A Multi Agent System for Autonomous Public Transport in Amsterdam[☆]

Boaz Vetter[a], Michelle Appel[b], Nils Hulzebosch[c]

[a] *Vrije Universiteit Amsterdam (VU)*
[b] *University of Amsterdam (UvA)*
[c] *University of Amsterdam (UvA)*

Abstract

This report presents a software implementation of a public transportation system using multi-agent system strategies. The goal is to create a system which functions as efficiently as possible in terms of travel time, costs and number of messages sent between agents. A score based decision method is proposed, where a score is computed which measures the favourability of travelling to a stop and picking up certain passengers, and is used to base which vehicles claim which passengers upon. To introduce new vehicles a voting mechanism is used. This multi-agent system was successfully implemented. Testing of the system showed that there is room for improvement, especially during peak times.

Keywords: Multi-Agent-Systems, MAS, Autonomous Transport, Artificial Intelligence

1. Introduction

This report presents a software implementation of a public transportation system using multi-agent system strategies. The intended public transportation system consists of multiple intelligent buses that pick up and drop off passengers at various bus stops around Amsterdam. These buses have their own logic and effectuate different kinds of strategies and actions autonomously, while communicating with each other. A basic framework in Netlogo was provided. The goal is to create a system that functions as efficiently as possible in terms of travel time, costs and number of messages sent between buses. This report describes a conceptual description of the system, various concepts used and finally the results are analyzed and discussed.

2. Environment Characteristics

Simulations are performed in NetLogo, a "multi-agent programmable modeling environment"[1]. Buses are treated as agents and can only use their local knowledge, or request other

[☆] Final assignment for the course Multi-Agent Systems (MAS) 2018 of the Master's Artificial Intelligence.

Email addresses: boazmvetter@gmail.com (Boaz Vetter), appel.michelle@gmail.com (Michelle Appel), n.hulzebosch@hotmail.com (Nils Hulzebosch)

[1] https://ccl.northwestern.edu/netlogo/

buses for knowledge (for example passengers in a bus). However, passengers at bus stops can be acquired through a central database, meaning buses are not required to be at bus stops to get information about the passengers at bus stops. Buses come in three types, each with their own passenger capacity, lease cost (in Euros), and cost per patch (in Euros), as shown in table 1. Lease costs are paid once when a bus comes into play, while patch costs are paid whenever a bus moves. Buses can pick up and drop off passengers at bus stops (no costs) and travel to bus stops (patch costs).

Bus type	**Capacity**	**Lease cost**	**Patch cost**
Small	12	6.000	1
Medium	60	12.000	1.5
Large	150	20.000	2

Table 1: Properties of the three types of buses.

For the simulation, a simplified (made-up) version of Amsterdam's bus system is used, including 24 bus stops, as shown in figure 1 (provided by the course instructors). The pink connections are the travel paths the buses can take. Furthermore, the distance is given in patches, a patch being a standard area in NetLogo. Larger distances (more patches) result in longer travel time and higher costs.

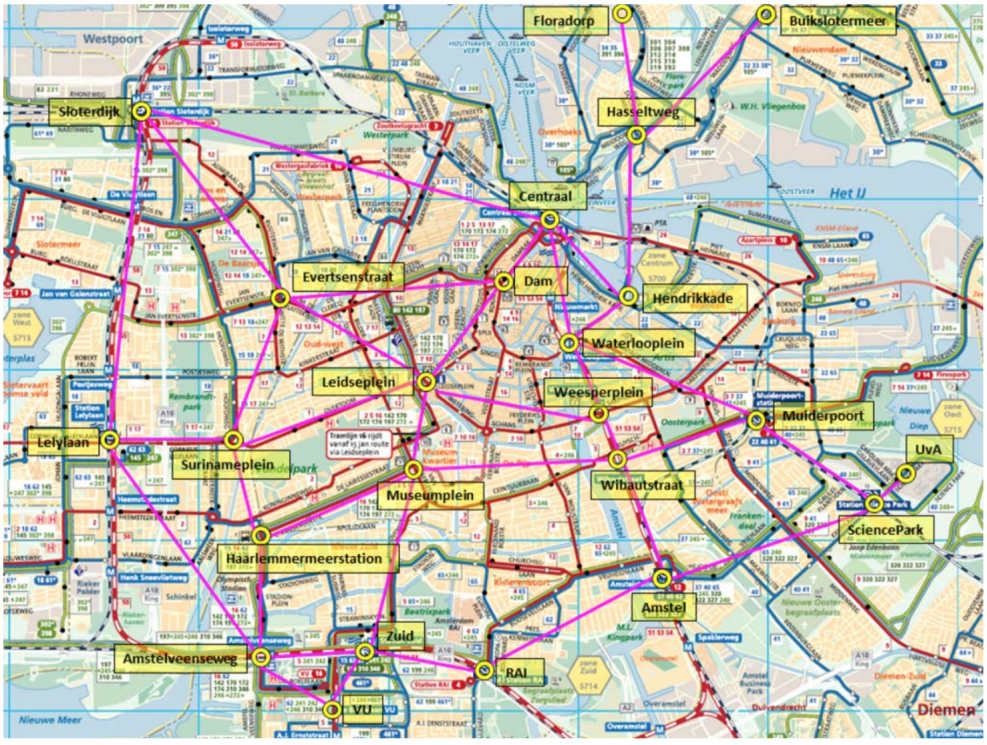

Figure 1: Simplified bus stops (yellow) and connections (pink) of Amsterdam.

3. Evaluation

The system will be evaluated based on three aspects (all after 'one day' of simulation):

1. The average travel time over all passengers
2. The total amount of money spent by the public transport company
3. The total number of messages sent.

Furthermore, at the end of simulation, each passenger not delivered at their destination gets a penalty of 180 ticks, which is added to calculation of average travel time.

All three criteria must be minimized to obtain a good solution, where the first (average travelling time) has the greatest influence on evaluation of final performance. To be able to evaluate the proposed model, the performance will be compared to a simple baseline model, in which buses randomly go to one of their connected bus stops, drop off all passengers that need to get off, and pick up as much passengers as possible given the capacity. For proposed model and the baseline model, the amount of buses will be equal to obtain a fair comparison.

4. Methods

This section describes the methods used to design a system of autonomous vehicles. The main feature for decision making is the calculation of a score for the next bus stop, based on short term and estimated long term goals. Furthermore, the claiming of passengers was added to improve this system (buses with the highest score can claim certain passengers). Thirdly, to handle peak moments, buses can vote for adding or not adding a new bus. Lastly, to facilitate these mechanisms, a communication protocol was created to handle communication between agents.

4.1. Decision making based on score

A score based decision making method is proposed, where the score gives measurement to the efficiency of travelling to a stop and picking up certain passenger groups. The highest score, given a specific passenger group (or combination of groups) at a specific bus stop, would be the most preferable bus stop to travel to and passengers to pick up. The computed scores are communicated with with other buses. All buses will compute scores for their connected bus stops and send a reply if they have a higher score. If the bus that sent the initial request has the highest score out for that specific passenger group (combination) at that specific bus stop, it will pick up these passenger groups. If another bus scores higher, the bus that sent the request will go to the bus stop with its second highest score.

The main goal is to minimize the passenger travelling time and the cost made by travelling, which is likely to be achieved by choosing the most efficient bus stops to travel to each time such a consideration needs to be made. To give some measurement to such decisions, some assumptions are made. The first assumption is that the more passengers are dropped off, the better, since they reach their destination and their travelling time stops increasing. Also, the more passengers are picked up the better, as it means that their travelling time is

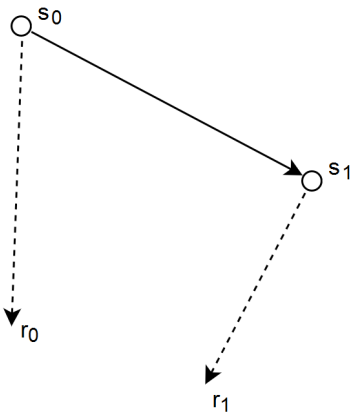

Figure 2: A vector representation of the current stop s_0, the next stop s_1, the estimated route of the passengers on board r_0 and the estimated route of the new passengers r_1.

now spent travelling instead of waiting at a stop. Another assumption made is that a new planned route is more efficient if it is similar to the original route, as the travelling time for the current bus passengers then has minimal change. Also, a major assumption made is that the route given the bus passengers can be estimated by taking the weighted average of their destination, which relies on the assumption that bus stops that are nearby are likely to be connected. This estimation is explained in section 4.1.1.

In figure 2, s_0 is the location of the current bus stop, s_1 the location of bus stop of arrival, r_0 is an estimation original planned route, given the current bus passengers (group g_0) and r_1 is an estimation if the the planned group g_1 is picked up. The estimated routes r_0 and r_1 are a weighted average of the passenger destinations.

4.1.1. Estimation of routes

The estimation of the route r from stop s with passenger groups g is made by

$$\vec{r}(s,g) = \frac{\sum_{p\in g} \|p\|(\vec{d_p} - s)}{\|\sum_{p\in g} \|p\|(\vec{d_p} - s)\|} \frac{\|\sum_{p\in g}(\vec{d_p} - s)\|}{\sum_{p\in g} 1}$$

, where d_p is the destination stop of passenger group p. The estimated route $\vec{r}$ uses as direction the weighted average of the destinations of the group passengers, with group size as weights. The length of $\vec{r}$ is the length of the average of destinations.

4.1.2. Score function

A score S for current bus stop s_0, next bus stop s_1 with passengers g_0 on board of the bus and passengers g_1 to be picked up can be given by

$$S(s_0, s_1, g_0, g_1) = \frac{1}{c_p}\left((1 + \sum_{p\in g_0}[d_p = s_1]) \frac{exp\left\{\frac{\vec{v}_{s_0 s_1}\vec{v}_{s_0 r_0}}{\|\vec{v}_{s_0 s_1}\|^2}\right\}}{\sqrt{\|\vec{v}_{s_0 s_1}\|}} + \|g_1\| \frac{exp\left\{\frac{\vec{v}_{s_1 r_0}\vec{v}_{s_1 r_1}}{\|\vec{v}_{s_1 r_1}\|^2}\right\}}{\sqrt{\|\vec{v}_{s_1 r_1}\|}}\right)$$

, where c_p is the cost per patch, d_p is the destination stop of passenger group p, $\vec{v}_{s_0 s_1}$ describes the route from stop s_0 to stop s_1, $\vec{v}_{s_0 r_1}$ describes the estimated route from s_0 with group g_0 on board, $\vec{v}_{s_1 r_0}$ describes the estimated route from s_1 with group g_0 on board and $\vec{v}_{s_1 r_1}$ describes the estimated route from s_1 with group g_1 on board. The first part of the function describes the favourability of driving towards the next stop s_1 and the second part describes of picking up passengers at the stop s_1.

4.1.3. Coordination through partial global planning

The decision making for the next stop could be seen as a form of *coordination through partial global planning*. As explained by Wooldridge (2009, p. 163-164), this consists of cooperating agents that inform each other to "reach common conclusions about the problem-solving process". In our setup, agents exchange their local plans / goals (the best score for a bus stop) to other agents, which determine whether their goals are common or dissimilar. If another agent has a higher score than the original sender, its local plan can be altered in order to better achieve its goals. Note that agents have the same long-term goals: minimizing average travel time, costs, and messages sent. The short-term goal however, is choosing the best bus stop to travel to, which is coordinated with other agents.

4.2. Power-set and heuristic search tree

Whenever a bus is located at a stop and needs to make a decision between next bus stops, for every connected stop the $score_s 1$ for that stop is calculated. The highest score corresponds with the most preferable next stop to travel to. This means that for every stop the power set of all passenger groups need to be considered, giving a worst-case complexity of 24!, which is impossible to compute. Therefore, after a certain threshold of amount of passenger groups, a heuristic search is performed instead. Table 2 shows the worst-case complexity for different thresholds; keep in mind that this is for one connected bus stop for one bus.

Threshold for power-set	Worst-case number of calculations
3	$3! = 6$
4	$4! = 24$
5	$5! = 120$
6	$6! = 720$
7	$7! = 5.040$
8	$8! = 40.320$
9	$9! = 362.880$
10	$10! = 3.628.800$

Table 2: Worst-case number of calculations for different thresholds for the power-set.

The heuristic search tree works in the following way. First the score for every individual group is calculated:

$$\{\{1\}, \{2\}, ..., \{24\}\} \text{ (24 scores)}$$

Then the maximum score is chosen, for example picking up the group with the second bus stop as destination (denoted as $\{2\}$) results in the highest score. Now we use this expand the possible passenger groups into combinations of passenger groups:

$$\{\{2, 1\}, \{2, 3\}, ..., \{2, 24\}\} \text{ (23 scores)}$$

Above, the combined group $\{2, 1\}$ represents a group that includes at least one passenger with bus stop destination 'one' and at least one with bus stop destination 'two'.

Again, the (combined) group with the maximum score is chosen. If this score is higher than the previous maximum, continue. Otherwise, stop and take the previous group as best score for this bus stop. This can be repeated for 24 iterations in the worst case, but probably stops earlier in the average case. The the worst case, the final calculation is that of all passenger groups combined:

$$\{\{1, 2, ..., 24\}\} \text{ (1 score)}$$

Using this method, only $24 + 23 + ... + 1 = 300$ calculations are needed in the worst case. In practice, only passenger groups that are present at a bus stop are taken into account. A disadvantage is that only a local optimum is found, there is no guarantee for a global optimum (which is guaranteed when computing the power set). Therefore, the assumption is made that the local optima are sufficient for a reasonable score.

For the final results and delivered code, the power-set was not used, due to the long runtime of the program. Nevertheless, the performance of the program did not visibly decrease, meaning the heuristic search tree method was sufficient for an estimate of the route.

4.3. Claiming passenger groups

To prevent buses from planning to pick up the same passengers or computing score for passengers that are already picked up by the time the bus arrives, the possibility to claim passengers is added. A bus may claim a passenger (group) when this bus is planning to pick up this passenger as a result of obtaining the highest score. Claiming a passenger ensures that this passenger is not taken into account when calculating scores for other buses.

4.4. Voting for new buses

Another mechanism is voting for new buses. Since the system is desired to be autonomous, the adding of buses should also be autonomous and not regulated by a central database. A system was implemented where buses can vote for or against adding a new bus. Note that this is a simple voting system with two outcomes: adding or not adding a bus. The (plurality) winner is the action that will be taken. Every 12 ticks, a voting takes place, where each bus votes for or against adding a new bus, by sending their preference to the first bus (the vote counter).

The vote is based on the 'weighted average filled capacity' (Abbr. WAFC), which is a score for how much of a bus' capacity is filled in the present and in the near past. At the beginning, the WAFC is set to zero. Then, at every tick, the current filled capacity is calculated: the amount of passengers in the bus divided by its capacity. The WAFC is then updated by taking 75% of the previous WAFC and adding 25% the current filled capacity. This way, distant past measures almost don't count, while near past measures have a high count. The 75%-25% distribution is again based on empirical research (weighing the current measure too high, it becomes too sensitive to filled capacity, while weighing it too low, it becomes too unresponsive to changes in capacity). At each voting, if the WAFC is 0.5 or higher, the preference will be to add a bus, and otherwise to not add a bus.

If 50% or more of the buses prefer a bus to be added, a bus will be added. Furthermore, the amount of ticks for the next voting is set to 6. If less than 50% of the buses prefer a bus to be added, no bus will be added and the amount of ticks for the next voting is set to 12. This is done to increase the frequency of votings in peak hours, because buses might be needed quickly, while in quiet hours votings aren't really necessary.

Finally, it might not be wise to add hundreds of buses: each bus in the simulation must remain there until the end and cannot be removed from the simulation. Adding many buses results in high lease costs, so the amount of buses should be reasonable. This is one of the reasons votings are not held every tick and only one bus is added per voting.

4.5. Communication protocol

For communication between buses, a very simple communication protocol was created, as shown in table 3. This protocol is inspired by the FIPA Performatives (FIPA, 2000; Wooldridge, 2009, p. 140-144), and mainly works by sending and requesting information, some of which is used for negotiation. The table shows the name of the form of communication, the description, and the type (at least one of the following four: passing info, requesting info, negotiating, and performing action). These methods of communication are used for all previous mechanisms, such as negotiating about scores, claiming passengers, and voting.

Name	Description	Type(s)
inform-cleared-claims	After calculating all claimed passengers that are already picked up, inform all buses about the updated claim list.	*Passing info*
inform-init	Inform all buses about its existence, so they can update their local variable *buses_count* (followed by request-buses-amount and request-claims).	*Passing info*
inform-score	Inform nearby buses about its maximum score for a given *busstop* and *passenger_group* (followed by request-score) OR answer a request-score from a bus.	*Passing info, negotiating*
inform-status	Inform buses about *current_stop*, *next_stop*, and *bus_passengers*.	*Passing info*
inform-vote	Inform the vote counter (the first bus) about its vote (adding or not adding a bus) maximum.	*Passing info, negotiating*
request-buses-amount	When a new bus comes into play, request the total *buses_amount* from one bus.	*Requesting info*
request-claims	When a new bus comes into play, request the list of *claimed_passengers* from one bus.	*Requesting info*
request-score	Request the informed buses about whether their score is higher than the given score for the given *busstop* and *passenger_group*.	*Requesting info, negotiating*

Table 3: Communication protocol for the Multi-Agent System.

5. Results

In this section, the results of simulating the proposed model and baseline model are discussed. Table 4 shows a comparison of both models after running one simulation day, with different parameters.

As can be seen in Figure 3, the average traveling time rises until roughly 05:00 AM, followed by a drop. After 08:00 AM the average traveling time sharply rises, as does the bus' expenses and the messages sent. This rise can be explained by the explosive growth of travelers from 07:00 AM onward. The bus' expenses increase incrementally and the messages sent increase exponentially with a linear increase in buses. From the graph 'number of passengers waiting', it becomes clear that the buses cannot cope with the stress created by a flood of passengers. After 10 AM the situation stabilizes, and the number of passengers waiting decreases slowly. This figure shows a scenario with 20 initialized buses, runs with different parameters yielded similar graphs.

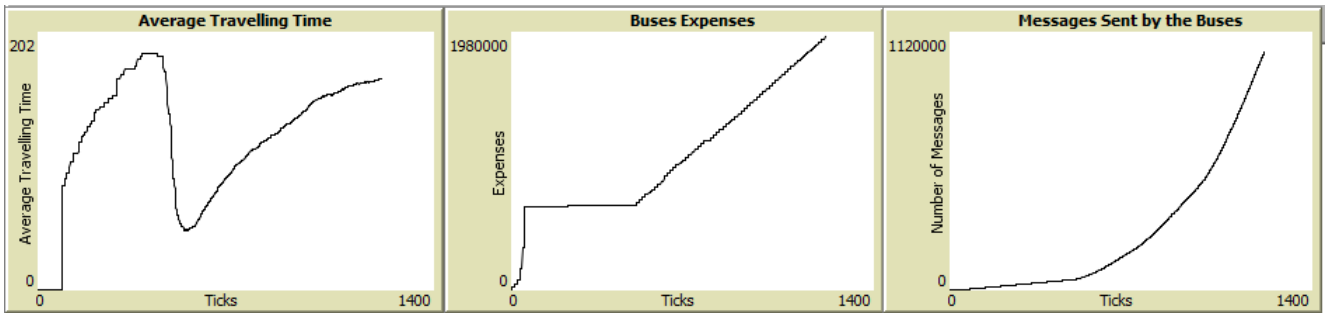

Figure 3: Various statistics of a simulation of one day(file:passengers-day1.csv), with 20 initialized buses.

	10 buses	**15 buses**	**20 buses**
Average travel time	186	187	166
Total expenses	1.508.694	1.633.242	1.952.040
Total number of messages	537.023	636.861	1.038.884
Num passengers waiting for bus	8139	6277	3565
Avg travelling time remaining	226	250	275
Final Average travel time	205	215	210
Final tick	1140	1200	1217

Table 4: Comparison of three evaluation criteria: 1) Average travel time over all passengers, 2) the total amount of money spent by the public transport company, and 3) the total number of messages sent, for both models. These results are measured after the given amount of final ticks, given the test data, using three different initializations (10 buses, 15 buses, and 20 buses).

6. Conclusion

A multi-agent system with intelligent vehicles which perform both autonomous actions and group decisions was successfully implemented. Route planning has been implemented by estimating scores for picking up groups of passengers at possible stops and communicating this score with other vehicles. Here, scores are based on short-term and estimated long-term goals. Vehicle behaviour is regarded intelligent because the buses cooperate by making group decisions through communication of scores, and secondly through the implementation of a voting system to add new buses.

It becomes clear in the previous section that there is room for improvement of the system. During peak hours the amount of travelers waiting to be picked up rapidly increases, sequentially the average traveling time increases.

7. Discussion and Future work

A multi-agent system has the potential to realize a system that operates in real-time. However, a different set of programming skills is required in a multi-agent modeling environment. Time complexity becomes increasingly important when many agents have to perform complex calculations repeatedly. Both the heuristic search tree and the voting system became rather time-consuming processes as these became more sophisticated.

A strong point of the system was the cooperative behaviour of the buses. When a bus travels to a bus stop to pick up a portion of the passengers, other buses register these passengers as claimed passengers. Buses react to stress in the system by hosting voting rounds more often when a new bus comes into the simulation.

Parameter tuning was limited because of the slow running time. The main parameters tweaked were the amount of buses into play and the use of power sets. One limitation of the current implementation is the inability to handle the high amount of traffic during peak hours. Efforts were made to solve this problem. For example, a maximum amount of destinations per bus was implemented, picking up of all passengers at a stop, combining smaller with larger buses, and emptying buses partially when full before picking up new travelers. However, these changes were made to no avail. Possibly, implementing transferring of passengers into the system might help solve this problem.

In order to reduce the amount of messages sent, messages could only be sent to buses that are nearby. In the present version, messages are sent to every other bus.

The current scoring method uses a weighted average of the destinations of passenger groups to estimate a route. This may be problematic when destinations are each others opposites with respect to some origin, as they cancel each other out, resulting in an estimated route of a length approximate to zero. In this case, the bus will travel to the nearest bus stop as this decision causes minimum cost. It seems that this results in the bus getting 'stuck' between two stops, going back and forth, which is undesirable behaviour as it increases the expenses and travelling time, while not picking up nor dropping any passengers off. What would be desirable behaviour is for the bus to choose one direction of the passenger destinations to drop them off, which could be an improvement on the score function.

References

FIPA Communicative Act Library Specification. (2000). Foundation for Intelligent Physical Agents - `http://www.fipa.org/`.

Wooldridge, M. (2009). An introduction to multiagent systems, p. 140-144, p. 163-164. John Wiley & Sons.