Transportation System Report

Sherida van den Bent, Pim Dijt, Daniel Maaskant (Group 12)
10428461, 11168056, 10326170

April 1, 2018

1 Introduction

Have you ever seen a collective of ants, who are inherently simplistic animals, make relatively smart decisions? This is because of the hive brain. Ants communicate with each other, and together, they are one single big smart entity. One multi agent system.

In this paper, the multi agent system principle is used on vehicles. More specifically; public transport buses. These buses travel through the city to pick up passengers, all the while communicating with each other to optimize the entire process. This communication exists of voting for extra buses on a particular line, auctions to win customers and a master-bus coordinating all of this.

2 Environment

The map used for this system is based upon the map of Amsterdam, simplified to the 24 most important public transport stops. The connections between these stops are simplified to "as the crow flies" distances. This results in an undirected, not fully connected graph. This graph, superimposed on the actual map of Amsterdam, can be found in figure 1.

2.1 Schedule

In order to stay close to the real-life situation, the driving schedule that was implemented for this research was based on the current schedule of the GVB for the buses in Amsterdam. To reach this schedule, firstly the GVB bus schedule was mapped to the bus stops and connecting edges in the simulation environment. This resulted in a partial schedule shown in figure 2.

This schedule is the basis for the final version. For the final version the unconnected bus stops were added to the graph by either expanding current bus routes or adding a new bus route. New

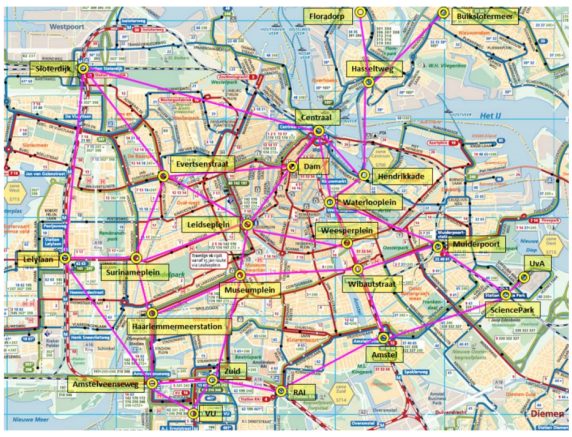

Figure 1: Map of Amsterdam with the simplified graph on top in pink

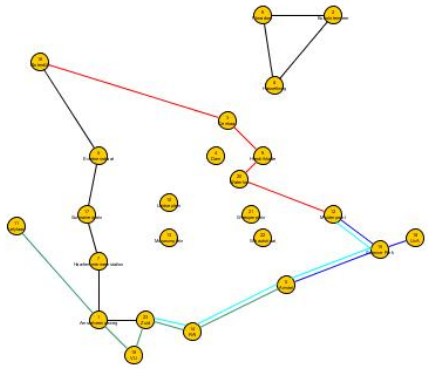

Figure 2: GVB schedule mapped to simulation bus stops

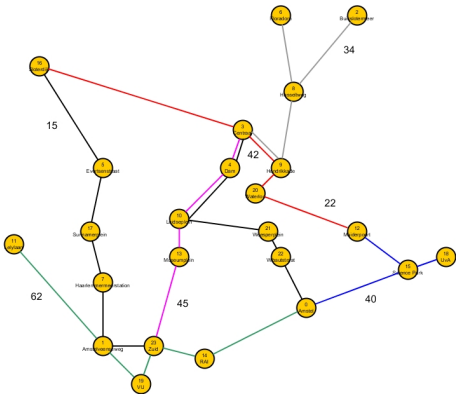

Figure 3: GVB schedule altered to fit simulation stops

routes were needed in the city center, because the GVB uses trams instead of buses there; hence, those tram routes were not part of the base bus schedule. The final schedule is shown in figure 3.

2.2 Passengers

The passengers in this simulation are little more than objects. They do not reason for themselves; it is the duty of the buses to decide which bus on which bus line picks up the passenger, and where that bus drops the passenger again.

Each passenger initiates at a bus stop, with another bus stop as its destination. The amount of passengers on each given bus stop at each given time with each given destination are scraped from Google maps; therefore, it can be assumed that they accurately represent a normal day in the public transport system of Amsterdam.

2.3 Buses

When a bus drives by a bus stop, it is able to pick up passengers. There are different scenarios for the passengers at the bus stop, and the bus will decide what to do.

The first scenario is where the passenger in question has a destination that is on the bus line of the current bus. In that case, the bus automatically picks up the passenger, as it is sure that it can bring the passenger to his or her required destination.

If the bus does not have the destination of the passenger on its line, it checks if there is another bus line with the current stop in its schedule that also has the destination in its schedule. If one such line is found, the bus does not pick up this passenger, as another bus can pick up the passenger and drop it off at its required destination without any other transfers.

If none of these two options hold, a transfer is in order. It is then checked whether the destination is in the schedule of a bus which the current bus connects to, in order to plan a transfer. The transfer is then added to the list of transfers of the bus and the passenger is picked up. Lastly, the connections are dense enough that a maximum of 2 transfers may be necessary, therefore the last check consists of checking if the destination is in the schedule of a bus that connects to another bus, which in turn connects to the current bus. This guarantees that every passenger will be picked up eventually, as the maximum of necessary transfers with these bus lines in this graph is two.

Next to that, a version is implemented which incorporates bidding for passengers by various buses. How this mechanism works will be explained in chapter 5.

The bus checks all of its passengers and their destinations to find whether the current stop is their destination and if so, drops them off. If not, then it will check if the current stop is their transfer destination and drop them off. If this is the case, the transfer will be deleted from the list of transfers of the bus.

3 Communication and Coordination

The agents in the transportation system (buses) have a certain knowledge about the world. This knowledge includes the various possible bus routes, the bus schedules and which passengers are waiting at which bus stops. This knowledge alone is not enough for the agents to coordinate with each other so that people get to their destination faster. To set the foundation for this, the agents must also communicate with each other to gather extra knowledge. Such knowledge includes asking for a schedule to let the bus initialize itself and other messages explained in later chapters.

3.1 Master-bus

For the buses to communicate with other buses on the same line, they need to know which buses ride on which line. A master-bus which stores this information is implemented. The master-bus will always stay at central station, and can be seen as a 'headquarters' for the bus company. Next to holding the configuration of the buses over the lines, the master-bus also has to power to add buses to lines which need more capacity. It will only do so if a bus requests for a vote, more on this will be explained in chapter 4.

3.2 Protocol

In the system it is important to communicate with the other buses on your bus route. Because the master-bus holds this information, all the communication goes through the master-bus. When an agent receives a message, it is stored in the inbox of that agent. This message contains the tick, sender and the actual content of the message. This content within the message can either be a request or a response. Both these messages are accompanied by the nature of the request or response and possibly a value if applicable. This value may have the form of a single integer or a list of integers. The resulting message has a structure of a tuple with the elements in the order just mentioned.

3.3 Requests and Responses

3.3.1 Buses

Request and response for initialization
When a bus is added to the environment, it is unaware of which bus line it has to operate in. It therefore asks the master-bus which bus line the bus will have to follow. It will then drive to the bus stop specified by the master-bus for that line after which it can function as a normal bus.

Request for vote
The protocol for this message be to set "vote" as the nature of the message in the message. Via this message a bus requests the master-bus to arrange a vote between the sender and the other buses in the bus line of the sender.

Response for vote
When a bus is requested to vote whether it wants an additional bus on its line, the bus will check its current capacity. This is then compared to the threshold and the bus will send a response to the master-bus.

Request for auction
In the version of the implementation where passengers are auctioned between different buses, a bus can request the master-bus to arrange this auction. The bus will request this auction if the second next bus stop is in fact a crossing at which other bus lines pass by as well.

Response for auction
When a bus receives a request to bid over passengers, it will respond with a list of biddings for each passenger.

3.3.2 Master-bus

Response for initialization
The master-bus will look up the configuration for that bus and then sends a response with this information.

Request for vote
The master-bus receives a request about a vote and sees this as an initial "yes" vote. It will then look up the bus line of the sender and then send a request to vote to each of the other buses on that same line.
All of the responses are recorded and once all of the expected votes have been received, the master-bus will check whether a majority has been reached. If so, it will add a bus to that line.

4 Voting

The buses are now able to vote on whether or not they want to have an additional bus on their line. A bus will now request a vote when it is 2/3 full and the master bus will coordinate this vote. When receiving a request, the master-bus also notes a vote for an additional bus. Then, the master bus will check with possible other buses on that line to ask for their vote. They will respond with either a 0 or a 1 for "no" or "yes", respectively. This number is then added to the one coming from the original request for the vote, eventually resulting in a final score. This final score is reached when all of the expected votes are in. If this final score is higher than the rounded down number of half of the amount of possible votes on that line, then a new bus will be added, resulting in a simple majority voting scheme.
A timeout has been introduced to prevent buses from sending a request for a new bus every tick. The bus must first wait 15 ticks after its last request, before it is allowed to make a new request.

5 Picking up passengers

5.1 Competition

When the second next stop of a bus is a transfer station, multiple bus lines have that stop in their schedule, the bus initiates a bidding for that stop. It sends a message to the master-bus, stating that it wants to start a bid by specifying the list of passengers currently at that stop. This specification ensures consistency of the passengers to vote on over multiple ticks. Furthermore, limiting the auctions to competitions between bus lines and not within bus lines themselves, reduces the amount of messages needed. Because within a bus line, the bus that requests the auction will be the closest to the bus stop in question.

5.2 Master-Bus

The Master Bus receives a request for a bidding for a particular bus stop, including the list of passengers to auction. It then sends a request to each bus in each bus line that passes by that stop. Each bus then replies with a list containing the amount it wants to bid on each passenger on that bus stop, accompanied by the bus stop, its bus ID and the amount of space left on the bus. The Master Bus then checks per passenger which bus wins that bid,, however, when the limit of space of a bus has been reached, the passenger will go to the second-best, and so on. Then it informs each winning bus by sending a list of passenger ID's that they may pick up.

5.3 Bidding amount

The amount that a bus bids for a passenger is reliant on both the amount of transfers the passenger would need to do to reach its destination if the bus would pick the passenger up, and the distance from the bus to the passenger. The amount of transfers determines its transfer score, which is 1, -1, -2, -3 or -4, when the total transfers are 0, 1, 2, 3 or the location is «»»««, respectively. This score is then multiplied by 1000, making the distance a factor that only distinguishes between buses that need the same amount of transfers to get the passenger to its destination. Possible improvement:

Also incorporate the distance from the location of the passenger to its (intermediate) destination, as the closest bus may go the wrong way on that line.

6 Experiments and Results

For the system to work well, enough buses need to be driving around to pick up passengers. The amount of buses driving around is influenced by the current buses, who vote if they want extra buses or not. This voting will only take place if a bus requests it. This happens if the occupancy of the bus is higher than a new-bus threshold. This threshold will be simulated with the values 0.25, 0.33, 0.50 and 0.67. Furthermore, the simulation will be tested with the bidding on passengers in place and without.

To assess the simulation, the average travel time, money spent, total messages and buses created are averaged over 30 simulations. The results for the simulations with bidding in place is shown in table 1. The results for the simulation without bidding in place is shown in table 2.

	average travel time	money spent	total messages	buses created
Voting threshold 0.67	88.09	860604	63479	32
Voting threshold 0.50	80.50	1136336	88523	45
Voting threshold 0.33	77.92	1525232	127551	63
Voting threshold 0.25	77.58	1882800	159017	80

Table 1: Table of results for different values for the threshold for creating a vote for an extra bus in the version with bidding.

	average travel time	money spent	total messages	buses created
Voting threshold 0.67	57.19	1207348	75512	48
Voting threshold 0.50	54.09	1508356	96580	62
Voting threshold 0.33	52.17	1871406	117712	79
Voting threshold 0.25	51.27	2504846	153637	109

Table 2: Table of results for different values for the threshold for creating a vote for an extra bus in the version without bidding.

It should be noted that the performance measure for this system will be done by ranking each of the scores, except for buses created, against the other systems produced during this class. The average of these ranks will be the final performance measure, with the average travel time weighing twice as heavy as the other two.

7 Discussion

When looking at the results and keeping in mind the performance measure, the simulation performs better when the auction mechanism is not in place. When varying the voting threshold, the total messages send stay roughly the same over the 2 types of algorithms.

The decrease in money spent and increase in average travel time when using the auction mechanism can be contributed to the fewer buses created. With fewer buses, the buses are full more often and the passengers have to wait longer. For more buses to be created, the majority of buses in a bus line have to be occupied for a larger portion than the voting threshold. The voting mechanism suppresses this amount, as the passengers are less sparsely distributed over the buses, meaning that the majority vote will be won less often.

When comparing similar amount of buses created, which is the case for threshold 0.67 without bidding and threshold 0.5 with bidding, it may be noted that the amount of messages are higher for with the auction system. This is probably due to the amount of extra messages needed for

the auctions. Furthermore, even though a similar amount of buses is created, the average travel time is lower. This suggests that next to fewer buses created, also the way the passengers are transported is less efficient when using the auction mechanism. This could be happening when passengers arrive at a bus stop whilst a bidding is taking place. If that happens the passenger has to wait until the next bidding will take place.

When comparing scores between the different voting thresholds it is clear that there is a direct effect on the average travel time, which decreases with the voting threshold. This in turn allows for more buses created, which explains the reduced travel times.

8 Conclusion

According to the performance measure presented by the course, it may be concluded that the version without the auction system has a better performance.

Another conclusion might be that an auction system while having set bus lines is not the most efficient implementation of that form of cooperation. It may be more beneficial to try such an implementation in a more free environment, where agents choose their own paths.

Also, in the current environment it was not possible to remove buses from the simulation once they were created. Instead the buses could stand still when fewer buses were needed to reduce driving costs. Also, buses could possibly change bus lines to more crowded bus lines and different bus types were available. These things were however not implemented.