Reward Definitions in Reinforcement Learning for Traffic Light Control

Cian Jansen[1] Miguel Suau de Castro[2] Frans Oliehoek[3]

[1] *Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands*

[2] *Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands*

[3] *Department of Intelligent Systems, Delft University of Technology, Delft, The Netherlands*

Abstract

Traffic congestion is a problem of tremendous size that affects many people. Using Reinforcement Learning to find a light control policy can ease traffic congestion and decrease travel time for vehicles. This paper specifically looks at the effect of using different reward functions for training agents. We highlight how the learnabilty of a reward function and its alignment with the final goal of the agent are the most important factors when designing a reward definition. Finally we propose a reward function to use for future applications.

***Keywords*—** Reinforcement learning, Traffic signal control, Reward functions, SUMO, Linear regression

1 Introduction

Traffic congestion is a large problem that brings with it enormous cost for the economy and immense frustration for the people it affects. The European Commission estimates this cost to be around 1% of the European Union's entire GDP (Barker, 2001). A viable method to alleviate this problem is by using smart algorithms to control traffic lights at intersections (Van der Pol & Oliehoek, 2016). One way to achieve this is by using Reinforcement Learning to learn optimal policies for deciding which lane should have a green light at each moment.

In Reinforcement Learning (RL), a reward function is used to describe the benefit (or reward) of moving from a certain state to another state by taking a certain action (Sutton & Barto, 1998). By tuning this reward function, one can prioritise different goals. In the domain of traffic light control, the ultimate goal is to minimize the travel time on a road network or intersection by maximizing the average speed of all cars, as explained in section 4.1. However, this goal itself can not be used as a reward function, since it can only be calculated after a simulation is finished. Reward functions need to be calculated at every step of a simulation. To bridge this problem, prior works researching RL for traffic control have assumed that certain factors, which *are* continually observable, are likely to give an estimate of the impact of actions on traffic performance (Van der Pol & Oliehoek, 2016) (Van der Pol, 2016) (Touhbi et al., 2017). Examples include the average speed of cars in the area of the intersection or the average time a car has to wait at the intersection. These factors have then subsequently been used to calculate the reward that should be given to an agent. However, a comprehensive overview of these different factors, whether they are actually accurate for estimating traffic performance and what kind of results agents can achieve when using them is missing. This will be surveyed in this paper at the hand of the following question:

What effect can different reward functions have on the performance of a Reinforcement Learning system for traffic light control?

The performance of policies obtained by training with different reward functions will be compared on their impact on the average speed of all cars in the system. section 4.1 will explain why this is the most suitable metric to answer "What defines success in a traffic control scenario?". Multiple reward functions will be described in section 4.2 by surveying "Which aspects of traffic can be represented in a reward function?" and using these aspects or weighted combinations thereof to grant rewards while training. This paper will use reward shaping techniques, as introduced by Ng, Harada, & Russell (1999), to substantiate the granting of rewards that reinforce behavior that is aligned with our goal of maximizing average speed. Trained models can be compared using the SUMO traffic simulator (Behrisch, Bieker, Erdmann, & Krajzewicz, 2011). These comparisons will be used in section 5 to conclude what influence an expertly designed reward function can have on the learning speed and final performance of an RL system.

This research differentiates itself from other research in the field by focussing on comparing different reward functions. In the works of Touhbi et al. (2017) and Van der Pol & Oliehoek (2016), reward metrics are chosen but no investigation is done to identify whether the chosen metrics actually estimate traffic performance. In the research done by Tumer & Agogino (2006), cars are used as agents, whereas this research will use intersections as agents. This shifts the incentives of agents from a selfish focus on a single car to a more social focus of improving traffic for all participants. Kuyer, Whiteson, Bakker, & Vlassis (2008) focus on rewarding coordination of multiple intersections, and use a relatively primitive reward function for individual intersections. In the research by Wiering (2000), minimizing waiting time of cars is used as the ultimate goal. Even though this metric is continually observable, it is flawed in that it rewards traffic lights that switch between red and green rapidly. This minimizes cumulative waiting time without creating a desirable traffic situation as explained by Van der Pol & Oliehoek (2016), and is thus not an ideal reward function.

2 Background

2.1 Reinforcement learning

In reinforcement learning, an agent is trained to map a state to an optimal action to perform, in such a way that total reward over the long term is maximized. Every timestep t, an agent receives information about (a representation of) the current state s_t of the environment. It then takes an action a to arrive in state s_{t+1}. Doing so, the agent receives reward r_t. A learning agent tries to find an optimal policy $\pi : S \mapsto A$ to maximize the discounted cumulative reward function $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, with $0 < \gamma < 1$ as the discount function. The value of a state-action pair $Q_t(s, a)$ is called the Q-value, which a q-learning algorithm estimates with iterative Bellman updates $Q_{t+1}(s, a) = Q_t(s, a) + a[y_t - Q_t(s, a)]$, with target $y_t = r + \gamma \max_{a'} Q_t(s', a')$
Deep Q-Learning algorithms (DQN) use deep neural networks to estimate the value of these State-action pairs. This is used to allow larger or even continuous state-spaces, which would otherwise be too large to have all possible Q-values memorized or stored. In turn, DQNs allow for using images to represent states.

2.2 Reward functions in reinforcement learning

The reward function R defines what reward $R(s, a, s')$ is given to an agent when transitioning from state s to state s' by taking action a. This reward is calculated by examining the state of the environment and granting reward based on the quality of said state. When designing a basic reinforcement learner for path-finding, reward could for example be granted when an agent reaches the goal location. In more complex scenarios, multiple aspects of the state are examined to determine the value of being in said state. This is then used to calculate the appropriate reward for moving from a certain state to another state by performing an action. There exist multiple aspects of the state of the environment (e.g. the average speed cars are currently driving, the cumulative waiting time of all cars currently in the system) that can be represented in the reward function. By choosing which aspects are represented, and how heavily to value them, one can choose what kind of behavior should be reinforced and what kind of behavior penalized. Thus selecting different aspects with different relative weights will lead to a different optimal policy $\pi : S \mapsto A$.

2.3 Reward shaping

Shaping rewards, as introduced by Ng et al. (1999), are additional rewards that are granted to actions that do not achieve a goal immediately, but nonetheless improve the state of the environment. They are used when there is no specific metric available that can be used to optimize for, or when rewards would otherwise be too sparse. In the case of this paper, reward shaping will be used to create reward functions that are strongly aligned with improving average speed in the system. The average speed of the system can not be used as a reward metric itself, since it can only be calculated after a simulation is done. This is too sparse of a signal for training agents, which need to be able to calculate the reward for possible actions during every step of a simulation.
When optimizing a policy for Markov Decision Process $M = (S, A, T, \gamma, R)$ by adding shaping rewards, optimization is done on MDP $M' = (S, A, T, \gamma, R')$, with $R' = R + F$. This means that the actual reward an agent receives is given by $R(s, a, s') + F(s, a, s')$, where $F(s, a, s') = r$ for some value r when moving in the direction of a goal, and $F(s, a, s') = 0$ otherwise. This shaping reward F can be used for guiding agents in a more informed way towards a preferable state. Shaping reward F is called "potential-based" when the optimal policy $\pi_{M'}$ in M' is also an optimal policy in M.

2.4 Learnability of reward

The learnability of a reward describes how well the reward an agent receives can be attributed to the action an agent performs, as opposed to other events that impact the environment. In a multi-agent scenario, it is also used to determine how much a reward is dependent on a certain agents action compared to the actions performed by other agents (Tumer & Agogino, 2006). A strict formula for the learnability of rewards can not be given for this single-agent research, as that would necessitate total knowledge of how every aspect of a state was formed by all actions of the agent combined with influence from the environment. This information is not available since the environment is randomized to counter overfitting, and thus no deterministic causal links can be made. Learnability of a reward can however be estimated by analyzing how the amount of reward gained from a policy changes depending on actions of an agent. Such an analysis is performed in section 6.

3 Reinforcement learning for traffic light control

Following the work done by Van der Pol & Oliehoek (2016), we will try to find an optimal reward definition for applying RL to the problem of controlling traffic lights at an intersection. We use SUMO (Behrisch et al., 2011) for testing and verifying results.

3.1 Model

We describe the problem as finding an optimal policy π_M for some Markov Decision Processs (MDP) $M = (S, A, T, \gamma, R)$ where S represents the state space, A is the action space, T is the transition function, γ is the discount value and R is the reward function.
State space: We describe the state around an intersection as a binary matrix containing the positions of vehicles in the lanes surrounding the intersection. The matrix also includes information about the position and current state (Green/Yellow/Red) of traffic light in the intersection.
Action Space: The agent decides whether to give a green light to one lane or the other on each time step.
Transition Function: The transitions when going from s_t to s_{t+1} are defined by SUMO and depend on the state of lights in environment and the location of cars on step t.
Discount Value: $\gamma \in (0, 1]$ defines the discount of importance given to older rewards.
Reward function: Multiple reward definitions will be used and compared as explained in section 4.2.

3.2 Traffic Scenario

For training and evaluation, a computer simulation of a traffic scenario is used. This scenario consists of a certain road network with some number of intersections, as well as presets for how many cars pass through the area at a given time. The scenario used in this research is a basic grid intersection, shown in Figure 1, where two road cross each other with a single set of traffic lights. The amount of cars that pass into the intersection and their corresponding routes are generated in a random fashion based on two random variables, car_tm and car_pr. Every timestep between $t = 0$ and $t = car\_tm$, a car has a chance equal to car_pr of spawning, with a route that includes passing the intersection. This traffic load is randomized so the agent can not overfit on a set input and develop a control policy which only works on that exact input set. Overfitting happens when the model of some learner loses abstraction and is highly tuned to the exact input dataset. This leads to the model having an unrealistic accuracy when working with data from the dataset it was trained on and performing worse on data it was not trained on (Hawkins, 2004).

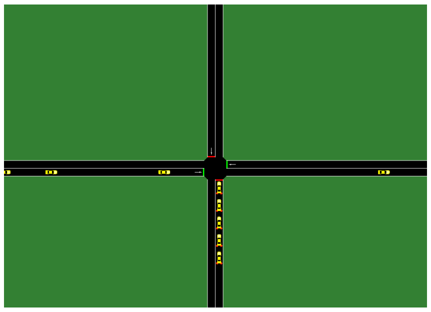


Figure 1: A graphic overview of the scenario used for training and evaluation

3.3 Experimental Setup

To identify the effect of different reward definitions, the following testing setup was created: A certain reward definition would be used as the reward function of an agent trained over the course of 2000 episodes on the basic traffic scenario. Every tenth episode is an evaluation episode, and a reward function is used that returns the average speed of all cars during that episode. This reward can only be used in evaluation and not in training since it is too sparse, only granting reward on the very last step of the episode. It can nonetheless be used in episodes when the agent is not learning, to evaluate the policy is has learned so far. The reward functions tested are discussed at length in section 4. After those episodes of training, the agent is evaluated for 150 episodes on three factors:

- What is the average speed of all cars in the scenario when using pure exploitation of the learned policy?
- How well is the actual reward given by said agent aligned with the ultimate goal of maximizing average speed?
- Did the agent converge to a policy or stay unstable?

The number 2000 for training episodes was chosen after trial and error, since most agents converged before 2000 episodes have passed. Agents that did not converge to a policy after 2000 episodes would most likely never experience convergence. 150 evaluation episodes were chosen because it is a large enough number to mitigate fluctuations caused by the random car spawning, which can sometimes be more favourable to agents.

3.4 Variability of Results

Reinforcement learning, especially when utilizing a DQN system, is a technique which can yield highly variable results on the same problem. Not only the reward function, but also the memory size, freeze rate, exploration rate, observation range, stack size and state representation can severely impact the result. The values used for these parameters in the experiments performed were held constant (except for the reward function) and can be found in appendix B. Also, using randomized inputs to avoid overfitting of the model on the exact values of the input dataset can add some randomness to results. Many graphs in this paper use trendlines showing the moving average of values attained instead of plotting the real values. The moving average shows the average value of the past n results for a window size of n. For highly variable results with a steadily improving average, the moving average will give better insight into the actual progress by filtering out the fluctuations in individual episodes. Whenever moving averages are used, standard deviation of the window mean σ is mentioned.

4 Reward Definition optimization

4.1 Defining success in a traffic setting

The ultimate goal when controlling traffic lights is to maximize the average speed of all vehicles.
Evaluation Performance: The average speed of all vehicles in the system
This can be computed by taking the average speed of each individual vehicle, and subsequently taking the average of all these values:

1. Average speed of a single vehicle:

$$Si_{avg} = \sum_{t=Ti_{enter}}^{Ti_{exit}} \frac{Si_t}{Ti_{exit} - Ti_{enter}} \tag{1}$$

2. Average speed of all vehicles:

$$\sum_{i=1}^{N} \frac{Si_{avg}}{N} \tag{2}$$

Definition of symbols used:

- N is the number of vehicles on the lanes the agent controls
- Ti_{enter} is the timestep on which vehicle i entered the simulation
- Ti_{exit} is the timestep on which vehicle i left the simulation
- Si_t is the speed of vehicle i at time t

The average speed of all vehicles in the system is a suitable metric because it gives the same importance to each vehicle, and it is inversely proportional to the average travel time in the intersection. This means that maximizing this average speed will minimize the average time needed to travel through the intersection controlled by the agent. Any traffic situation that would be seen as undesirable (driving slowly, standing still, repeatedly speeding up and braking to standstill) will decrease the average speed of vehicles. Furthermore, it is robust since it can not be tricked by traffic lights flipping the light rapidly, which is a problem when minimizing waiting time (Wiering, 2000). To compare the performance of different agents we will always compare them on the average speed of all vehicles, which will be referred to as "average speed" or otherwise as "eval performance" to indicate the performance of an agent during evaluation. However, the metric of average speed can not be used for the actual training of agents. An agent needs to be able to map a reward value to every state transition during every step of training, and the average speed of all vehicles in the system can only be calculated after the entire episode of training is finished. Because of this, there is a need for metrics that can be calculated for any state of the environment at every timestep, whose values estimate the average speed of all vehicles.

4.2 Metrics for estimating traffic success

Metrics to use as reward definitions were selected based on former research and a general understanding of the traffic control problem to estimate which metrics might be indicative of an intersection performing well (or poorly). Multiple metrics and combinations of metrics were drafted to be tested against each other.

The Speed Change metric grants rewards when vehicles speed up, and punishes the system for vehicles braking. When using a reward definition that corresponds to a certain metric, reward can be given in two ways. Directly using the performance metric as reward function (real rewards), or giving fixed rewards (penalties) when the metric is above (below) a certain threshold (bounded rewards). To gain insight into which method yields the best results, this metric was tested in two ways; a function where the positive/negative reward given was directly related to the speedup/slowdown of vehicles, and a function where a vehicle speeding up or slowing down gave a set reward.

$$RealSpeedChange : r_t = \sum_{i=1}^{N} rsc_i = \sum_{i=1}^{N} Si_t - Si_{t-1} \tag{3}$$

$$BoundedSpeedChange : r_t = \sum_{i=1}^{N} bsc_i = \sum_{i=1}^{N} \begin{cases} 0 & Si_t > Si_{t-1} \\ -0.5 & Si_t = Si_{t-1} \\ -1 & Si_t < Si_{t-1} \end{cases} \tag{4}$$

Where N is the number of vehicles on the lanes the agent controls and Si_t is the speed of vehicle i at time t.

In the work by Van der Pol & Oliehoek (2016), a reward function was introduced that combines the metrics of Delay (d), Hard Brakes (e), Jams (j) and Waiting Time (w), in addition to a constant penalty of -1 for flipping the traffic light (c) (to discourage rapid switching). Similar metrics are present in the work by Touhbi et al. (2017).

- Delay is defined as 1 - $\frac{vehiclespeed}{allowedspeed}$, giving a higher penalty for a lower speed.
- Hard Brakes are defined as moments when a vehicle's speed changes quickly (at a deceleration of more than $4.5m/s$).
- Jams are moments when multiple vehicles are standing still bumper to bumper. Jams were not investigated in this paper since they were very rarely observed and their occurrence correlated very strongly with a high waiting time in the system.
- Waiting Time is given as a reward of -0.5 for vehicles standing still for a single timestep, and a reward of -1 for vehicles standing still for more than that.

In addition to the individual metrics, the combination of factors as described by Van der Pol & Oliehoek (2016) was drafted, using different weights for each metric to account for the fact that the Jams metric was not used. The resulting reward functions look like:

$$Delay : r_t = \sum_{i=1}^{N} d_i = \sum_{i=1}^{N} 1 - \frac{Si_t}{Si_{allowed}} \tag{5}$$

$$HardBrakes : r_t = \sum_{i=1}^{N} e_i = \sum_{i=1}^{N} \begin{cases} -1 & Si_t - Si_{t-1} \leq -4.5 \\ 0 & otherwise \end{cases} \tag{6}$$

$$WaitingTime : r_t = \sum_{i=1}^{N} w_i = \sum_{i=1}^{N} \begin{cases} -1 & Si_t = 0 \wedge Si_{t-1} = 0 \\ -0.5 & Si_t = 0 \wedge Si_{t-1} > 0 \\ 0 & otherwise \end{cases} \tag{7}$$

$$ParameterCombination : r_t = 0.1c + 0.2\sum_{i=1}^{N} e_i + 0.3\sum_{i=1}^{N} d_i + 0.3\sum_{i=1}^{N} w_i \tag{8}$$

Where N is the number of vehicles on the lanes the agent controls.

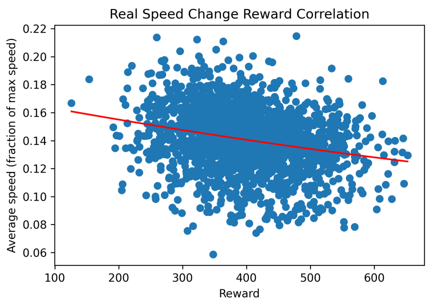
After promising results from the Speed Change and Waiting Time metrics, a combined metric using both values was introduced.

$$WaitingTime + SpeedChange : r_t = 0.1c + 0.5\sum_{i=1}^{N} bsc_i + 0.5\sum_{i=1}^{N} w_i \tag{9}$$

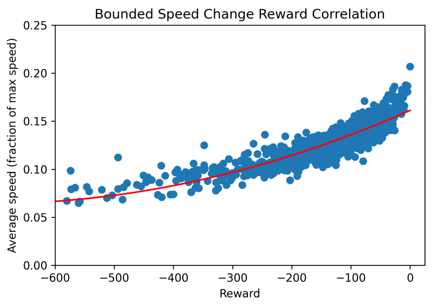
5 Results

5.1 Metric Reward Correlation

The following section contains reward correlation graphs. These figures show a scatter plot where each episode of training represents a blue dot, with its x-axis value representing the reward the agent got during said episode, and the y-axis value representing the average speed of cars during said episode. The red line is a 2nd degree polynomial automatically fitted to the dataset using a least squares equation. This graph can be used to compare how well the reward an agent receives is proportional to the actual average speed of all cars. A straight diagonal line means that more reward being granted directly contributes to a higher average speed of cars, which is ideal. This means that optimizing the agent's policy to maximize reward gained also directly optimizes the average speed in the system. Such a reward definition would be a perfect potential-based shaping reward. To identify which reward functions are accurate at estimating the average speed of the system, all reward functions used where plotted in this way.
The first functions tested in this way were the Real Speedchange and Bounded Speedchange functions discussed in section 4.2. In literature, it is more common to use rewards for individual observations that are bounded $r \in (-1, 0)$. This test shows the correlation between reward given by the Real Speed Change function and the average speed at the end of the episode. It also shows this information for Bounded Speed Change function.



(a) Real Speedchange Reward correlation

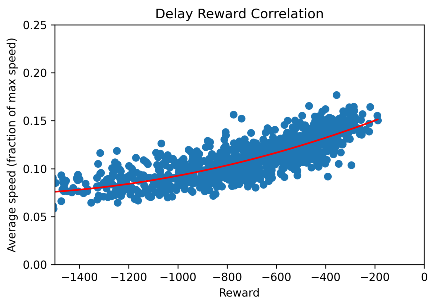


(b) Bounded Speedchange Reward Correlation

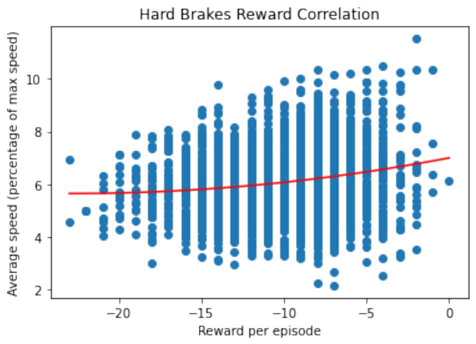
Figure 2: The reward correlation of Real Speedchange and Bounded Speedchange

Figure 2 shows that the bounded reward function has a much stronger correlation between the average speed in the system (y-axis) and the reward the agent receives (x-axis) than the real function. The real, unbounded function suffers from a lot of noise since a single car entering the system and speeding up can compensate for multiple slow driving cars having to stop because of a bad agent action. This suggests that bounded rewards, which have a stronger correlation with the average speed metric used in evaluation, should yield better results when used for training agents. This same noise also diminishes the learnability of the unbounded metric. Figure 5 confirms this in the sense that a more steady convergence can be observed for the Bounded Speed Change metric (5a), whereas the Real Speed Change metric (5b) does not see consistent improvement after the first dozen of episodes. It also suffers from unpredictabilty since the training rewards received remain unstable.

For all other reward definitions used, the correlation between reward gained from the reward function and actual average speed in the system was investigated following the same method.



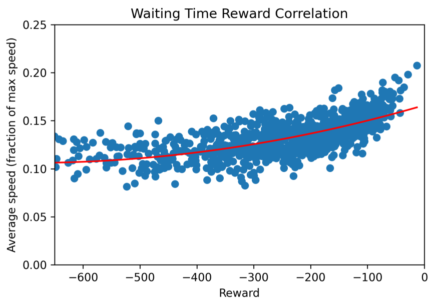
(a) Delay Reward correlation



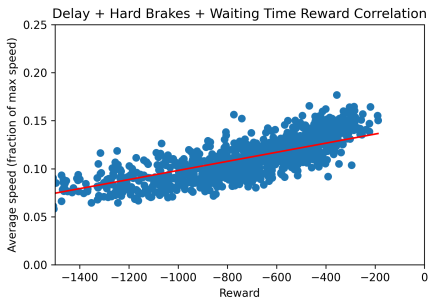
(b) Hard Brakes Reward Correlation

Figure 3: The reward correlation of Delay and Hard Brakes

It becomes apparent from Figure 3 and Figure 4 that Delay (3a), Waiting Time (4a) and the Parameter Combination (4b) reward functions all have rewards that estimate the performance of the agent in total average speed to a certain degree of accuracy. This can be seen in the fact that a higher reward per episode correlates strongly with a higher total average speed. For Hard Brakes (3b), there seems to be no apparent correlation between the reward given to the agent and the performance of the system. This suggests that using the Hard Brakes metric as a reward function for training an agent is unlikely to give a good result. This was later confirmed by the fact that the Hard Brakes metric as a reward function achieved the very worst average scores, as can be seen in Section 5.3.



(a) Waiting Time Reward correlation

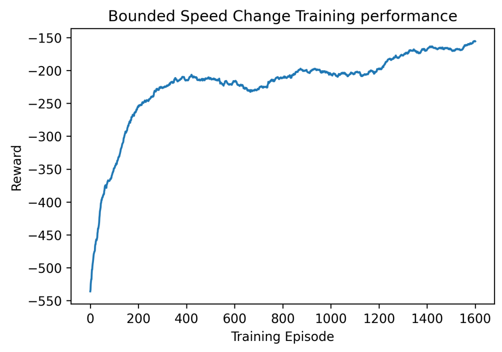


(b) Parameter Combination Reward Correlation

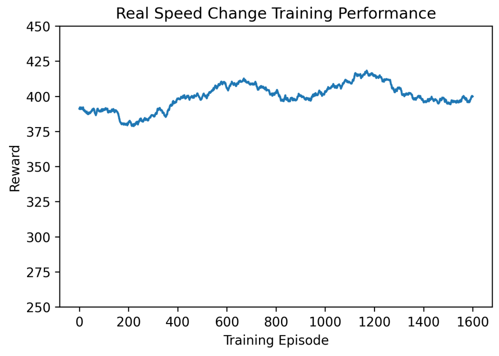
Figure 4: The reward correlation of Waiting Time and Parameter Combination

5.2 Metric Training Performance

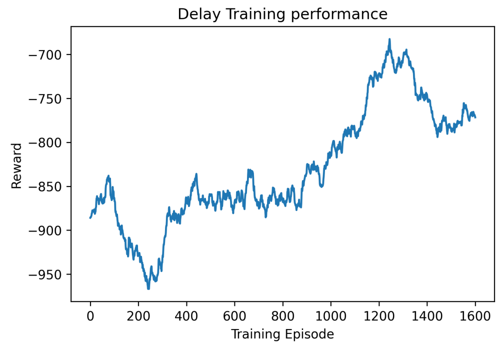
The training patterns of agents are shown in Figure 5. Agents were trained with the reward function listed for 2000 episodes (as described in detail in section 3.3). When an agent manages to quickly increase the reward it gains per episode, followed by a period during which its reward stays relatively stable, it has almost certainly converged to a policy. This is desirable as it means the agent can consistently respond to input in a proper manner and thus run the traffic in its intersection smoothly. Agents that do not seem to converge, such as the agent using the Delay metric as a reward (figure 5c) stay unpredictable and keep changing their policy.



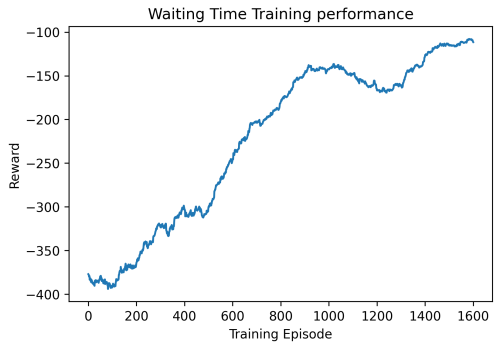
(a) Bounded Speed Change, $\sigma = 115$



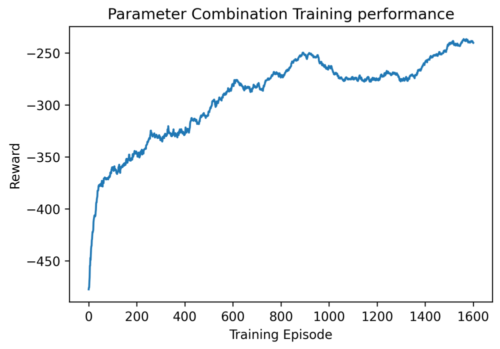
(b) Real Speed Change, $\sigma = 81$



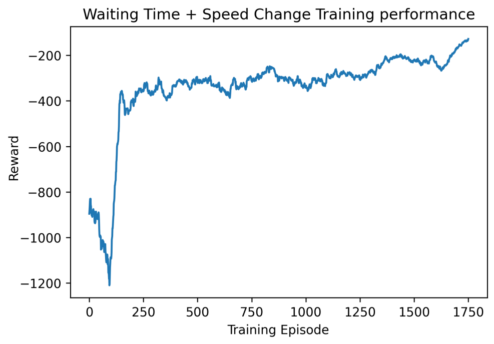
(c) Delay, $\sigma = 387$



(d) Waiting Time, $\sigma = 165$



(e) Parameter Combination, $\sigma = 141$



(f) Waiting Time + Speed Change, $\sigma = 222$

Figure 5: Moving average (window size = 100) of training performance for different reward metrics

5.3 Metric Evaluation Performance

Arguably the most important information, the change in average speed performance during training is shown in Figure 6. The average and best scores achieved during evaluation of the learnt policy is shown in Figure 7. Agents did not use the average speed in the system as a reward function during training, as it can only be calculated at the end of an episode. Agents optimized for different reward functions that were assumed to estimate the performance in terms of total average speed.

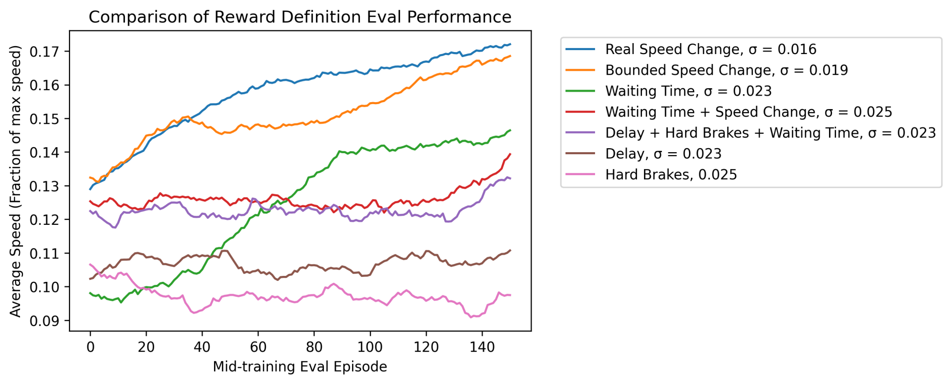


Figure 6: Change of Evaluation Performance during Training (moving average, window size = 50)

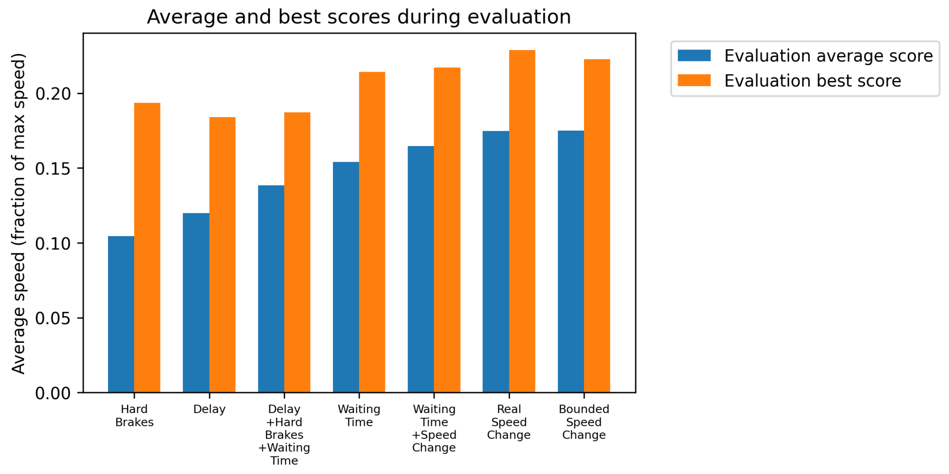
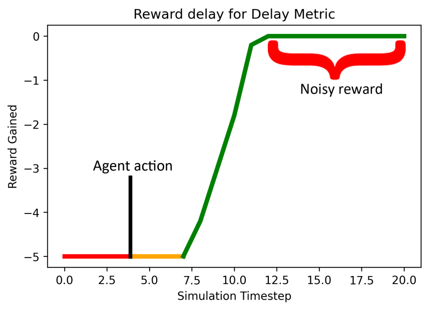


Figure 7: Average and best performance during Evaluation

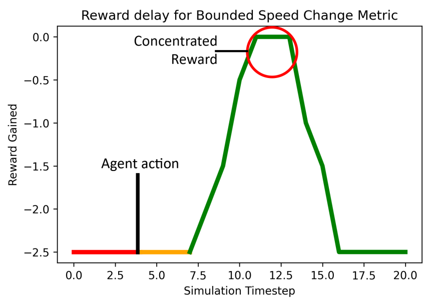
6 Discussion

Most metrics show convergence to some policy over the course of training, which can be observed when the training reward an agent receives stabilizes after some number of training episodes. The metric of Delay (Figure 5c) does not converge, which leads it to achieve the 2nd lowest average speed (as seen in Figure 7). Not reaching convergence suggests that the learnability of this metric is not high enough. When learnability is low, the agent can not tell how to choose actions that optimize reward, making it difficult or impossible to find a policy that consistently produces high reward. This can be explained by the fact that the delay between an agent's action (flipping the light) and the speeds of cars going up is rather large.

After observing low convergence of the Delay metric, a small scale custom scenario was created to investigate its learnability. Figure 8 shows the reward granted by the Delay and Bounded Speed Change metrics when five vehicles that are waiting at a traffic light get a green light. Figure 8 highlights that when an agent decides to switch which lane has a green light, it takes a number of timesteps (during which the light is orange), and then the vehicles on the new lane get to drive. However, it also takes these cars an additional couple of timesteps to accelerate to their maximum speed. After this acceleration is done, these cars keep granting rewards every timestep, since they are driving their maximum speed. This means that the reward gained from performing an action could be 10 or more timesteps in the future. Vehicles also keep contributing reward long after they have stopped being impacted by the agent. This is hard to optimize a policy for, since a large chunk of agent reward will be granted by cars that are not being directly impacted by the agent anymore. When comparing to a metric like Bounded Speed Change (Figure 5a), the impact of the delay from flipping the lights is halved because cars do start braking immediately when the light turns orange. Braking means speed going down, and these cars immediately start contributing negative reward. Additionally, cars start contributing their maximum positive reward as soon as they start driving and stop contributing reward after they have left the intersection and are cruising at the speed limit again. This concentrates any reward given to the system around cars that are either braking for a red light or speeding up when the light turns green, which are both closely linked to the agent's action. The learnability of the Waiting Time (Figure 5d) metric is high for this same reason, since the only cars contributing anything to the reward function are cars waiting at the traffic light.



(a) Delay metric



(b) Bounded Speed Change metric

Figure 8: Reward delay and noise for Delay and Bounded Speed Change metrics

From these results one can tell that agents using a reward function with a high learnability leads to more steady convergence to a policy. Learnability is defined by how much the reward an agent receives is dependent on the action it takes as opposed to environmental noise. Thus when learnability is high, agents can observe how their actions influence the reward they receive and more selectively pick actions that have high reward potential. Secondly, the results suggest that reward functions where the reward granted is strongly correlated with the total average speed achieve better results in terms of this average speed when agents converge to a policy. Thus, a good reward function is one that combines learnability with strong correlation to average speed.

The learnability of some reward function is difficult to directly calculate, as that necessitates being able to attribute every aspect of every observable state to the actions of the agent *or* to environmental factors *or* some combination thereof. It is thus more viable to theorize, like done above, whether a reward function has a high learnability or not. Correlation of granted rewards with the goal of maximizing average speed, on the other hand, *can* be calculated and observed quite easily, like was performed in section 5.1. The next section of this paper will explain how in further research, machine learning techniques could be used to find a reward function whose reward is perfectly correlated with maximizing average speed.

7 Future work

7.1 Runtime constraints

A strong -or even direct- correlation between reward granted by an agent and the average speed in the system guarantees that optimizing an agent's policy for high reward also directly optimizes the average speed of vehicles. This is an ideal potential-based shaping reward. In this research, the correlation between a function's reward values and the average speed in the system are determined empirically. Simulations were performed where both the reward value granted and the average speed in the system were saved. A scatter plot can then be made to get a graphical overview of how well a reward function estimates the total average speed. This gives a good visual overview but is far from conclusive for a number of reasons. First of all, there is a lot of variance in results achieved by a DQN when optimizing for traffic control. Even when using a very small traffic scenario with relatively predictable vehicle routing, an agent using the same reward function can converge to multiple different policies or none at all. Furthermore, training and testing an agent is very computationally heavy.

This means that accurately testing the impact of small tweaks, such as changing the individual importance of different factors in a combined reward function, is unfeasible. To gain insight into the effect of such a change, multiple DQN agents using the new configuration have to be trained and tested for thousands of episodes to filter out variations and randomness. Due to the high runtime involved with such a method of analysis, techniques like Sequential Model-based Bayesian Optimization as described in the work by Snoek et al. (2012) and extended on by Feurer et al. (2015) are not computationally feasible.

7.2 Linear regression on empirical data

To circumvent runtime constraints, a setup is needed where the training and testing of an agent is not needed to analyze the correlation between its reward and the average speed. In the final phase of this research, such a setup was devised. Results are limited, but to inform future research into this area, details will be described in this section. Using the same setup as described in section 3.3, an agent controls the intersection using a fixed-time policy. This means that one lane has a green light for 20 timesteps, the light switches to yellow for 3 timesteps and then the other lane has a green light for 20 timesteps, followed by yellow again. This pattern continually repeats itself. A fixed time strategy is chosen as it is highly predictable, and is likely to expose the traffic environment to most possible situations that would also occur when an agent would be controlling the intersection. In prior research, an agent with a fixed time policy is often used as a baseline to compare agent performance to (Shelby, 2004). This agent was put in charge of controlling the intersection for 3000 episodes, during which data was collected to determine:

1. The average speed during an episode
2. The reward values that *would be* granted to an agent by the four most promising reward metrics at the end of every episode (the Light Flip, Waiting Time, Bounded Speed Change and Delay metrics)

If the actual average speed could be expressed as some linear combination of reward values granted by different functions, there must exist some combination of individual weights that could be combined to form a reward function that is directly correlated with average speed. If this combination exists, it can be found using Linear Regression on a large enough dataset. Linear Regression is used to determine how one or more so-called response variable(s) change when predictor variable(s) are varied (Weisberg, 2005). Since this method of analysis is not the main focus of this research, linear regression itself will not be investigated in great depth. For readers wanting to know more, the work by Weisberg (2005) is extensive and contains many explanations and practical advise.

7.3 Setup

Using the PyTorch framework (Paszke et al., 2019), a simple neural network for linear regression was created. An entry i in the input dataset looked like:

$$([c_i, w_i, bsc_i, d_i], AS_i) \tag{10}$$

Symbols used:

- c_i is the reward granted by the Light Flip penalty metric at the end of episode i
- w_i is the reward granted by the Waiting Time metric at the end of episode i
- bsc_i is the reward granted by the Bounded Speed Change metric at the end of episode i
- d_i is the reward granted by the Delay metric at the end of episode i
- AS_i is the average speed in the system during episode i

The neural network model was trained to predict the average speed AS_i of a certain episode i by multiplying the values of the input set by some weights $[w_c, w_w, w_{bsc}, w_d]$ while adding bias term B to predict AS_i like

$$PredictedAS_i = B + (w_c * c_i) + (w_w * w_i) + (w_{bsc} * bsc_i) + (w_d * d_i) \tag{11}$$

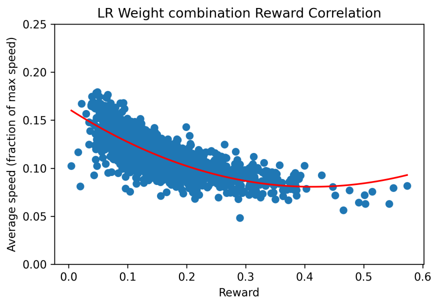
to minimize the Mean Squared Error (MSE) over the entire dataset as calculated by

$$MSE = \sum_{i=0}^{I} \frac{(PedictedAS_i - AS_i)^2}{I} \tag{12}$$

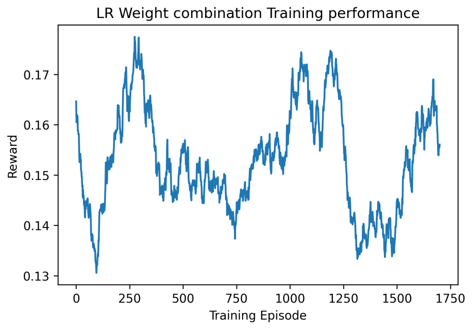
where I is the number of episodes in the dataset.

After running this model for 10000 epochs of training, the MSE was reduced to $MSE <= 0.0001$. The resulting weights and bias term can be found in appendix C, Table 4.

However, when training an agent with a reward function using the metrics and individual weights as calculated by the linear regression model, agent performance was sub-par, achieving an average speed of 0.1213, and the correlation between the reward given by the function and the actual average speed was inverse, meaning higher reward resulted in slightly lower average speed. This most likely means that the model came up with weights to overfit on the dataset, and the resulting weights do not achieve noteworthy performance in any other setting. This can be seen in Figure 9.



(a) LR Model Weight combination Reward Correlation



(b) LR Model Weight combination Training results

Figure 9: Training performance of an agent trained using the weights found by LR and Correlation of reward with average speed

7.4 Possible improvements

The linear regression model described above was introduced in the final phase of this research, mainly as a proof of concept. Deeper exploration of using regression for finding suitable weights is needed before conclusions can be drawn. Possible improvements to the model would include logging the reward achieved by the selected reward functions on **every** timestep instead of only at the end of the episode to detect how each timestep contributes to the average speed at the end of the episode. Furthermore, a larger dataset with more diverse policies controlling the traffic light could be used to reduce the risk of overfitting.

8 Conclusion

When embarking on this research, the central question was whether we could create a reward function that estimates the average speed performance of an agent. This is needed because, in the problem of traffic light control, the ultimate goal as explained in section 4 is to minimize travel time. Since this goal (favourable state) can fundamentally not be used as a reward function since it can not be calculated at every step of the simulation, every reward function used is trying to estimate it. A good reward definition for the traffic light control problem is thus a reward that is highly aligned with the goal, so that a policy that optimizes it also optimizes for travel time. A perfectly potential-based shaping reward function would be one where there is a fully linear correlation between the reward gained from the function and the average speed in the system. Such a reward function was not found in this research, but suggestions and proof-of-concept analysis was done in section 7 on how such a function could potentially be found.

8.1 The Optimal Reward Definition

In this research, no single "one size fits all" perfect reward definition was found. However, a recommendation can still be made. When designing a reward function for any kind of reinforcement learning setting, the learnability of the reward function and its alignment with the ultimate goal are the most important. Learnability was analyzed by looking at how steadily agents using a reward function would find convergence. A higher rate of convergence suggests a function with higher learnability. When a function would not cause an agent to converge, investigation was done manually by exploring the reward signal in different states of the environment to conclude what could hamper learnability. Alignment with the goal, in this research expressed as a correlation between reward gained from a function and the average speed of vehicles, *could* be systematically investigated. Most metrics discussed had quite strong correlations with average speed. The metric of Real Speedchange, although it performed well occasionally, was discounted on account of its low correlation and unpredictable rate of convergence.

$$WaitingTime + SpeedChange : r_t = 0.1c + 0.5\sum_{i=1}^{N} bsc_i + 0.5\sum_{i=1}^{N} w_i \tag{13}$$

In this research, the reward definition that ultimately had the best combination of stability and performance is the definition described above in Equation 13, as drafted in section 4.2, which combines the Waiting Time, Bounded Speedchange and Light Flip Penalty metrics. This reward definition achieved the 3rd highest average speed in the empirical tests (Figure 7), and by far the strongest convergence (Figure 5f). This is a new and more scientifically founded definition than was available in prior research, where reward definitions are not extensively compared for performance.

9 Responsible Research

This research paper surveys the effect of employing different methods of Reward Specification when using Reinforcement Learning to optimize the Traffic Light Control problem. All tests and experiments were performed as simulations only, in the publicly available SUMO (Behrisch et al., 2011). The traffic loads used in said simulations were created randomly based on parameters. The simulations performed by SUMO only simulate cars, and do not take into account any information about the driver, car type, model or make. The code used for running the experiments is available on request from Delft University of Technology, and was inspired by publicly available research papers referenced throughout this work. The seed used to generate randomness in the experiments is available in Apendix B. The results in this paper should be fully reproducible by any researcher, granted code access is given by Delft University of Technology.

References

Barker, M. M. (2001). *White paper european transport policy for 2010: time to decide.* Commission of the European Communities Brussels.

Behrisch, M., Bieker, L., Erdmann, J., & Krajzewicz, D. (2011). Sumo–simulation of urban mobility: an overview. In *Proceedings of simul 2011, the third international conference on advances in system simulation.*

Feurer, M., Springenberg, J. T., & Hutter, F. (2015). Initializing bayesian hyperparameter optimization via meta-learning. In *Twenty-ninth aaai conference on artificial intelligence.*

Hawkins, D. M. (2004). The problem of overfitting. *Journal of chemical information and computer sciences*, *44*(1), 1–12.

Kuyer, L., Whiteson, S., Bakker, B., & Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 656–671).

Ng, A. Y., Harada, D., & Russell, S. (1999). Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml* (Vol. 99, pp. 278–287).

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems 32* (pp. 8024–8035). Curran Associates, Inc.

Shelby, S. G. (2004). Single-intersection evaluation of real-time adaptive traffic signal control algorithms. *Transportation Research Record*, *1867*(1), 183–192.

Snoek, J., Larochelle, H., & Adams, R. P. (2012). Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems* (pp. 2951–2959).

Sutton, R. S., & Barto, A. G. (1998). *Introduction to reinforcement learning* (Vol. 135). MIT press Cambridge.

Touhbi, S., Babram, M. A., Nguyen-Huu, T., Marilleau, N., Hbid, M. L., Cambier, C., & Stinckwich, S. (2017). Adaptive traffic signal control: Exploring reward definition for reinforcement learning. *Procedia Computer Science*, *109*, 513–520.

Tumer, K., & Agogino, A. (2006). Agent reward shaping for alleviating traffic congestion. In *Workshop on agents in traffic and transportation.*

Van der Pol, E. (2016). Deep reinforcement learning for coordination in traffic light control. *Master's thesis, University of Amsterdam.*

Van der Pol, E., & Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016).*

Weisberg, S. (2005). *Applied linear regression* (Vol. 528). John Wiley & Sons.

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *Machine learning: Proceedings of the seventeenth international conference (icml'2000)* (pp. 1151–1158).

A Test Results

Table 1: Average and best performance during evaluation of agents with different reward functions

Test Results - Grid Scenario	Average speed	Best speed
Bounded Speed Change	.175	.222
Real Speed Change	.174	.229
Waiting Time + Speed Change	.165	.217
Waiting Time	.154	.214
Delay + Hard Brakes + Waiting Time	.138	.187
Delay	.119	.184
Hard Brakes	.119	.193

B Parameters

Table 2: DQNAgent parameters

Parameter	Value
num_frames	4
gamma	.0.99
learning_rate	2.5-4
batch_size	32
train_frequency	1
epsilon	0.1
double_dqn	True
encoder_type	large
memory_size	25000
freeze_interval	25000

Table 3: Grid Environment parameters

Parameter	Value
car_tm	100
car_pr	0.2
env seed	gridenv-2000

C Linear Regression Model Results

Table 4: Linear regression model

Parameter	Value
w_c	-9.5125e-06
w_w	7.3750119e-04
w_{bsc}	-1.0657244e-03
w_d	-3.3444860e-05
B	0.15374058
Evaluation average speed	0.1213
Mean Squared Error	0.00008