Multi-Agent systems - Assignment 3 (Communication)

Emma Machielse, Federico Tavella, Alessandro Tezza

February 25, 2018

1 Overview

During the previous week, we implemented the first version of the transportation system, in which the agents were moving according to a fixed schedule. The firstly created agent was responsible of creating new buses based on the amount of people waiting at the bus stops and the total capacity of the bus fleet. This week, we focus on improving the first version implementing a communication system between agents.

2 Improvements

- **Different fixed schedules:** the buses created with an even ID are moving according the schedule that we defined last week, whereas the bus created with an odd ID are moving through the *reverse version* of the same schedule;
- **Direction-based pick up:** each bus picks up only the passengers whose destination is closer in its schedule compared to the reverse one. If this is not the case, the passengers are waiting for a bus moving in the opposite direction. In this way, we try to minimize the travelling time of the passengers. The waiting time is not heavily affected by this choice thanks to the similar distribution of the buses in the two directions;
- **Roles:** every bus can have one or more roles. The roles define the *behaviour* of a bus. In this way, we can easily change the behaviour of a bus by changing its role (promotion/demotion). We define the following roles: (i) **Coordinator:** a special bus that is allowed to create new buses. At the moment, only the first created bus is a coordinator; (ii) **Scout:** a bus that is always moving through the schedule. It is not allowed to stop and wait; (iii) **Simple:** a bus with no special aims.
- **Bus stop-and-wait**: once a bus has delivered every passengers and reaches an empty bus stop - if its role permits it - it does not move anymore. This is useful when the passengers waiting for a bus are not a lot and the situation can be managed by other buses. In this way, we can decrease the travelling costs.

3 Communication

We implement the communication as follows. Every time a bus reaches a bus stop, it checks the incoming messages. It can retrieve the new messages from the inbox history comparing the ticks of the messages with a *locally stored variable* that represents the last time (i.e., last tick) in which the agent checked the messages. The content of a message is a string that represents an action to do or an information request. At the moment, the *coordinator* (that locally keeps track of the created buses and their capacity) can send a "promotion" message to the other buses: every *simple* bus that receives it becomes a *scout*. In this way, the *coordinator* can check for critic situations using its own local information and manage them ordering to the other buses to never stop (i.e., to be *scouts*). Once the danger situation has been managed, the *coordinator* can send other messages to remove the demote the buses. Consequently, they are allowed to stop and wait (i.e., their role change from *scout* to *simple*).