

TDK-dolgozat

Zoltán Barta

Smart Traffic Control

EÖTVÖS LORÁND UNIVERSITY

FACULTY OF INFORMATICS

DEPT. OF ARTIFICIAL INTELLIGENCE



Author:

Zoltán Barta

Computer Science MSc (2021-2024)

Supervisor:

Szilárd Kovács

PhD Candidate

Budapest, 2023

Contents

1	Introduction	4
2	Background	6
2.1	Traffic signal control	6
2.2	Reinforcement learning	8
2.2.1	Deep Q-Network	9
2.2.2	Advantage Actor-Critic	10
2.2.3	Proximal Policy Optimization	10
2.3	Decentralized RL-Control Systems	10
3	Methods and literatus review	12
3.1	Related work	12
3.2	Future Directions	14
4	Our approach	16
4.1	Actions and rewards	16
4.2	Observations and architecture	19
4.3	Experimental modifications	20
5	Implementation	22
5.1	Traffic Simulation	22
5.2	Reinforcement Learning models	23
5.3	Enivronment for the agents	23
5.4	Training Methods	23
6	Results	25
6.1	The test maps	25
6.1.1	Metrics	26
6.2	Single agent results	27

6.2.1	cologne1 and ingolstadt 1	29
6.3	Multi agent testing	31
6.3.1	3x3grid	31
6.3.2	4x4loop	33
6.3.3	4x4arterial	35
6.3.4	4x4grid	36
6.3.5	cologne3	36
6.3.6	cologne8	38
6.3.7	ingolstadt7	40
6.3.8	ingolstadt21	41
6.4	Future plans	43
7	Conclusion	44
A	The maps	45
B	Usage of AI tools	47
	Bibliography	48
	List of Figures	51

Abstract

Urban traffic congestion is an increasingly pressing issue and advanced solutions like intelligent traffic control systems are becoming unavoidable. This paper explores the application of reinforcement learning to enhance traffic flow and reduce congestion, using the Traffic Control Interface with the Simulation of Urban Mobility tool. Our goal was to develop an reinforcement learning based model that adapts to varying traffic conditions in real-time. The focus of our research is on scaling up the model and improving overall traffic management by using a method of sharing information that is similar to a cellular network. The results demonstrated notable enhancements in the efficiency of traffic flow, with reduced wait times and smoother movements of vehicles at key intersections. These findings underscore the potential of reinforcement learning in revolutionizing urban traffic control systems. The comprehensive analysis showed that smart traffic lights significantly enhance the efficiency of traffic systems, boasting improvements ranging from 10% to 80%. These intelligent controllers not only reduce waiting times but also contribute to environmental protection by reducing the carbon footprint. Combining smart traffic lights with advanced traffic management systems can lead to more adaptive, responsive, and sustainable urban traffic networks. These findings show that using smart technology can make a big difference in traffic management. It's a big step forward in transforming urban life, fostering cities that are more intelligent, environmentally sustainable, and well-connected.

Chapter 1

Introduction

Today there are approximately 1.4 billion vehicles on the road which number was estimated to be around 670 million in 1996 [1]. The enormous amount of vehicles and the ever increasing urbanization, puts intensive pressure on the transportation system, which leads to increased fuel consumption and air pollution. Most vehicles use fossil fuels, which are a major source of greenhouse gas emissions. Air pollution can cause serious health problems and can be responsible for up to 4.2 million [2] premature deaths each year. The increasing number of vehicles is accompanied by an increase in the number of urban traffic accidents. The escalating demand for public infrastructure is outpacing what can be feasibly achieved, given the existing social and economic limitations, leading to traffic congestion. Congestion also has serious economic consequences. Lost time, wasted fuel and increased operating costs all contribute to the burden on urban economies. In addition, congestion increases stress levels, which negatively affects people's health and well-being.

The only way to lessen the negative impacts of traffic is by making the most out of our existing infrastructure and planning our cities ahead of time to reduce the reliance on car use, to be more pedestrian friendly and walkable. To tackle this optimization problem Intelligent Traffic Management Systems (ITMS) have already been established, using technological tools and solutions to improve traffic flows, reduce congestion, and increase the efficiency of transport networks. ITMS typically systems uses sensors, cameras, and other data collection devices to monitor traffic conditions in real time. Using the data collected in this way, the systems can react quickly to changes in traffic conditions, for example by using dynamic signs and beacons to inform drivers of traffic conditions and rules. One major difficulty is that the task is extremely limited, since the minimum and

maximum duration of the green signal must be taken into account, and the regulation policy must be fair to all traffic directions, including side directions, in order to cope with the so-called "starvation" problem, when a particular lane doesn't get a green light for a long period of time.

To achieve this optimal policy reinforcement learning is a popular approach. In this type of solution, we differ two application methods in traffic management: decentralized distributed systems and centralized systems. In a centralized traffic management setup, one agent takes charge of all traffic lights, theoretically facilitating the attainment of a global optimum due to the uniform control of lights throughout the network. However, the practical implementation of centralized systems presents challenges, especially as the span of the area covered broadens, leading to potential slowdowns in the system and scaling difficulties. Managing expansive areas and numerous traffic lights demands substantial computing resources. Moreover, the central coordination may result in delayed system responses to the ever-changing traffic conditions. Furthermore, a malfunction in the central agent can disrupt the entire system, thereby elevating the risk to the system's reliability.

In a decentralized system, different agents control the lights, where each agent manages a single light, creating a multi-agent reinforcement learning (MARL) situation. In this type of system, the agents make decisions primarily based on local traffic conditions, which allows for flexibility in responding to changing traffic conditions and is generally easier to scale, since the addition of new lamps or groups of lamps does not significantly increase the central computational burden. However, because agents have limited information about global traffic conditions, decentralized systems have difficulty achieving the global optimum, and coordination between lamps can be challenging, especially when agents need to share information about global traffic conditions. In addition, decentralized systems have lower computational requirements for each agent as they only manage a smaller area, which allows for faster decision-making.

Overall, decentralized traffic management systems favor local optimization and flexibility over global optimization and require added communication and coordination mechanisms for efficient cooperation between agents. In this paper, we will deal with a decentralized solution that achieves such agents, that can approximate the global optimum, maintain their low computational cost, flexibility, and scalability, and ready to be applied in a real-world scenario.

Chapter 2

Background

In this chapter, we will introduce the underlying concepts and technologies of this paper focusing on signal control and aspects of reinforcement learning.

2.1 Traffic signal control

Traffic signal control systems are required to address various rules and standards to ensure safety and efficiency. These regulations are geared towards promoting road safety, enhancing the flow of traffic, and maintaining legal adherence. Below, visualized by Figure 2.1, are some crucial aspects and principles that a traffic signal management system needs to consider:

1. **Safety Rules:**

- The timing of the traffic signal shall ensure the safe passage of pedestrians and vehicles at intersections.
- A sufficient source of light shall be provided for traffic signals to allow drivers and pedestrians to detect signals in a timely manner.
- Timing should comply with local and national road safety regulations.

2. **Congestion optimization:**

- Traffic signal cycles and timing should optimize traffic flow and reduce congestion.
- The system must be able to adapt to changing traffic conditions, such as peak traffic periods.

- Traffic signal systems should be able to communicate with each other and with other traffic management systems.

3. Technological Integration and Data Gathering:

- The system has to be capable to examine data closely to measure performance and support continuous improvements.

4. Emergency Accessibility and Responsiveness:

- The system should have the capacity to swiftly react to emergency situations, for instance by ensuring priority access for law enforcement, and fire or rescue vehicles.
- The system should endorse emergency accessibility and prompt response mechanisms.

The principles mentioned earlier are fundamental to the design and functioning aspects of a traffic signal control system, contributing to enhancing the safety and efficiency of transportation networks.

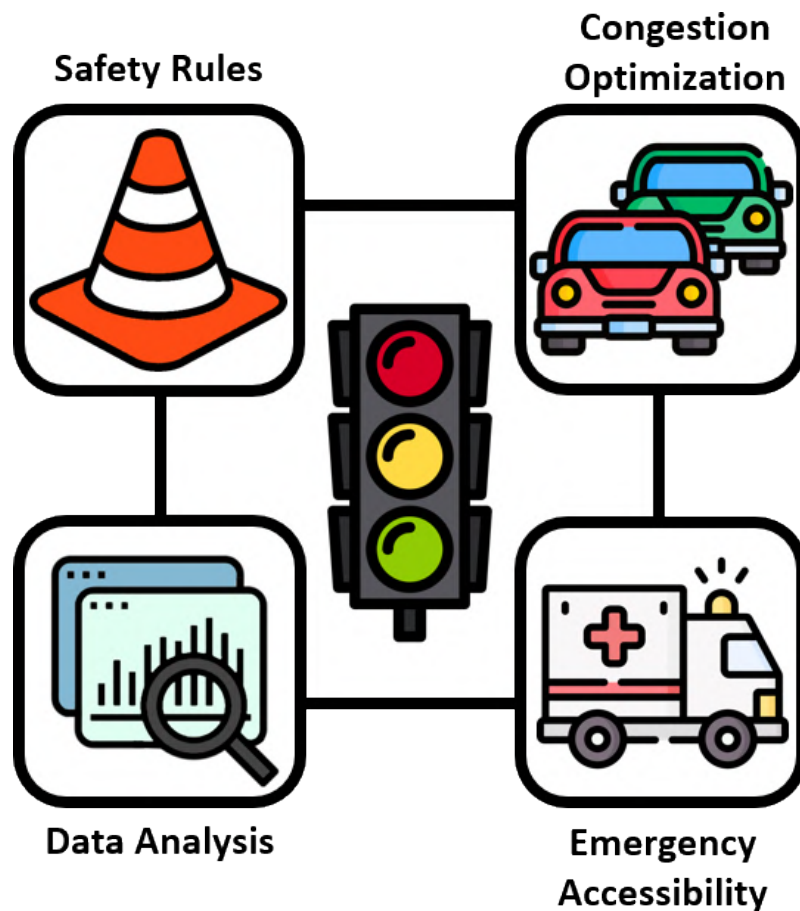


Figure 2.1: The main components of a traffic control system.

2.2 Reinforcement learning

Reinforcement Learning (RL) is the branch of machine learning where a smart agent engages with an environment to learn the strategy of maximizing the reward over a period while executing a specified task. The core concepts of RL include:

1. Agent:

- An agent represents a learning and decision-making entity capable of acting within an environment.

2. Environment:

- The environment signifies the scenario in which the agent interacts and learns.

3. State:

- A state provides a snapshot of the environment at a particular instance.
- Can be a picture or a vector describing the environment.

4. Action:

- The possible step(s) or decisions that the agent can undertake in a particular state.
- Can be discrete or continuous.

5. Reward:

- Reward is feedback presented to the agent concerning the efficacy of its actions with the objective of RL being the maximization of cumulative rewards over time.

6. Policy:

- A policy outlines a strategy dictating the choice of action the agent makes in a given state. It can be deterministic or stochastic.

7. Q-value:

- The value of a given state-action pair.
- Initially not known, but approaches the real value as the agent learns.

8. Learning Rate and Discount Factor:

- The learning rate decides the degree to which the agent revises its value estimations based on new experiences.
- The Discount Factor evaluates the current worth of upcoming rewards and helps the agent in deciding whether to prioritize short-term rewards or long-term rewards.

A reinforcement learning agent learns, similar to human babies, by trial and error, or even learn from other agents previous experiences. The agent's main goal is to maximize the expected reward. The initial policy of the agent is usually random, due to general usability and stronger exploration. To achieve the maximal reward and approximate the optimal policy, the agent has to explore and use the gathered knowledge. For this the most popular strategy is the ϵ -greedy strategy, where the agent takes a random action with the probability ϵ for the sake of exploration, and otherwise takes the action with the maximum Q-value, to exploit its knowledge.

In this paper, we focus on three algorithms, with three different approaches to learn the optimal policy. All three will be further discussed below. Every one of them can handle discrete action spaces.

2.2.1 Deep Q-Network

In Q-Learning, we try to estimate a given Q-value by storing the experienced state-action pairs in a Q-table and updating it continuously. When the Q-table is too extensive and infeasible to compute, a neural network can be used to predict the value of a state-action pair, hence the name Deep Q-Network (DQN). This method is model-free, which means that the agent does not have to know every state of the environment to achieve an optimal solution. Also off-policy method which means that the agent does not use the same policy for learning and acting, in this way it can learn from other policies, not only from the one it uses to maximize the reward function. A pivotal feature of DQN is experience replay, which employs a memory buffer to retain past experiences, utilizing them to refresh Q-values in small batches to enhance learning.

2.2.2 Advantage Actor-Critic

Advantage Actor-Critic (A2C) is also a model-free solution, but an on-policy algorithm that uses an actor and a critic component. The on-policy means that the agent uses the same policy for acting, and continuously updating and optimizing it during learning. The actor is responsible for choosing actions in states, while the critic evaluates these actions and provides feedback to the actor. A2C tries to improve the speed of learning by balancing between evaluation and policy updating.

2.2.3 Proximal Policy Optimization

The Proximal Policy Optimization (PPO) is a model-free, on-policy algorithm that attempts to facilitate the learning of policy gradient algorithms. PPO tries to learn a new policy that does not deviate too much from the previous policy and constrains the rate of policy update by a hyperparameter called the clipping factor. This helps avoid too large jumps in the policy space and improves the stability of learning. PPO is easy to implement and scale and works well in different environments and tasks.

2.3 Decentralized RL-Control Systems

In Intelligent Traffic Control Systems (ITCS), such as in Figure 2.2, a very popular approach is to redistribute the time between green phases in a full cycle. A full cycle

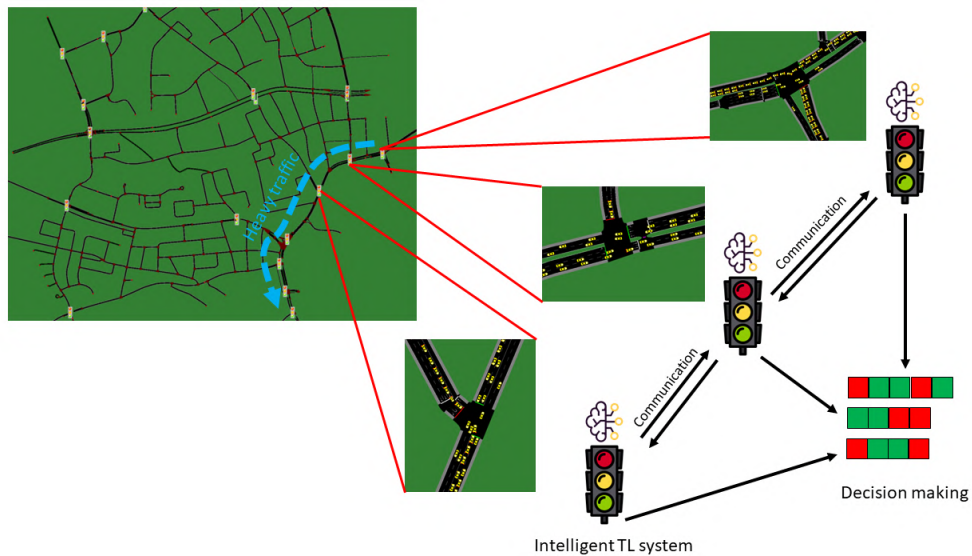


Figure 2.2: An intelligent traffic management system in action.

means that all of the green phases will be selected in the same order, ensuring that every lane gets a green light occasionally. This method is more dynamic than the original fixed traffic signal cycle, one degree less predictable but yielding better results as seen in [3], however, better optimization can be achieved by less rigid frameworks. In this paper we let our agents select the next phase in the next tick of the simulation. A fair comment that in this case, it is possible to induce a situation where one lane never gets its green phase, but that is less likely to happen since the main goal is to optimize uneven load distribution at intersections, where really high demand is present for every green phase.

Chapter 3

Methods and literatus rewiew

In this chapter we go through the state of the art methods for intelligent traffic controls and explore the relevant papers of the field.

3.1 Related work

Intelligent Traffic Control Systems have undergone significant advancements with the integration of sophisticated learning techniques and the growing availability of urban traffic data. Contemporary research in this field is predominantly centered around the application of RL to optimize urban transportation efficiency and adaptability. This section compiles pivotal findings from recent studies, showcasing innovative methodologies and their potential impact on the future landscape of ITCS.

Starting with the work of Mădălin-Dorin Pop [4], we see an exploration of traffic light optimization using an agent-based modeling approach with AnyLogic. This study focuses on reducing congestion and crossing times at intersections, demonstrating the potential of finding the most optimal durations for traffic light phases, for various intersections.

Vicente N. de Almeida, Ana L. C. Bazzan, and Monireh Abdoos introduce a multi-agent reinforcement learning approach with the k-nearest neighbors (k-NN) [5]. Their approach, avoids state discretization, by estimating Q-values based on the weighted average of the nearest previously visited states. This study introduces a novel approach to traffic signal control using MARL and k-NN, successfully addressing the challenges posed by large and continuous state spaces. The method demonstrates efficient learning and adaptability, suggesting significant potential for real-world traffic control applications.

The research by Guanjie Zheng et al. [6] examines the use of RL for traffic signal control, specifically focusing on optimizing the design of reward and state representations. The paper proposes re-examining RL approaches in traffic signal control through the lens of classic transportation theory, resulting in a method named "Light-IntellighT" (LIT). LIT is designed to be simple yet effective, theoretically grounded in classic traffic signal control methods such as SCATS and SCOOT [7], and demonstrates superior performance in experiments

Guanjie Zheng in another research, developed the Flip and Rotation and considers All Phase configurations (FRAP) model [8], based on phase competition, where priority is given to traffic signals with higher demand. One of the notable advantages of FRAP is its invariance to traffic flow variations, such as flipping and rotation, common in real-world scenarios. For instance, traffic patterns often reverse between morning and evening commutes. Experiments show that FRAP models trained on a specific traffic flow can adapt well to reversed or rotated flows without needing re-training, unlike previous RL methods. This feature significantly increases the model's efficiency and adaptability, reducing training costs and time.

Chacha Chen et al. [9] paper on decentralized deep reinforcement learning introduces a novel approach for multi-intersection traffic signal control, by designing RL agents (based on the FRAP model) that utilize the concept of 'pressure' for signal coordination. The study includes extensive experiments, including a real-world scenario with 2510 traffic lights in Manhattan, New York City.

A less complex model by Selim Reza et al.'s study [10] focuses on adaptive traffic signal control (ATSC) for autonomous vehicles using multi-agent systems and the SARSA(λ) model. It addresses inefficiencies in weight updating mechanisms of conventional SARSA(λ) models by proposing a Gaussian function for regulating the eligibility trace vector's decay. Additionally, it suggests using MaxAbs scaling for state values instead of MinMax normalization for better efficiency and robustness. The study demonstrates significant performance improvements in traffic simulation using the SUMO package, with substantial reductions in average total waiting time and stops, and an increase in average trip speed. The resulting algorithm showed a 40-80% improvement in total stops at traffic lights.

MetaLight is a novel framework for traffic signal control by Zang [11] using deep reinforcement learning. It enhances learning efficiency in new traffic scenarios by leveraging of real-life traffic data available from sources like surveillance cameras. Utilizing

a gradient-based meta-learning algorithm, MetaLight incorporates individual-level and global-level adaptation, significantly outperforming traditional methods and state-of-the-art RL models like FRAP in traffic signal control.

Another take on a more explainable policy was by James Ault in his paper [12], where he introduces policies, combining the efficiency of RL with the interpretability of traditional control functions, show potential in approximating optimized policies as effectively as deep neural network-based controllers. The paper presents and analyzes several online optimization techniques for tuning understandable control functions, demonstrating that a properly designed policy function can be as effective as a deep neural network in optimizing signal actuation policy. A variant of the Deep Q-learning algorithm, Deep Regulatable Hardmax Q-learning, is shown to be particularly effective, reducing vehicle delays by up to 19.4%.

Pravin Varaiya's paper [13] discusses the control of networks of signalized intersections using a Max Pressure (MP) control strategy. In this model, vehicles arrive at intersections as independent identically distributed streams. The network evolves as a controlled store-and-forward queuing network, where the control at each intersection selects a 'stage' that actuates a set of simultaneous vehicle movements. The MP control stabilizes the network if the demand is within a feasible set, thus maximizing network throughput. This approach requires local information at each intersection and provably maximizes throughput, distinguishing it from other traffic control methodologies.

PressLight [14], by Hua Wei, is a deep reinforcement learning based traffic signal control system that integrates the MP control method. The approach seeks to optimize traffic network throughput, effectively minimizing overall travel time. PressLight's design is grounded in transportation theory, focusing on maximizing network throughput by minimizing the pressure at intersections, defined as the difference in vehicle numbers between incoming and outgoing lanes. Comprehensive experiments demonstrate that PressLight outperforms both conventional transportation methods and existing RL-based approaches, offering more efficient traffic signal plans and shorter learning processes.

3.2 Future Directions

Current research indicates a strong trend towards hybrid approaches that blend advanced RL techniques with established traffic control theories, like max pressure control, for better adaptability and efficiency. Moreover, decentralized methods have proven to

be much more effective and usable compared to centralized methods, as they are more modular and easily handle changes in traffic patterns, such as the addition of a new lane.

Chapter 4

Our approach

In this chapter we write about the formulation of the RL task and the theoretical aspects of our proposed solution for this optimization problem.

4.1 Actions and rewards

The first task at hand is to formulate the optimization problem to a RL-compatible one. At first, we thought that we could train the agent to predict the length of the next phase, but as the timing is a continuous variable, much harder to estimate. However, we can achieve essentially the same effect, by predicting the phase for the next moment. This is computationally more expensive than the previous method, but more responsive too. Since the possible phases are finite we can map it to a discrete action space, making the learning easier. We can further prune the action space by only considering the non-conflicting green phases. For example, a two-way intersection has $|A| = 4$ actions, which can be seen in Figure 4.1. Between every phase change, there is always a yellow phase, with a fixed length, which the agent cannot control. For the sake of less arbitrary modifications,

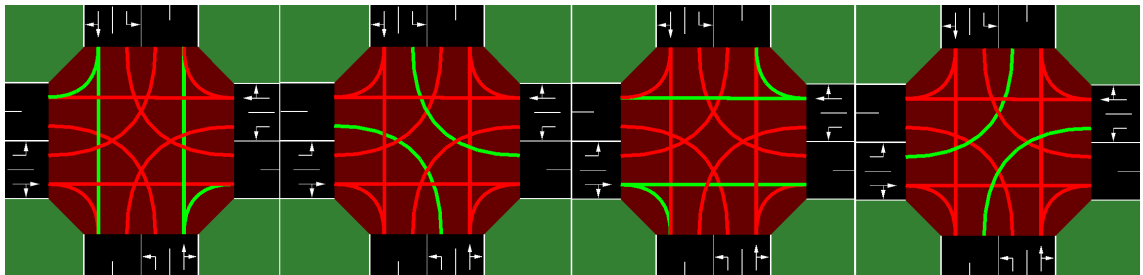


Figure 4.1: Non conflicting green signals [15]

at first we let the agent control all of the phases, even the transitions. However we later

scraped the idea, because as we began the initial training of our agents, they achieved the anticipated results, outperforming traditional controllers in situations where the traffic is mild, and the flow is more dynamic, but as soon as a real traffic jam is occurred, the agents started to prefer the yellow phases. This reaction was understandable, since some reward functions are calculated with the help of temporal differences, and in the case of a traffic jam, there is little change from one moment to the next, so the reward was also close to zero. Instead of letting the agents have control over the transition phases, we introduced a parameter to indicate whether a yellow phase was currently active during a decision-making moment

For reward functions, we used 4 different metrics:

1. **Difference in Waiting Time (*diff-waiting-time*)**: The reward is proportional to the difference in accumulated waiting time since the last measurement. It calculates this by taking the difference in total waiting time accumulated between measurement intervals.
2. **Average Speed (*average-speed*)**: The reward is proportional to the average speed of vehicles passing through the intersection. If there are no vehicles, the average speed value is maximal, i.e., 1.0.
3. **Queue Length (*queue*)**: This reward is the negative value of the total number of halted vehicles in the intersection.
4. **Pressure (*pressure*)**: The reward is calculated as the difference between the number of vehicles leaving and the number of vehicles approaching the intersection, thus measuring the 'pressure' of traffic.

During the initial training we saw that the agents tend to prefer longer phase durations, to mitigate this problem and to encourage more frequent phase changes we introduced a penalty for maintaining a phase for too long. This penalty acted as a wrapper for any of the four reward functions. The penalty formula, shown on 4.1 and 4.2, was carefully designed to allow a minimum phase duration to avoid too frequent switching, but beyond this threshold, the penalty grew exponentially which can be seen in Figure 4.2. This approach ensured a balance between phase duration and the need for phase changes, optimizing traffic flow and preventing the agents from getting trapped in less efficient traffic light patterns.

$$p_{duration} = -\max(0, (d - d_{min})^3 \times m) \quad (4.1)$$

$$r_{total} = r_{base} + p_{duration} \quad (4.2)$$

where

- d is the duration of the current phase.
- d_{min} is the minimum duration of a green phase.
- m penalty multiplier.

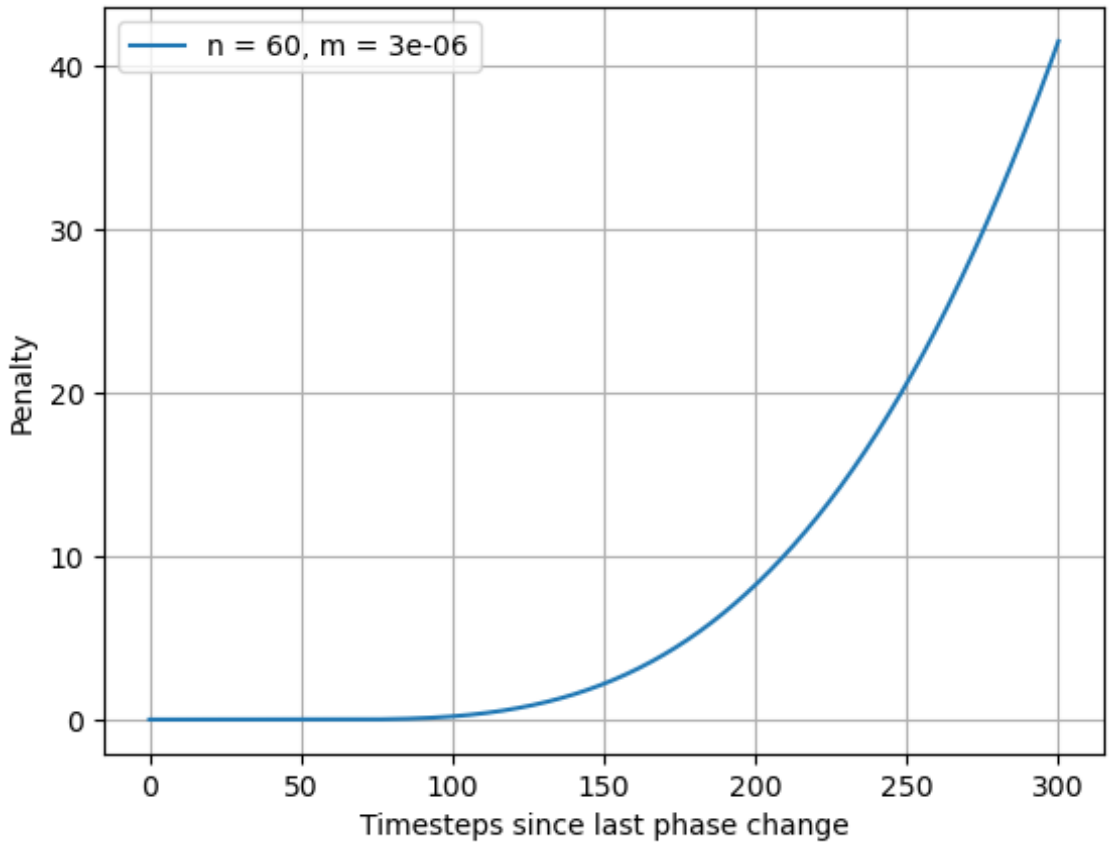


Figure 4.2: The penalty for not changing phase.

4.2 Observations and architecture

The observation space in most RL-based solutions are the incoming lanes at an intersection and various information about them, number of cars currently waiting, lane length etc. Usually this type of observation space is customized to individual traffic lights and can be trained relatively quickly, typically within about 30,000 simulation steps. While these agents are excellent for optimizing traffic flow at their specific locations, their initial focus on local optimization does not necessarily result a reduction in overall system load. To address this, we expanded the observation space of the agents to resemble a 2x2 intersection with as many lanes as possible. This design created a more universal observation space, allowing the same observation space model to be applied effectively to various traffic lights, because it can map the most complex intersections as well as the simple ones. The architecture can be seen in figure 4.3. Although this method extended

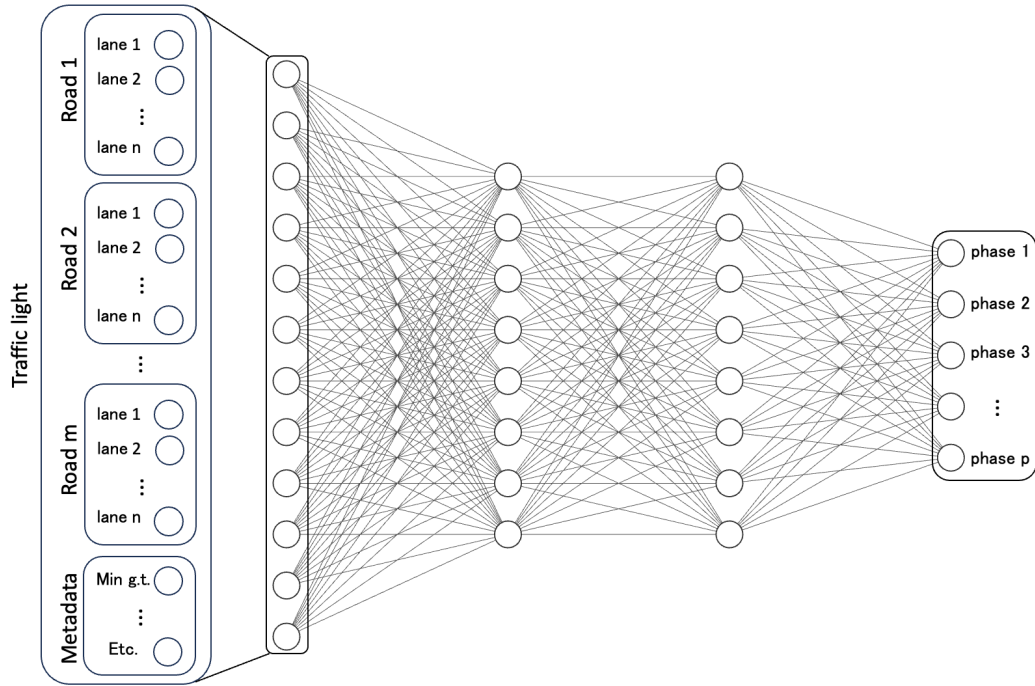


Figure 4.3: The architecture with the padded input.

the training time by a little bit, its unavoidable for the next step towards decentralization. To encourage communication between agents, we expanded the data input for each traffic light beyond its own observation space to include the observation spaces of nearby lights. This strategy was implemented using the previously mentioned padded observation space concept. As a result, traffic lights were equipped not only to make decisions based on

local traffic conditions but also to consider the surrounding traffic flow as well. The final proposed architecture is shown in figure 4.4.

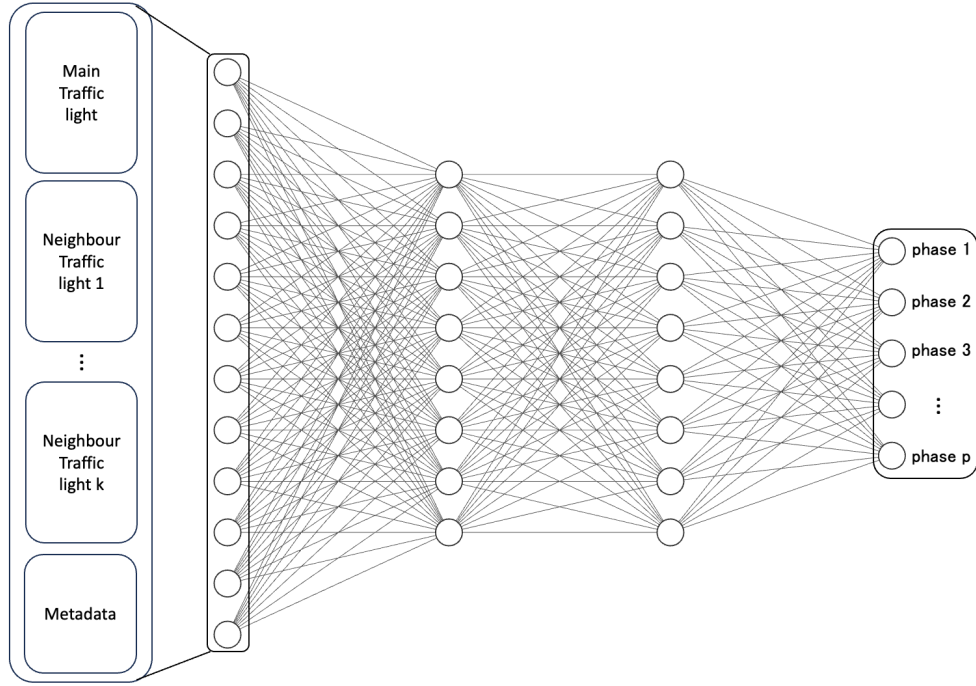


Figure 4.4: The final architecture.

4.3 Experimental modifications

During the research we experimented with other representations of the action space, by trying to mask the possible actions or map an action into another action space. The goal with this representation was to achieve an universal type agent, that can be trained and applied to any type of traffic signal. Unfortunately this was impossible to achieve with the current framework, because of some inner workings of the simulation software. The main problem was that we constructing the network, and the traffic lights, the order of incoming roads is not consistent, it depends on how the network file was structured. This order determines also the phase encoding structure for the traffic light. We can reorder the intersections by direction shown on figure 4.5, even the trickier ones like on figure 4.6, still the links wont be updated in the phase structure, so the actions wont be consistent through different types of intersections.

A real solution would be to correctly link the phase encoding to the given lane similar to fig 4.7, but this reaches the technological limitation of the current framework, which we working with.

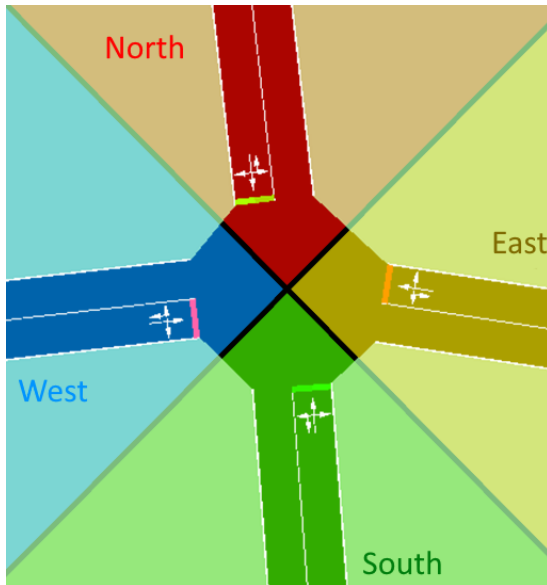


Figure 4.5: Order by direction case 1.

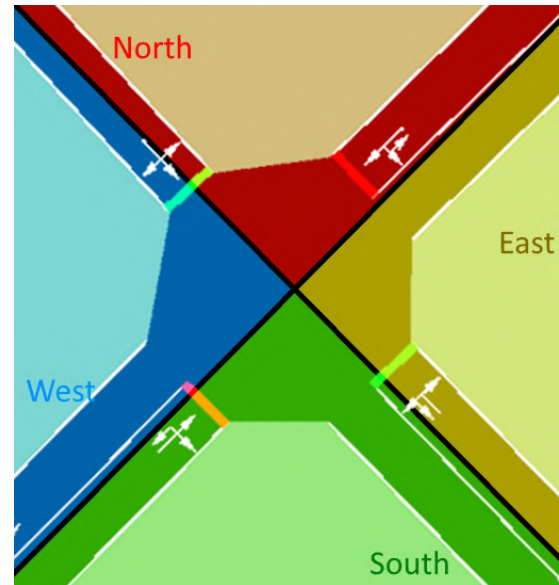


Figure 4.6: Order by direction case 2.

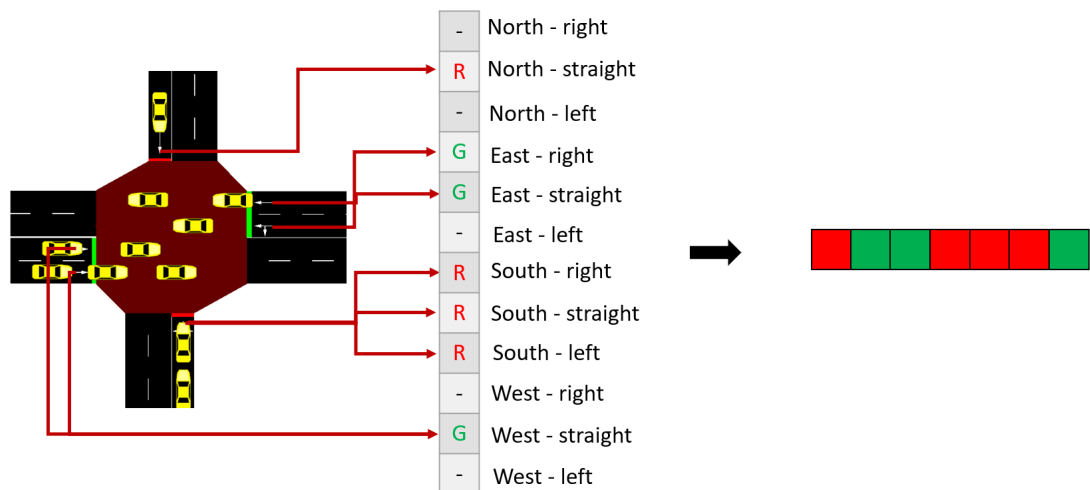


Figure 4.7: An encoding that links the lanes with the phases

Chapter 5

Implementation

In this chapter we present the technical implementation of the previously introduced intelligent traffic control system.

5.1 Traffic Simulation

Simulation of Urban MObility (SUMO) [16] is an open-source traffic simulation tool used in various domains like traffic planning, transportation system analysis, and research. A critical component of SUMO is the Traffic Control Interface (TraCI), which significantly enhances the software's capabilities by allowing dynamic interaction during simulations. This flexibility is vital for testing and implementing traffic management strategies under various conditions. TraCI is capable of data retrieval from simulations, like vehicle positions, speeds, emission levels, and the status of traffic signals. TraCI supports multiple programming languages, including Python, Java, and C++. This feature enables the automation of simulation tasks and allows for the integration of SUMO with other tools or custom algorithms. This data is crucial for comprehensive traffic analyses, understanding vehicle behaviors, and assessing the overall network performance. Despite its numerous advantages, TraCI is not without limitations. The quality of the simulations heavily relies on the accuracy of the input data and the underlying models in SUMO. There's also a learning curve associated with effectively utilizing TraCI's capabilities, especially when it comes to integrating it with other software or developing complex control algorithms.

5.2 Reinforcement Learning models

Stable Baselines3 (SB3) is versatile and rich reinforcement learning library, offering a range of PyTorch-based implementations of RL algorithms. It is known for its simplicity and ease of use, making it accessible for both novices and experts in RL. The library includes the A2C, PPO, and DQN algorithms, which we used in our research. SB3 supports custom environments, which was essential for our case. The Tensorboard [17] integration makes possible to visualize various training and evaluation processes. For hardware utilization, we can use parallel processing, if the environment supports it. Currently the library, doesn't support multi agent reinforcement learning, we also experimented with other libraries like Ray RLlib [18], which supports multi-agent training, but integrating it with SUMO proved to be challenging. Instead, we found a workaround in SB3 using custom modules.

5.3 Environment for the agents

The SUMO-RL [15] framework by Lucas Alegre played a big part in our research, because it made it possible to integrate various RL libraries with SUMO, using Python. This repository provides an environment compatible with OpenAI Gym [19], and with PettingZoo [20]. The basic structure of the environment in SUMO-RL includes the simulation itself as an environment, and the traffic lights as the agents. We implemented various custom modules to this framework to expand our possibilities, for example: PaddedObservation, SingleMultiAgentEnv (where all lights are intelligent, but only one learns), etc.

5.4 Training Methods

We trained our agent progressively, by slowly increasing the scale of the traffic in the simulation. With this technique, the agents grasped the baseline patterns in the initial training steps, and as the situation became more difficult, the agents could develop and fine tune a more sophisticated and optimal control policy. Besides this progressive training, we also trained the agents on different maps, on different intersection types, that were compatible. We call two intersections compatible, if their actions space is the same and their lane layout is similar, shown on figures 5.1, 5.2 and 5.3. To increase the effectiveness

of the training, we also applied learning rate decay, and early stopping if the evaluation results were not increasing after a set number of episodes. Initially the agents were trained in environments, where the other traffic lights are not intelligent, and were fine tuned in an environment where all the traffic lights were controlled by other agents, using our previously mentioned custom modules. This way we could achieve a multi agent-like training with SB3.



Figure 5.1: Similar intersections with $|A| = 2$

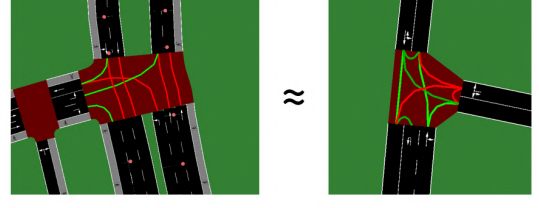


Figure 5.2: Similar intersections with $|A| = 3$

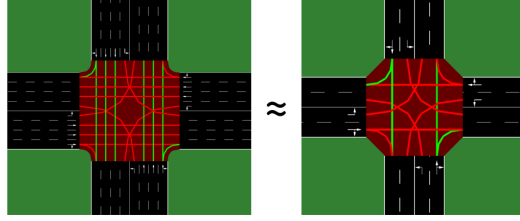


Figure 5.3: Similar intersections with $|A| = 4$

Chapter 6

Results

In this chapter we present the test maps and compare the results of our agents against the fixed original controllers as the baseline.

6.1 The test maps

As the final test for our agents and comparing with the state-of-the-art methods we used RESCO: Reinforcement Learning Benchmarks for Traffic Signal Control [21]. This repository is a significant resource for researchers and engineers working in the field of traffic signal control. It provides a comprehensive benchmarking toolkit for the development and comparison of RL-based traffic signal controllers. The simulations are specifically configured in SUMO, and representing real world scenarios in Cologne, Ingolstadt with different versions and some other specific maps. Beside the RESCO, other maps were used too for extensive testing and evaluation. On Table 6.1 we can see the most important setup parameters for the maps.

Important to note that the scaling parameter does not necessarily correlate with the density of traffic of different maps, we manually adjusted the scaling for each map. This discrepancy is largely due to the varying number of lanes in different maps. Consequently, while a scaling value of 2.0 might result in total gridlock in some maps, it might barely affect traffic in others. To ensure consistency, we calibrated the scaling such that the starting value represents normal traffic conditions, and the final value is set at a level where congestion becomes evident on the specific map. We used similar measurements as in Figure 6.1 to determine the scale. When the traffic is low, the waiting time only starts to accumulate when a particular road receives a red light, and it immediately decreases

Net file	Simulation length (sec)	Begin time (sec)	Scale initial	Scale final	Number of traffic lights
3x3grid	2500	0	1.1	1.8	9
4x4loop	750	0	0.7	4.0	7
arterial4x4	2500	0	0.2	1.2	14
grid4x4	1500	0	4.0	8.0	16
cologne1	4000	25700	1.0	2.0	1
cologne3	4000	25700	1.1	2.5	3
cologne8	4000	25700	1.7	3.7	8
ingolstadt1	4200	57600	1.0	4.0	1
ingolstadt7	4200	57600	1.7	2.8	7
ingolstadt21	4200	57600	1.0	1.5	21

Table 6.1: Example Traffic Simulation Parameters

as soon as there's a green light, which we can see as spikes, illustrated by Figure 6.1a. However, in the case of a traffic jam, as shown in Figure 6.1c, the waiting time does not start to decrease at the green light, because the vehicles cannot move. Therefore, the spikes completely disappear, and we only see an exponential increase.

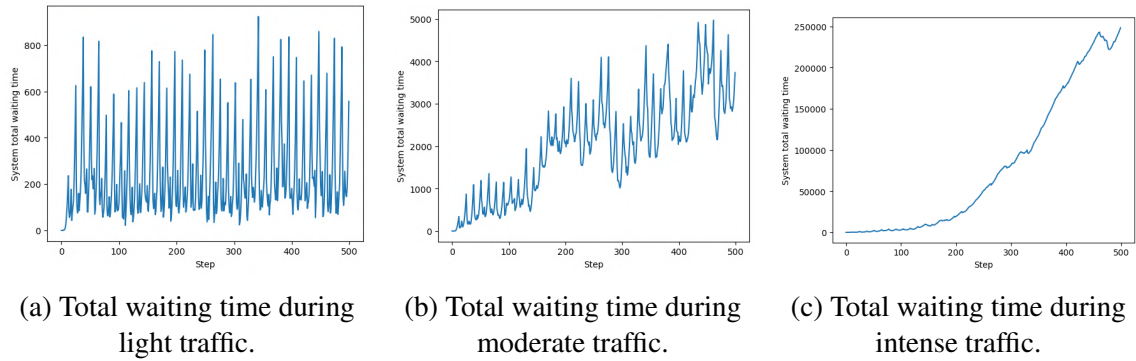


Figure 6.1: Total waiting times under different traffic conditions.

6.1.1 Metrics

The following metrics were used to evaluate an agents performance. We considered the metrics individually and also systemwise.

- **Avg Speed:** The average speed of the vehicles passing through in a 0-1 scale, with 0 being no movement, and 1 being the vehicle moving at maximum possible speed.
- **Avg Waiting Time:** The average waiting time per vehicle.
- **Pressure:** The overall pressure of the intersection.

- **In-Lanes Density:** The average number of vehicles on incoming lanes.
- **Out-Lanes Density:** The average number of vehicles on outgoing lanes.
- **Total Queued:** The total number of vehicles queued across all lanes.

6.2 Single agent results

We tested how would the overall performance improve if we only apply one intelligent controller. In Figures 6.2, 6.3, 6.4, 6.5, 6.6 and 6.7 shown the comparison of the metrics in these situations.

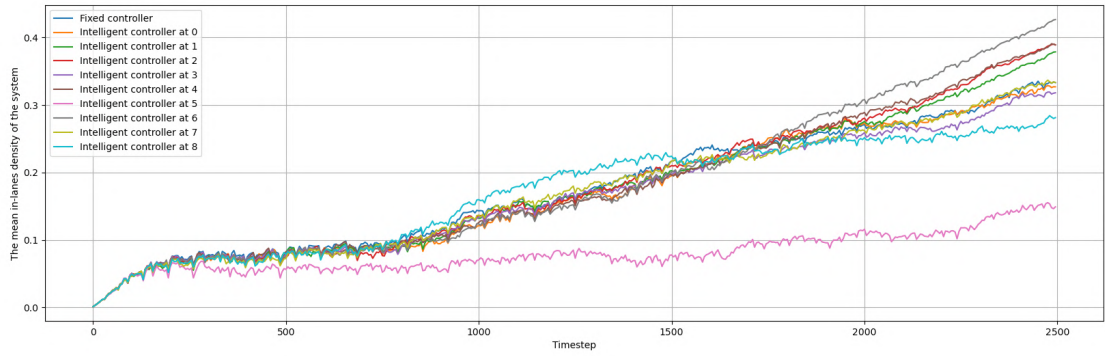


Figure 6.2: Mean of the number of incoming vehicles in the system.

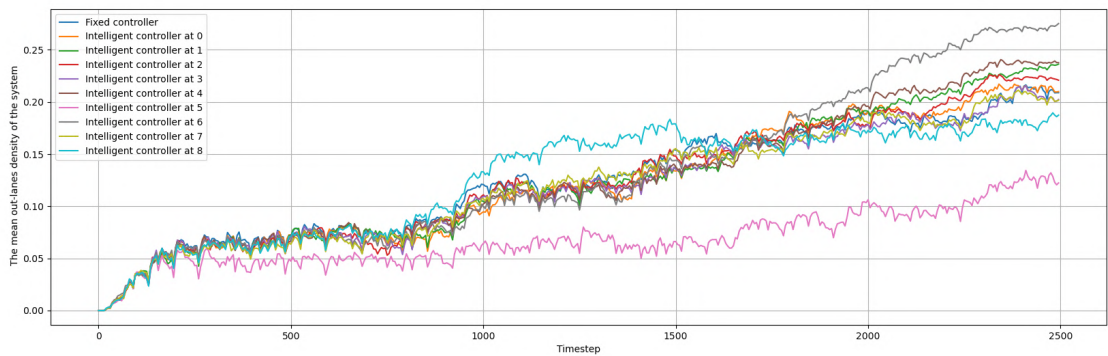


Figure 6.3: Mean of the number of outgoing vehicles in the system.

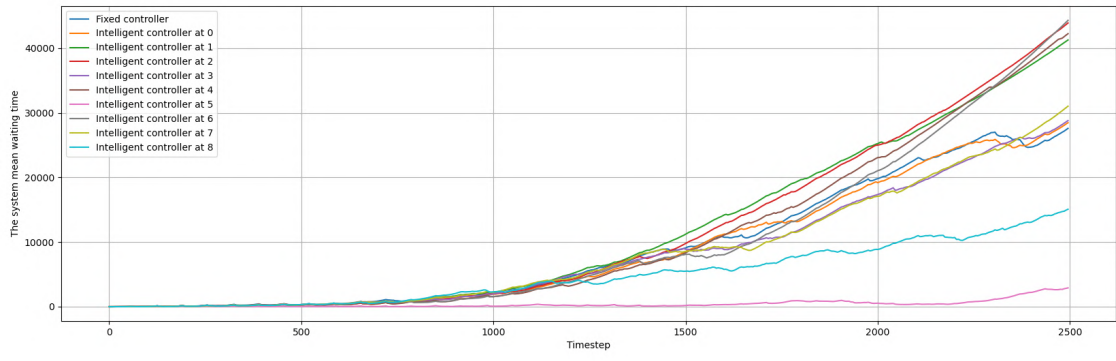


Figure 6.4: Mean waiting time of vehicles in the system.

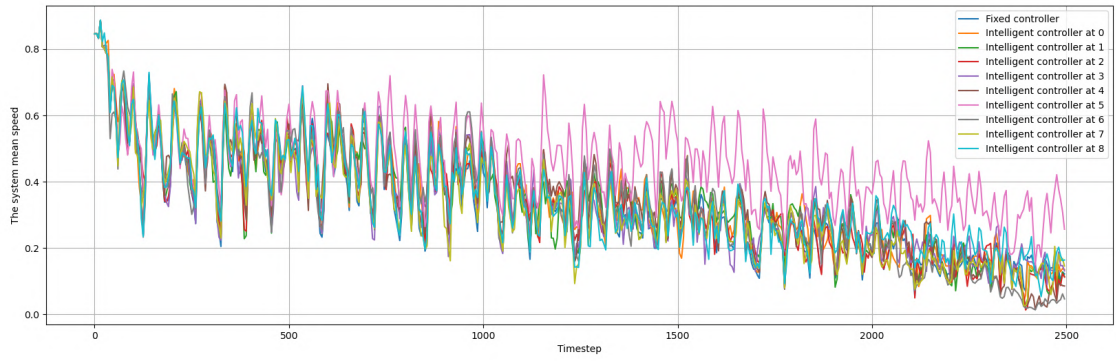


Figure 6.5: Mean speed of vehicles in the system.

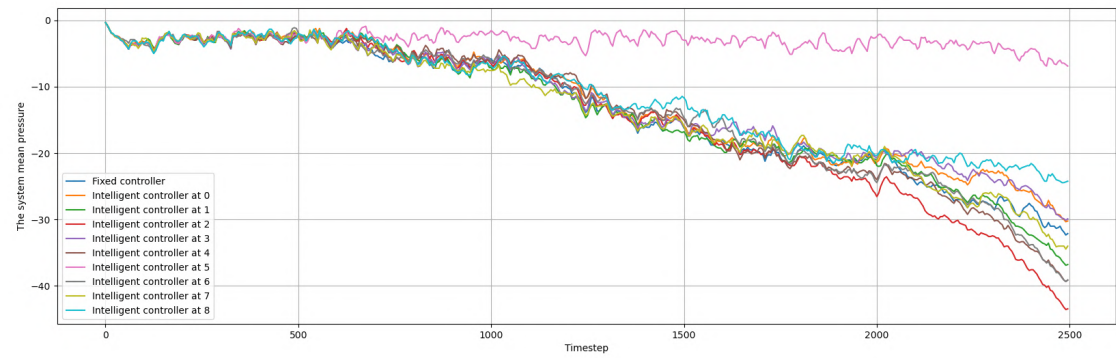


Figure 6.6: Mean pressure at intersections in the system.

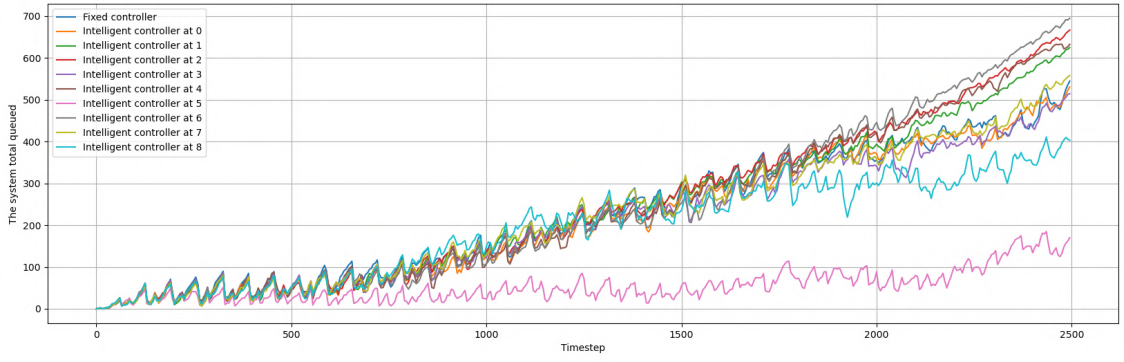


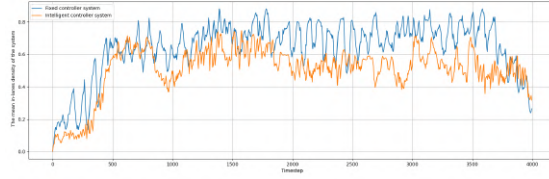
Figure 6.7: Total of cars queued on every lane.

Overall the performance was similar as the baselines, somewhere worse somewhere better. The takeaway is that only one intelligent controller can help to mitigate the traffic, if it was placed strategically, but to get the best results, we should apply as many as possible.

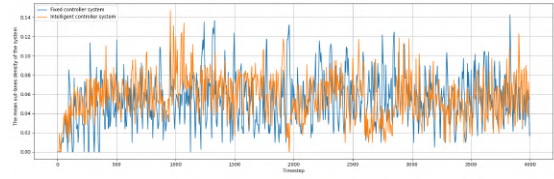
6.2.1 cologne1 and ingolstadt 1

This two maps, only have one traffic light, so we can only look it as a single agent problem.

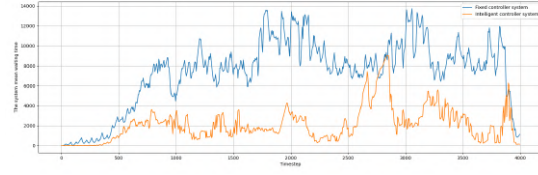
6. Results



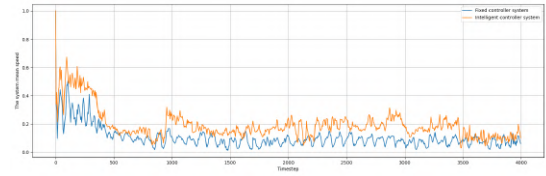
(a) Mean of the number of incoming vehicles in the system.



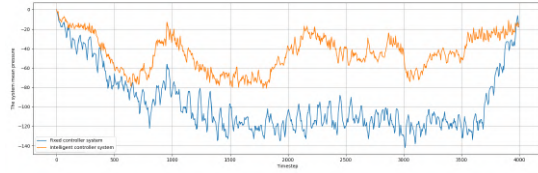
(b) Mean of the number of outgoing vehicles in the system.



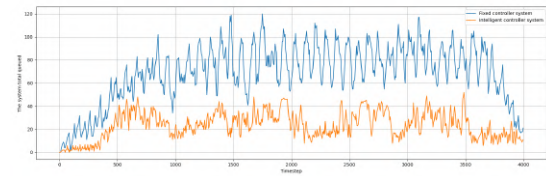
(c) Mean waiting time of vehicles in the system.



(d) Mean speed of vehicles in the system.



(e) Mean pressure at intersections in the system.



(f) Total of cars queued on every lane.

Figure 6.8: Comparison of fixed vs intelligent controls with different metrics.

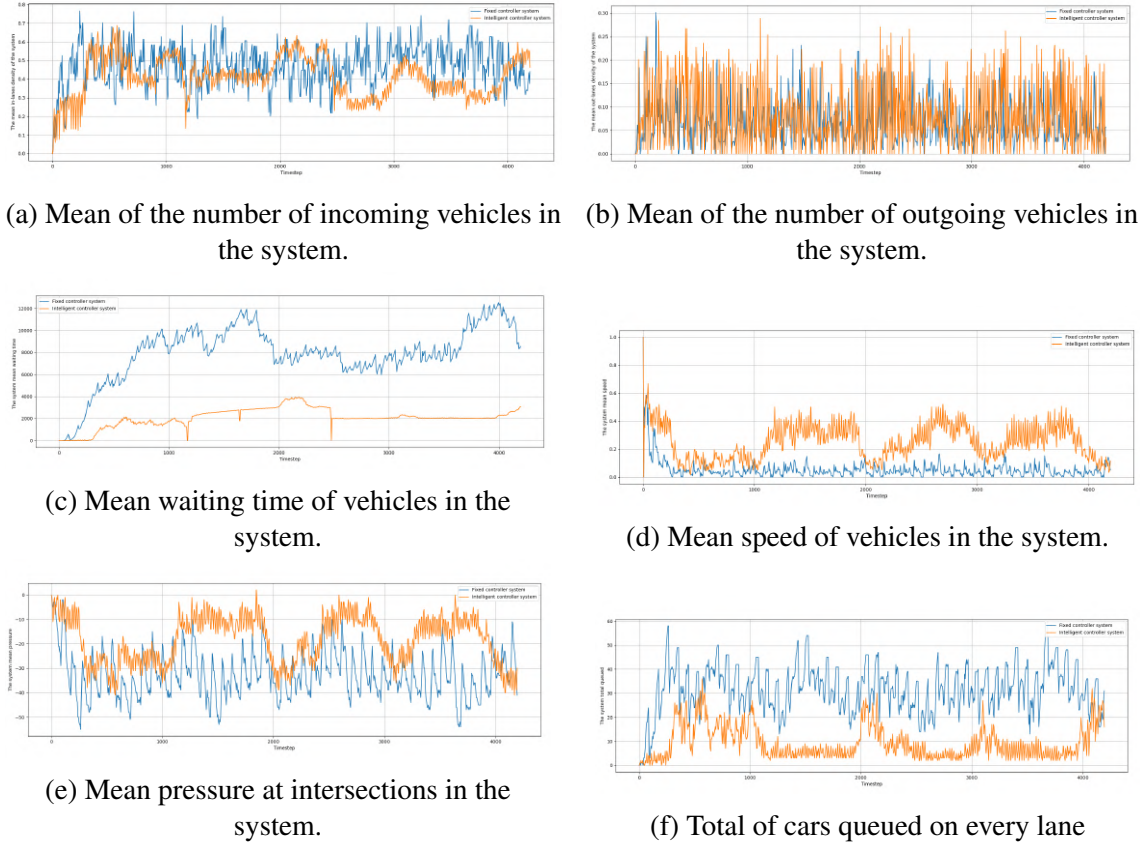


Figure 6.9: Comparison of fixed vs intelligent controls with different metrics.

With these results we can see, how impactful one intelligent traffic light could be alone.

6.3 Multi agent testing

To see the system full capability, we ran experiments on the maps, where every agent is intelligent, and communicates with the reachable neighbours. We write in detail about every map's results in the subsections. The traffic scale value in every test situation equals final scale mentioned in Table 6.1, if not stated otherwise.

6.3.1 3x3grid

Since this map consists of identical intersections in a grid layout, we trained only one agent, but varied it's position. On Figure 6.10 we can see that the total number of queued cars is 90% less in a case of multi agent system than in the case of a fixed phase system. Also if we apply only a single agent in the middle, the whole systems performance is

similar as if it was fixed, because one lamp cannot optimize the whole controller system alone.

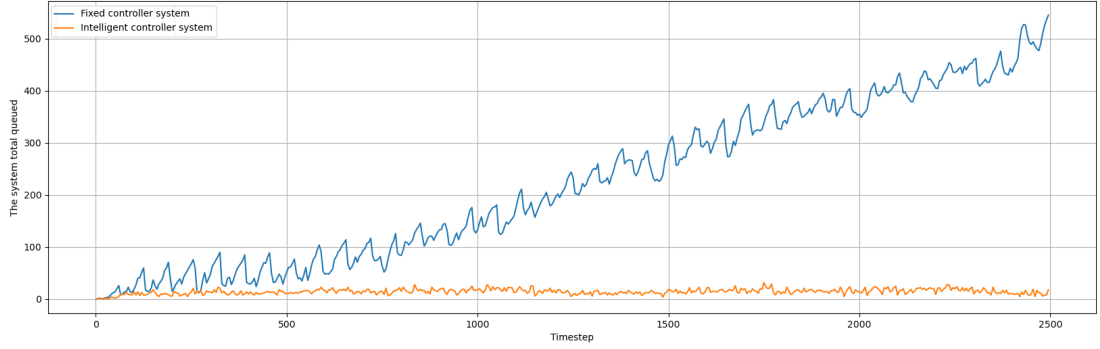
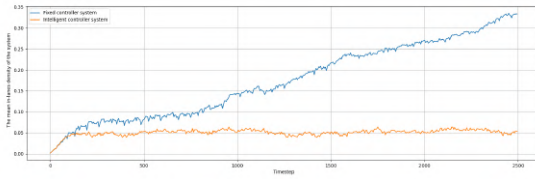
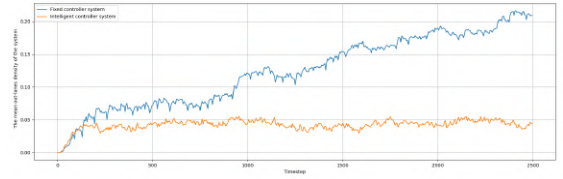


Figure 6.10: Sum of cars queued on every lane

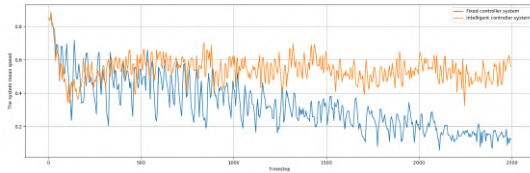
The other metrics show a similar increase in performance show on Figure 6.11



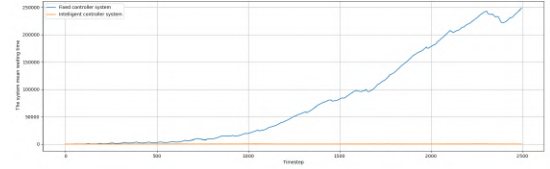
(a) Mean of the number of incoming vehicles in the system.



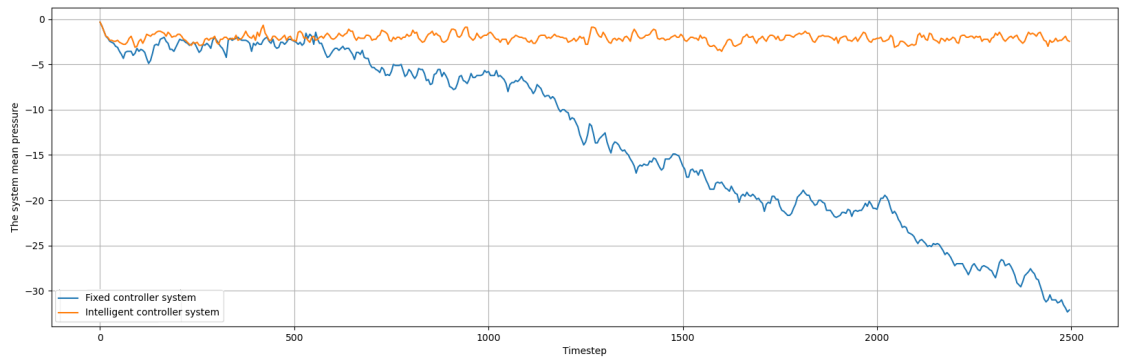
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.11: Comparison of fixed vs intelligent controls with different metrics.

Comparing the different learning algorithms, the DQN' and PPO's performance being almost the same, and A2C's slightly worse, as shown on Figure 6.12.

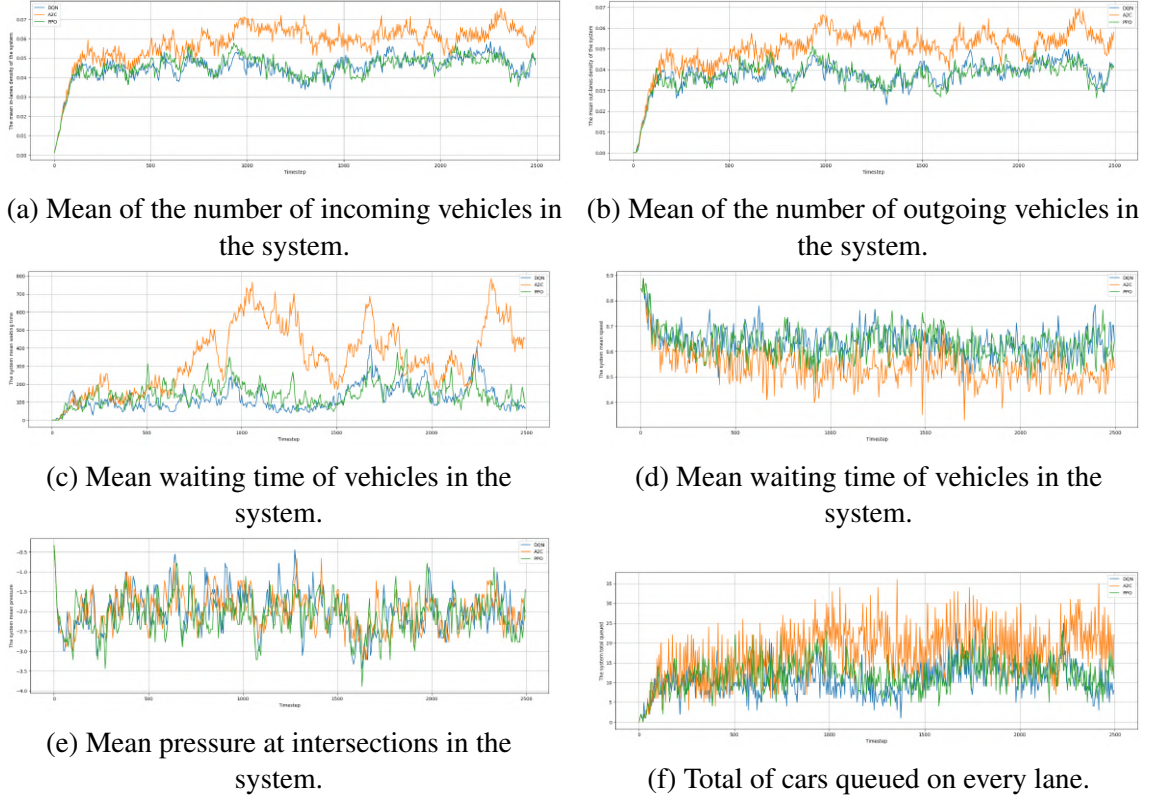


Figure 6.12: Comparison of DQN, PPO and A2C algorithms.

Grid-like environments, such as seen in Manhattan, are ideal for optimization and are common in major cities. We'll also explore maps of with less ideal layouts later.

6.3.2 4x4loop

The interesting aspect of the 4x4loop map is that initially there is traffic on one main line, and around the middle of the simulation, participants start to split and some begin to take a detour. Here too, we can see similar results to the previous ones. The comparison on Figures 6.13 and 6.14 is between the fixed controllers and a PPO agent, since both DQN and A2C provided similar results.

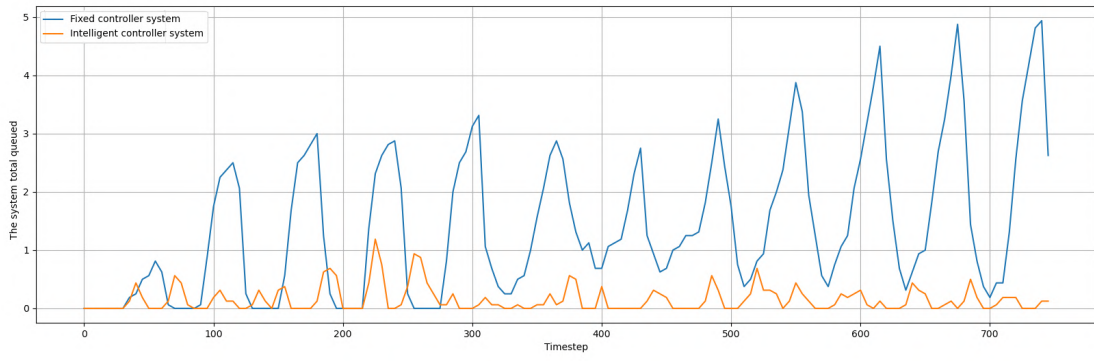
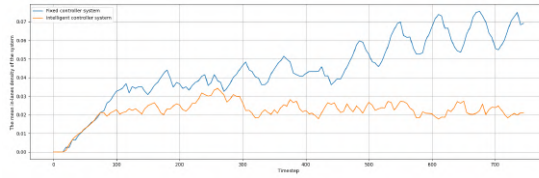
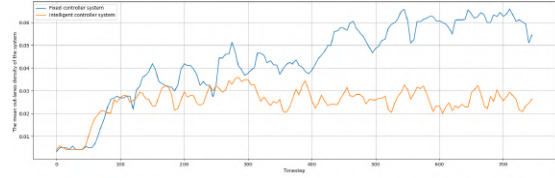


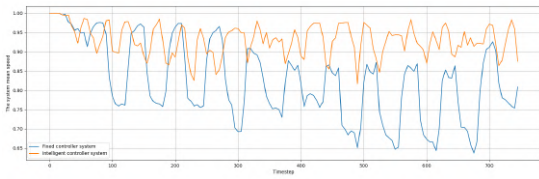
Figure 6.13: Sum of cars queued on every lane



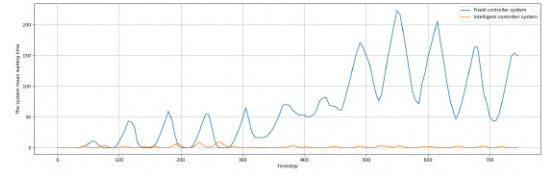
(a) Mean of the number of incoming vehicles in the system.



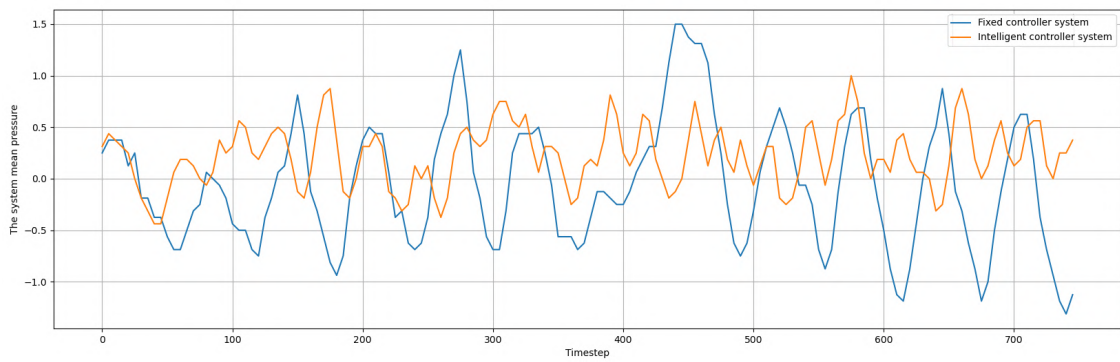
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.14: Comparison of fixed vs intelligent controls with different metrics.

6.3.3 4x4arterial

The 4x4arterial is the first RESCO map, it's a simple 4x4 grid with identical intersections.

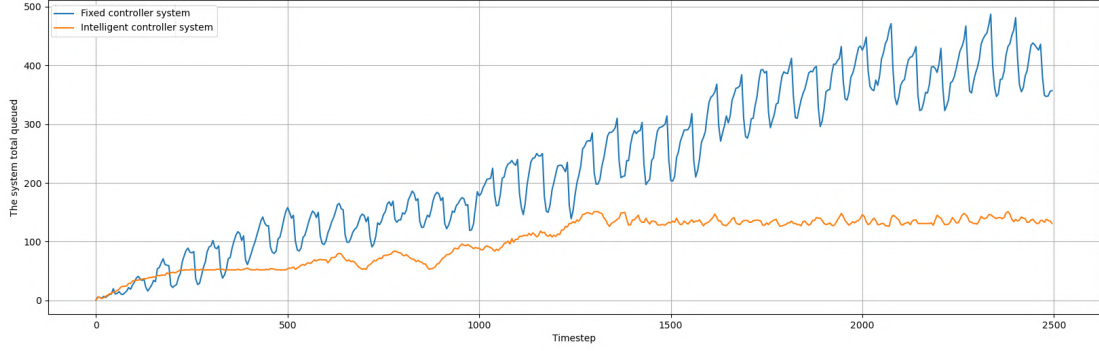
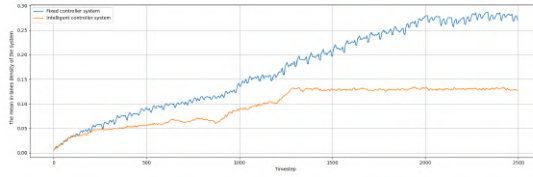
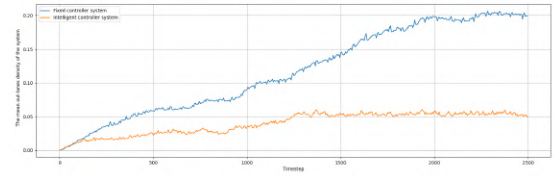


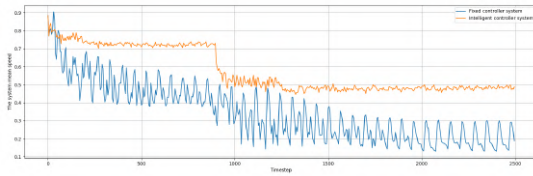
Figure 6.15: Sum of cars queued on every lane



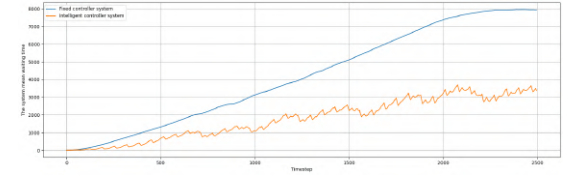
(a) Mean of the number of incoming vehicles in the system.



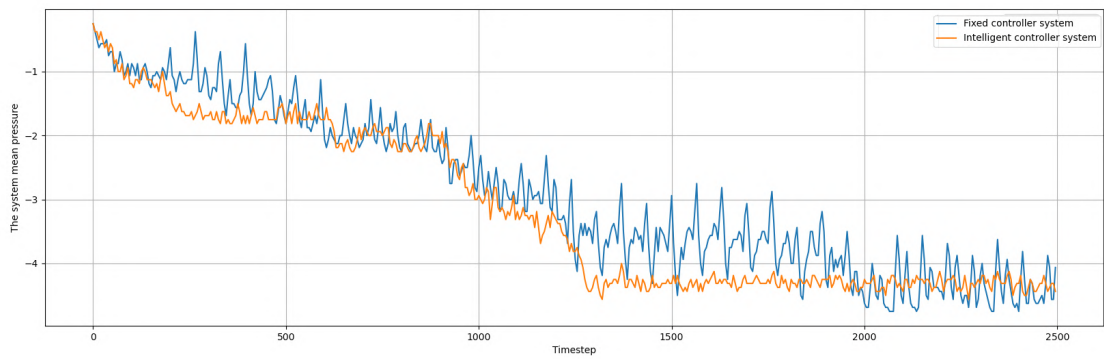
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.16: Comparison of fixed vs intelligent controls with different metrics.

As before, the three algorithms yielded similar results, so we only show the PPO model in the Figures 6.15 and 6.16.

6.3.4 4x4grid

This grid is almost the same as the 3x3grid, but this time the connecting roads, have much more lanes, and dedicated turning lanes at every intersection. The traffic lights are fully utilizing the padded architecture.

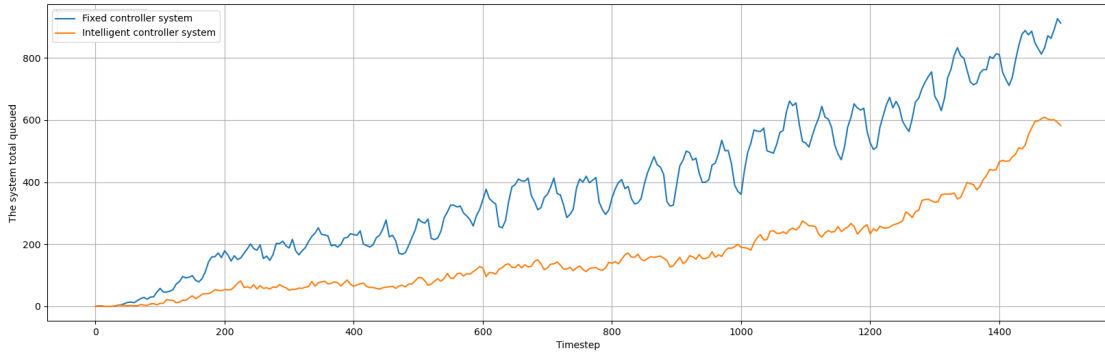


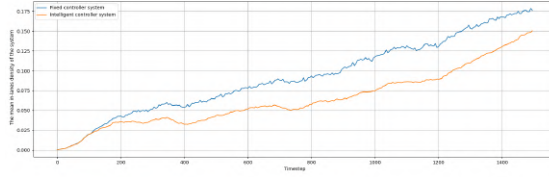
Figure 6.17: Sum of cars queued on every lane

At this map, we could only achieve 20% average performance improvement, shown in Figures 6.17 and 6.18, because the number of lanes, and dedicated turning lanes already optimized the whole system somewhat.

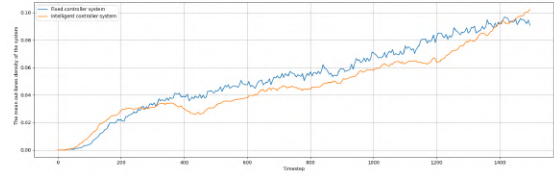
6.3.5 cologne3

The cologne3 map is a real-life simulation compiled based on previous data. We have already seen the single agent version earlier. By expanding that map with two additional lights, we examined it as a multi-agent environment. The results are shown in Figures 6.19 and 6.20

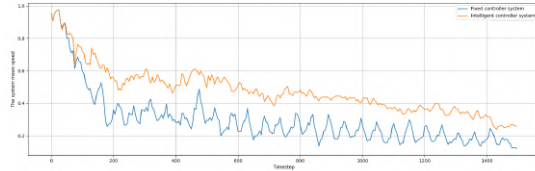
6. Results



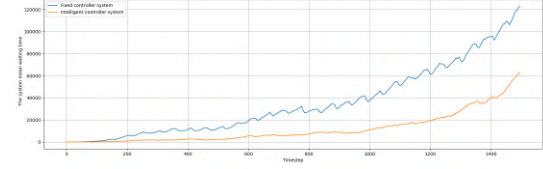
(a) Mean of the number of incoming vehicles in the system.



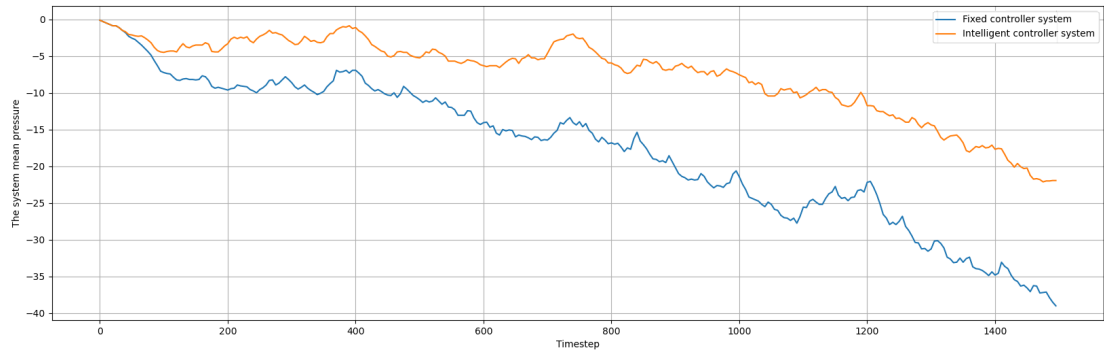
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.18: Comparison of fixed vs intelligent controls with different metrics.

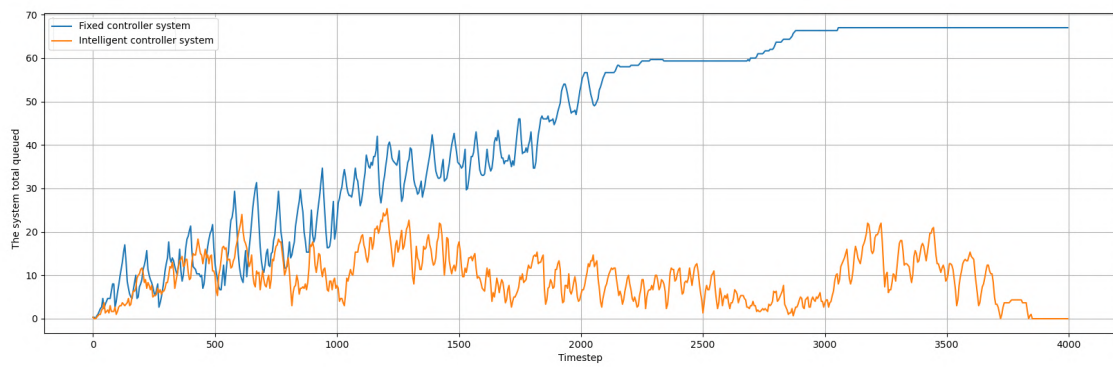
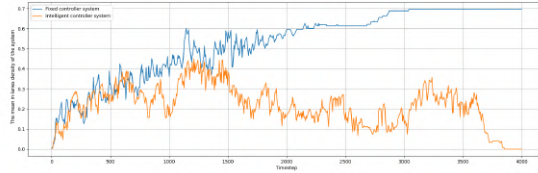
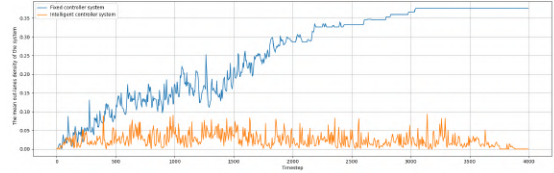


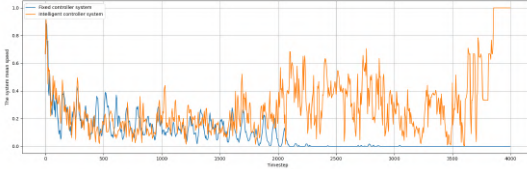
Figure 6.19: Sum of cars queued on every lane



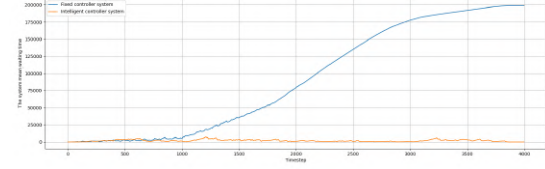
(a) Mean of the number of incoming vehicles in the system.



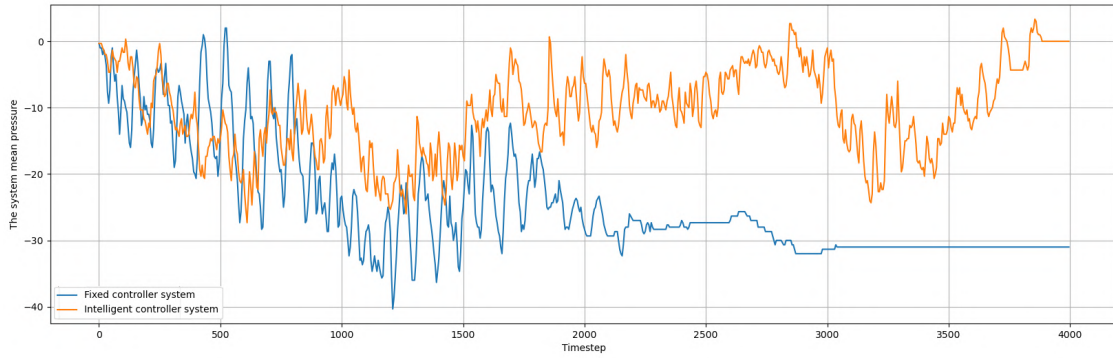
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.20: Comparison of fixed vs intelligent controls with different metrics.

As the time goes on the improvement becomes more evident. Since this is a fairly small map, our system clearly outperforms the fixed controllers.

6.3.6 cologne8

The cologne8 further expands the previously mentioned cologne3 map with an additional five lights.

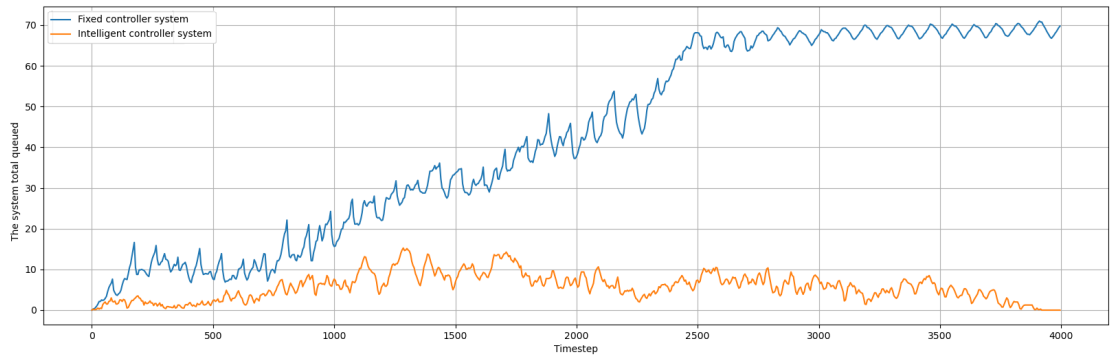
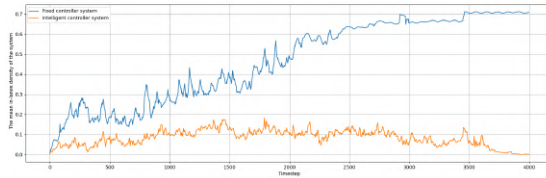
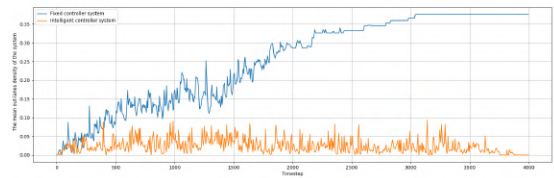


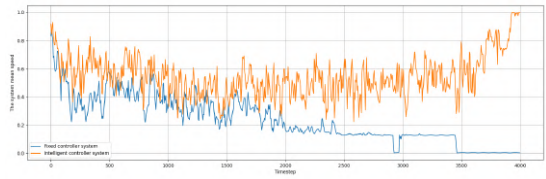
Figure 6.21: Sum of cars queued on every lane



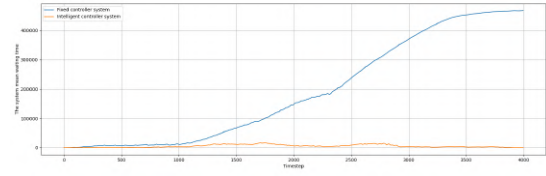
(a) Mean of the number of incoming vehicles in the system.



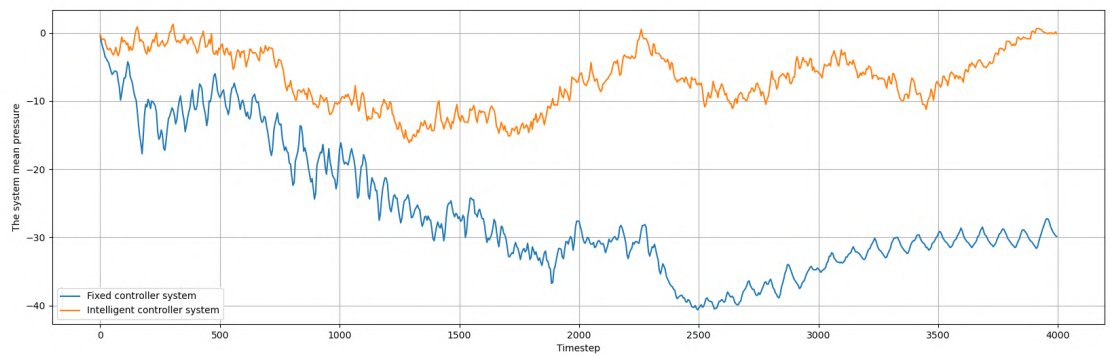
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.22: Comparison of fixed vs intelligent controls with different metrics.

On Figures 6.21, 6.22 visible, that as area of interest grows, it's harder to optimize, but still achievable.

6.3.7 ingolstadt7

The ingolstadt7 map is also a real-life example, depicting a corridor of traffic lights, where the significant number of vehicles are using the the main road.

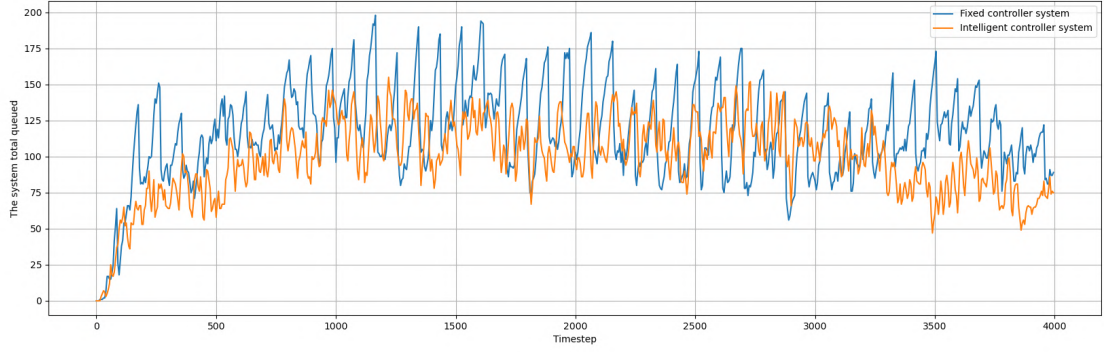
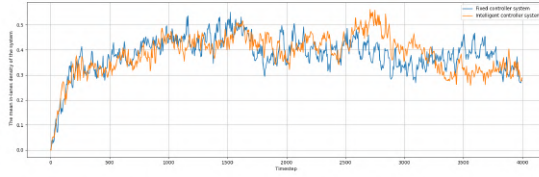
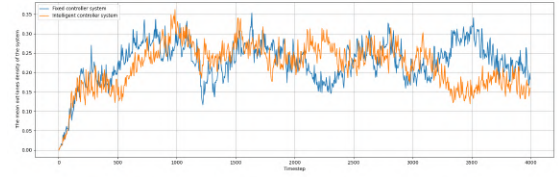


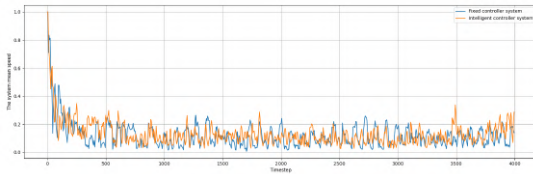
Figure 6.23: Sum of cars queued on every lane



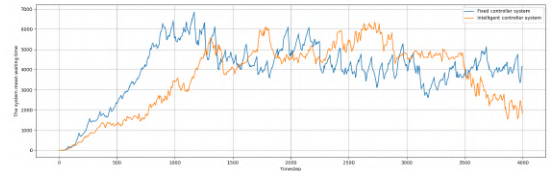
(a) Mean of the number of incoming vehicles in the system.



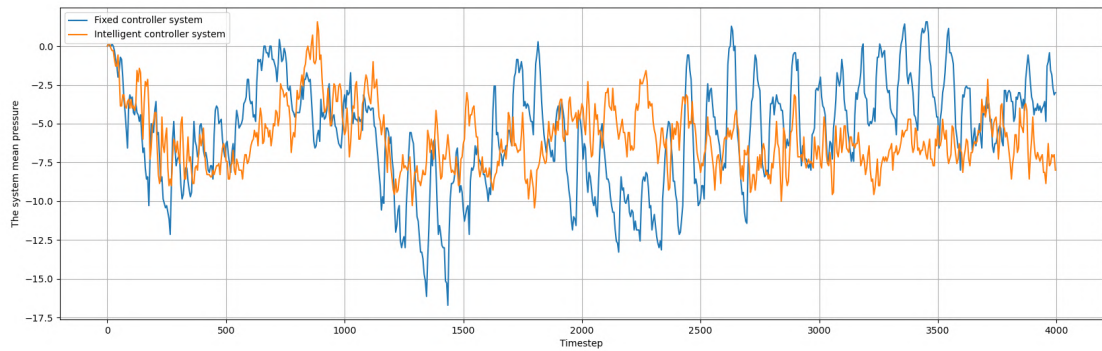
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.24: Comparison of fixed vs intelligent controls with different metrics.

The results, Figures 6.23 and 6.24, show that it was much harder to further optimize the fixed controllers, but overall the intelligent system performed better.

6.3.8 ingolstadt21

This map is the biggest and most complex one what we tested our agents on. Tripled the number of controllable traffic lights of ingolstadt7 map.

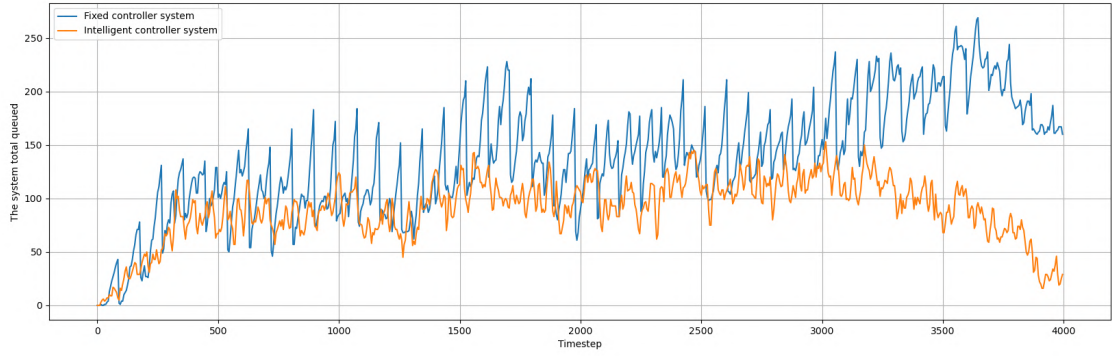
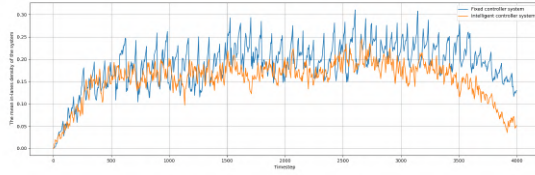
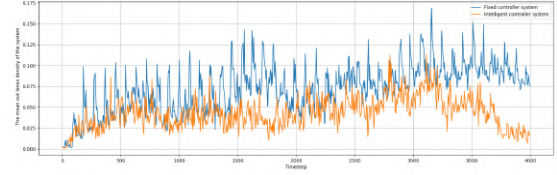


Figure 6.25: Sum of cars queued on every lane

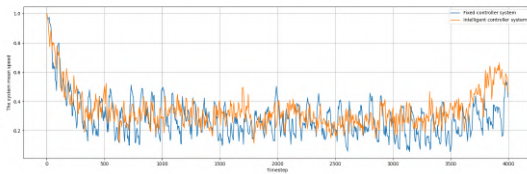
The results show a 45% average improvement in mean waiting time over the fixed controllers in Figure 6.26d. The other metrics unfortunately now showing such a big positive difference, but they are still noticeable towards the end of the simulation, in Figures 6.25, 6.26. This final simulation proves that applying intelligent traffic lights, can make a impact in real life situations.



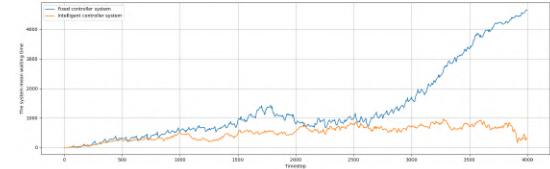
(a) Mean of the number of incoming vehicles in the system.



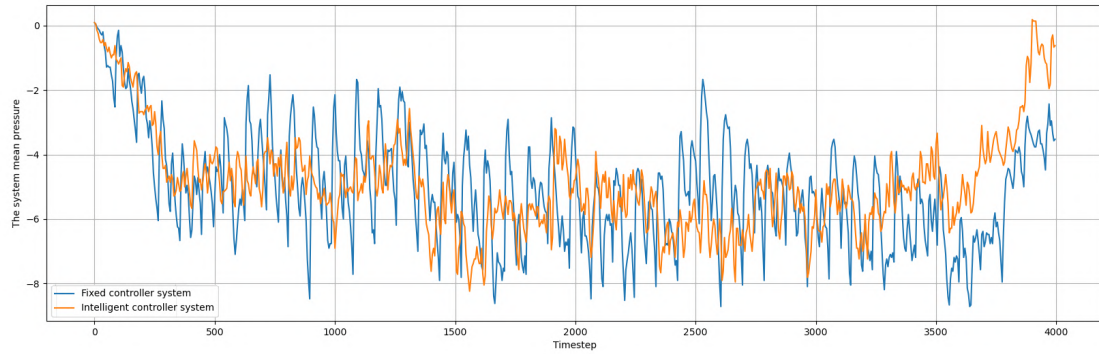
(b) Mean of the number of outgoing vehicles in the system.



(c) Mean speed of vehicles in the system.



(d) Mean waiting time of vehicles in the system.



(e) Mean pressure at intersections in the system.

Figure 6.26: Comparison of fixed vs intelligent controls with different metrics.

6.4 Future plans

Our future plans definitely include continuing our research. We primarily see the greatest potential in laying down a new framework that will be much more flexible than the current one, enabling us to conduct the experimental works we have written about earlier. With the new framework, we would be capable of determining a switching pattern for each lane, allowing agents to learn new patterns that have not been seen before. Of course, this requires the integration of various safety features to ensure that we do not receive contradictory schemes. This would definitely improve the quality and overall adaptability of our current system. Additionally, we aim to move towards a universal model by employing various encoding techniques, which could characterize a network much more effectively and compactly. For the sake of decentralization, we also plan to explore and further develop other global optimizations methods such as convolution, in our current cellular model. We're also preparing agents for different scenarios that did not fit into the current research scope. Furthermore, we want to investigate situations where, in addition to smart traffic lights, we have the opportunity for route planning and thereby examine the improvements we can achieve. Indeed, with route planning/recommendation, we can really come close to achieving a globally optimal traffic system.

Chapter 7

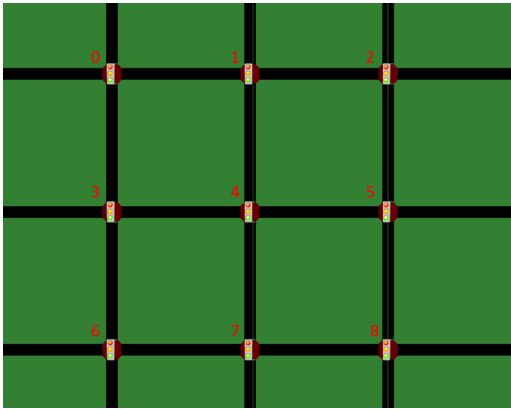
Conclusion

In our comprehensive analysis, we found that the deployment of smart traffic lights greatly enhances the efficiency of traffic systems, ranging from 10% to 80% improvement. These advanced lights not just decrease waiting times; they also play a crucial role in reducing our carbon footprint, thereby protecting the environment. Our research suggests that combining smart traffic lights with advanced traffic management systems can lead to more adaptive, responsive, and sustainable urban traffic networks.

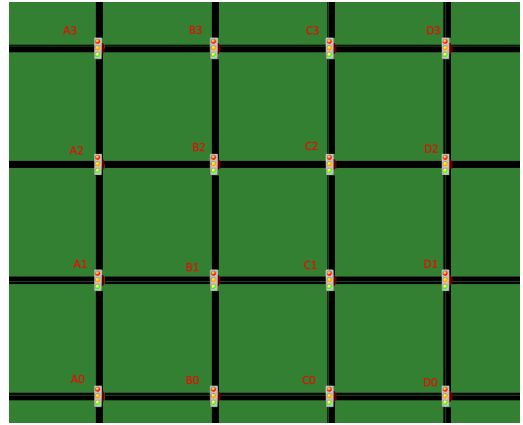
We believe that one of the greatest challenges in the future will be the sensible and effective integration of learned policies into the real world. It will be crucial that the guidelines formulated by algorithms not only work in theory but are also applicable and adaptable in practice to the specific challenges of urban traffic systems. This will require special attention to dynamically adapt to changing traffic conditions, as well as strict adherence to safety and sustainability considerations. In addition to these challenges, it is important that the solutions developed are transparent, safe, and demonstrably optimal. We know that in a delicate area such as urban traffic management, employing a "black box" solution is not advantageous. Therefore, it is essential to keep in mind these essential principles: transparency, reliability, safety, and optimization. The application of these principles will help ensure that the advancements are not just technological progressions but also contribute meaningfully and responsibly to the field. As these technologies continue to evolve, they will play a crucial role in shaping the future of urban living, making cities smarter, greener, and more connected.

Appendix A

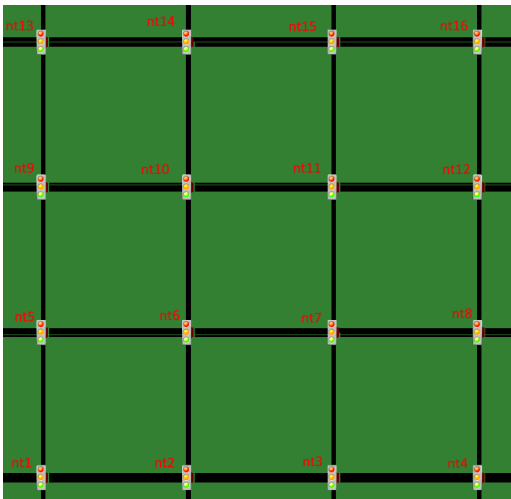
The maps



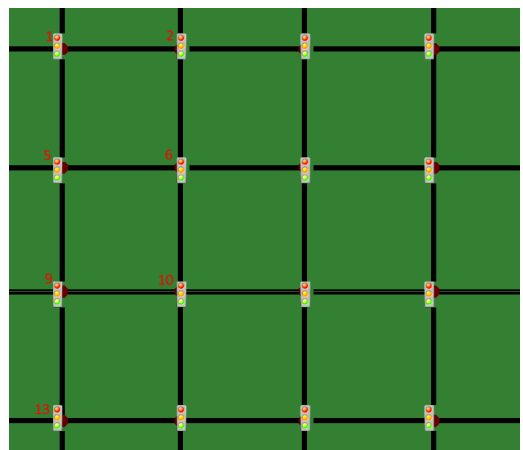
(a) Aerial view of 3x3grid



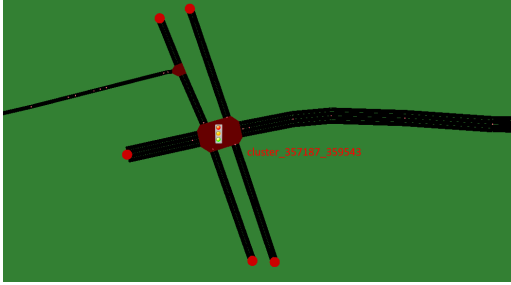
(b) Aerial view of grid4x4



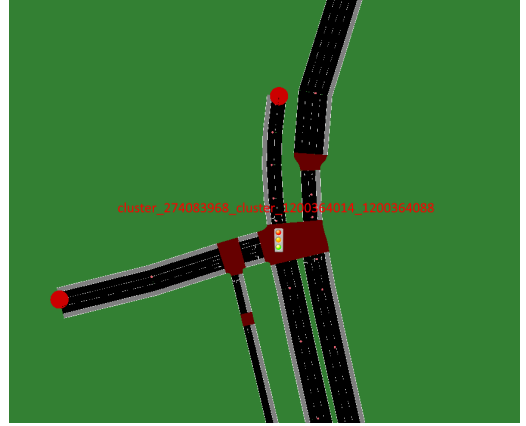
(a) Aerial view of arterial4x4



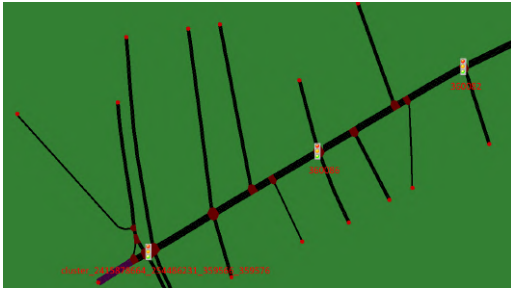
(b) Aerial view of 4x4loop



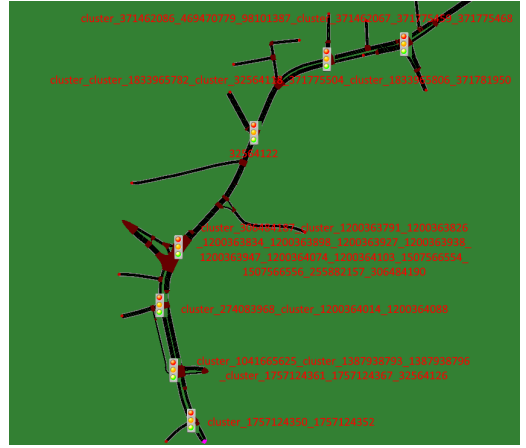
(a) Aerial view of cologne1



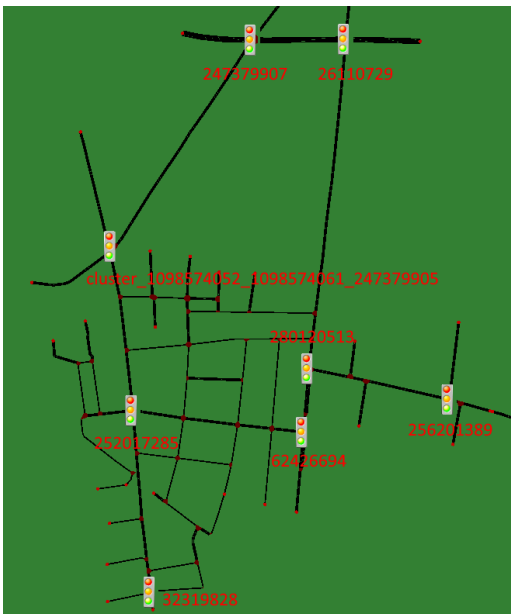
(b) Aerial view of ingolstadt1



(a) Aerial view of cologne3



(b) Aerial view of ingolstadt7



(a) Aerial view of cologne8



(b) Aerial view of ingolstadt21

Appendix B

Usage of AI tools

During the preparation of the thesis, we utilized ChatGPT [22] to assist with checking and ensuring the correctness of language usage, which included verifying grammar and syntax.

Bibliography

- [1] Sarah Petit. *World Vehicle Population Rose 4.6% in 2016*. Accessed: 2023-09-14. 2017. URL: <https://wardsintelligence.informa.com/WI058630/World-Vehicle-Population-Rose-46-in-2016>.
- [2] Soheil Sohrabi, Joe Zietsman, and Haneen Khreis. “Burden of Disease Assessment of Ambient Air Pollution and Premature Mortality in Urban Areas: The Role of Socioeconomic Status and Transportation”. In: *International Journal of Environmental Research and Public Health* 17.4 (2020). ISSN: 1660-4601. DOI: 10.3390/ijerph17041166. URL: <https://www.mdpi.com/1660-4601/17/4/1166>.
- [3] Szilárd Kovács, Zoltán Barta, and János Botzheim. “Traffic Optimization by Local Bacterial Memetic Algorithm”. In: *Computational Collective Intelligence*. Ed. by Ngoc Thanh Nguyen et al. Cham: Springer Nature Switzerland, 2023, pp. 484–496. ISBN: 978-3-031-41456-5.
- [4] Mădălin-Dorin Pop. “Traffic Lights Management Using Optimization Tool”. In: *Procedia - Social and Behavioral Sciences* 238 (2018). Challenges and Innovation in Management and Entrepreneurship, pp. 323–330. ISSN: 1877-0428. DOI: <https://doi.org/10.1016/j.sbspro.2018.04.008>. URL: <https://www.sciencedirect.com/science/article/pii/S1877042818300387>.
- [5] Vicente Nejar de Almeida, Ana Lúcia Cetertich Bazzan, and Monireh Abdoos. “Multiagent Reinforcement Learning for Traffic Signal Control: a k-Nearest Neighbors Based Approach”. In: *ATT@IJCAI*. 2022. URL: <https://api.semanticscholar.org/CorpusID:251258226>.
- [6] Guanjie Zheng et al. *Diagnosing Reinforcement Learning for Traffic Signal Control*. 2019. arXiv: 1905.04716 [cs.LG].

- [7] Aleksandar Stevanovic, Cameron Kergaye, and Peter Martin. “SCOOT and SCATS: A Closer Look into Their Operations”. In: Jan. 2009.
- [8] Guanjie Zheng et al. *Learning Phase Competition for Traffic Signal Control*. 2019. arXiv: 1905.04722 [cs.LG].
- [9] Chacha Chen et al. “Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control”. In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (2020), pp. 3414–3421. DOI: 10.1609/aaai.v34i04.5744. URL: <https://ojs.aaai.org/index.php/AAAI/article/view/5744>.
- [10] Selim Reza et al. “A citywide TD-learning based intelligent traffic signal control for autonomous vehicles: Performance evaluation using SUMO”. In: *Expert Systems* n/a.n/a (), e13301. DOI: <https://doi.org/10.1111/exsy.13301>. eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1111/exsy.13301>. URL: <https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.13301>.
- [11] Xinshi Zang et al. “MetaLight: Value-based Meta-reinforcement Learning for Traffic Signal Control”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020.
- [12] James Ault, Josiah P. Hanna, and Guni Sharon. *Learning an Interpretable Traffic Signal Control Policy*. 2020. arXiv: 1912.11023 [cs.LG].
- [13] Pravin Varaiya. “Max pressure control of a network of signalized intersections”. In: *Transportation Research Part C: Emerging Technologies* 36 (2013), pp. 177–195. ISSN: 0968-090X. DOI: <https://doi.org/10.1016/j.trc.2013.08.014>. URL: <https://www.sciencedirect.com/science/article/pii/S0968090X13001782>.
- [14] Hua Wei et al. “PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network”. In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. KDD ’19. Anchorage, AK, USA: Association for Computing Machinery, 2019, 1290–1298. ISBN: 9781450362016. DOI: 10.1145/3292500.3330949. URL: <https://doi.org/10.1145/3292500.3330949>.
- [15] Lucas N. Alegre. *SUMO-RL*. <https://github.com/LucasAlegre/sumo-rl>. 2019.

- [16] Pablo Alvarez Lopez et al. “Microscopic Traffic Simulation using SUMO”. In: *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. URL: <https://elib.dlr.de/124092/>.
- [17] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from [tensorflow.org](https://www.tensorflow.org). 2015. URL: <https://www.tensorflow.org/>.
- [18] Eric Liang et al. *RLlib: Abstractions for Distributed Reinforcement Learning*. 2018. arXiv: 1712.09381 [cs.AI].
- [19] Greg Brockman et al. *OpenAI Gym*. 2016. arXiv: 1606.01540 [cs.LG].
- [20] J. K. Terry et al. *PettingZoo: Gym for Multi-Agent Reinforcement Learning*. 2021. arXiv: 2009.14471 [cs.LG].
- [21] James Ault and Guni Sharon. “Reinforcement Learning Benchmarks for Traffic Signal Control”. In: *Proceedings of the Thirty-fifth Conference on Neural Information Processing Systems (NeurIPS 2021) Datasets and Benchmarks Track*. 2021.
- [22] OpenAI. *ChatGPT*. <https://www.openai.com/chatgpt>. 2023.

List of Figures

2.1	The main components of a traffic control system.	7
2.2	An intelligent traffic management system in action.	10
4.1	Non conflicting green signals [15]	16
4.2	The penalty for not changing phase.	18
4.3	The architecture with the padded input.	19
4.4	The final architecture.	20
4.5	Order by direction case 1.	21
4.6	Order by direction case 2.	21
4.7	An encoding that links the lanes with the phases	21
5.1	Similar intersections with $ A = 2$	24
5.2	Similar intersections with $ A = 3$	24
5.3	Similar intersections with $ A = 4$	24
6.1	Total waiting times under different traffic conditions.	26
6.2	Mean of the number of incoming vehicles in the system.	27
6.3	Mean of the number of outgoing vehicles in the system.	27
6.4	Mean waiting time of vehicles in the system.	28
6.5	Mean speed of vehicles in the system.	28
6.6	Mean pressure at intersections in the system.	28
6.7	Total of cars queued on every lane.	29
6.8	Comparison of fixed vs intelligent controls with different metrics.	30
6.9	Comparison of fixed vs intelligent controls with different metrics.	31
6.10	Sum of cars queued on every lane	32
6.11	Comparison of fixed vs intelligent controls with different metrics.	32
6.12	Comparison of DQN, PPO and A2C algorithms.	33
6.13	Sum of cars queued on every lane	34
6.14	Comparison of fixed vs intelligent controls with different metrics.	34

6.15	Sum of cars queued on every lane	35
6.16	Comparison of fixed vs intelligent controls with different metrics.	35
6.17	Sum of cars queued on every lane	36
6.18	Comparison of fixed vs intelligent controls with different metrics.	37
6.19	Sum of cars queued on every lane	37
6.20	Comparison of fixed vs intelligent controls with different metrics.	38
6.21	Sum of cars queued on every lane	39
6.22	Comparison of fixed vs intelligent controls with different metrics.	39
6.23	Sum of cars queued on every lane	40
6.24	Comparison of fixed vs intelligent controls with different metrics.	40
6.25	Sum of cars queued on every lane	41
6.26	Comparison of fixed vs intelligent controls with different metrics.	42