北航中法工程师学院
Centrale Pékin

CentraleSupélec

SystemX
INSTITUT DE RECHERCHE TECHNOLOGIQUE

AnthroPOLIS
HUMAN CENTERED MOBILITY

Anthropolis Chair and Future Cities Lab

Joint Seminar Series 2021-2022

September 29th, 2021

Anthropolis Chair and Future Cities Lab Joint Seminar Series 2021-2022

- 14 seminars during this 2nd edition
- Presentations from the Anthropolis Chair, the Future Cities Lab and more
- Full programme available on our website www.chaire-anthropolis.fr
- Subscribe to our mailing list to stay informed about our events
- We are also on twitter: @CAnthropolis
- Stay tuned with the Future Cities Lab : www.futurecitieslab.city

04/10/2021

Session 1: Anthropolis Chair and Future Cities Lab Joint Seminar

Implementing and Using Reinforcement Learning for Empty Empty Shared Vehicle Rebalancing in Multi-Agent Transport Simulation MATSim

Presented by

Tarek CHOUAKI
PhD Student
tarek.chouaki@irt-systemx.fr

Jakob Puchinger
Supervisor

Sebastian Horl
Co-Supervisor

Please welcome AMoD

- The advent of autonomous vehicles offer a whole range of new opportunities
- Autonomous Mobility on-Demand promises to improve transportation quality for users and fleet usage efficiency for operators

But we should not forget that:

- Autonomous on-demand vehicles are not arriving in an empty space
- They have to efficiently collaborate with other services to achieve an attractive overall transportation offer
- The manner in which an autonomous on-demand vehicles fleet is operated strongly impacts its performance

Challenges to tackle for AMoD

- Adaptiveness
  - The mobility context is highly dynamic and constantly evolving
  - The offer of mobility can drastically change in a short period of time (before and after the Covid-19 crisis for example)
- Proactivity
  - An AMoD system can rely on past experience to predict future demand and act beforehand in order to better satisfy transportation queries
- Integration
  - An AmoD system interacts, directly or indirectly, with other mobility systems. A good integration of these different services together is crucial for overall performance

Potential of using Reinforcement Learning for operating AMoD

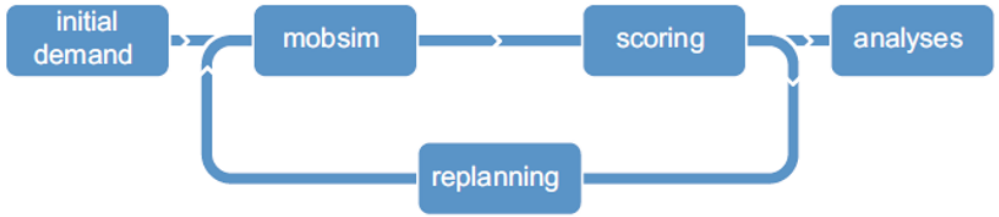
- A system that continuously learns from the consequences of its decisions can adapt to changes in a situation and act in a proactive way. And if these actions are evaluated regarding the overall mobility performance, a good integration of AMoD in its environment can be achieved.
- Hence our PhD topic : **Reinforcement Learning and Stochastic Optimization for the design of on-demand mobility services by simulation**

About the PhD subject

- Scientific Goals
  - Design on-demand mobility services and simulate them in combination with classical public transports using MATSim
  - Use techniques of decision under uncertainty to operate such a service in an efficient manner
  - Take into account the infrastructure supporting the service; i.e the charging stations and the parking areas
- Our Practical Objective
  - Arrive to a simulation of the future Paris-Saclay area (horizon 2030) mixing existing and future public transports alongside AMoD services with the infrastructure supporting it
  - Be able to forecast the impact of the upcoming Grand Paris Express
- Also involves :
  - Develop the required MATSim modules to simulate such a service
  - Implement scenarios on various other regions.
  - Maintain the reproducibility of the simulations

Agent-Based simulation of mobility using MATSim

- MATSim is an open-source activity based, multi-agent simulation framework implemented in JAVA
- It simulates each agent individually, i.e microscopic
- It uses a queue based model for travelling through links of the network
- MATSim performs iterations of the simulation applying the co-evolutionary principle : each agent tries to optimize its schedule while competing with other agents
- The agents changing their plans to maximize individual scores allows us to assess service impacts on modal shares

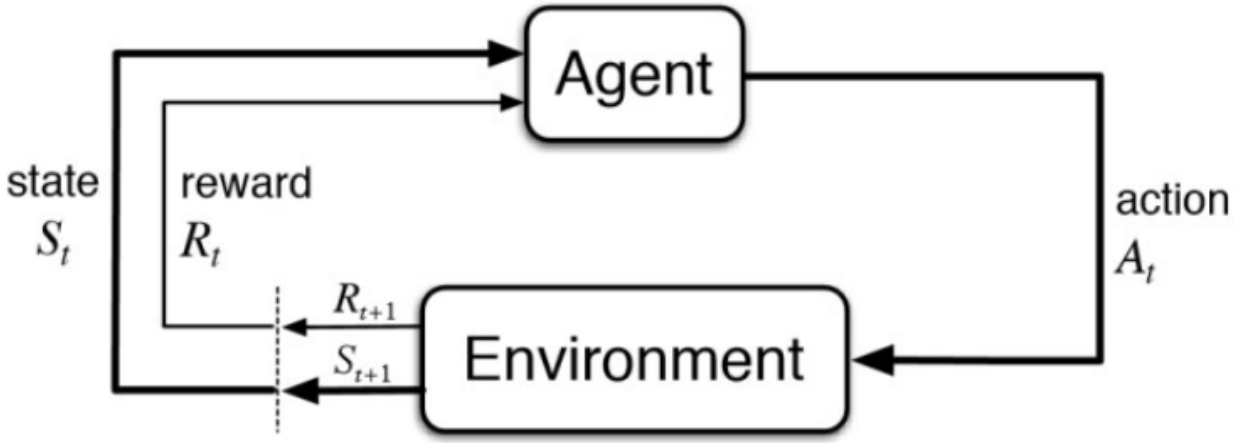


State of the art

- MATSim provides various rebalancing algorithms
- One tries to estimate the next demand [1]
  - Modelling the rebalancing problem as a transportation problem
  - Estimating the number of requests during the next time slot from the same time slot of previous iterations
- Existing studies use RL for rebalancing & dispatch in simulation
  - The tools used for the simulation often do not allow to assess the impact on travelers' choices and modal shares (**[2][3]**)
  - Some of the studies are performed on built-in simulators and are hardly **reproducible** (**[3]**)
  - **[4]** Uses a centralized RL approach for empty vehicle rebalancing
  - Electric vehicles and **recharging** decisions are not considered
- Our study
  - Introduces a **first decentralized RL** algorithm for empty vehicle rebalancing in **MATSim**
  - Is fully **reproducible**
  - Serves as a foundation for next studies

Reinforcement learning

- The considered agent(s)
  - The agent has a set A of possible actions
  - A value function V indicating the desirability of being in a given state
  - A policy P mapping environment's states to actions
  - A model of the reinforcement (how the agent learns)
- The environment in which the agent evolves
  - The environment has a set S of possible state
  - The environment can be fully or partially observable by the agent(s)
  - It can be static (changes only when agent performs an action) or dynamic
  - After each action, the agent receives a reward value by the environment



How to implement Reinforcement learning in MATSim

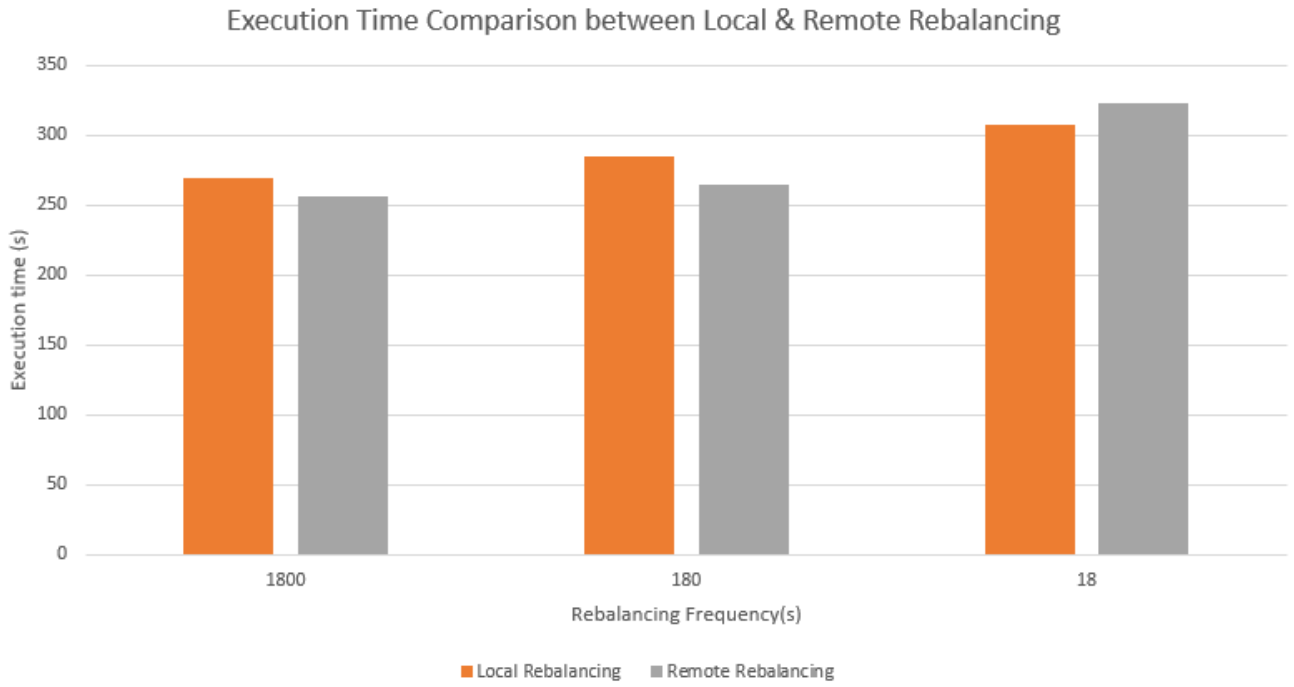
- First implement a rebalancing strategy that uses RL
- Rebalancing is usually performed periodically considering discrete zones of the network
- Exploring how rebalancing strategies are implemented in MATSim's DRT module
- An existing functionality estimates the next demand from the previous iteration
- Technical questions that arise:
    - How to position the reinforcement learning loop relatively to the simulation loop ?
        - Apply reinforcement after each MATSim iteration
        - Apply reinforcement after iterating in MATSim and then simulate again
    - Where to implement the RL algorithm ?
        - Directly inside the DRT module, in JAVA
        - As an external API that is called by the DRT module

How to implement Reinforcement learning in MATSim

- First implement a rebalancing strategy that uses RL
- Rebalancing is usually performed periodically considering discrete zones of the network
- Exploring how rebalancing strategies are implemented in MATSim's DRT module
- An existing functionality estimates the next demand from the previous iteration
- Technical questions that arise:
    - How to position the reinforcement learning loop relatively to the simulation loop ?
        - Apply reinforcement after each MATSim iteration
        - ~~Apply reinforcement after iterating in MATSim and then simulate again~~
    - Where to implement the RL algorithm ?
        - ~~Directly inside the DRT module, in JAVA~~
        - As an external API that is called by the DRT module

Remote Rebalancer

- An external API to be requested by MATSim's DRT module
  - A **Remote Rebalancer** tool written in Python
  - Communication with MATSim through TCP requests
    - Allows to have the rebalancing part running in another machine/server
    - Allows to test various algorithms without the need to touch MATSim's code
  - Is it reasonable to perform the rebalancing out of MATSim and have it wait for answers ?
    - The benchmarks say yes !



A first RL algorithm in MATSim:

- We implemented a Q-Learning rebalancing algorithm
  - Decentralized : Each vehicle learns on its own
  - An agent's state is comprised of:
    - Its discretized position on the road network (current zone)
    - The discretized current time
  - The action space corresponds to the zones that the network is discretized to
    - A vehicle chooses one zone to rebalance to and moves to its centroid
  - The reward received by a vehicle is the number of passengers since last rebalancing
  - After each action **a** from state **s** leading to state **s'**, the vehicle updates its Q-Table based on the reward **r**

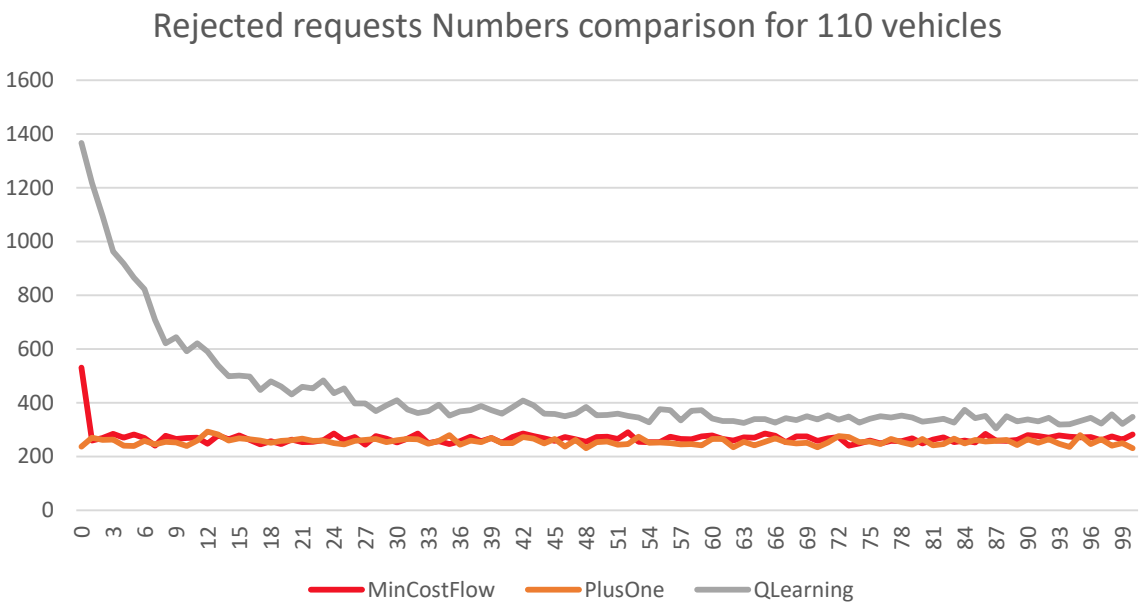
$$Q[s,a] \leftarrow (1-\alpha)Q[s,a] + \alpha(r + \gamma \max_{a'} Q[s',a'])$$

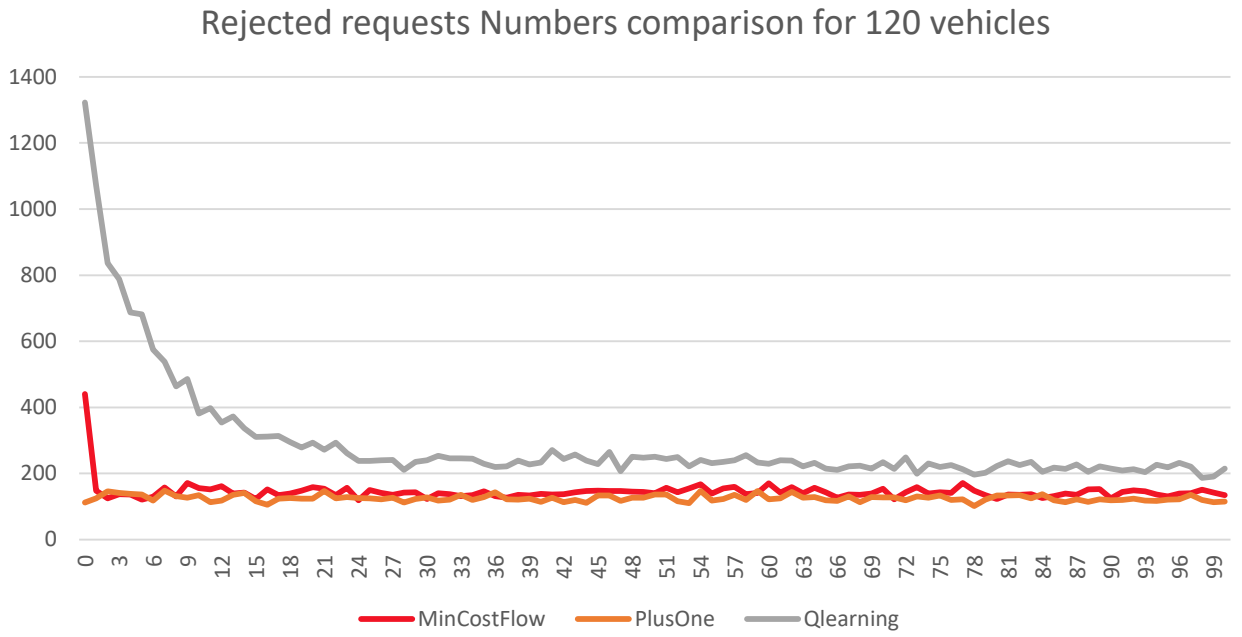
  - At each step, there a probability that the vehicle choses a random action (ε-greedy)
- The implementation was done as an external software MATSim calls through TCP requests

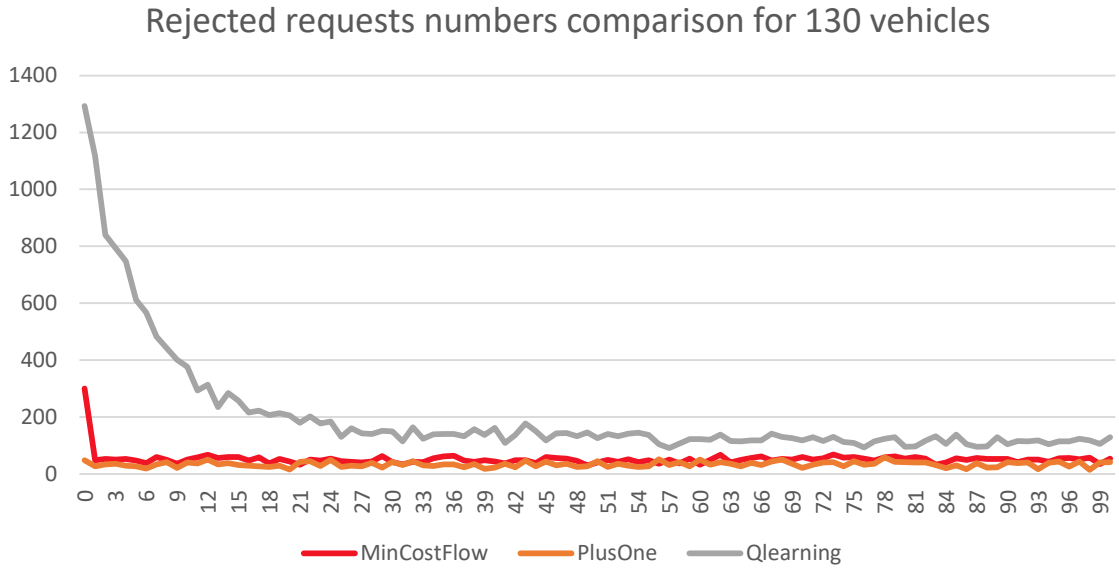
Our experiments:

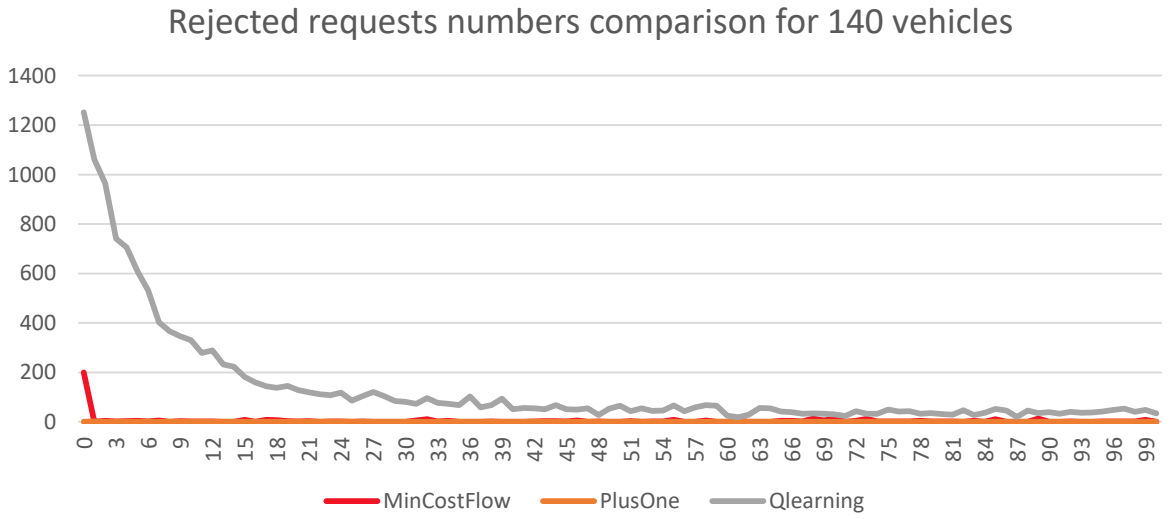
- Tested on an open MATSim Scenario (Cottbus)
  - The replanning of the agents was disabled
- Alongside MATSim's default vehicle assignment
  - A request is rejected if the expected max wait time for the users is over a certain threshold
  - The objective is then twofold : reduce waiting times and the number of rejected requests
- First tests of the QLearning algorithm with 200 vehicles
- We tested the rebalancing algorithms already present (MinCostFlow and PlusOne) in MATSim with various fleet sizes until they showed rejected requests
- We then tested the QLearning algorithm with different values for the parameters (alpha, gamma and epsilon) and different fleet sizes and compared the best settings

Our results:

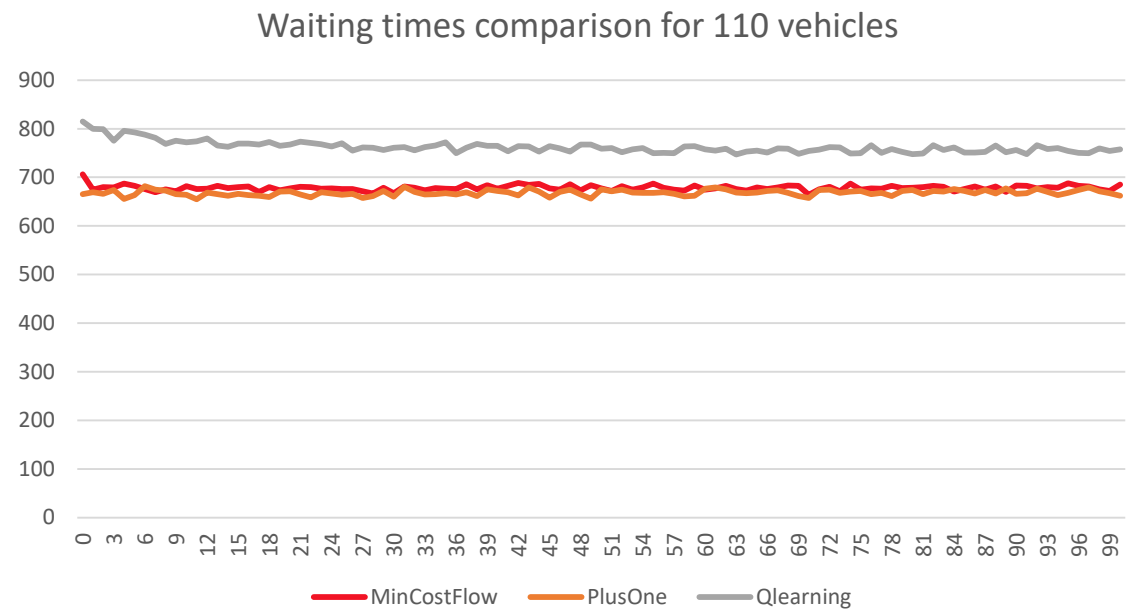

Rejected requests Numbers comparison for 110 vehicles
1600
1400
1200
1000
800
600
400
200
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
QLearning

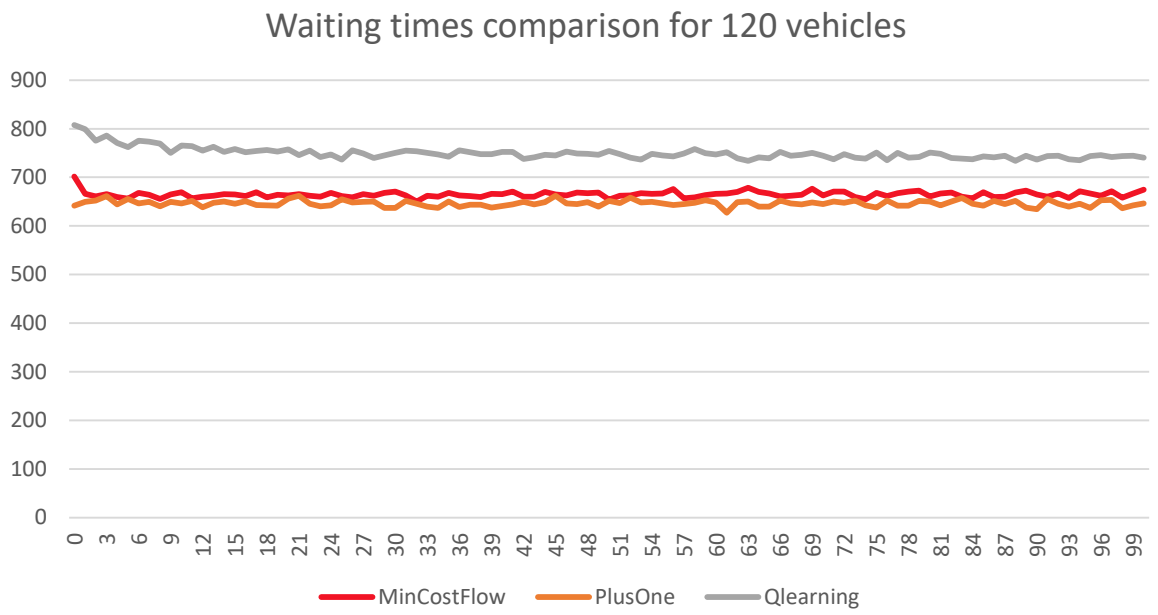

Rejected requests Numbers comparison for 120 vehicles
1400
1200
1000
800
600
400
200
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning

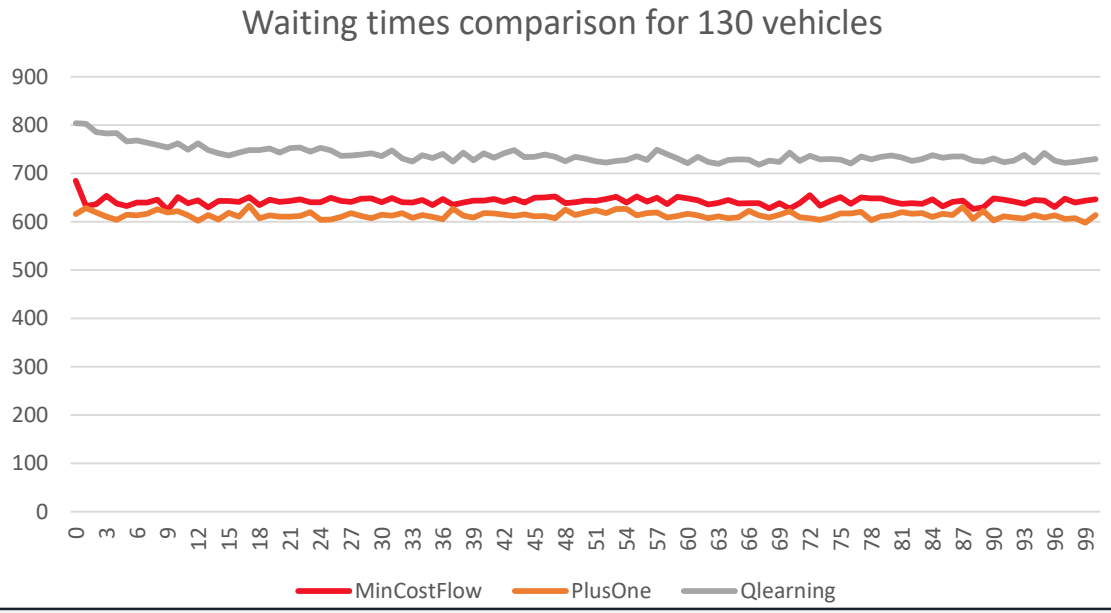

Rejected requests numbers comparison for 130 vehicles
1400
1200
1000
800
600
400
200
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning

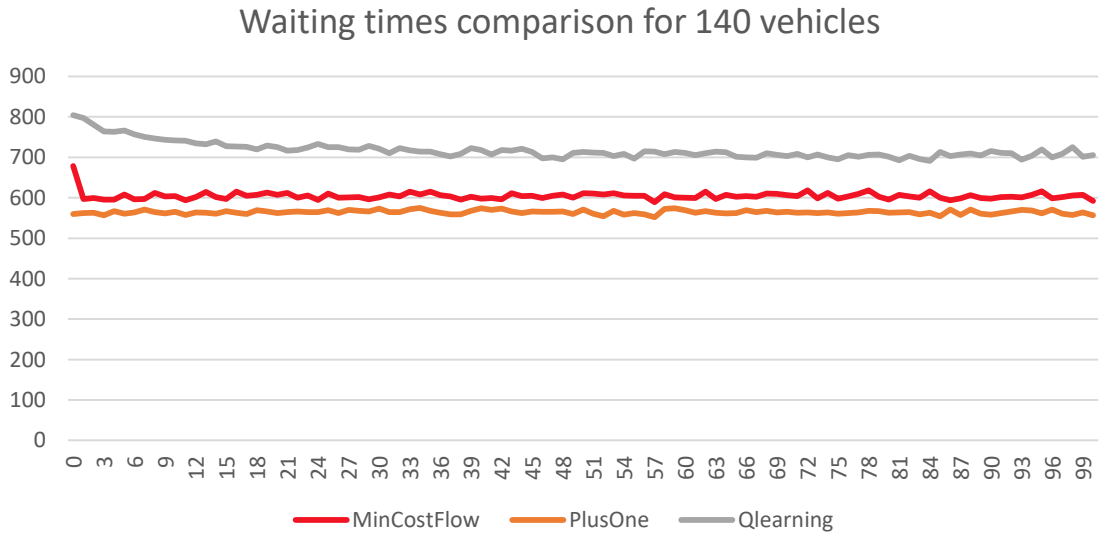

Rejected requests numbers comparison for 140 vehicles
1400
1200
1000
800
600
400
200
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning

Our results:


Waiting times comparison for 110 vehicles
900
800
700
600
500
400
300
200
100
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning


Waiting times comparison for 120 vehicles
900
800
700
600
500
400
300
200
100
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning


Waiting times comparison for 130 vehicles
900
800
700
600
500
400
300
200
100
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning


Waiting times comparison for 140 vehicles
900
800
700
600
500
400
300
200
100
0
0 3 6 9 12 15 18 21 24 27 30 33 36 39 42 45 48 51 54 57 60 63 66 69 72 75 78 81 84 87 90 93 96 99
MinCostFlow
PlusOne
Qlearning

Conclusions from out first results

- A decentralized QLearning rebalancing algorithm is able to show good results on a static demand
- A more dynamic testbed is needed to benchmark the algorithm

Other reward signals for the algorithm are being tested

- A reward reflecting the cost/revenue of the service
- A lexicographic reward with the number of passengers and waiting times
  - The first results show similar performance to the first QLearning algorithm

Research perspectives

- Take into account **electric vehicles** in the RL algorithm
  - Including the state of charge in the state and charging stations in the possible actions
  - Curse of dimensionality, need to use more compact representations
- Test on a scenario of the Paris-Saclay area
  - More realistic and more dynamic demand
  - **Alongside public transports** with a focus on **intermodality**
  - Then test on other scenarios
- Combine MATSim simulations with **Cost-Benefit Analysis**
  - In collaboration with another PhD student

Referefences

- [1] Regue, Robert, et Will Recker. « Proactive Vehicle Routing with Inferred Demand to Solve the Bikesharing Rebalancing Problem ». Transportation Research Part E: Logistics and Transportation Review 72 (décembre 2014): 192-209. https://doi.org/10.1016/j.tre.2014.10.005.
- [2] Gueriau, M., F. Cugurullo, R. A. Acheampong, et I. Dusparic. « Shared Autonomous Mobility on Demand: A Learning-Based Approach and Its Performance in the Presence of Traffic Congestion ». IEEE Intelligent Transportation Systems Magazine 12, nº 4 (winter 2020): 208-18. https://doi.org/10.1109/MITS.2020.3014417.
- [3] Al-Abbasi, Abubakr O, Arnob Ghosh, et Vaneet Aggarwal. « DeepPool: Distributed Model-Free Algorithm for Ride-Sharing Using Deep Reinforcement Learning ». IEEE Transactions on Intelligent Transportation Systems 20, nº 12 (décembre 2019): 4714-27. https://doi.org/10.1109/TITS.2019.2931830.
- [4] Fluri, C., C. Ruch, J. Zilly, J. Hakenberg, et E. Frazzoli. « Learning to Operate a Fleet of Cars ». In 2019 IEEE Intelligent Transportation Systems Conference (ITSC), 2292-98, 2019. https://doi.org/10.1109/ITSC.2019.8917533.

04/10/2021

Thank you for your attention

Systemx
INSTITUT DE RECHERCHE TECHNOLOGIQUE

LGi
Laboratoire Génie Industriel

CentraleSupélec

AnthroPOLIS
HUMAN CENTERED MOBILITY

Nest Seminar

DOES THE LABOR COMPETITION REALLY MATTER TO URBAN AGGLOMERATION DEVELOPMENT ?

Han WANG, PhD Candidate, Beihang University, Beijing

WEDNESDAY, OCTOBER 13th, 2021 | 10-11 AM CEST

Link to the seminar