems*. John Wiley & Sons.