Learning Control Policies in Smart Cities from Physical Data

Mykhaylo Marfeychuk
mykhaylo.marfeychuk@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

November 2020

Abstract

Motor vehicle emissions are the primary contributors to the increase in ambient pollution levels. Rapid urbanization and lack of a good solution to manage the traffic are forcing cities to take drastic measures against the automotive industry. In this thesis, we build a case study of the traffic in the Norwegian city of Trondheim and create a realist simulation based on real world data, that simulates the traffic and the emissions. We then propose a Reinforcement Learning based solution that controls the access to the different regions of the city to optimize the traffic given a desired metric. We also take a look at different improvements, like using a multi-agent system and using pre-generated data for the training phase. We compare the obtained results with the baseline and against a reactive agent. At the end we assess the solution's strengths and weaknesses, and propose possible future improvements.
Keywords: Deep Learning, Reinforcement Learning, Multi-Agent System

1. Introduction

Motor vehicle emissions contribute to ambient pollution levels with carcinogen toxins. Exposure to these toxins can also cause non-cancer health effects, such as neurological, cardiovascular, respiratory, reproductive and/or immune system damage. Many of today's traffic lights and road accesses, are controlled by a timer or a predefined pattern. This solution may have a lot of overhead in situations where the traffic doesn't follow the pattern. This in turn has a direct correlation to the pollution increase within the city.

The goal is to implement a solution that reduces the pollution in the area by optimizing the traffic flow, using Deep Learning and Reinforcement Learning (RL) based algorithms. The solution, given the readings from the air pollution sensors and the traffic density, needs to reduce the dimensionality of the feature set, i.e. use only relevant features, and determine the current environment state. In many cases the sensors data is plagued with noise and uncertainty, so the solution needs to compensate for that, with the added requirement that it is able to work as desired with as little information as possible, as it is hard to collect perfect knowledge from the real world. With the states, a Deep Neural Network needs to learn an optimal policy, on how to control the traffic. The policy dictates which actions to be executed, in this case, by opening and closing access to certain roads. The objective is to optimize the traffic flow in such a way that it lowers the pollution levels caused by the traffic, while still maintaining an efficient traffic flow.

The AI4EU consortium was established to build the first European AI On-Demand Platform and Ecosystem, by leveraging industry-led pilots. ISR collaborates with one of the pilot projects, with the goal of implementing solutions to better citizen's lives by improving the city traffic in the city of Trondheim. The abundance of traffic and pollution data and that the AI4EU partner in charge of the pilot, Telenor, is located in Trondheim, are the reasons why the city was chosen as the main subject of the thesis case study.

2. Related Work

2.1. RL Based Traffic Control

Lin et al proposed[1] a Deep RL based approach to optimize the traffic flow by better controlling the traffic lights. The approach uses an Advantage Actor-Critic (A2C)[2] based deep learning model. The model receives as input a 3D tensor that represents the entire traffic grid. The model's output is a simplified discrete action space, in a shape of $< N_{tls}, 2 >$, where 2 indicates the number of the discrete probabilities of choices, either maintaining or switching to the next phase for each intersection, and N_{tls} is the number of traffic lights.

In the paper, the agent's reward was divided into two parts. One part is a global reward that leads the agent to optimize a global behaviour of the whole network. For this, the net outflow of the whole network was used. The net outflow is calculated by subtracting the input vehicles $||Veh_t^{(in)}||$ from

the output vehicles $||Veh_t^{(out)}||$ within the selected area at each time step t (1). When a congestion or collision occurs, the simulator often teleports the vehicles to the position where the vehicle should have been. Because of this only the vehicles that weren't teleported are counted for the net outflow.

$$r_t^{Global} = ||Veh_t^{(out)}|| - ||Veh_t^{(in)}|| \quad (1)$$

The other part is a local reward, that helps the agent learn a local behaviour relative to the intersecting. It is defined as the absolute negative difference between queue length (2).

$$r_t^{TLS_i} = -|\max q_t^{WE} - \max q_t^{NS}| \quad (2)$$

For each intersection TLS_i, q_t^{WE} is the number of halting vehicle in lanes from west to east or vice versa.

The complete hybrid reward function is formed by summing both the global and local rewards (3).

$$r_t = \beta r_t^{Global} + (1-\beta)\frac{1}{N_{TLS}}\sum_i^{N_{TLS}} r_t^{TLS_i} \quad (3)$$

2.2. Multi-agent Based Road Traffic Control

Arel et al [16] introduced a control system where the model-based reinforcement learning approach is utilized to optimize traffic signal in a network aiming at minimizing travel time.

The team proposes a multi-agent approach, where in a five-intersection system where each intersection has an agent deciding the traffic light state. There are two types of agents where the only difference is the type of information they receive. The outbound intersection agents only have access to the local traffic statistics, while the central intersection agent has access to all states of its neighbouring intersections.

The outer intersection agents employ the longest-queue-first algorithm, while the inner intersection agent uses the RL based approach. When arrival rates are low, the longest-queue-first scheduling algorithm performs slightly better than the multi-agent Q-Learning system.

For the reward, the traffic delay difference between the current and previous timesteps was used, this may be positive or negative (4). D_{new} and $D_{current}$ are the previous and current intersection total delays.

$$r = \frac{D_{last} - D_{current}}{\max[D_{last}, D_{current}]} \quad (4)$$

Alegre [3] implemented a multi-agent approach using RLlib and SUMO (Simulation of Urban MObility)[4] to control Traffic Lights. In the approach, each traffic light has it's own independent agent. Each agent receives the states of the neighboring lanes, which includes the amount of cars, density, waiting time, and the traffic lights current state. At every few steps the agents have to decide to which phase they should change the Traffic Light.

Given the flexibility of the RLlib library, he also created a variation where only a single agent is used to control all the traffic lights. RLlib having many RL algorithms already implemented, allows easily to easily change the used algorithm with minimal changes.

Chu et al [5] propose a Multi-agent RL scalable approach for the adaptive traffic control problem. The proposed approach uses a synchronous multi-agent A2C algorithm to train the agents, where each intersection has an agent which decides the red-green combinations of the traffic lights. The agent receives the *wait* which measures the cumulative delay of the first vehicle, *wave* which measures the total number of approaching vehicles, and the neighbor policies[6].

For the reward the authors propose to optimize for the *queue* which measures the queue length along each incoming lane, the *wait* and a which is the trade-off coefficient (5). This reward only contains a local reward, and not a global reward like other proposed approaches.

$$r_{t,i} = -\sum_{ji \in E,\ l \in L_{ji}} (queue_{t+\Delta t}[l] + a \cdot wait_{t+\Delta t}[l]) \quad (5)$$

The approach was compared to a Deep Q-network (DQN)[7] based and single A2C agent based approaches. The DQN wasn't able to converge, and the multi-agent approach converged faster than a single agent.

3. Environment

SUMO provides many necessary features, such as traffic flow simulation, traffic lights control and vehicle emissions, which are necessary for this case study.

To guarantee the credibility of the simulation, real world sampled data is used to tune the simulation for it to be as close as possible to the real world.

3.1. Trondheim Map

OpenStreetMap (OSM) is an open source collaborative project to create a free editable map of the world. The project tracks the topology of the environment, geocoding of addresses, place names, and route planning. provides a REST API that can be used to programmatically download the desired information. OSM Web Wizard is a tool that SUMO provides that allows to generate a map using OSM.

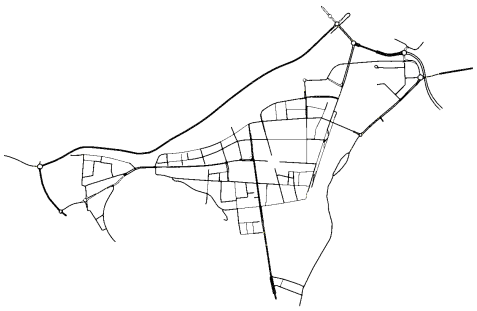


Figure 1: Generated SUMO Simulation Map

A grid system was used on the map to track the environment state and simplify the interactability with the environment components. The map is split into 17×16 cells, where each cell is 200 m^2 and is composed of multiple roads. Furthermore the cell tracks the number of vehicles present in it at any current simulation step, as well as the pollution values, and the travel time. Another state of the cell is related to the action that may be performed on the cell. A cell may be open or closed, which propagates the state to the road. If a cell is closed, the vehicles are prohibited to travel on the roads that lay inside of the cell.

3.2. Traffic Modelling

An inductive-loop traffic detector is a detection system which uses the principle of electromagnetic induction to detect or measure objects. It is widely used in cities to detect passing vehicles or to control traffic signals at an intersection of roads.

The Norwegian Public Roads Administration [1] host a variety of induction loop sensors located in relevant entry points of the city of Trondheim. The sample rate is per hour basis and counts how many vehicle passed through the sensors, and logs the estimated size of the vehicle. The size distinction is relevant, as differently sized vehicles represent different emission categories.

By having the number of vehicles passing through each induction loop, and the relative location of the induction loop in the simulated environment, DFrouter tool was used to generate the traffic. DFrouter is a tool that tries to create traffic where the simulated induction-loop vehicle count matches the desired vehicle count.

The vehicle routes are originally set up based on the real world data. When the environment state changes, for example when a road is closed, the vehicles need to react to the change and generate a different route to guarantee the arrival at the destination. By default the dynamic routing is not enabled, and the simulator manually adds and removes vehicles to maintain the specified induction loop count. When enabled, at each simulation step all the vehicles check if all the edges in their route are reachable, if not then a new route is calculated.

3.3. Pollution Modelling

In many areas, vehicle emissions have become the dominant source of air pollutants, including CO, CO2, CHs, NOx, and PM. Many of the particles are affected heavily by the external environment sources, and some don't have any data available, so the primary focus of the simulation will be to study and simulate the NOx particles.

The NILU[2] hosts a variety of air quality sensors located throughout Norway, including three located in Trondheim. The data is available online and will serve as the baseline to model the simulated emissions.

SUMO simulator provides various open source and commercial models. The HBEFA[8] v3.1 model is used, as it is the latest more accurate open source model available.

The SUMO simulator only provides instantaneous emissions for the vehicle and the road, meaning that the values are only available for the last step of the simulation. Because of this, a custom emission tracker and simulation needs to be implemented. At each time step the vehicle contributes to the emissions of the cell it's located in, by concatenating the vehicle pollution values to the current cell value.

To make the emission behaviour more realistic, two extra features are used, pollution propagation and decay. At each step the pollution values are reduced by a certain amount, to simulate the particle settling down and decaying. A Gaussian Blur is also applied, after the value reduction, to simulate the propagation through the neighboring cells.

The per-pollutant decay and dissipation values can be seen in Table 1.

	CO	**CO_2**	**CHs**	**NO_x**	**PM**
Decay	0.9999	0.8	0.999914	0.996	0.995
Dissipation	0.3	0.4	0.3	0.3	0.3

Table 1: Per-emissions type decay and dissipation values

3.4. Environment State Tracking

The SUMO simulator only handles the traffic simulation and provides a Traffic Control Interface (TraCI) API to interact with the simulation. However this is only the most basic and necessary, so it doesn't have the functionality that is needed. To solve this, an environment was created that wraps the SUMO simulation and adds extra functionality. The built environment follows a modular methodology, where a core system exists, and optional mod-

[1] Norwegian Public Roads Administration website, https://www.vegvesen.no/

[2] Norwegian Institute for Air Research website, https://www.nilu.com/

ules can be provided to augment and provide more functionality.

The core of the environment is called *BaseEnv*, and it is responsible for handling the SUMO simulation and optional modules. *BaseEnv* receives the simulation map, start/stop time steps, update frequency, logging directory, and additional modules. Another class called *BaseRLEnv* was created that wraps the *BaseEnv* and implements the *Gym.Env* interface. This is necessary because the used RL library requires a gym environment.

During the simulation environment creation, a list of modules are passed as one of the parameters. The environment takes care of calling the module base functions, but it is the module's responsibility to have the desired functionality implemented. The modules implemented for the project are the following:

- **Cells Module:** This module creates the cell matrix of the map, where each cell corresponds to each region of the map. The module keeps track of certain information like, all the roads that belong to the cell, the number of vehicles, the travel and waiting time, and the current state. The module provides functions which are used by the agents to change the state of the cells to open or closed.

- **Emissions Module:** The module keeps track of the per cell emission values. For each emission type, the module creates a cell grid matrix which contain the emission type values. At every step it gets the current step emission values of all the vehicles and adds them to the corresponding matrix position.

- **Induction Loops Module:** The module keeps track of how many vehicles pass through every induction loop for every hour. At the end of the simulation the module writes the vehicle count to a CSV file.

- **Tracking Module:** The module tracks general statistical information about the simulator, like the number of closed cells, aggregated emission values, max emission value, wait and travel times, number of created and arrived vehicles. This module samples the information every 15 minutes and writes them to a CSV file.

4. Implementation

An agent is taught a policy by interacting with the environment. This policy is then used to choose which action to perform at a specific state. A2C was chosen as the training algorithm as it is parallelizable and showed to converge faster compared to the other algorithms.

Every 15 minutes of the simulation, the agent has to perform actions on the environment. The agent goes over every cell of the map and decides if the cell should stay open or closed. After this, a binary grid map is generated which represents which cells should be closed or open.

4.1. Environment

At each step, the agent needs to decide the action to perform for each cell. The way the problem is formulated, each cell will have it's own observation, action and reward.

The proposed observation has a mix of both, the state of all the cells as well as the current action cell's specific information. This way the agent will choose the best action for the current cell while taking into account the state of all the surrounding cells.

Due to the fact that the city is represented as a $m \times n$ grid, and that the state of one road may directly affect the state of another, a Convolutional Neural Network (CNN)[9] is proposed for the initial layers of the model.

The matrix input is comprised of the following data:

- **Emissions:** Represents the current per cell emission values.

- **Number of vehicles** Contains the number of the vehicles that are present in each cell.

- **Closed cells:** Is a binary matrix that represents the currently closed cells.

- **Action Cell:** This matrix represents on which cell the agent is acting upon. One value is set to 1 and the rest is set to 0. This matrix changes for each acting cell.

For each cell, the agent can perform one of two actions, open or close the cells. Changing the cell state also changes the state of all the roads inside the cell.

4.2. Deep Learning

Each agent has a neural network that when given a state, produces a action. The primary reason for using a neural network is that the current problem can't be represented as a q-table.

There is no clear way of finding the optimal performing network structure without trial and error. The proposed model is comprised of two consecutive convolutional layers. The task of these layers is to find spatial correlation between the input data. After each of the convolutional layers, there is a Max Pooling Layer which performs down-sampling, which helps to reduce the dimensionality of the data. After this, the max pooling layer connects to a fully connected network, with an additional input. This extra input contains categorical data

about the general state of the environment. The fully connected network is comprised of two dense layers and a Softmax layer. For each dense layer a Dropout regularization technique is used, which randomly disables certain neurons from training.

The model uses ReLU as the primary activation function, this is because the input data can have abstract data ranges, meaning that activation functions like Sigmoid or tanh would perform poorly. The generated neurons are initialized with a normal distribution. The detailed layer parameters are shown in Table 2.

Layer Name	Activation Shape
Input 1	(16, 17, 4)
Conv 1 (f=16, k=[3,3], s=1)	(16, 17, 16)
Max Pooling 1	(15, 16, 16)
Conv 2 (f=32, k=[4,4], s=1)	(15, 16, 32)
Max Pooling 2	(14, 15, 32)
Input 2	(16, 1)
FC 1	(144, 1)
FC 2	(64, 1)
Softmax	(2, 1)

Table 2: Neural Network Parameters (f=filters, k=kernel, s=stride)

4.3. Multi-agent Framework

Ideally each cell would have it's own agent which would learn a policy only applicable to that cell, and not a global one. Taking this approach would require a lot of computational power as, thus compromised approach is proposed where the map is split into 4 regions, represented in Figure 2. Each region has its own agent that acts upon the cells in that region. Each agent has its own network which needs to be trained independently. For each step, each agent iterates over the cells in its region and decides if the cell should be open or closed. All the agents receive the same input where the only variation is the location of the cell.

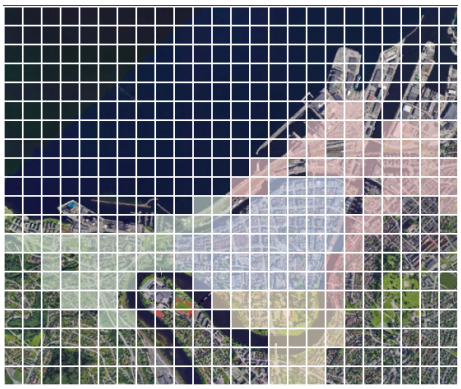


Figure 2: Map split into multi-agent regions. Each color corresponds to an agents actionable cells.

4.4. Reactive Agent

As there are no available implementations that run on the same environment, an ad hoc agent is proposed. The proposal in question is a reactive agent. The only functionality that the agent has, is to check the pollution levels in each cell and close the ones that exceed a specific threshold, as represented in Algorithm 1. The reason for such an agent is because every city in Europe has a limitation of pollution that an area in any given time cannot surpass. Two tests are performed, one where the threshold is 100 NOx (g/m^3) per cell, and the other where the threshold is 50.

Algorithm 1 Reactive Agent Decision Flow

```
procedure REACTIVE_AGENT(threshold, cells)
    for cell in cells do
        get cell emissions
        if cell_emissions >= threshold then
            Close cell
        else
            Open cell
        end if
    end for
end procedure
```

4.5. RL Agent

The goal is to develop an RL agent that reduces the pollution in the city. For this an adequate reward function needs to be found. For this an iterative search was performed, until the correct reward was found, as shown in (6).

$$reward = \frac{(t - mce)}{200} + \frac{av}{eav} \tag{6}$$

where:

t = emissions threshold
mce = current highest cell emission
eav = expected step arrived vehicles
av = step arrived vehicles

Having a reward function, the final experiments can be conducted. First, a experiment is done to test the performance of a single agent against a multi-agent system. The best approach will then be used for the further experiments. Two additional experiments are conducted where the only variation is the threshold value, one with 100 and 50 NOx.

5. Results

Most of the days showed little variation between one another. The training was performed on a single day, and tested on a different one. This is to test the agents ability to generalize. The training is limited to a single day because of the time a training sessions takes.

5.1. Vehicle Throughput

While testing the agent for the arrived vehicles optimization, in some cases the agent learned a different pattern of behaviours. In general the agent learned that opening all the cells maximizes the traffic throughput, but in some cases, the agent learned that some cells are irrelevant for the traffic, and keeping them closed doesn't affect the traffic flow. This is because, when those cells are closed, there is always a different route that the vehicles could take. The cells in question can be seen in Figure 3.

Figure 3: Cells that are not relevant for the traffic

5.2. Pre-gerenated Data

Training the agent on pre-generated data sped up the training drastically, which in turn allowed more testing and rapid prototyping. This unfortunately didn't work, as the agents always overfitted on a wrong solution or didn't learn a solution at all, compared to a normally trained agent. This may be due to the lack of training data, as the agent would require more examples because of the lack of exploration.

5.3. Multi-agent

A test was made comparing a single agent, against a multi-agent system. As shown in the Figure 4, the multi-agent approach was able to learn a better policy compared to a single agent. With different reward functions, the multi-agent approach was able to converge faster on a solution compared to a single agent. Because of this, all of future trainings were performed on the multi-agent system.

5.4. Comparison

Taking a look at Figure 5, we can see some interesting patterns. Looking at the agents with 100 NOx (mg/m^3) threshold, we can see that the RL agent is able to reduce the emissions more than the reactive agent. This is because the RL agent leverages the emissions more than the arrived vehicles, thus a small decline in arrived vehicles.

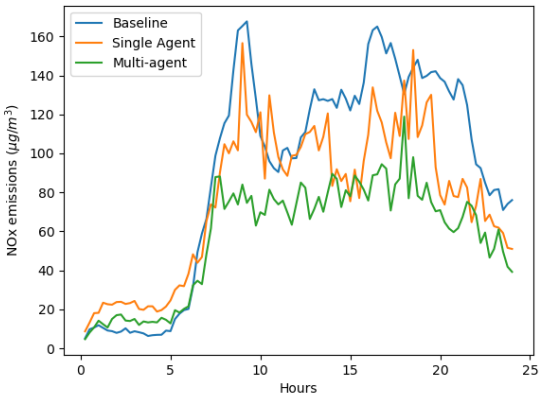


Figure 4: Max cell emission comparison between Single Agent and Multi-agent approaches

For the 50 NOx (mg/m^3) threshold, both agents struggle to maintain the below the threshold. The reactive agent can't lower the emissions more due to the poor management of the cells, while the RL agent is due to the emissions-arrived equilibrium. It is also relevant to notice, that with this threshold, the agents prioritize lowering the emissions more than maintaining vehicle flow. In general both agents have similar results, except for the waiting time. The Reactive agent closes the cells with the vehicles, blocking them from moving, this in turn increases the waiting time, while the RL agent tries to avoid this behaviour. The RL agent is able to learn this behaviour because of the arrived vehicles reward part. When the vehicles are stuck in a cell, they are not able to arrive to the destination, in turn lowering the reward.

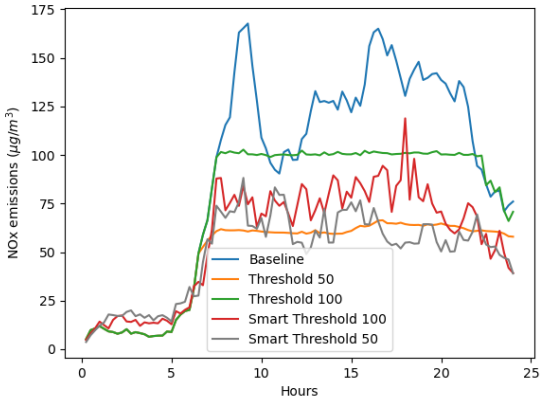
It may seem counter-intuitive that vehicles taking longer routes don't increase the pollution. This is because when the vehicles take longer routes, they in fact produce more emissions, but they produce the emissions in less concentrated areas which allows for the emissions to dissipate faster.

Table 3 shows a more statistical results comparison.

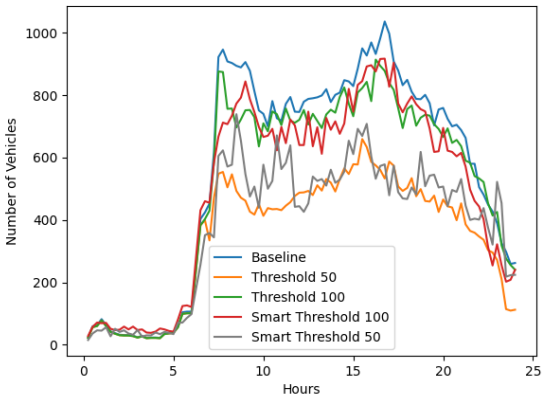
6. Conclusions

Since it's not possible to test a solution on the real world, a simulator is required. A simulation based on SUMO was created that generates traffic based on real world sampled data. Additionally vehicle emission propagation was modelled based on real data. The simulator was also extended to implement the OpenAI Gym format which is a common RL practice. Using a Gym environment allows to use existing RL frameworks.

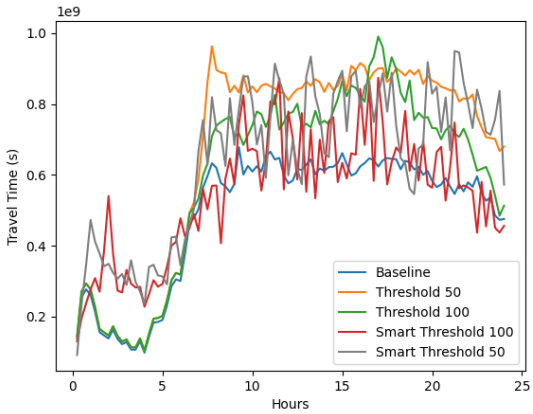
We provide an RL solution to reduce the city level emissions by managing the access to specific



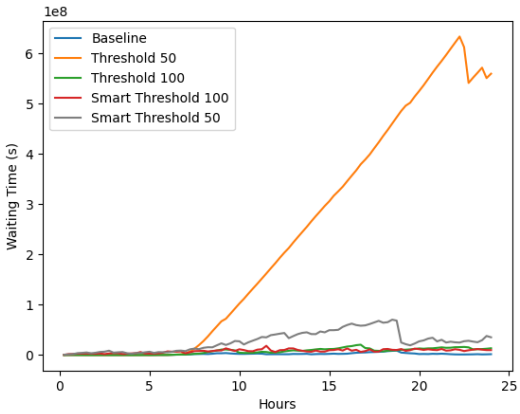
(a) Max Cell Emissions Value



(b) Vehicles arrived to Destination



(c) Travel Time



(d) Waiting Time

Figure 5: Results comparison between Reactive Agent, RL Agent and Baseline

Table 3: Agent differences relative to the Baseline

	Avg. Max Emissions	**Avg. Arrived Vehicles**	**Avg. Travel Time**	**Avg. Waiting Time**
Reactive Agent Limit 100	-19.13%	-8.21%	15.69%	192.40%
Reactive Agent Limit 50	-47.39%	-37.67%	28.14%	9073.79%
RL Agent Limit 100	-36.97%	-10.53%	11.51%	275.15%
RL Agent Limit 50	-45.99%	-31.07%	31.70%	1195.48%

areas of the city. The proposed approach splits the map into a cell grid and uses multiple agents that assess the state of each of the cells. Each of the agent is responsible of managing a specific region of the city, which are comprised of multiple cells. Each of the agents contains a CNN which is used to find the patterns on the input data, and learn a desired policy. The multi-agent approach was used for the experiments as testing showed that it performed better than a single agent solution. This is because a single agent can't generalize well on so many actionable spaces.

An approach using pre-generated data was proposed. Here multiple simulations with random actions were run and the intermediate states were stored. The agents were then trained on this data. This approach would allow a more controlled environment and faster training. Unfortunately the agents were not able to learn a valid policy and overfitted on a specific behaviour.

The agents were trained on different reward functions and compared with a reactive agent and the baseline. The RL agents perfomed better than the reactive agent at minimizing the emissions below the threshold. When more weight was given to minimizing the emissions, a bigger reduction in number

of vehicles was seen, similar behaviour to the reactive agents behaviour. The RL agents also showed that some regions of the city are not necessary to the traffic throughput, as there are alternative routes that the vehicles could take.

Acknowledgements

I would like to thank my partner for their friendship, encouragement and caring over all these years, for always being there for me through thick and thin and without whom this project would not be possible.

I would also like to thank my parents, for the opportunity to attend University and for their support throughout all these years.

I would also like to acknowledge my dissertation supervisors Prof. Pedro Lima and Dr. Tiago Veiga for their insight, support and sharing of knowledge that has made this Thesis possible.

Last but not least, to all my friends and colleagues that helped me grow as a person and were always there for me during the good and bad times in my life. Thank you.

To each and every one of you – Thank you.

References

[1] Y. Lin, X. D. andLi Li, and F.-Y. Wang, "An efficient deep reinforcement learning model for urban traffic control," *CoRR*, vol. abs/1808.01876, 2018. [Online]. Available: http://arxiv.org/abs/1808.01876

[2] V. R. Konda and J. N. Tsitsiklis, "Actor-critic algorithms," in *SIAM Journal on Control And Optimization*. MIT Press, 2001, pp. 1008–1014. [Online]. Available: https://papers.nips.cc/paper/1786-actor-critic-algorithms.pdf

[3] L. N. Alegre, "Sumo-rl," https://github.com/LucasAlegre/sumo-rl, 2019.

[4] P. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[5] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, 2019.

[6] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, "Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 3, pp. 1140–1150, 2013.

[7] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *DeepMind Research*, December 2013. [Online]. Available: https://arxiv.org/pdf/1312.5602v1.pdf

[8] "The handbook of emission factors for road transport," Jan 2010. [Online]. Available: https://www.hbefa.net/e/index.html

[9] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–44, May 2015.