Multi-Agent System - Final Week

Group 32

Ya-Chu Yang 11571276
Jiun-Han Chen 11649712
Michael van der Werve 10565256

Abstract

This report is written for final project of the course Multi-Agent System of University of Amsterdam(UvA). In this project, we must design a multi-agent system to handle daily bus transport requirement and simulate the bus transport system of Amsterdam with simulation software NetLogo. We use **Floyd-Warshall algorithm** to calculate shortest path between bus stops, **BDI framework** to model what bus should do, **communication between agents** to coordinate, **dynamic route** to carry passengers, group decisions with **Plurality** and **Borda Count** to decide number of buses and **negotiation** to swap passengers.

Keywords

Floyd-Warshall algorithm, BDI framework, communication between agents, dynamic route, coordination, Plurality, Borda Count, Negotiation.

1 Introduction

1.1 Define Question

In this project, we are given bus map of Amsterdam, connections between stops, distance between bus stops, passenger flow of one day as shown in Figure 1. Our goal is to design a multi-agent system and also bus transport system to carry all the passengers to their destinations. But in the same time, we must try to **minimize the cost expense**, **average traveling time of passengers** and **number of messages exchange between buses**. The cost expense is determined by Table 1

Table 1: The cost expense

vehicle type	capacity	lease cost	cost per patch
small	12	6000	1
medium	60	12000	1.5
large	150	20000	2

There are some limitations, firstly, buses must communicate through function "send-message" to create the exchange of messages, we can't use global variables to do that. Secondly, buses only can move to connected bus stops.

We must handle one problem, the flow of passengers is not fixed. So we can't give a deterministic solution of this problem. The basic idea of our design is to provide a greedy solution based on shortest path, and the shortest path can be calculated by destinations of passengers on the bus. With this main greedy solution, we improved our solution with communication, group decision and negotiation.

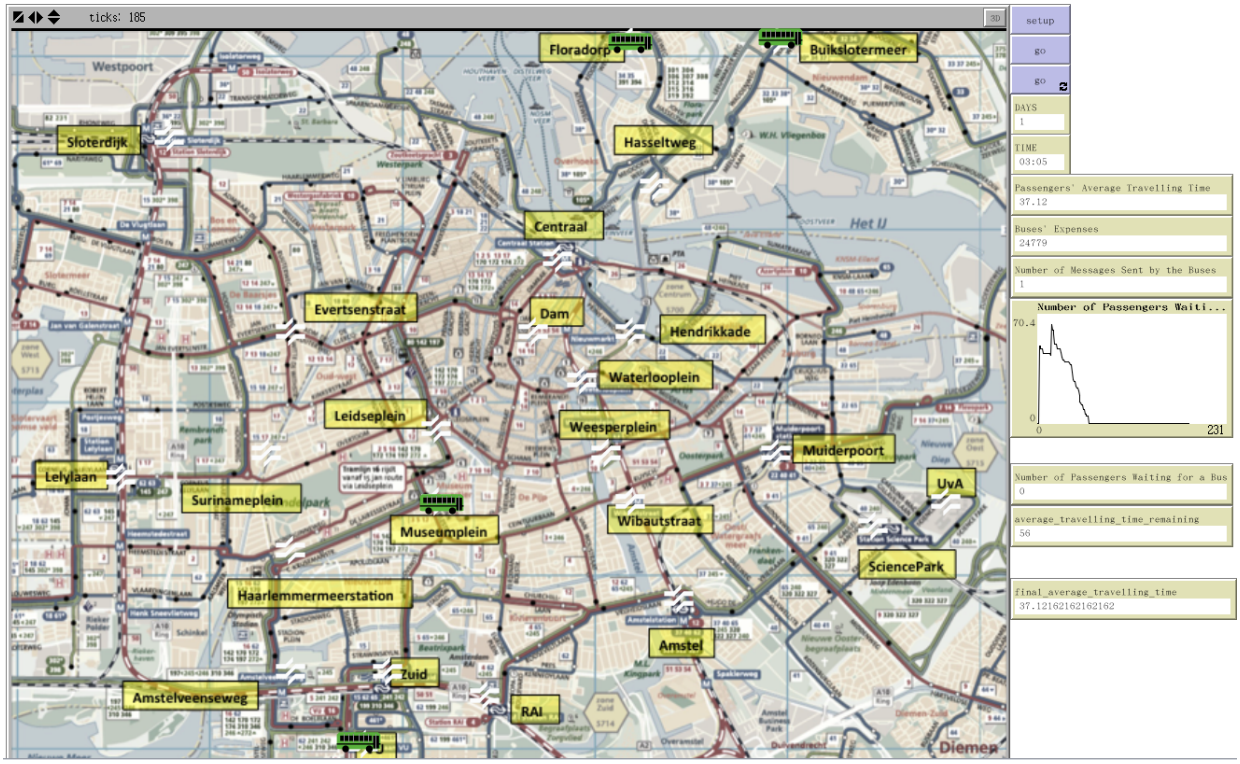

Figure 1: Given map of Amsterdam bus stops

2 Detail of implementation

2.1 Shortest Path

We implemented Floyd-Warshall algorithm to calculate the shortest path between two unconnected bus stops. Each bus initializes its shortest path matrix, which can be precomputed[1], and save this matrix to variable "Floyd_path". So when a bus wants to go to a specific stop, the bus can calculate shortest path between current stop and destination stop with function "find_path". This implementation is the basic of our greedy solution.

2.2 BDI framework

A vehicle will have the desire to travel to bus stops according to the dynamic route, pick up as more passengers as possible, and drop passengers to destinations. The belief of a bus is the information of bus stops and passengers waiting at bus stops. A bus has two type of intention, one is "PICK_UP", if there is no passenger on the bus, another is "RUN" which is trigger when there are some passengers on the bus.

2.3 Dynamic Route

To make buses coordinate with each other, dynamic route is easier to achieve our goal than fixed schedule. So we design our dynamic route according to destination of each passenger on the bus. If there is no passenger on the bus, the bus will find out the nearest stop with passengers. If there is passenger on the bus, the bus will pick its destination with shortest path, according to all destination stops of passengers on the bus. So all the buses will coordinate together to achieve the goal which is carrying all the passenger to there destination. To this part, our basic greedy solution is completed.

2.4 Communication Between Agents

Our basic idea of communication is simple, the buses must exchange the unknown properties between buses. We try to list all the unknown properties which are passengers picked by which bus, the locations of buses, the destinations of buses and the size of buses.

We define some tags to exchange important information:

Table 2: Self-defined Protocol and Messages

Tags	Description	Format
pick_up	Which bus picks who. Information exchange can be triggered only when a bus pick some passengers.	pick_up bus_id passenger_ids
drop_off	Which bus drop off who. Information exchange can be triggered only when a bus drop off some passengers.	drop_off bus_id passenger_ids
target_stop(*)	Destination stop of the bus. Information exchange can be triggered only when a bus change its destination.	target_stop bus_id stop_id
bus_size	Size of the bus. Information exchange can be triggered only when a bus is created.	bus_id bus_size (1=small, 2=medium, 3=large)

(*) In final project, we only use target_stop to minimize number of messages.

Our buses won't go to the same stop to pick up the passenger. To achieve this strategy, each bus will tell the destination stop to other buses via communication. We used "target_stop" tag to exchange this information. When a bus is planing its route dynamically, with the "target_stop" tag, the bus will be able to read the information from variable "inbox" to avoid from moving to the same stop for picking passenger.

2.5 Group Decision

When should we add new bus (Plurality)

The timing that we should add new bus is when the buses are close to unable to handle the transportation problem. The passenger loading rate $p \geq 50\%$ is how we define the transportation situation beyond the ability of buses can handle. So each bus as a voter, if the passenger loading rate of a bus $p \geq 50\%$, the bus will vote "Yes" on adding new bus, otherwise, it will vote "No". After each voter submitting their preferred options, the voting scheme, plurality voting, will come out a winner, a decision describing whether adding a new bus.

What's the size of new bus (Borda Count)

After deciding to add a new bus from the previous step, borda count, in which each bus stop are voters, is used here to decide the size the of the new bus. We simply assume that the three options, adding small, medium, and large vehicles, are ranked in four kinds of order of preference:

- small>medium>large (number of waiting passengers at a bus stop < 6)
- medium>small>large (6 $\leq$ number of waiting passengers at a bus stop $\leq$ 12)
- medium>large>small (12 < number of waiting passengers at a bus stop $\leq$ 24)
- large>medium>small (number of waiting passengers at a bus stop > 24)

The other two orders are meaningless. Because we can't define a continuous range to include these two situations. The bus stop votes on the order of preference according to which range is the number of waiting passengers on that bus stop falling within (see the descriptions inside the parentheses above). Thus, the borda count will determine a winner, deciding what's the size of the new bus.

2.6 Negotiation

Negotiation Timing, Topic, and Target

The timing that negotiation only happens when two buses meet each other at the same station. If there are more than two buses in the same stop. There will be multiple negotiations with pair buses. The topic that buses negotiate about is "Should buses swap their passengers?". And the target of each bus that bus wants to achieve is to minimize the total route distance with the same number of passengers.

Procedure of Negotiation

When buses meet each other at the same stop, the negotiation will start. For example Bus_A has passenger A = [5, 6, 7, 8] and Bus_B has passenger B = [9, 10, 11, 12, 13], the numbers in the array are passengers' id.

- Step 1. Swap passengers with index i, which are 5 and 9 if i equals to 1.
- Step 2. Both buses calculate the route distance change after swapping with shortest path algorithm.
- Step 3. If one of bus thinks the swap is not good. The bus is able to cancel the deal and then swap back.
- Step 4. Back to step 1 with index i + 1, until index > min(sizeof(A), sizeof(B)).

We expect that, with this strategy, the destination of passengers of one bus should be more coherent to lower the average traveling time and expense. The reason we don't swap passenger with different index(e.g. passenger 7 and 12) is because that bus picked passengers in order, if we swap some passengers to lower priority to be transported, which are not fair to those passengers.

3 Experiment Result

3.1 Tuned parameters

In our design, there are some parameters can be tuned to get the different results. Here we list some of important parameters:

- MAX_NUM_BUSES: There is a limitation number of maximum number of buses. If we set this number higher, we can get lower average travel time, but higher cost expense. Multiple buses can be added during one tick. So it might excess MAX_NUM_BUSES.
- CAPACITY_VOTE: CAPACITY_VOTE_S, CAPACITY_VOTE_M and CAPACITY_VOTE_L, these parameter determine whether a bus votes yes for the question: "Should we add a new bus?" The bus will check its number of passengers, if number of passenger is larger than this parameter, than the bus votes a yes.
- RANGE_BORDA_COUNT: s_limit, m_limit, and l_limit, these parameters determine how does a bus rank its preference for the size of new added bus. They define each bus stops should be classified to which preference for Borda Count.

We consider "passengers-location_day1.csv" as our validation set to tune the parameters, and consider "passengers-location_day4.csv" and "passengers-location_day5.csv" as our test set. After tuning parameters, we get the following parameters shown in Table 3 which consider minimize the cost expense, average traveling time of passengers and number of messages exchange between buses:

Parameter	Value
MAX_NUM_BUSES	16
CAPACITY_VOTE_S	5
CAPACITY_VOTE_M	30
CAPACITY_VOTE_L	75
s_limit	6
m_limit	12
l_limit	24

Table 3: Tuned parameters according to day 1

3.2 Final Performance

We represent the experiment result of day 1, day 4 and day 5 in the following Table 4:

	Day 1	Day 4	Day 5	Average
Average Traveling Time	3.033	2.303	1.408	2.248
Buses' Expense	418451	735926	913848	689408
Number of messages	4440	11521	16808	10923
Number of small buses	2	12	3	5.67
Number of median buses	14	8	18	13.33
Number of large buses	0	0	8	2.67
Total number of buses(*)	16	20	29	21.67

Table 4: Experiment result
(*) Multiple buses can be added during one tick. So it might excess MAX_NUM_BUSES.

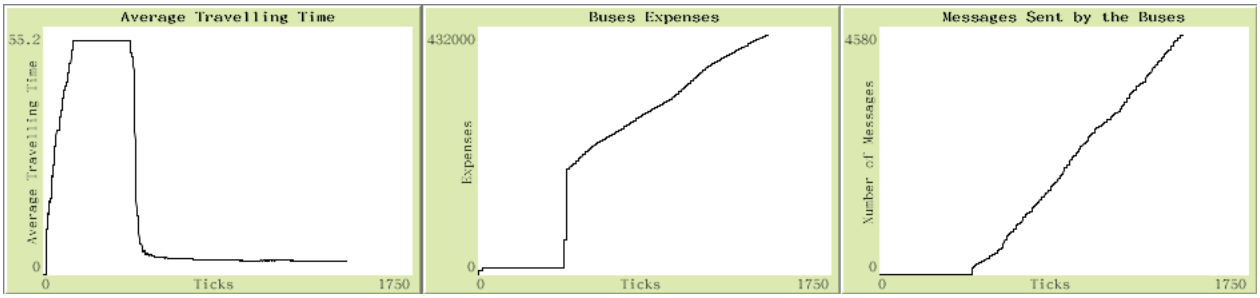

Figure 2: day 1 graph

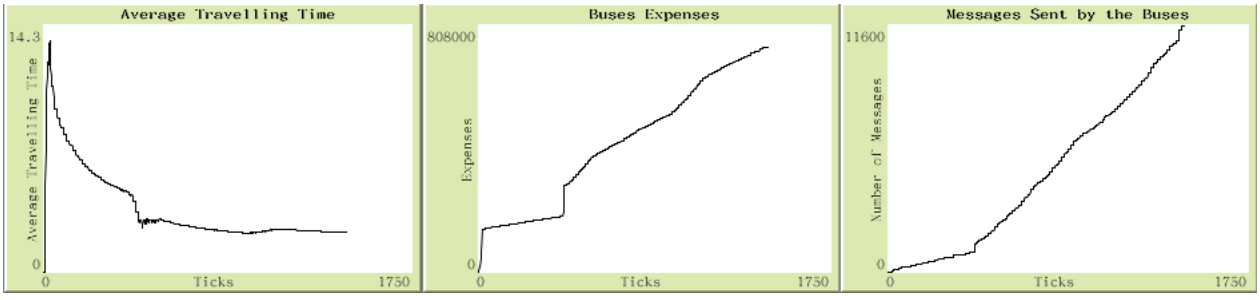

Figure 3: day 4 graph

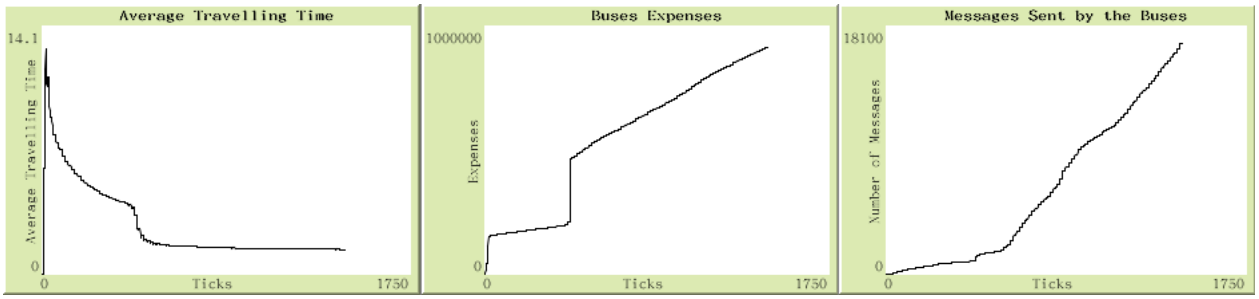

Figure 4: day 5 graph

In the final version, our average of average traveling time is 2.248 minutes, buses' expense is 689408 and number of messages is 10923. In general, if we use more buses, we get lower average traveling time, but higher buses' expense. We can also draw the same conclusion for we set different MAX_NUM_BUSES for data of Day 1. Day4 and Day5 is similar data, according to Figure 3 and Figure 4, they both have more passengers at early stage, so they got similar graphs.

3.3 Improvement of different strategies

According to data of previous weeks, we are able to check the improvement of each strategy. We used fixed number of buses and only use small bus before week 5.

Table 5: Final average traveling time of different strategy

Strategy	Final average traveling time
Fixed Schedule (week2)	25.842
Dynamic route (week4)	1.268
Coordinate with communication(week4)	1.267
Group decision(week 5)	0.760
Negotiation(week 6)	0.626

The lower average traveling time is based on high number of buses. As you can see, the greedy strategy(dynamic route) played the most important part of this multi-agent system. It provides a basic solution of this problem. And group decision to decide the appropriate timing to add bus and size of new bus, which also got a good improvement. With group decision, it became easier to decide the timing dynamically and adapt to different situations.

4 Conclusion

To solve this bus transport system problem, the dynamic route with greedy strategy has huge improvement with relative to fixed schedule. Group decision is able to help us dynamically decide different parameters to fit complex and undetermined situation. Negotiation between agents is able to maximize the reward of each individual but also improves the reward of whole group. Communication between agents can uncover unknown information, further agents are able to make better decision. We combined all these techniques to provide a appropriate solution of this problem.

In general case, to solve a complex multi-agent problem, we can apply greedy strategy to each agent to get an appropriate solution, and then based on this solution, using various techniques that we learned from multi-agent system, like group decision, negotiation and communication to improve this greedy strategy. Make the greedy strategy closer to optimal solution with some simple rules for each agent.

References

[1] Chandler Burfield. Floyd-warshall algorithm. *Massachusetts Institute of Technology*, 2013.