Delft University of Technology

Civil Engineering
Transport & Planning

Bus management using multi-agent reinforcement learning

Author:
George Weijs (4953436)

July 21, 2021

Preface

Before starting my thesis I was sure that it was going to be a challenge for me to bring it to a good result. The fact that you have to work alone on a project of this duration was daunting for me. Luckily, several people made sure that I did not have to do it alone and that I was able to keep my energy up throughout the project.

I want to thank each of my supervisors and the people at Lynxx for their sincere interest in my thesis. The help, time, insight, experience and energy that I gained from these interactions made this thesis, and my experience making it, better.

I want to thank my daily supervisor, Adam Pel, for being such a good brainstorming partner. Our sessions were always good moments for me to take a step back and decide what direction I want to take. You were always interested in what I was doing and why. With your feedback I was able to get the vague story line in my head to a clear story line on paper. I am sure this has been a huge benefit for both my research and the people reading it.

Next, I want to thank Oded Cats for the support he gave to the whole group of Transport & Planning thesis students. Our weekly catch up sessions provided me with a good start of the week and a good base to connect with fellow students during the Covid lockdowns. Furthermore, your feedback on my thesis improved the quality and its relevance to the rest of the research field. You were easy to reach out to and always willing to help out, thanks.

Besides two supervisors from my own faculty, I wanted to have a supervisor with specific reinforcement learning knowledge on board. Luckily, Matthijs Spaan, wanted to help me out. Thank you for the brainstorm meetings that we have had, they provided me with new ideas to try out in my training procedure. This new input gave me the energy to keep on experimenting until I had created a stable training procedure.

In addition to all the supervision that I got from the TU Delft, Lynxx also gave me support in many ways. Having a place to go to every now and then is a luxury in Covid times and I am very grateful that I had the opportunity to go to the Lynxx office. Moreover, it was a pleasure to be part of such an open and energetic team. Additionally, the supervision from my company supervisor, Peter Nijhuis, gave me insight on how to take on big projects, like a thesis, and manage the interest of all my stakeholders. I have learned a lot from seeing you work and I want to thank you for that.

Along with the wonderful team surrounding me, I am also grateful that my supervisors and Lynxx saw potential in my topic idea. The topic was really interesting to me and it gave me the chance to learn many things in a field that I am interested in.

Last of all, I want to thank my friends and family for their love, distraction, and support. Writing a thesis from home is not an easy feat, however, working from home with my girlfriend Hanne and my dog Knoet, it is not so bad at all, and I am deeply grateful for their constant support.

Executive Summary

Introduction

Bus systems are used all over the world and, compared to the public transport (PT) modes of train, tram and metro, they are more flexible by nature. This mainly stems from the fact that they do not need dedicated infrastructure. This flexibility is one of the benefits of bus systems, but it has a downside too. The downside of not having dedicated infrastructure is that bus systems are often affected by external conditions such as traffic conditions. Additionally, the passenger travel demand for bus systems affects the operational speed of the busses, since the number of boarders and alighters determines the dwell time of busses per stop. The passenger travel demand can vary through time, and with that the operational speed of the busses. This means that when the busses are in operation it can be beneficial to coordinate the system as a whole to make sure the fluctuating external conditions do not result in big disruptions of the bus system. Such disruptions usually come in the form of bus bunching, i.e. two busses of the same line driving very close together, followed by a large gap until the next bus.

To avert small disruptions from building up to large disruptions, coordination among the busses is needed, referred to as bus management in this study. This can be done in various ways, for example: skipping stops, temporarily adding / removing busses in / from the system, adjusting driving behaviour through communication with the bus drivers, manipulating traffic signals on the bus routes and possibly demand steering. In this study adjusting driving behaviour and manipulating traffic signals are chosen since (if the necessary (digital) infrastructure is in place) they are less intrusive measures than other possible measures. It is thus focused on the PT side of intelligent transport system (ITS), specifically intelligent bus systems. Moreover, the proposed solutions are drawn from the field of reinforcement learning (RL), specifically multi-agent reinforcement learning (MARL).

Abdulhai et al. [5] present several different topics RL could be used for in the transport sector, among them are intersection control and traffic management. They suspect that RL can play an essential role in an ITS. This is a result of the many intelligent entities in such a system, which have to communicate in a manner that takes the objective of both the individual entity and the system into account. In RL, such entities are called agents and MARL provides a framework to find effective behaviour for all the agents. RL frameworks have proven to be effective in many single-agent settings [48], [38], [47]. For multi-agent settings it is harder to find good policies since the interactions between agents create many different solutions and often cooperation is needed for good solutions.

RL is already extensively used in the development of self driving cars and it would be a big loss for the PT industry if RL would not be adopted to benefit them as well. This research aims to provide knowledge and inspiration for one possible application of RL in the PT industry: real-time bus service management.

The potential of applying RL in ITS has been identified [5] and has also already been applied to headway management [56]. Yet, the implementations so far have been relatively simple, and as a result they do not provide sufficient information about the applicability of an RL framework for bus management in real day-to-day scenarios. This study builds on previous studies by increasing the realism in the scenarios that are investigated and adding agents to the MARL algorithm that control the traffic lights. From this research, the understanding of RL frameworks in the bus management setting can grow and thereby the field can come one step closer to the actual implementation for operators of an RL framework for bus management.

As mentioned before, the current state-of-the-art [56] uses an idealised simulation environment to train the MARL algorithm in for headway management. This is not a realistic setting and this study aims to progress to a more realistic setting by adding stochasticity and route heterogeneity to the problem. There are several aspects that can be added to the simulation environment of an idealised bus route (currently used by the state-of-the-art) to make it more realistic. Yet, these aspects will make the route also more complex. Therefore, they are referred to as environmental extension. Besides the possible additions to the route, the number of possible actions in a control algorithm can be expanded beyond holding only as well. In this study traffic light control is added to further manipulate the bus system. The route extensions that can be introduced are listed below:

1. Route sections with different lengths
2. Stochastic driving times
3. Stochastic boarding and alighting

4. Stochastic dwell times
5. Introducing traffic lights to route sections

Main research question

With the goal to investigate the potential of a MARL algorithm in a simulation setup that resembles real world bus operations, the following research questions are defined.

The main research question of this research is: **How does a MARL algorithm perform under different types of (combinations of) environmental and control complexity?**

Here, the environmental and control complexities are combinations of the extensions mentioned above.

Research Sub Questions

The following sub research questions are drawn up to support the main research question:

1. What is the current state-of-the-art of applying MARL systems to both headway management and intersection control?
2. What are valuable extensions to the current state-of-the-art in terms of simulation environment and control complexity?
3. For which (combinations) of the above mentioned extensions is it feasible to implement them in a simulation environment?
4. What is the performance of a MARL algorithm under the above mentioned environment complexities?
5. What is the benefit of using a MARL system for the problem of bus management compared with rule based management?
6. To what extent is the used implementation transferable to other situations?
7. To what extent can the used implementation be translated from a model environment to the real world?

Literature Review

Several lessons can be drawn from the literature review. First of all, it is clear that using RL for headway management is a fruitful concept that has already been implemented with promising results. However, the current state of the art is not yet ready for real world implementation since the studies done use rather idealised settings to train their algorithms in. It is a valuable extension to these studies to train an RL algorithm in a more realistic, and thus complex environment.

The second lesson drawn is that RL algorithms are a suitable fit for intersection control as well. In this field the implementations are somewhat further developed compared to the headway management implementations.

These two lessons combined appear to be a good first step in the direction of an ITS that uses RL since the behaviour of the traffic lights on route can improve or hinder the performance of a bus system. Furthermore, traffic lights are very common on bus routes which makes a solution using them applicable in many regions.

The third lesson concerns the RL implementations used for headway management. From previous studies that incorporate a MARL algorithm it appears to be beneficial to include the forward and backward headway into the observation as local information. Moreover, the research from Cats et al. [10] shows that the best performing holding scheme is based on the mean of the preceding and succeeding busses.

Lastly, in terms of the algorithmic setup Proximal Policy Optimisation (PPO), Actor Critic (A2C) and Deep Q-Learning (DQN) have provided good results in the past. Furthermore, it is important to incorporate an incentive into the algorithm to make sure the agents strive for the system optimum. This can be done by trying to approximate the system state and its value and reward based on that. Another option is to create a reward function based on a global system metric such as the passenger travel times.

Research Methods

A simulation model is developed to be able to investigate the performance of a MARL algorithm under different circumstances. This simulation model simulates a one-directional circular bus line. There are many configuration possibilities in the simulation model, but for this study a bus route of 12 stops is created with 6 busses. The busses have an unlimited capacity, the passengers arrive at the stops according to a Poisson distribution and their destination is uniformly distributed over the coming 10 stops. The dwell time of the busses is the maximum of the boarding and alighting time of the passengers. The dwell time function can be deterministic or stochastic. The travel times on the route sections can be changed per section and can be either deterministic or stochastic. Further, the demand per stop can be specified and each section can contain a traffic light or not. Lastly, the traffic lights follow a predefined cycle which may result in a delay for the busses passing. This delay can be increased, decreased or kept the same by the RL agent controlling the traffic light. For the experiments of this study five scenarios are created with these variables. An overview of the scenarios and their configurations is given in table 1.

Table 1: An overview of the scenarios used for the experiments.

Name	**Travel time**	**Route sections**	**Demand stops**	**Dwell time**	**Traffic lights**
Idealised (ID)	Deterministic	Identical	Identical	Deterministic	No
Stochastic (ST)	Stochastic	Identical	Identical	Stochastic	Yes - Uncontrolled
Stochastic -Traffic Light control (TL)	Stochastic	Identical	Identical	Stochastic	Yes - Controlled
Deterministic - Heterogeneous (DH)	Deterministic	Different	Different	Deterministic	No
Realistic (RE)	Stochastic	Different	Different	Stochastic	Yes - Controlled

These scenarios in the simulation model are used as the environment to train the MARL algorithm in. The Proximal Policy Optimisation (PPO) implementation from RLlib [1] is used for the MARL algorithm. For the observation of the RL agents the difference between their forward and backward headway is provided. After receiving their observation the bus agents can select an action from the following action space: they can hold for 0, 30, 60, 90, 120, 150 or 180 seconds after the boarding and alighting process is completed. For the traffic light agents the action space is described above; increase or decrease the bus delay by 10 seconds or do nothing.

For each of the five scenarios, ten policies are trained with an unique seed and the five best performing are selected for analysis. The analysis is done by executing these five policies in the simulation environment. Of the resulting simulation runs, the averages of the following metrics are stored: the holding time per stop, the passenger waiting time at the stop, the passenger travel time, the experienced crowding, the excess waiting time and the coefficient of variation of the headways.

With the metric scores the performance of the MARL algorithm is compared with two benchmarks; a no control benchmark and a holding control benchmark. The holding control is based on the difference between the forward and backward headway. When the forward headway is more than 15 seconds shorter than the backward headway, the bus will hold for 30 seconds. When it is shorter than 45, 75, 105, 135 or 165 seconds shorter than the backward headway, a holding action of 60, 90, 120, 150 or 180 seconds is applied respectively. This holding strategy is selected because it uses the same information as the MARL algorithm. Furthermore, it performs well on headway equalisation.

Results

The learning process is stable and converged for most seeds. All the policies that converged have learned an effective policy, i.e. they outperform the no control baseline. However, as the scenarios increase with complexity so does the number of seeds that does not converge; from the ten different seeds used, 10, 8, 7, 7 and 6 of them converged for scenario 1 to 5 respectively.

Figure 1 compares the performance of the best policy of each scenario with the holding benchmark. From this figure it is clear that the MARL algorithm manages to outperform the holding benchmark in most scenarios and on most metrics. The obvious exception to this is the deterministic - heterogeneous scenario, with a deterministic and heterogeneous route. Another interesting aspect of these results is that the MARL algorithm keeps the headway very well equalised in the deterministic scenarios (one and four), but not so much in the stochastic ones. In the stochastic scenarios the improvements in performance stem from shorter headways, instead of more reliable headways. From the other results it is also clear that the stochasticity is not a problem for the MARL algorithm, in fact the stability of the performance between policies increases. The heterogeneity of the bus route does add extra difficulty to the problem for the algorithm, as seen in scenario four.

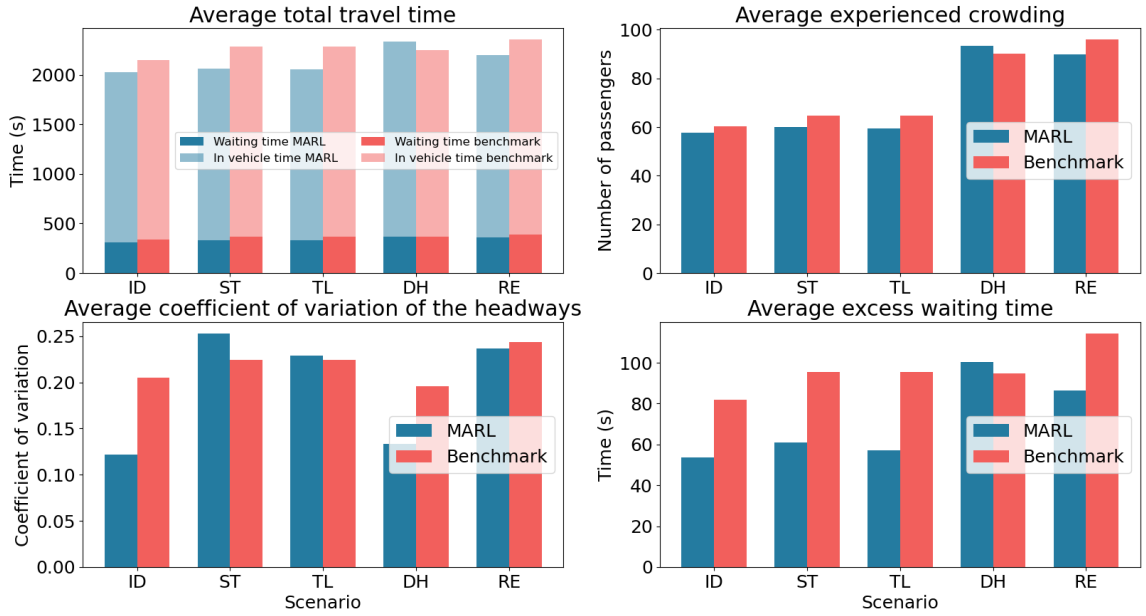

Figure 1: The average metric scores in all five scenarios of the best MARL policy (blue) and the benchmark (red). The total travel time consists of waiting time (blue / red) and in vehicle time (green / yellow).

In all but the deterministic - heterogeneous scenario, the MARL policy manages to use less holding time. In combination with its effective use, this results in a large improvement on the average excess waiting time. Finally, from the difference between scenario two and three it is clear that the traffic light agents are able to add value to the system by improving the average coefficient of variation of the headways, while maintaining equal performance on the other metrics. Even though the traffic lights were able to improve the performance, learning a policy that cooperates well between the busses and the traffic lights proved to be hard, as only a few of the seeds manage to achieve this.

Discussion & Conclusions

The work from Wang et al. [56], where they build a MARL framework to avoid bus bunching in an idealised simulation environment, can be seen as the starting point for this study. The main contributions from this study are expanding the simulation environment in which the MARL algorithm is trained to a more realistic environment. This is done by making several of the processes in the simulation stochastic and creating a heterogeneous bus route.

From the results of the experiments it is clear that a MARL algorithm is able to deal with more complex and realistic bus simulation environments. Specifically, stochastic processes are not a problem for the MARL algorithm, but a heterogeneous bus route does increase the difficulty. Yet, even when both the stochasticity and the heterogeneity are combined, the MARL algorithm manages to outperform the holding benchmark.

It is also clear from the results that the MARL algorithm opts for a different control strategy in stochastic scenarios than it does in deterministic ones. This shows that adding the stochasticity and heterogeneity to the simulation model are useful extensions since they have an impact on the performance and the behaviour of the MARL algorithm. Furthermore, the addition of the traffic light agents turns out to be useful as well, since the performance of the system as a whole is improved.

Additionally, this research shows that using the passenger travel time as the basis for the reward function can be effective with a simulation length around twice the average travel time. This has two main advantages: it is a global variable, which means the agents are inclined to optimise the system as a whole. The second is that this reward function eliminates the need to find a good balance between headway equalisation and driving speed.

Furthermore, the results of this study indicate that training on multiple seeds is useful since the performance and the behaviour is not exactly the same. Especially when more types of agents have to cooperate, like in scenario three and five, the cooperation strategy may develop differently in each learning trajectory.

These results are promising for the use of a MARL algorithm in real world cases. However, there is still more research needed to achieve this goal. With the simulation model used in this study it is possible to simulate a wide variety of bus routes and train a MARL algorithm in this environment. In future research this could be extended to a simulation that can simulate a whole network. This could potentially improve the performance of a bus network since coordination between lines is possible. Yet, modelling and controlling a whole bus network is a challenging task since it is more complex than a single line. Additionally, the action space, the observation space and the number of different actors could be extended as well. Such a study could go more in depth into one scenario instead of exploring different scenarios like this study.

Other topics that should be investigated in future research are the performance of a MARL algorithm on resilience and the robustness of such an algorithm. Robustness concerns the quantitative relation between a disruption of the bus system and its effect on the passengers. While resilience concerns the recovery of such a disruption. Furthermore, the behaviour of the policy should be reliable before it can be implemented in practice, which means that this should be incorporated into the design of the training procedure or be checked after the training procedure.

Together with these research topics, a case study could be executed to discover and solve the practical issues that arise when a MARL algorithm is implemented in practice. Some of the possible issues are already taken into account in this study i.e. the step size is set to 15 seconds to prevent data overload and be robust to data latency. Also, the actions are clear and easy to execute and the information needed for the observation is widely available.

If these obstacles can be overcome and MARL can be used for bus management in practice it has the potential to improve the performance of the bus system. This can be achieved by making the objective of the system the objective of the algorithm, which eliminates the need for heuristics. Additionally, a MARL algorithm can potentially take more information into account than conventional control strategies. Furthermore, due to the learning characteristic of MARL a control policy can be easily updated and adjusted to new situations.

Contents

List of Tables

List of Figures

Abbreviations

ITS intelligent transport system

IV intelligent vehicle

MARL multi-agent reinforcement learning

MAS multi-agent systems

ML machine learning

NN neural network

OD origin destination

PT public transport

RL reinforcement learning

SGD stochastic gradient descent

VDN value decomposition network

1 Introduction

1.1 Problem Description

Bus systems are used all over the world and, compared to the PT modes of train, tram and metro, they are more flexible by nature. This mainly stems from the fact that they do not need dedicated infrastructure. This means that bus systems can easily change routes on a day-to-day time scale as well as on a concession-to-concession time scale. Furthermore, frequencies can be changed on both time scales as well. The downside of not having dedicated infrastructure is that bus systems are often affected by external conditions such as traffic conditions. Additionally, the passenger travel demand for bus systems affects the operational speed of the busses, since the number of boarders and alighters determines the dwell time of busses per stop. The passenger travel demand can vary through time, and with that the operational speed of the busses. This means that when the busses are in operation it can be beneficial to coordinate the system as a whole to make sure the fluctuating external conditions do not result in big disruptions of the bus system. Such disruptions usually come in the form of bus bunching, i.e. two busses of the same line driving very close together. The phenomenon of bus bunching is further explained in section 1.1.1. To avert small disruptions from building up to large disruptions, coordination among the busses is needed, referred to as bus management in this study. This can be done in various ways, for example: skipping stops, temporarily adding / removing busses in / from the system, adjusting driving behaviour through communication with the bus drivers, manipulating traffic signals on the bus routes and possibly demand steering. In this study adjusting driving behaviour and manipulating traffic signals are chosen since (if the necessary (digital) infrastructure is in place) they are less intrusive measures than other possible measures. This study is thus focused on the PT side of ITS, specifically intelligent bus systems. In the coming sections the scientific research related to these measures will be discussed.

1.1.1 Headway Management

Adjusting driving behaviour in bus systems can be done with the aim of maintaining the headways between busses from the same line as even as possible. This is called headway management. Headway management is a problem that has been studied extensively. This is done to avoid the problem of bus bunching. Bus bunching occurs when a bus is delayed and, as a result of its delay, encounters more passengers at its bus stops, which in turn increases its delay due to the longer dwell times. For the following bus on the line the opposite is true: the delay of the first bus results in a smaller time interval between both busses, which means fewer passengers have accumulated at the bus stop, this leads to shorter dwell times. The result of bus bunching is that two busses of the same line drive very close to each other, followed by a large gap, and then, again, two busses close together.

As discussed in [16], headway control can lead to a more reliable PT system and shorter travel times. The shorter travel times are achieved by shorter average waiting times. For example, when a bus line has a frequency of six busses per hour and passengers arrive at a constant rate at the bus stops, the average waiting time would be five minutes when the headways are perfectly even. However, in a completely bunched, system the average waiting time would be ten minutes, since the busses are driving in pairs of two and thus the effective frequency is reduced to three busses per hour. Furthermore, with even headways, the occupancy of the busses is also better spread out [49]. This means fewer passengers will experience an overcrowded bus.

The research field of headway management aims to minimise bus bunching and in this way increase the reliability, quality and comfort of bus systems.

1.1.2 Intersection Control

The manipulation of traffic signals at intersections fits in the field of intersection control and, similarly to headway management, intersection control is not a novel topic. It has been studied for a long time with the general goal of improving traffic flow. This research field started out with optimising the traffic flow around a single intersection [30] and optimising the flow of a whole network based on static origin destination (OD) patterns [60] or link flows [11]. Over time it has evolved to more complicated settings such as dynamic and cooperating controllers [59] and the current state-of-the-art treats topics like intersection control with AV's and without traffic lights [21] [18].

If so desired, a specific role can be created for PT in intersection control. This traditionally comes in the form of transit priority [36], but more complex settings have also been studied where PT only gets priority when

it facilitates sticking to the timetable of the PT system in place [12] [14].

The incentive behind these studies has remained the same over the years: if the traffic flow can be improved, it means less time spent in the transport system for its users. Combined with a value of time (VoT) of the transport system users, this can be converted to a decrease in economical loss for a society.

1.2 Reinforcement Learning

In this study the proposed solutions are drawn from the field of RL, more specifically multi-agent. Even though the RL framework has been around for a while, recently it gained momentum due to advances in the amount of data and computation power available. Abdulhai et al. [5] present several different topics RL could be used for in the transport sector, among them are intersection control and traffic management. They suspect that RL can play an essential role in an ITS. This is a result of the many intelligent entities in such a system, which have to communicate in a manner that takes the objective of both the individual entity and the system into account. In RL, such entities are called agents and MARL provides a framework of coming up with effective behaviour of all the agents. The basic idea behind RL is to train an agent that learns what to do given a situation, called the state, by letting the agent interact with an environment [53]. The agent is thus learning to map states to an action. The agent can interact with the environment in steps. This means that it can select an action, then it has to wait a certain amount of time before it gets a new observation of the environment and can select its new action. This step size determines the frequency of the interaction between the environment and the agent; the environment is continuously evolving and at every step size interval the agent can selects its action. For every action it takes, it is rewarded or punished based on the so called reward function. The reward function has to be designed in such a way that the agent optimises the objective when it tries to obtain as much reward as possible. The resulting mapping of situations to actions for an agent is called the policy of the agent. This RL framework has proven to be effective in many single-agent settings [48], [38], [47]. For multi-agent settings it is harder to find good policies since the interactions between agents create many different solutions and often cooperating is needed for good solutions. The research on MARL systems is ongoing and more research is still needed to put the many different possibilities into practice.

RL is already extensively used in the development of self driving cars and it would be a big loss for the PT industry if RL would not be adopted to benefit them as well. This research aims to provide knowledge and inspiration for one possible application of RL in the PT industry: real-time bus management.

In the real-time bus management setting, every bus is an agent and this can be extended with other agents. All agents will have to be trained to learn what behaviour is good for the system.

1.3 Societal Relevance

The potential of applying RL in ITS has been identified [5] and has also already been applied to headway management [56]. Yet, the implementations thus far have been relatively simple, and as a result they do not provide sufficient information about the applicability of an RL framework for bus management in real day-to-day scenarios. This study builds on previous studies and increases the realism in the scenarios that are investigated. From this research, the understanding of RL frameworks in the bus management setting can grow and thereby the field can come one step closer to the actual implementation for operators of an RL framework for bus management.

An RL framework that is ready for real world deployment can potentially yield various benefits for society. The most evident one is increased reliability of a bus system and a decreased passenger travel time. Other traffic can also profit from this solution when the traffic lights are included in the RL framework. This profit comes as a result of the potential improvement in the way the traffic lights can 'guide' the busses through the remaining traffic. Another promising benefit is the learning aspect of RL, which can yield better long-term performance with less 'maintenance' in the form of redesigning the best solution. This can save valuable time for those constructing the bus management algorithm.

A less work intensive long-term solution can bring the costs of a bus management system down. In turn, the decreased costs can translate to a broader affordability of such a system. This results in a larger share of society that can benefit from an improved bus system.

Along with the potential benefits for bus systems, the acquired knowledge in the field of applied MARL can aid society in a more general sense, as it can be applied in other future cases as well. For example, Abdulhai et al. [5] regard RL as a key ingredient for multiple aspects of an ITS, not just bus management.

This study thus aims to bring the benefits of a future proof, well managed bus system one step closer to reality. While the focus is on bus systems, it can provide inspiration for various other PT systems, such as tram, train, metro or on-demand transport systems.

1.4 Research Questions

1.4.1 Objective

As described in section 1.3, the goal of this study is to improve the reliability and travel time of a bus system in a future proof manner. This could potentially be done by using a MARL framework for bus management. As Wang et al. [56] have shown, such a framework can outperform commonly used benchmarks in an idealised environment; a circular bus line where all the stops and trip sections have the same characteristics and the driving and dwell times are deterministic. This is not a realistic setting and this study aims to progress to a more realistic setting by adding stochasticity and route heterogeneity to the problem. However, it is not clear whether a MARL framework still performs well in more realistic environments, since these tend to be more complex. In this study the intention is to get an idea whether a MARL framework is also a promising solution in more realistic environments or that this is still too complex to be able to train a MARL algorithm with better performance than the current state-of-the-art.

There are several aspects that can be added to the current state-of-the-art of an idealised bus route to make it more realistic. Yet, these aspects will make the route also more complex. Therefore, they are referred to as environmental extension. Besides the possible additions to the route, the number of possible actions in a control algorithm can be expanded beyond holding only as well. These are referred to as control extensions. The aspects that can be introduced are listed below:

- Route extensions
    1. Route sections with different lengths
    2. Stochastic driving times
    3. Stochastic boarding and alighting functions
    4. Stochastic dwell time functions
    5. Introducing traffic lights to route sections
- Control extensions
    1. Traffic light control
    2. Speed control

This means it is possible to create a scenario where not all stops and sections will have the same characteristics. In this way it can provide a more realistic setting and thus should come closer to a useful MARL algorithm for the problem.

Every aspect listed above can be either added or left out to create a unique scenario for the MARL framework to operate in. Next, for each of these unique scenarios the MARL framework can be configured as well. Some of the most important aspects of the configuration are:

- The MARL algorithm used
- The reward function used
- The observation available to the agents
- The action set available to the agents
- The step size (i.e. the frequency of interaction between the agents and the environment, as explained in section 1.2)

Investigating a non-ideal bus line will be the research gap of this study and therefore other research topics surrounding a MARL framework are left for future research. One of such aspects is the performance under disruptions. This study focuses on bus management under daily stochastic variations, with the aim to improve the reliability of the bus system. The performance of the MARL algorithm under disruptions is not investigated, meaning that improving aspects like the robustness and resilience of the system are still open research topics. Considering that these aspects are not investigated, it is evident that the result of this study can not be used as a standalone controller on a real-world system. It should rather be used as an advisory system for the people or system in place. Another option would be that it is used directly, but it is overruled by a different controller system when a disruption occurs.

Moreover, the focus in this study is not on finding novel RL algorithm, but rather on creating methods to make RL applicable for bus control. This relates better to the expertise of civil engineering. Furthermore, finding novel MARL algorithms is a less relevant research gap since a lot of research effort is already focused on this. However, for future research it may be interesting to investigate tailor made (MA)RL algorithms for transport applications. The objective of this study is thus the following:

To investigate the potential of a MARL algorithm in a simulation setup that resembles real world bus operations.

1.4.2 Main Research Question

The main research question of this research is: **How does a MARL algorithm perform under different types of (combinations of) environmental and control complexity?**

Here, the environmental and control complexities are combinations of the extensions brought forward in section 1.4.1.

1.4.3 Research Sub Questions

The following sub research questions are drawn up to support the main research question:

1. What is the current state-of-the-art of applying MARL systems to both headway management and intersection control?
2. What are valuable extensions to the current state-of-the-art in terms of simulation environment and control complexity?
3. For which (combinations) of the above mentioned extensions is it feasible to implement them in a simulation environment?
4. What is the performance of a MARL algorithm under the above mentioned environment complexities?
5. What is the benefit of using a MARL system for the problem of bus management compared with rule based management?
6. To what extent is the used implementation transferable to other situations?
7. To what extent can the used implementation be translated from a model environment to the real world?

In this sense, the aim of this study is to first find the current state-of-the-art in terms of MARL application for bus management. Then, this state-of-the-art is extended with the most promising, widely applicable and feasible control extensions. Next to this extension, the learning environment is made more realistic by introducing more realism into vital processes of the bus system. The first addition yields a more elaborate MARL bus management implementation than currently used in the literature. The second addition moves from an idealised simulation environment to a more realistic one. This is done to get a better indication of the usefulness of a MARL algorithm for bus management in real world scenarios. The performance of the used MARL setup is analysed along with the transferability of the setup. The outcome of this analysis is used to identify topics for future research that should be covered in order to make real world MARL bus management a reality.

2 Literature Review

A literature review is conducted to be able to do research that aligns well with the current state-of-the-art. This literature review is done to answer the first two research sub questions as formulated in section 1.4. This thus comprises of the state-of-the-art of the use of MARL in headway management and intersection control and the most valuable extensions to this in terms of the environment and the control measures used. Additionally, the design choices for the simulation environment are discussed since this has an indirect impact on the performance of the MARL algorithm. Lastly, the topic of MARL is discussed to give an intuition for the decision that have to be made in this research.

2.1 Modelling Bus Systems

Simulation models are often used to train and test both intersection control and headway management algorithms. In this study the this is also the case. Therefore it is paramount that this model simulates the aspects of the bus system that are important for the problem appropriately. Furthermore, the simulation model should be fast since it will be evaluated many times when training the algorithm. Due to these requirements the model should adhere to a famous quote of Albert Einstein: "Everything should be made as simple as possible, but no simpler".

Two important aspects for the simulation model of this study are the passenger arrival function and the dwell time function. Without a proper implementation of these two, bunching will not be observed. Furthermore, to come to a complete simulation model, proper driving times should be implemented for all the sections on the bus route and realistic green time restrictions (e.g. a minimal green time per direction) should be used for the intersections.

In high-frequent PT systems the passenger arrival function is often assumed to be a homogeneous Poisson process [45], [31]. This assumption is based on the idea that the passengers do not consult the timetable of the PT service due to the high frequency of the system. There are numerous variations on this, such as the non-homogeneous Poisson process used in [31], which improve the accuracy or the computation time. These variations become more relevant when there are reasons to believe that the passengers do not arrive at random at the stops. This could be the case when the frequency of the system is not high enough or when the bus system is is fed by another low frequent PT system.

The dwell time function is often assumed to be linear with the number of boarders and alighters [4]. This function contains a constant that represents the time needed for stopping, opening and closing the doors and driving away, [4] recommends to choose this constant between 10 and 15 seconds. In the study by [51] they used a modification to the linear function by creating two separate functions; one for boarding and one for alighting, the dwell time function is then the maximum of the two. Evidently, such dwell time functions remain an approximation of reality and the dwell time of busses in a bus system is influenced by multiple factors.

Driving times of the busses on their different route sections can be modelled in numerous ways. Since there are several factors that influence the driving times (e.g. road conditions, hour of the day, weather, etc...) it is important to define which of these factors are held constant in the model and which are variables that influence the driving times. The choices made in this regard will determine how the driving times are distributed. The work of Uno et al. [54], Srinivasan et al. [50] and Li et al. [34] has shown that the distribution of driving times along a route section often be approximated by a log normal distribution.

2.2 Control Measures

2.2.1 Headway Management

As discussed in section 1.1.1, headway management in bus systems aims to increase the quality of the system by improving the reliability and comfort. Generally this is achieved by keeping the headways between busses equally distributed and thus addressing the problem of bus bunching.

The characteristics of a bus system that lead to bunching are the characteristics of a high frequency and high demand bus system [19]. Only in these circumstances the passengers will arrive at random at the bus stops and only with large amount of passengers will the dwell time be significantly influenced by the passengers.

It is not exactly defined when a system can be called a high frequency system, but passengers only start to arrive randomly at the stops when the headway is 10 minutes or smaller according to [24]. Yet, Ingvardson et al. [29] finds that with headways of 5 minutes 43% of the passengers still arrive in a timely manner at the stops.

There is a variety of methods to achieve regular headways. Some of them focus on specific settings, such as long headway services [55]. However, most of them focus on high frequency and high demand systems, since these are most prone to bus bunching. The basic idea behind many state-of-the-art headway regularisation methods today is laid out by Daganzo et al. [17]. These methods are often mainly based on holding; the idea to hold a bus at a given location to make sure its following bus does not get too far behind. Or, put in other words, to hold a bus at a given location to make sure it does not get too close to its preceding bus. As brought up in [17] and [49], with holding schemes it is paramount to keep the holding to a minimum. Too much holding can lead to a low average driving speed and thus *"this medicine (slack) is sometimes worse than the illness (irregular headways)"* [17]. Moreover, low average driving speeds also increase the operating costs and the required fleet size [49]. An example of a holding scheme implementation is given in [58] by Xuan et al. Cats et al. test several different holding schemes in [10] and they conclude that the best performing strategy is the one that is based on the mean of the preceding and succeeding busses.

This type of headway regularisation is currently most used in literature and is based on simple or complex rules. Further, most holding schemes are station based and thus only allow for control decisions on a subset of the bus stops. A disadvantage of station control methods is *"that decisions can be taken only at stops, resulting in a significant time lag between observations and control actions. This can play a vital role if the system experiences various uncertainties both in time and space, which is the case en route (congestion heterogeneity) and at stops (passenger demand heterogeneity)"* [49]. Despite being rule based, several of these methods allow the optimal headway to "emerge" by itself, thus not needing a pre-specified headway [8].

In addition to holding schemes, headway regularisation can also be achieved by speed control (e.g. [7]) or stop skipping. They are not as common as holding, since their implementation is not as straightforward as with holding. Speed control for example, is easy to implement when a bus-only lane is in place, but on shared infrastructure it becomes harder. Stop skipping on the other hand is simple to implement, but the difficulty lies in effective communication to the passengers. As expected, all these methods can also be combined in more complex headway management schemes (e.g. [23]).

One flaw in many of the regularisation studies is that they are tested in ideal simulation environments, when the reality is far from ideal. Congestion and traffic lights (often negatively) influence the headways of a bus system as demonstrated by Hans et al. [28]. Studies such as [20], who take this into account, yield a more realistic picture of the performance of a regularisation scheme.

2.2.2 Intersection Control

IC has been an active research field for many years and is being transformed by emerging technologies such as smart vehicles, as discussed in section 1.1.2.

Classically intersection control was the science of creating a good static traffic light controller for one or more intersections in a road network. This was done by simulating the observed or projected traffic demand in a simulation model of the road network and optimise the traffic controllers in this simulation model. The optimal controllers are then thus based on link flows and OD patterns, such as in [11] and [60]. This can also done on the scale of one intersection to investigate different types of intersections, an example of such a study is [30].

With the advent of road sensors and other forms of traffic data, it was possible to move away from static controllers and move towards dynamic ones [39]. With dynamic controllers it is possible to optimise an intersection controller better with respect to the current conditions, which often fluctuate throughout the day.

The next development that will change the field of intersection control is the arrival of intelligent vehicle (IV)'s. intersection control in a setting with IV's does not need physical traffic lights anymore [21]; whether or not a car is allowed to cross an intersection can be communicated to the car directly by the controller in place. Even in the transition period, when not all of the cars on the road are intelligent, it would be possible to use

such systems to improve the physical traffic light systems [59].

In traffic scenario's where (most of the) vehicles are intelligent it is possible to use more complex objectives for the intersection controller than the traffic flow only. An example of such an additional objective is the driver experience or comfort, which is taken into account by [18]. With intelligent controllers in place it becomes more accessible to give PT a special role in the intersection control. Examples of special PT roles are investigated by [14], [23], [25], where busses get priority at the intersections based on several different aspects like their delay and the capacity at the succeeding bus stops. MARL is a commonly used framework for intelligent intersection control, especially when cooperation between different intersections becomes important [26].

2.2.3 Connecting Intersection Control to Headway Management

Given the extensive research on intersection control and on headway management, it is interesting to investigate whether these two topics can strengthen each other. After all, waiting for a red light at an intersection can be seen as randomised holding for busses. For a PT system it is advantageous if this randomised holding can be transformed to optimised holding. Moreover, when all the busses of a bus line have equal headways, the line is in a fragile equilibrium. This equilibrium can easily be disturbed by a red light at an intersection, causing bunching [23].

Early attempts in this direction have been done by Liu et al. [36] and Chandrasekar et al. [12]. In [36] they concentrate on transit priority at intersection control, [12] takes this a step further by explicitly taking the headways into account. Chow et al. [14] extend this, they incorporate a kinematic wave model to simulate the traffic situation and minimise the deviation of busses from their desired headway, while trying to minimise the delay of the regular traffic as well. In the work from Estrada et al. [23] this concept is taken even further, in their research intersection control is combined with speed control and holding to optimise headways and system costs. They propose a solution where some slack is built into the bus schedules and additional measures can be taken real-time with green light extension (at intersections) and speed control. Additionally, they note that real time control is especially useful when the vehicle departures are not synchronised with the traffic light settings, since this is a more unstable setting. This unveils the potential benefit of coordinating bus schedules with intersection control. The research of Estrada et al. [23] is not done with the aid of RL, but, given the vast amount of control options in such a system, it seems probable an RL based controller could outperform a rule based controller.

2.3 Reinforcement learning

2.3.1 Key Concepts of Reinforcement Learning

The environment in which line busses have to operate is a complex one, namely a PT system on the (urban) road. This environment has many influencing factors like the other traffic, the weather, traffic lights and the passengers in the PT system. Furthermore, the influencing factors do not influence the environment independently, they may have correlations. Previous headway management studies came up with rule based algorithms to be able to control busses in an effective manner is this complex environment. However, these rule based algorithms are essentially a heuristic for the true optimal behaviour of the busses in this complex environment. With the advent of massive computing power and big data it is now possible to search for the true optimal solution for every specific situation, not a heuristic. The problem then becomes finding the optimal solution in the enormous set of possible solutions. RL can provide a framework to find and continuously update the (locally) optimal solution for every specific situation.

It is important to have an idea of what the term RL entails since the solutions of this research are drawn from the field of RL. With an understanding of the key aspects of RL it will be easier to see the discussed problems from the perspective of RL and the transition towards the creation of the RL framework will be smoother. Therefore, RL will be shortly introduced in the this section and the choices within an RL framework will be discussed.

As mentioned in section 1.2, RL, like most machine learning (ML) methods, RL is gaining attention due to the continuous improvements in computing power and the increasing availability of data. However, RL is different from other machine learning methods in the sense that it is interactive [53]. RL will learn by trying, similar to the way humans learn. Given a 'world' or environment and a set of possible actions, an RL agent will try an action, observe the resulting state of its environment and then try a new action. The RL agent will then learn from its past experience and converge to better behaviour. An RL agent will thus create its own

experience by interacting with an environment. This is not the case in most machine learning methods where the experience is a given and is usually static. The environment of an RL agent can be the real world; for example, a robot can be taught to clean a floor. This can work well since the robot in this situation would be free to make errors and clean the floor badly during training. However, making errors and exhibiting inefficient behaviour becomes expensive and possibly dangerous when the RL agent is not a floor-cleaning robot but a bus managing controller. Therefore, a simulation environment is usually used for the training of the RL agent and the RL agent is only used in the real world when it is fully trained and exhibits reliable behaviour. Another benefit of training in a simulation environment is that it can be much faster than the real world. For example, the duration of modelling a full day of bus operations can be on the scale of seconds or minutes using a simulation model. This means that an RL agent can gain experience faster when using a simulation model as environment instead of the real world. It is evident that this simulation model should be an appropriate representation of the world if the RL is to be used in the real world at a later stage. This is where big data sets come into play: they can help a modeller calibrate the model by making it data driven. In this way data drives the learning of an RL agent, but in an indirect way, which is different from most ML methods.

A simulation environment or the real world can thus be used for training an RL agent. This training process consists of two types of behaviour: exploratory and greedy behaviour [53]. The exploratory behaviour is characterised by trying new actions irrespective of the other possible actions given the state. Greedy behaviour on the other hand is characterised by behaviour according to the agents policy: the agent will thus choose the action that he considers to be the best given the state. In general, exploration is necessary during training, but the percentage of exploratory actions taken is slowly decreased. When a trained agent is deployed in the real world it is often programmed to exhibit greedy behaviour only. However, this is case dependent and it may be important to keep exploring, especially when the environment will change over time. The option to keep exploring during deployment can be an advantage of RL solutions since it means the agent continuously adapts its behaviour to a changing environment. Nonetheless, in the case of a bus system it is probably more useful to update the simulation environment with new data than to train an RL agent in the real world. The main reason for this is the speed of learning; in a simulation environment a full day of operations can be modelled on a much smaller time scale.

To be able to judge what is good, an RL agent needs a reward function [53]: a function that rewards the agent for every action according to how good or bad it was. Defining the reward function is thus a paramount part of RL, since it will steer the behaviour of the RL agent. For example, when a bus gets rewarded for every transported passenger, it will learn to optimise its efficiency with respect to transporting passengers to get as much reward as possible. Without the incentive of rewards (positive or negative) an RL agent will not learn a meaningful policy.

By using this reward function it becomes unnecessary to define what the desired behaviour looks like, only the end goal of the behaviour needs to be defined. This goal can usually be defined exactly and rule based heuristics are thus no longer required. This means that an RL agent will look for the optimal behaviour for every situation, not just for the optimal behaviour in the majority of the situations, which is the case with a heuristic.

When a transport system transitions from a regular transport system to an ITS, more aspects of the system become controllable in real time. This means more aspects are optimisable in real time. Finding a good heuristic when there are many factors in play can be a challenging task. Therefore, there is a growing need to move away from solutions based on heuristics and towards solutions driven by the end goal.

Humans are able to come up with very good solutions (desired behaviour of busses) when the problem of bus management is relatively simple e.g. headway management on a ideal bus line with homogeneous demand, without intersections and where the action space consists of holding or not holding a bus at a stop. However, when the action space becomes larger (e.g. adding speeding up / slowing down of the busses, traffic light control, continuous feedback, etc...) or the environment becomes more complex (e.g. intersections, heterogeneous demand, interactions with other traffic, etc...) it becomes hard for humans to define the desired behaviour of a bus for every given situation. This is the case because the solution space is enormous and the dynamics of the problem are generally not fully understood. In such a situation it is usually still relatively straight forward what is desired of the busses (e.g. minimise overall travel time), but is has become unclear what behaviour will lead to achieving this. RL is essentially a framework to move through this solution space, try solutions, learn and eventually converge to (locally) optimal behaviour. When RL is combined with function approximation,

it can even relate new states of the environment to states it has already seen and use the experience from the similar states to exhibit appropriate behaviour.

In this study the performance of the RL algorithm is investigated under different types of environmental complexity. The less complex environments are used to see if RL is a suitable framework for the bus management problem, while the more complex environments are used to see if RL can be an asset in real life bus management. In these environments several busses, and in the complex environments traffic lights as well, have to interact and cooperate with each other. This means the environment has multiple actors. Therefore, a MARL algorithm is chosen to tackle the bus management problem.

2.3.2 Multi-Agent Reinforcement Learning

The RL concepts explained in section 2.3.1 are explained from the angle of single agent RL since this is a less complicated setting than the MARL setting. This is also the reason that the field of single agent RL is further developed than MARL. Single agent RL has seen many successes [48], [38], [47] and is already used in production on several products like self driving vehicles [57]. MARL on the other hand is not as far developed since it quickly becomes more complex than the single agent setting. In this section several MARL frameworks are explored and their pro's and cons are discussed. Important to note is that the focus here is on the possible methods to get multiple RL agents to cooperate given a task they have to execute as a team. The algorithms that enable the individual agents to learn are thus not discussed since this does not contribute to the understanding of the problems dealt with in this study. Furthermore, in this section a new term is used: 'learner' which represents an RL agent in a MARL setting. The term 'agent' is not used here since in an environment there may be multiple acting agents that are all controlled by the same RL agent which may lead to confusing descriptions.

MARL can be seen as a subsection of the research field of multi-agent systems (MAS). The field of MAS has influences from various other fields and concepts like graph theory, game theory, control theory, swarm intelligence and MDP [44]. Here the focus is on MARL, but MARL too is influenced by the other fields just mentioned.

In MAS there are several difficult problems that arise. The major problem is making sure that the performance of the team comes before the individual performance. This involves cooperation, which is often hard to incorporate in a MARL framework. One of the reasons this is difficult is due to the predictability vs. exploitation dilemma. Here predictability can be explained as 'stick to the plan', making sure the other learners in the system know what to expect from you. Exploitation on the other hand is explained by using the knowledge that a learner has at the moment to get as much reward for him self as possible given the current situation. The latter is local optimal behaviour, but the former is often, but not always, globally better. This dilemma holds for every learner in the system.

A straightforward implementation of a MAS using RL is to use team learning [41]. In team learning there is one learner in the system that controls all the agents. The central learner is the only RL agent in this case and operates like a 'puppet master'. This central learner is rewarded according to the team performance, this means the performance of the team as a whole matters, not the performance of the individual agents in the system. This is a considerable advantage of team learning. However, the downside is that the action space of the central learner and the state space of the environment can become prohibitively large [41]. This is because the size of the action space of the central learner is equal to the size of the action space of the individual agents to the power of the number of individual agents. Furthermore, when a team learning system is deployed there should be a central point of communication, due to the presence of the central learner. All the necessary data has to be available in this central point, which can create numerous challenges. "Moreover, in the sensor-network applications, in which the information is observed across a large number of scattered centers, collecting all this local information to a centralized unit, under some limitations such as energy limitation, privacy constraints, geographical limitations, and hardware failures, is often a formidable task" [40]. Additionally, this can make a system vulnerable since there is a single point of failure.

For these reasons it is desirable to have a distributed system. A way to do this is by creating a learner for every agent in the system. This setup can solve some of the challenges that come with team learning, but provides a new challenge; a non-stationary environment. In a single learner setting, the learner learns the 'rules' of the environment, meaning every time a learner is in state A and takes action B, the probability of ending up in state C is the same. [40] The environment is thus stationary. This is not the case with MARL, because there are other learners in the environment whose behaviour is non-stationary. This makes it harder for a learner to

converge to a good policy.

Nonetheless, this method is used in some settings and is called 'independent learners'. With this method the other learners are thus modelled as the environment. A variation on the independent learners method is one where all the learners use and update the same policy. By using this 'shared policy' the learning goes faster because all the agents contribute their learning to the same policy. Another method that can be used to speed up learning is transfer learning [15]. With transfer learning the knowledge learned by one agent can be (partially) transferred to another agent. This method can speed up the learning process in situations where training an individual agent takes a lot of computational power. However, the problem remains with these methods that it is hard to learn to cooperate for the individual learners. Additionally, due to the non-stationary environment, there is no guarantee that the learners will converge to a locally optimal policy [43]. One of the algorithms that has proven to be able to handle these difficulties well is the proximal policy optimisation (PPO) algorithm [46]. PPO uses "multiple epochs of stochastic gradient ascent to perform each policy update" [46]. Furthermore, each update is done within a "trust region"; i.e. each step can only change the current policy within the predefined trust region around the current policy.

The problem of a non-stationary environment is addressed by a method called 'fully observable critic' [40]. In this method there is a central entity called the critic, which observes all observations and actions of the learners in the system. The observation of the central critic, or an approximation of the observation is available to all the learners. Because the global information of the critic is added to the observation of the individual learners, the environment becomes stationary for the individual learners; given observation of state A and action B the probability of ending up in state C is always the same. Nonetheless, the global observation makes the observation space very large and therefore it becomes difficult to find the best action. This issue is often referred to as 'hard greedification', since it is hard to find the greedy action.

An alternative MARL framework has been developed to address the difficulty of cooperation and hard greedification: 'actor critic'. In this framework there is a central critic that learns to approximate the value of a global state given the current state, the selected actions of the individual learners and possibly other inputs. This is similar to the fully observable critic framework, but the key difference is that in an actor critic framework the central critic is not only approximating the global state, it is also learning to approximate its value. Every individual learner has an actor that takes decisions based on its observation and after taking the decision it is criticised or rewarded by the central critic. The central critic should thus learn to assess the value of an action of an individual learner for the system as a whole, and reward him accordingly. The decision making process for the individual actor is uncomplicated in this framework since it is only based on its own (local) observation, yet it is rewarded on its value for the system as a whole.

The actor critic framework has proven its worth [27], but, similar to several other methods, an important challenge that comes with the framework is an accurate approximation of the global value function. It is hard to define the value of a state accurately since there are many factors contributing to it. The approximation can be made easier by breaking the global value function up into a set of local value functions. This is done by learning a value function for a subset of the agents and combining the value functions of all subsets to form the joint value function of the global state. This technique is called value function factorisation [40]. By using value function factorisation less parameters have to be learned, which makes the learning easier. Furthermore, if the set of parameters of the local value function is small enough, it becomes possible to find the maximum and the argmax of it, similar to single agent settings. This in turn allows to find the maximum and the argmax of the joint value function, since the joint value function is simply the sum of all the local value functions. The downside to this approach is that it can oversimplify the situation in the real world since the value function factorisation assumes that all the local value functions are independent of each other. Often this is not a realistic assumption. This property can block interactions, and thus cooperation, between agents that optimise a different local value function.

An extreme example of value function factorisation is given by [52], where they come up with value decomposition network (VDN). In this study the joint value function is split up into local value functions containing only one learner. Hence the joint value function is a sum of the value functions of all the individual learners. The argmax of the joint value function then becomes the greedy action for every learner on its own value function. This is a very useful property since it makes the learning more straightforward. However, this extreme factorisation means that this method assumes that all agents are independent. Due to this assumption the learners are no longer incentivised to coordinate their actions.

A variant on VDN is QMIX [43]. The difference with VDN is that in QMIX the value of the joint value function is not simply the sum of the local value functions. The local value functions are mixed together to form the joint value function. This means that all the values of the local value functions are multiplied by a specific factor to form the joint value function. This factor depends on the current state of the system, but not on the actions chosen by the learners.

QMIX mixes the values of the local value functions under the condition that the derivative of the joint value function with respect to each local value function is positive. This comes down to the condition that $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0$, where Q_{tot} stands for the global value function and Q_a for the value function of an individual learner [43].

VDN has to make sure that the joint value function is a strict sum of the local value functions. As a result of this limited flexibility of VDN, it makes errors in approximating the joint value function or the local value function or both if the true joint value function is non-linear.

Due to the mixing, QMIX can better approximate non-linear joint value functions than VDN. Making an error in the value function approximation is not a big problem; as long as the learners choose the right action, the reward yielded will be the same no matter the approximation of that value. However, an error in the estimation of the joint value function can have negative consequences in the preceding states. The actions chosen in the preceding states are based on the value estimation of their following states. An over- or underestimation can thus lead to erroneous choices by the learners. QMIX does not have this problem with non-linear value functions. An example of a joint value function that QMIX can learn and VDN cannot, is given in table 2.

Table 2: Joint value function, reward depends on the actions taken by both agents. [43]

		Agent 2	
		A	B
Agent 1	A	0	1
	B	1	8

Yet, due to the condition $\frac{\partial Q_{tot}}{\partial Q_a} \geq 0$, QMIX still has the advantageous property of VDN that the argmax of the joint value function is given by the argmax of all the local value functions. This keeps the learning straightforward while the action selection is improved due to the added flexibility of making the joint value function more than simply the sum of all local value functions. QMIX can thus better represent the added value of coordination among the learners. Using the quote of Aristotle "the whole is greater than the sum of its parts". This is better reflected by QMIX than by VDN.

Methods like QMIX and actor critic, decentralise the learners of the MAS system to an extent, but there is still a central entity (the mixing network and the central critic respectively) in the system. This central entity is necessary while learning, yet, when all learners have learned a good policy the central entity can be discarded. This means the agents can be trained in a setting where the central entity is available to all agents, but they can act decentralised in an execution phase. This property is valuable since it can solve the earlier discussed problems of real time central data storage and the system having a single point of failure.

Another approach to the MARL problem is to let the learners not only cooperate, but communicate too. This strategy is used in the 'consensus' and 'learn to communicate' methods [40]. In the consensus method a learner can communicate with its neighbouring learners during training and execution. The information exchanged can be observations or policy weights/gradients or both. In this way each learner only uses its own observations and information from its neighbours, hence the problem is still tractable. The consensus method can be used with different learning schemes like actor critic or Q-learning [40]. The effectiveness of the consensus method depends on the problem setting. For the 'learn to communicate' method the communication is less rigid. With learn to communicate the learners learn during training to send effective messages among each other. In this manner they learn what to send, when to send it and to whom to send it. The resulting communication policies are then used during execution.

2.4 The Prospects of Multi-Agent Reinforcement Learning for Bus Management

With an idea of the key concepts and challenges of RL and MARL it is interesting to see how these concepts can be applied in the fields of intersection control and headway management or combinations of both. As discussed in section 2.3, the use of RL or MARL can be a benefit to these fields since they pose problems that should be looked at from the system point of view and both usually have a large solution space.

Evidently, this is not the first study that aims to apply the concepts of RL and MARL to intersection control and headway management. Therefore, in the coming sections the preceding research on the combination of these fields will be explored.

2.4.1 RL for Intersection Control

For this research topic RL can provide an alternative to rule based intersection control. With the RL framework the agent that controls the intersection is able to learn an effective control policy based on its experience. Furthermore, the control policy can be state dependent if there is real-time data available like the number of waiting cars for every traffic light. Such implementations have been developed in [36], [32] and several others. Moreover, the controller implementation by Liu et al. [36] also contains transit priority. Some of these RL implementations were developed almost 20 years ago. Nevertheless, the research in this area is still active due to the successes booked.

One of these successes is shown by El-Tantawy et al. [22], who used RL for their intersection control solution. They reduced congestion and further improved their controller systems by making several controller systems cooperate. *"The results show unprecedented reduction in the average intersection delay ranging from 27% in mode 1 [non-cooperative controllers] to 39% in mode 2 [cooperative controllers] at the network level and travel-time savings of 15% in mode 1 and 26% in mode 2, along the busiest routes in Downtown Toronto" [22].*

These are satisfying results that demonstrate the potential of RL for this field. The performance can probably be improved further as the field of RL develops.

2.4.2 RL for Headway Management

As discussed in section 2.3.1, it can be difficult to come up with a rule based headway regulation system that is robust in small and big disruptions and that is able to take the heterogeneous traffic conditions, traffic lights, intersections and demand heterogeneity into account. To this extent RL can be of benefit: with RL one has to define what should be optimised without defining how to attain the optimal value for every specific situation. This benefit is highlighted in [56], an RL agent can take all the details into account that are relevant while keeping the focus on optimising the system as a whole. The RL agent will thus choose which headway is best for the system and may in some cases not steer the busses towards equal headways if that is better for the overall performance of the system.

Several implementations of MARL for headway management are investigated by Alisiani et al. [6], Chen et al. [13] and Wang et al. [56]. From these studies it is evident that there are various algorithmic setups that can be effective for MARL in headway management and that the research on MARL algorithms is continuously evolving. While all of them take the multi-agent approach, the learning algorithms applied vary among them. As examined in section 2.3.2, each MARL algorithm comes with its own pros and cons. In the observation space all three studies incorporate the forward headway or at least the information to calculate the forward headway. Chen et al. also incorporates the number of onboard passengers, whereas Wang et al. incorporate the number of passengers at the coming stop. All observations are thus local. To achieve global cooperation Alisiani et al. created a reward function that is based on global metrics. Wang et al. aimed for global cooperation through a centralised joint action tracker. It is hard to compare the performance of each configuration with the other one. Yet, it is clear that a MARL algorithm should aim for global cooperation to outperform the current state-of-the-art.

2.4.3 Investigating More Complicated Environments

As mentioned in section 2.2.1, many of the headway management studies have been tested in a fairly idealised setting. This also applies for the studies that used RL methods to address the headway management problem

([6], [13] and [56]). This raises a clean and clear test environment with easy to understand results. However, in reality, bus lines with these ideal routes are very rare. It is more common for a bus route to encounter heterogeneous traffic conditions, traffic lights and heterogeneous demand. Additionally, busses often drive through different types of environments. This is because bus systems are often applied in moderately densely populated areas and rural areas. In very densely populated areas metro, tram or train systems are often in favour. However, many bus lines start and terminate at big transport hubs, which are often located in very densely populated areas.

The transitions along the bus route call for distinct bus management on the diverse route sections. While equal headways are relevant no matter the characteristics of the route, the level of control needed to attain this may differ throughout the bus route. For example: in rural areas or on a separate bus lane it may be easy for a bus to speed up or slow down in order to stick to its desired headway, yet in a city centre this may prove to be more difficult. Along the same lines: around a big bus station in a city centre it may be worthwhile to control the traffic lights in the area to optimise the traffic flow, while in a rural setting this may be less likely to be economically feasible.

Nevertheless, the control of the dense urban area and the other parts of the route are not two separate problems, they depend on each other. For example, when a bus station in the dense urban area is at full capacity and no more busses can enter the station area, then this will affect the optimal policy on the route section before the station area. This relation and the best way to optimise the two sections together are investigated in this research. A bus route that goes through dense urban areas and rural areas as well, is not exceptional, meaning results from this research have many applications. A previous study going in this direction is done by Gao et al. [25], where they use intersection and speed control to minimise delay at intersections and bus stops. Accompanying this, they make sure the capacity of the bus stops is never exceeded.

Rule based headway management is generally based on one rule for the whole system. This means it will use the same regulating rule in rural areas as in dense urban areas and during on-peak and off-peak periods. It would be a laborious task to define specific rules for every type of environment along a bus route, yet with RL this is a given. With RL the objective is defined and is the same along the route, but the optimal behaviour is not predefined and will be discovered by the MARL framework. The RL agents are not inclined to use static rules and will try to find the best behaviour for every specific situation. Therefore investigating an RL control algorithm is especially interesting in more complex environments such as a heterogeneous bus route.

On top of a more complex simulation environment, it can be interesting to give the MARL agents more information. As suggested in [56], real-time information or predictions on the number of passengers that are waiting at bus stops may improve the performance of the controller. This is confirmed by Berrebi et al. [9], where they find that busses need less holding time when this information is available.

2.5 Synthesis

Several lessons can be drawn from the literature review. First of all, it is clear that using RL for headway management is a fruitful concept that has already been implemented with promising results. However, the current state of the art is not yet ready for real world implementation, since the studies done use rather idealised settings to train their algorithms in. It is a valuable extension to these studies to train an RL algorithm in a more realistic, and thus complex environment.

The second lesson drawn is that RL algorithms are a suitable fit for intersection control as well. In this field the implementations are somewhat further developed compared to the headway management implementations.

These two lessons combined appear to be a good first step in the direction of an ITS that uses RL since the behaviour of the traffic lights on route can improve or hinder the performance of a bus system. Furthermore, traffic lights are very common on bus routes which makes a solution using them applicable in many regions.

The third lessons concerns the RL implementations used for headway management. From previous studies that incorporate a MARL algorithm it appears to be beneficial to include the forward and backward headway into the observation as local information. Moreover, the research from Cats et al. [10] shows that the best performing holding scheme is based on the mean of the preceding and succeeding busses.

Lastly, in terms of the algorithmic setup PPO, A2C and DQN have provided good results in the past. Additionally, QMIX may be an interesting option since it is specifically created for multi agent scenarios. Furthermore, it is important to incorporate something into the algorithm to make sure the agents strive for the system optimum. This can be done by trying to approximate the system state and its value and reward based on that. Another option is to create a reward function based on a global system metric such as the passenger travel times.

3 Model

With the information from the literature in mind, a simulation model is developed on which several configurations of a MARL framework are tested. The simulation model is build from scratch, for this reason the first research effort is dedicated to investigating whether a MARL framework is be able to use it as an learning environment. In the coming sections the decisions on the type of model are discussed along side with all the aspects of the simulation model that can be configured for a specific scenario.

3.1 Environment Simulation Model

3.1.1 Modelling Choices

Like discussed in section 1.4, in this study multiple extensions are considered to the base simulation model of an ideal bus model. Based on the complexity of such MARL problems, as mentioned in section 2.3.2, the algorithm is first tested on the base scenario without extensions. From that starting point the complexity of the simulation model is increased to see if the algorithm is still able to learn and converge to reasonable behaviour in this environment. In this section the choices made with respect to the modelling framework and the type of model are described, in the next sections the possible configurations and extensions are discussed.

As mentioned, a custom built environment is created for the experiments in this study. This environment should simulate the busses in operation. As found in the literature study, it is paramount that the simulation model has all the necessary properties to simulate the bunching effects of busses in a realistic manner. This means the dwell time should have a component that depends on the number of passengers that board or alight the bus. Additionally, the dwell time should have a constant for stopping as discussed in section 2.1. Furthermore, the passenger arrival functions and the travel times of the busses should be realistic.

There are numerous ways to model such a system e.g. with an agent based model or event based model. In this study an event based model is selected because the focus of this research is on the busses, not the passengers. The busses are the focus of this research since they are the controllable factor in the system, not the passengers; they are modelled as a given. The busses are few in number and their behaviour is controlled by the MARL algorithm. This is not a typical use case for an agent based model, where the behaviour of one agent is usually known, but the behaviour of a group of such agents is not well understood. Therefore, an event based model is selected.

Furthermore, the model is dynamic because the interaction between the busses is important. The events in the simulation can be both deterministic or stochastic, depending on the scenario created. This holds for the events like travelling along a route section between stops and dwelling at the stops. Regarding the level of detail in the simulation model; every passenger and every bus in the simulation is of importance and is thus simulated microscopically. However, the other traffic on the road has only a secondary effect on the busses; namely the maximum driving speed attainable on a route section. Therefore, the other road traffic is not simulated, but the driving times of the busses can be adjusted and made stochastic per section to be able to reflect the true conditions on a specific route section. As mentioned in section 1.4, this study is focused on improving the reliability of a bus system, not its robustness or resilience. This means that, even though several elements in the simulation model can be stochastic, large disruptions of the system are not simulated.

There are many tools to create such a simulation environment, in this study SimPy [2] is selected. "SimPy is a process-based discrete-event simulation framework based on standard Python" [2]. This has the direct advantage that it is written in the same programming language as the RLlib [1] framework, which is used for the MARL algorithms in this study. This makes it easier to connect the two frameworks. Another advantage of SimPy is the level of freedom in the framework, since it is a low level simulation framework.

3.1.2 Base Environment

A circular bus line is taken as the basis of the simulation model. This is done because it provides freedom in terms of the simulation duration; the busses can run perpetually, but a short simulation run is also possible. Furthermore, on a circular line there is no first bus nor a last bus which simplifies policies or calculations that incorporate the headways of the busses. When re-creating a non-circular bus line in the simulation model, it

is assumed that the busses turn over and the end points of the line. This creates the possibility to model a non-circular line in a circular fashion, since the busses are essentially driving in circles between the two "end points". An example of this is given in figure 2.

As discussed above, the simulation model consists of a circular topology. This topology is built up like a network; there are several nodes (the bus stops), connected by edges (route sections). The number of nodes and edges and their characteristics are needed as an input to the simulation model. The same holds for the number of busses, their capacity and their start position. Lastly, an end trigger for the simulation has to be defined. This trigger determines when the simulation run is finished. The trigger can be set at a certain point in time, but can also depend on the state of the simulation e.g. when all headways are equal or when all passengers are processed. With this input the simulation model is initialised and the simulation starts.

When the simulation starts, all busses start driving in the same direction (note that two-directional lines are modelled as a circular line). The simulation is used to model only a single bus line, this means all busses will attend to all stops. The travel time of each edge (a route section) is a property of each edge and can thus differ between edges. Also, edges can contain traffic lights which can create additional travel time for the bus. When a bus is finished travelling across an edge, it will arrive at a node (a bus stop). The bus is able to 'enter' the stop if there is still an unoccupied platform at the stop. At the bus stop all passengers with that stop as destination will alight the bus. Additionally, all passengers at the stop will try to board the bus, if this is possible or not depends on the capacity of the bus. The number of passengers waiting at the stop depend on the passenger arrival function of that stop and the time passed since the last bus stopped there. The destinations that the waiting passengers have also depends on the destination distribution of the stop. Two simplifications are made in the model that concern the stopping process. The first is that a bus will always stop at a stop, even if no passenger wants to board or alight the bus. The second simplification is that the passengers that arrive at a stop when the boarding and alighting process has already started cannot board the bus anymore. The simulation will run in time steps of a predetermined size until the simulation is finished. At every start of a step there is a possibility for information exchange between the simulation and the RL agents (an observation, action or reward). A conceptual framework of the operations in the simulation model is given in figure 2.

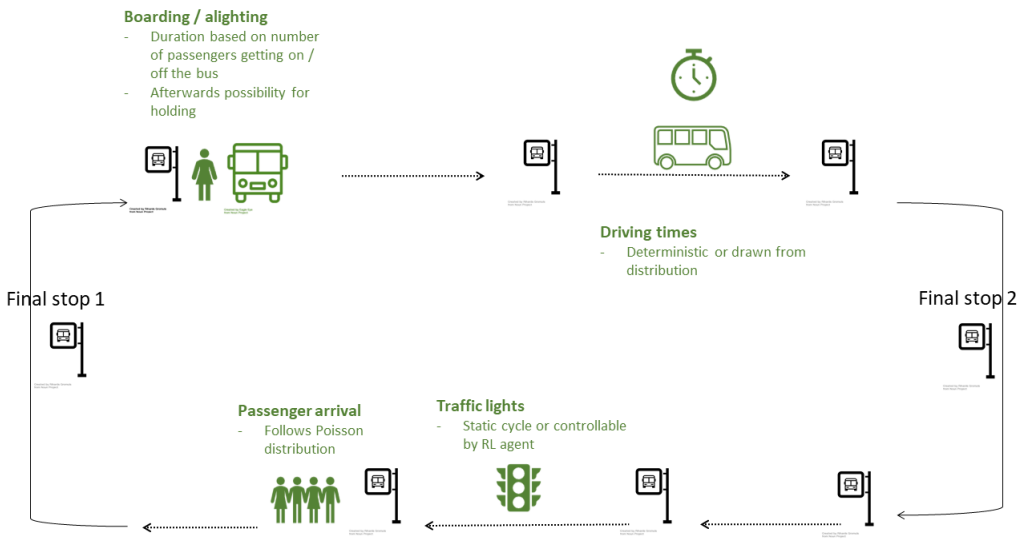

Figure 2: Conceptual framework of the simulation model.

3.1.3 Simulation Configurability

As mentioned in section 3.1.2, several aspects of the simulation model are configurable. In the coming section, all the configurable settings are discussed in detail. An overview of all these settings can be found in table 3 and 4.

First off, the size of the bus line can be adjusted, meaning the number of nodes and edges that compose the

Table 3: An overview of all the settings in the simulation environment that can be configured to create a specific scenario.

Setting	Options
Length bus line	Set the number of nodes and edges can be set.
Number of busses	Set the number of busses that drive on the bus line.
Travel time edges	Set the driving time or the driving time distribution characteristics for every edge.
Passenger arrival rate stops	Set the average passenger arrival rate for every stop.
Destination distribution passengers stops	Set the destination distribution for every stop.
Number of platforms stops	Set the number of platforms that every stop has.
Dwell time function	Select whether the dwell time is based on the sum of boarding and alighting time or the maximum of the two.
Capacity busses	Set the passenger capacity for every bus.
Including traffic lights	Select if there are traffic lights in the simulation.
Edges with traffic lights	For every edge set whether or not it has a traffic light.
Traffic lights are controllable	Select whether the traffic lights are random or controlled by an RL agent

line. As long as it is a circular line, it will work. Secondly, the travel time along each edge can be configured. This can be a deterministic travel time, but it can also be a stochastic one. The stochastic travel times are drawn from a distribution. This travel time distribution is a predefined property of an edge. As a result, every edge can have the same travel time distribution, but they can also differ between edges. It is also possible to cap the used distribution at certain values to make sure extreme events are not modelled. Furthermore, each edge can have a traffic light, with a configurable red and green phase cycle. Busses have to wait until the traffic light is green before they can continue their journey. In the scenarios where the traffic lights are controllable by RL agents, the green phase can be extended or shortened by the RL agents to improve the performance of the bus system. Thirdly, the nodes (stops) have several properties that need configuration as well. Their first property is the arrival function. The passenger arrivals are assumed to be independent of each other and therefore their arrival is assumed to follow a Poisson distribution for every bus stop. The average arrival rate is not assumed to be the same for every bus stop, this parameter has to be set for every stop. The second property of the nodes is the destination distribution. This distribution gives the probability that a certain stop on the bus line is the destination of a passenger starting at the current stop (the probability for the current stop is always 0). The last property of the nodes is the number of platforms they have. In this study all nodes have just one platform, but this feature could be used to model larger stations. The number of platforms determines how many busses can board and alight at the same time at the node in question. If all platforms are in use, a bus has to wait until one of them becomes available.

There is one more property that has to be defined for the network as a whole: the dwell time function. The dwell time is a function of the number of boarding and alighting passengers and can, again, be deterministic or stochastic. Another configuration option of the dwell time function is whether it depends on the total number of boarders and alighters combined or that it is the maximum of the expected boarding and expected alighting time. For this last functionality, both the boarding rate and alighting rate need to be specified.

Apart from the possible configurations on the bus line, there are also several properties on the busses and the passengers that need to be set. The busses are initialised at the start of the simulation and have a route that they are following. Furthermore, they have a maximum passenger capacity and a start node (the node that they are initialised on). The passengers are initialised at the nodes continuously as the simulation progresses and only need a start node and a destination.

The final setting that has to be defined for the simulation model to run properly is the **end trigger**, as is brought up in section 3.1.2. The end trigger defines the end of a simulation run. This is needed for an RL learning process. Some problems already have a natural end trigger e.g. an RL agent that has to find the exit of a maze has the natural end trigger when it has found the exit. However, for the bus management problem

this is less obvious, since it is not clear when the problem is 'solved'. One option to frame a headway management problem is to start with a state of the environment that is sub optimal, i.e. the headways are not well distributed. The MARL agents can then interact with the system to reach a better state of the environment. The problem is then 'solved' when the headways are well distributed. This way of addressing the problem suits RL well; there is a clear task that the agent should try to solve in as little steps as possible. However, this way of addressing the problem is not the most logical when looking at the headway management problem; once the headways are well distributed they should be kept that way, because they can be disturbed at any moment. From the perspective of the bus management problem, the MARL agents should be able to manage the busses continuously, irrespective of the end of the problem. This is because the end of the problem (the end of the day in practice, but the end of the simulation when learning) has nothing to do with the performance of the MARL agents; the problem is not 'solved', it has simply ceased to exist. Reasoning this way makes a predefined end time a logical choice for the end trigger. When a specific timetable is simulated, the stop moment will be the end of the timetable. Yet, such a setup brings one difficulty to the table: with each simulation run the MARL agents will think that they solved the problem with their last action even though this is not the case. This means the value estimation of the agents of the last action of the simulation run will be very bad since no more rewards / punishments follow after this action. This gives the agents the idea that the expected future reward of this action is zero, but this was only the case by coincidence. With enough training this effect should eventually vanish, but initially it is a disturbance in the learning process. A way to counter this issue is by making the agents 'time aware' as discussed by Pardo et al. [42]. Time aware agents know how much time they have left in the current simulation run, which means they will learn that when the time runs out, the simulation will stop, irrespective of their actions. Making an agent time aware as suggested by Pardo et al., is done by adding a normalised time value to the observation of the agents between -1 and 1, where -1 stands for the start of the simulation and 1 for the end. Evidently, many more options are possible when it comes to creating an end trigger for the simulation. However, the discussed two methods are tested in the preliminary experiments of this study and the predefined end trigger is used for the final experiments as it fits the problem of headway management the best. Furthermore, the time awareness is not added in the final experiments since there was no positive effect visible. Further, the 'warming up' phase, discussed in section 4.1, alleviates the disturbance that the end of the simulation causes.

The length of a simulation run is set to roughly twice the length of an average total travel time. This is done because the RL agents are rewarded at the end of the simulation based on the average travel time of the passengers, as discussed later in this section. Twice the average passenger travel time is a balance between too short simulation lengths and too long simulation lengths. Too short simulation lengths are unfavourable, since only a few passengers have arrived, which yields unreliable results. On the other hand, too long simulation lengths are not useful since it becomes hard to distinguish the good from the bad behaviour within one run for the agents, due to one reward at the last step. This may lead to slower learning or not learning at all.

The discussed factors determine the functioning of the simulation model, but they have no influence on what the RL agents in the simulation can see, do and how they are rewarded. These aspects are important for the performance of the MARL algorithm in the simulation environment and they have to be configured in the simulation environment. In the coming paragraphs the options related to the observation space, the action space and the reward function for the RL agents are discussed. It is important to note that on each of these three aspects the number of possible configurations is infinite. Clearly, not all options are considered. Instead, the configurations often used in literature are discussed along with a few other configurations that appear to have potential.

First off, the **observation space**; as mentioned in 1.2, the observation of an RL agent is the information it receives on the current state of its environment. Based on this information, the agent will take its decision as to what action to take. A good observation provides the RL agent with all the information that it needs to select the right action, but contains no redundant information. Redundant information only introduces noise in the learning process, may slow down learning and possibly decrease the final performance. Constructing a good observation is more art than science, yet, some basic intuitions can help design a good observation space. First of all, the essential information should be presented in such a way that it is compact and easy to interpret for the RL agent. For example, in a headway problem it may be better to provide a bus agent with its forward and backward headway instead of providing its own location and the ones of the busses behind and in front of him. This saves the RL agent the time of deducing the headways, if that is the most relevant information for him and it means it has only two variables to observe instead of three. Secondly, if it is possible, it is helpful to the RL agent if the observation space aligns well with its action space and its reward function. In the case of the headway problem, this could mean that the bus agents can observe their forward and backward headway, can decide to hold, to increase and decrease their forward and backward headway respectively and are rewarded

Table 4: An overview of all the settings in the simulation environment that can be configured, relating the RL capabilities.

Setting	Options
Observation traffic lights	Set the observation space for the RL agents that control the traffic lights. I.e. include or exclude bus positions, headways, number of passengers at stops, number of passengers in the busses, etc.
Action space traffic lights	Set the action space for RL agents that control the traffic lights. I.e. decide what they can alter in the green phase time and whether they have discrete or continuous options.
End trigger	Select the moment when the simulation environment will return that the simulation is done. I.e. based on simulation time, rewards or state of the system.
Observation busses	Set the observation space for the RL agents that control the busses. I.e. include or exclude bus positions, headways, number of passengers at stops, number of passengers in the busses, etc.
Action space busses	Set the action space for RL agents that control the busses. I.e. decide what discrete holding options they have or what the range is of the continuous holding.
Reward function busses	Set the reward function for the busses. I.e. when will they receive their rewards and what are the rewards based on (headways, passenger travel times, etc.).

based on their forward and backward headway. In this situation it is clear that the RL agents has influence on the state that he observes and the relation between the observed state and its received reward is also clear. If instead, the bus agents would be rewarded based on the waiting time of the passengers at the stop, it would be less clear for the bus agents what states, and thus what behaviour, is desirable and what is not, since the reward is also dependent on the behaviour of the preceding bus. However, aligning the reward function with the action and observation space is not always the best choice. The downside with rewarding an agent directly is reward shaping, this is discussed more in depth in the reward function paragraph.

Apart from the information that the bus agents need to select the right action, general state information may also be needed. For example, as mentioned above, a time observation can be added to help the agent understand that when the time runs out, the simulation ends. In this study different observation spaces are tested; these are combinations of the forward headway, the backward headway, the difference between the forward and the backward headway, time awareness, position on the route and the number of passengers at the stops. After some preliminary experiments the difference between the forward and the backward headway is selected to be the only information available to the bus agents. This choice is motivated by the research from Cats et al. [10], which indicates that the best performing holding scheme is based on the mean of the preceding and succeeding busses. Moreover, this observation space gave the best results in terms of learning stability and performance. Additionally, this information is easy to obtain in real world bus systems. The observations space is continuous to allow for enough detail. A discrete observation space is also tested in the preliminary experiments, but this results in either too course information to be useful or too many options to yield the benefits of a discrete observation. For the experiments where the route is heterogeneous, the position of the bus is added to the observation as well.

The calculation of the forward and backward headway is done at each stop. The elapsed time since the previous arrival is set as the forward headway for the arriving bus and as backward headway for the preceding bus. This way of calculating the forward and backward headway is chosen since it is often used in the PT industry and therefore adds realism to the experiments. The position of the bus is represented by a discrete observation that has a value between 0 and 45. Here 0 means the bus is waiting to enter stop 0, 1 means it is boarding and alighting at stop 0, 2 means the bus left stop 0, 3 means the bus is waiting to enter stop 1 etc.

For the **action space** there are often fewer sensible options than for the observation space. Still, some choices have to be made. For the case of the holding action of the bus agents the first element that has to be set is the moment of choice, i.e. when can the bus agents decide to hold or not? This is not part of the action

space, but influences the performance of the algorithm. The second element that has to be chosen is whether the action space is discrete or continuous. Most often, a discrete action space leaves the RL agent with fewer options, which may lead to an easier learning process. However, the relation between actions can get lost with discrete actions. For example, if a bus is given the option to hold either 0, 30, 60 or 120 seconds, this will be represented as action 0, 1, 2 or 3. For the RL agent it is not clear that all actions are holding actions and that they differ only in duration. Neither does the RL agent know how much bigger or smaller the actions are compared to the others. It has to be learned by the RL agent that these four actions are essentially a different value for the same parameter. In a continuous action space the RL agent is free to chose any value within a given range and therefore immediately knows that all possible values affect the same parameter. Additionally, the relation in magnitude between the possible values is also more obvious. In this study the busses can hold after the boarding and alighting process is finished. A discrete action space is used since this binds the MARL algorithm to realistic actions: 0, 30, 60, 90, 120, 150 or 180 seconds of holding time. Whereas with a continuous action space it would be possible to hold for a few seconds only, This may be difficult to implement in reality. Furthermore, a continuous action space led to less stable learning curves in the preliminary experiments.

The last factor is the **reward function**, which determines how the agents are steered towards desirable behaviour. Again, there are numerous elements to base the reward function on. In this study it is based on either the headways of the busses or the travel times of the passengers. For the final experiments the average travel time of the passengers is selected as the basis for the reward function of the busses and the traffic lights.

The passenger travel times are selected because, in the end, this is what a bus system should do: transport passengers from A to B as fast as possible (within some boundaries). The idea behind this is that the shorter the average travel time is, the better the behaviour of the agents was. This gives the agents a lot of freedom to find the best behaviour. Naturally, the average travel time may not be the only important performance indicator, but since the bus routes are predefined and circular, it is assumed to be a good approximator of the performance. A benefit of this reward function is that there is no need to find a good balance between the speed of the busses and the headway distribution. The RL agents will have to find this balance by themselves.

After selecting the basis of the reward function, the moment of the reward should be selected. It is possible to reward the agents for every step they take, so they have continuous rewards during the training episodes. Another option is to reward the agents only at the end of an episode based on their overall performance. For the first option it has to be clear what behaviour or what intermediate states of the environment are desirable. This tends to lead to reward shaping: shaping the behaviour of an agent with continuous rewards. This is not necessarily bad, but it takes away a key advantage of RL; only defining the desired outcome, not the desired behaviour. With reward shaping the RL agents are thus no longer incentivised to find the optimal behaviour themselves, instead they are pushed to follow the behaviour predefined by the reward function. This predefined behaviour may be suboptimal. This disadvantage is not present when the RL are rewarded at the end of the episode based on their overall performance. However, with only a reward on the end, it may be hard for the RL agents to trace back the good and the bad behaviour of an episode. Depending on the rest of the setup of the simulation one method may be favorable over the other. For example, when the passenger travel times are used as the basis of the reward function it makes more sense to reward the bus agents at the end of the episode. This is because it takes a while until the first passengers have arrived at their destination and the travel times depend on the OD of the passengers as well, which means the learning is more stable when rewarding with the average travel time of the episode. On the other hand, when the headways are the basis of the reward function, it makes more sense to reward continuously. At every moment it is clear how well the headways are distributed, so a continuous reward can be calculated.

It is also possible to construct a reward function that is a combination of continuous rewards and a final reward. An example of this is the case where the bus agents are rewarded based on passenger travel times, but every time they bunch, they get a negative reward. This may speed up learning without restricting the freedom of the RL agents too much. However, the downside is that policies that lead to bunching are actively discouraged without looking at the effect on the system as a whole. Nevertheless, this approach is used in the reward functions for the agents in the final experiments.

Finally, a completely different angle could be used when the end trigger of the simulation is set on well distributed headways. In this case the agents have to 'solve' the headway problem as fast as possible i.e. get to a state where the headways are well distributed instead of the uneven distribution of the start state. This means giving a negative reward for every step the agents need, with a possible positive reward when they succeed can be an effective reward function.

Considering the options of the end trigger, observation space, action space and reward function, it becomes clear that these elements should not be configured independently. It is important to make sure all four elements fit together, yet, it can be a challenging task to do so.

3.2 RL Implementation

3.2.1 RLlib

In this study the Python library RLlib [1] is used for the implementation of the MARL algorithms. This means all the learning is done within this library, only the environment in which the agents should learn has to be provided. There are several environments available online, and these are often used to benchmark new algorithms. However, in this study a novel, custom build environment is used, as introduced in section 3.1.2.

3.2.2 Interaction RLlib and Simulation Environment

RLlib can handle any type of environment as long as it meets the following requirements: the environment class has a reset and step function, than can be called and returns the right values. Both functions will be called by RLlib with no arguments and the agent actions respectively. The reset function resets the simulation environment to its begin state and returns an observation for all agents that need to take an action in the first step. The step function 'takes a step in the simulation environment', in this study this comes down to running the simulation environment for a certain amount of time. After this time the necessary observation and rewards are returned. The argument of the step function contains actions for (a subset of) the agents in the system. Consequently, it is not necessary that all agents in the system take an action in each step. For each action that is taken, a reward should be returned to the agent afterwards (this can be in the next step function call). Finally, the step function returns a boolean for each agent with the information whether this agents is done or not. If an agent is done, it means it will no longer need to take an action during that episode. When all agents are done, it means the end of the episode and the simulation will no longer continue. All agents have an agent ID to identify which observation, action and reward belong to which agent.

3.2.3 Configurations

The way of communication between RLlib and the simulation environment is predefined, but there are several parameters to be set on the side of RLlib. With the observation space, the action space, the reward function and the end of each episode as input, the RL agents will try to learn the optimal policy. The parameters on the RLlib side influence the process of going towards that optimal policy.

Apart from a set of hyperparameters there are two to make in RLlib: which algorithm is used to improve the policy and which agents will use which policy. For the first choice, there are numerous algorithms implemented in RLlib and a subset of them is also available in multi agent mode. All these algorithms serve the same general goal of improving the policies of the agents, but there are differences between them as explained in section 2.3.2. In this study for every scenario the following algorithms are considered: PPO, actor critic and QMIX. PPO and actor critic are selected because of their popularity which stems from the fact that they have proven to be effective in many cases. QMIX is selected since it may provide an advantage in the more complicated scenarios since it is specifically developed for multi-agent settings.

For the second point of choice, the most straightforward option is that there is only one policy for each RL agent type, i.e. an RL bus agent or an RL traffic light agent. This means all the agents of one type use parameter sharing, this means they all learn on and draw from the same policy, as discussed in 2.3.2. The other option is to give each individual agent its own policy.

Aside from the choices on the algorithm and the policies, there are a number of hyperparamters that need to be set. An overview of the hyperparameters that have to be set for every algorithm is given in table 5. These hyperparameters mainly influence the stochastic gradient descent (SGD) process of the algorithms. SGD is used by the algorithms to find out in which direction the parameters of the neural network (NN) should be adjusted to get to a policy with higher rewards.

Table 5: An overview of the hyperparameters in RLlib and their functionality.

Hyperparameter	**Functionality**
Learning rate	This defines the learning rate of the algorithm, affecting the 'step size' the algorithm takes in SGD toward new policies.
Model	This determines the architecture of the NN that has to learn the policy.
Train batch size	The number of training steps that defines a training batch. This defines the size of each SGD epoch.
SGD mini batch size	The number of training steps that defines the size of the mini batches in SGD.
Number of SGD iterations	The number of SGD iterations in each outer loop, i.e. the number of epochs to execute per train batch.

4 Experimental setup

To answer the main research question, as proposed in 1.4.2, several scenarios are designed in the simulation model. The approach taken with the scenario design is elaborated on in the coming sections. Furthermore, the procedure to analyse the experiments in these scenarios is discussed. Lastly, a case study is discussed with an analysis of the observed driving time data.

4.1 Experiment Options

As brought forward in section 3.1.3, there are many things in the simulation environment that can be configured to create the environment that is desired. In this research such a configuration of simulation environment settings is called a scenario. The full list of configurable settings that describe a scenario can be seen in table 3. The configuration of such a scenario determines the complexity of the scenario for the MARL algorithm. The amount of change in the difficulty for the MARL algorithm when a parameter is changed is unknown. Yet, it is clear that some parameters effect the complexity of the RL problem, whereas others have a limited effect on the complexity. The interaction between the different parameters and the scenario is shown in figure 3.

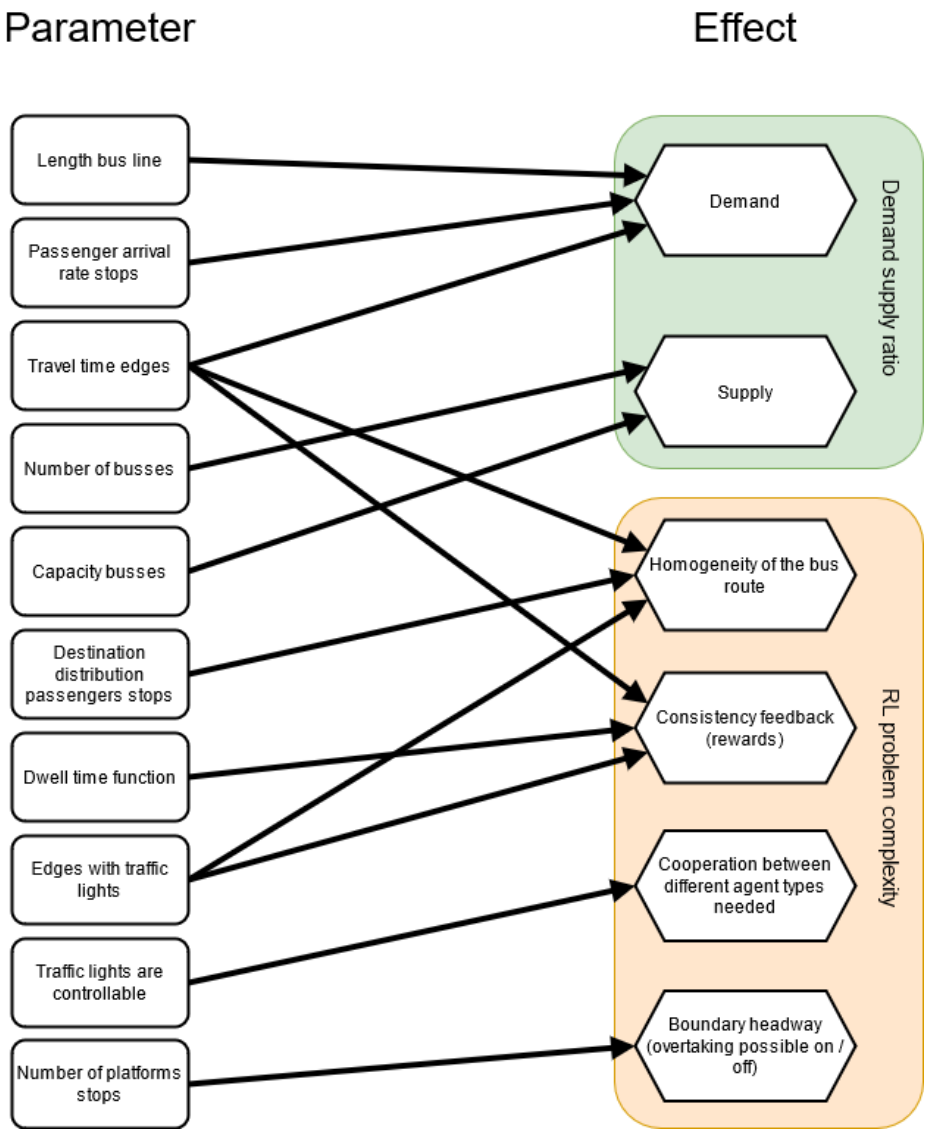

Figure 3: An overview of the relation between the scenario parameters in the simulation environment and the effect on the RL problem.

It is not possible to quantify the change in difficulty for the MARL algorithm when a parameter is changed.

Next to these scenario settings there is also a number of settings in the simulation environment that relate to the RL capabilities in the environment. These are also explained in section 3.1.3 and an overview of all of them can be found in table 4.

Lastly, there are a number of settings in the MARL algorithm itself. These hyperparameters do not affect the simulation environment in any way, but they determine how the gained experiences in the simulation environment lead to policy improvements. These hyperparameters are examined in section 3.2 and an overview of the most important ones can be found in table 5.

All these settings create an extensive set of possible configurations that can be experimented on. This is broken down to a manageable set with the most interesting combinations. These scenario designs are based on previous research (the current state-of-the-art) and the day-to-day reality of bus systems (the end goal). Next, the RL settings (in the simulation and in RLlib) are selected that are believed to be most effective. In this selection process several of the possible configurations are tested, but far from all, due to the finite amount of time and computation power available. This means insight is needed to select the most interesting combinations. As an illustration: when a scenario is designed where a large observation space is deemed necessary, but the NN in the algorithm that is expected to learn from these large observations is small, it will never be able to consider all the given information in its decisions. A sufficiently large NN is needed in a scenario like this, eliminating all possible configurations with small NN's since they are unfit. Another example could be a scenario where all the agents are meant to learn their own unique behaviour to achieve the best solution. In such a scenario it is fruitless to use parameter sharing, since this, by definition, prohibits unique behaviour per agent. It is a complicated task to put such insight onto paper, but figure 4 attempts to do this, giving the main correlations between parameters and their effect within the framework. With this as a guide, the most suitable configurations are selected per scenario.

For the final experiments the PPO algorithm is selected with a shared policy for the bus agents and a shared policy for the traffic lights. The choice for separate bus and traffic light policies instead of one central policy is made since this is a more scalable setup. If, for example in a future research or application, the action space, observation space, the bus line or the bus network are extended, one central policy may no longer be a feasible option. Additionally, the found literature all used decentralised options. Moreover, this setup performed best in the preliminary experiments. The hyperparameters shown in table 5 are tested for several values that are commonly used.

The learning rate adheres to a learning rate schedule that starts at 0 for a warm-up phase. After 200.000 steps the learning rate increases linearly to 5×10^{-4} at 600.000 steps after which it declines linearly to 5×10^{-5} at 1.000.000 steps. The learning rate is then held constant. The warm-up phase is incorporated to make sure the learning starts from a batch that represents the training data well and to prevent the algorithm from giving too much weight to the first steps. For the train batch size, a value of 3000 is chosen after experimentation as a balance between learning stability and learning speed. For the SGD mini batch size and the number of SGD iterations, the common values of 32 and 10 are used respectively. The NN's architecture is set to a single hidden layer containing 256 nodes.

The configuration of the simulation parameters (see table 4) are chosen based on experimentation. Firstly, a discreet action space is selected of 0, 30, 60, 90, 120, 150 or 180 seconds holding time for the busses. This is done because it results in faster learning than a continuous action space and because it assumed to have more practical value. This assumption is based on the fact that agents in a continuous action space often use very small holding times, which may be hard to execute for a bus driver. Secondly, the observation space of the busses and the traffic lights the difference between the forward and backward headway is used. This is the most simple observation space option and it gives the best results in the experiments executed. Thirdly, the reward function selected is one based on the travel times of the passengers. This reward function rewards the agents only at the end of the simulation. This reward function not only gives the best results, it also gives the most freedom to the agents to learn the optimal policy. The only exception to this is when the forward and backward headway of a bus differ a lot, then a bus receives a small negative reward. This is thus the only reward shaping in this reward function. Lastly, the end trigger option selected is a predefined duration of the simulation. This option is chosen since the it reflects the continuous nature of the bus management problem well. As discussed in 3.1.3, a long duration with only a final reward means it is hard for the RL agents to identify which behaviour was good and which was bad. Whereas a short duration does not reflect the continuous nature of the problem well. The duration chosen is selected as a balance between these two cases.

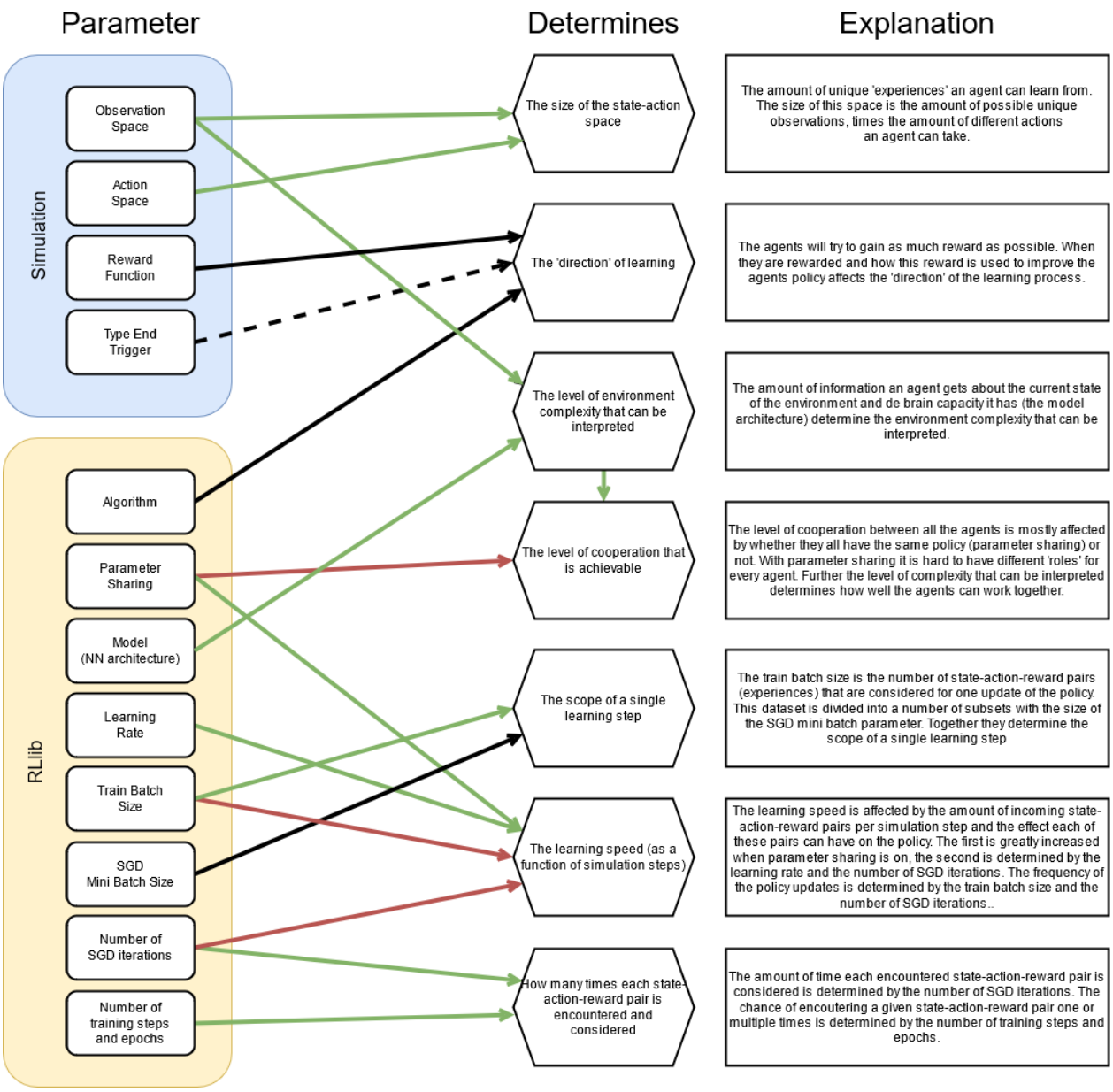

Figure 4: An overview of the correlations between parameters in the simulation environment, in RLlib and their effect on the learning process. A green arrow represents a positive correlation, whereas a red arrow represents a negative correlation. The dashed arrow indicates an indirect relation.

In section 4.3 the selected scenario designs are discussed in detail.

4.2 Performance analysis

As described in section 1.4, this study focuses on using MARL for the problem of bus management, not on improving MARL algorithms. This means most of the research effort is invested in testing configurations of the simulation environment. To get the optimal performance in each scenario an extensive parameter study should be executed. However, due to the exploratory nature of this study, optimal performance is not a crucial aspect. Therefore, the scope in this aspect is limited in this study. Several configurations are tested on the base scenario. Following this, the best performing configuration is used to assess the other scenarios.

For each scenario the performance of the MARL algorithm will be compared with two benchmarks; a no control benchmark and a holding control benchmark. The holding control is based on the difference between the forward and backward headway. The headways are calculated at each stop and the actions are executed at stop level as well. When the forward headway is more than 15 seconds shorter than the backward headway, the bus will hold for 30 seconds. When it is less than 45, 75, 105, 135 or 165 seconds shorter than the backward headway, a holding action of 60, 90, 120, 150 or 180 seconds is applied respectively. This holding strategy is selected because it uses the same information as the MARL algorithm. Furthermore, it performs well on headway equalisation.

There are five different scenarios that are used in the experiments. Each of the scenarios is discussed in detail in section 4.3. For every scenario the process of training and evaluating is the same. This process starts with setting the simulation environment to the settings that correspond with the concerning scenario. Next, the MARL configuration is set and the learning is started. For every scenario the learning is done ten times, each with a different seed. Using ten seeds gives a good impression of the spread in performance, at the same time the computing time needed is still manageable. The ten seeds used are the same for every scenario to make comparison possible. Each of these ten runs will produce its own unique learning trajectory, which means the resulting policies will also differ. When the training is done, the performance of the policy throughout the learning process is assessed based on the minimum, the maximum and the mean reward of the policy in each training iteration. Each training iteration consists of multiple simulation runs with each their own reward. The mean, minimum and maximum reward give an indication of the performance and the spread in performance of the used policy. Based on this information, the moment is selected when the policy was at its best. This is often the last iteration, but not always. The minimal, mean and maximal reward of an example learning process are shown in figure 5.

The result of this selection process is a set of ten policies, each from a different learning trajectory. Not every learning trajectory converges, which means not every learning trajectory yields a useful policy. From the learning trajectories that converged, the five best are selected based on the mean reward. This selection thus excludes the learning trajectories that did not converge, but still gives an idea of the spread in performance between learning trajectories. This spread is interesting, since it gives a more reliable indication of the difficulty of a scenario than a single learning trajectory. These five policies are then executed greedily for several runs, meaning they are no longer trying to learn at this stage. Running the policy multiple times is done to get an accurate estimate of the performance of the policies. The results are then analyses based on several metrics, show in table 6. With these metrics the performance of the MARL algorithm can be compared with the benchmarks in each scenario.

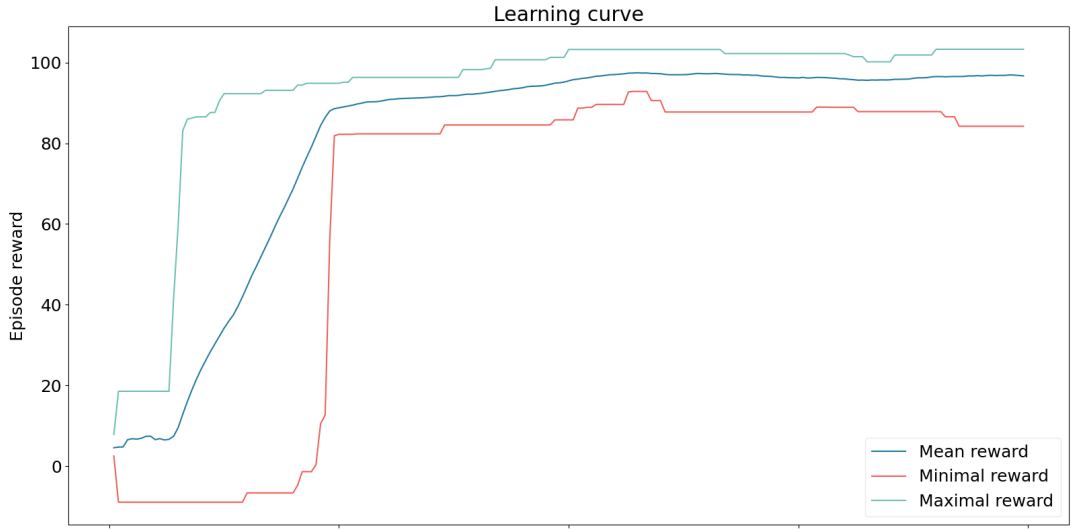

Figure 5: An example of a learning curve. The minimal (red), mean (blue) and maximal (green) reward per episode are plotted for every step in the learning process of the MARL algorithm.

Table 6: An overview of the metrics used to evaluate performance per scenario.

Metric	**Explanation**
Average holding time	The average holding time of the busses, including the zero second holding actions.
Average holding time per stop	The average holding time of the busses per stop, including the zero second holding actions.
Average waiting time passengers	The average waiting time of the passengers until they can board a bus.
Average in vehicle time passengers	The average in vehicle time of the passengers, from boarding the bus until the arrival at their destination.
Experienced crowding	The average crowding in the busses weighted by the number of passengers in each bus. See equation 1
Excess waiting time	The difference between the actual waiting time an the scheduled waiting time. See equation 4
Coefficient of variation headways per stop	The standard deviation of the headways per stop divided by the mean.

The experienced crowding metric is calculated using the following formula:

$$Experienced\ Crowding = \frac{1}{P} \times \sum_{B} C_b^2 \tag{1}$$

Where P is the total number of passengers, B the total number of busses and C_b the crowding of bus b.

The excess waiting time metric is calculated as done by Leong et al. [33], which is by subtracting the scheduled waiting time from the actual waiting time, see equations below.

$$Scheduled\ Waiting\ Time\ (SWT) = \frac{\sum_N scheduled\ headway_n^2}{2 \times \sum_N scheduled\ headway_n} \tag{2}$$

$$Actual\ Waiting\ Time\ (AWT) = \frac{\sum_N actual\ headway_n^2}{2 \times \sum_N actual\ headway_n} \tag{3}$$

$$Excess\ Waiting\ Time\ (EWT) = AWT - SWT \tag{4}$$

Where N is the total number of busses in the system.

The calculation of the metrics for comparison with the benchmarks is as follows:

1. Per simulation run the average value for each metric is calculated.
2. Each of the five different policies per scenario is evaluated during approximately 35 simulation runs.
3. The average values over these 35 are calculated for each policy and each metric.
4. For each policy and each metric the score is compared with the holding benchmark.

4.3 Scenario design

In the coming paragraphs the details of, and motivation behind the design all five scenarios that are used for the experiments are explained. A general overview of the scenarios with their properties is given in table 7. An exact overview of the scenarios with all their settings is given in appendix A. Note that since all scenarios are based on the same simulation model, the only difference between the scenarios will be in the settings that are discussed in section 3.1.3.

The most idealised scenario, called 'Idealised' (ID), used in this study is inspired by the study done by Wang and Sun [56]. This means there is nothing stochastic in the environment except for the arrival pattern of passengers at the stops, which follows a Poisson distribution. The destinations of the passengers are evenly distributed over the ten stops following the origin stop. The travel times along the edges, the number of stops, the number of busses, the boarding rate and the alighting rate are all the same as in the study by Wang and Sun. This means the travel time between stops is 4 minutes, there are 12 stops, 6 busses, the boarding rate is 3 s/pax and the alighting rate is 1.8 s/pax.

In contrast to Wang and Sun the arrival rate is the same at each stop along the way and is set at 1 pax/min. This is done to make the route as 'simple' as possible. Evidently, there are some other differences since the scenario is created in a different simulation environment than used by Wang and Sun. The most notable difference here is the resolution (one step) of the simulator: 15 seconds instead of 1. This is done because this is assumed to reflect reality better. If the MARL were to be implemented, the communication between the busses would likely be less frequent than every second.

The experiment in the base scenario serves two main purposes: to see if the MARL algorithm is able to learn in the given setup and to get a reference point for the performance.

The second scenario, 'Stochastic' (ST), brings more stochasticity into the simulation environment. The scenario is identical to the idealised scenario except for three stochastic elements. In this scenario the travel times along the edges are stochastic: they are drawn from a predefined log normal distribution with a given mean, standard deviation, minimum value and maximum value. The reason for selecting a log normal distribution is explained in section 4.4. The log normal distribution used is the same for every route section. The second stochastic element is the dwell time. First the dwell time is calculated as is done in the deterministic setting, then the stochastic dwell time is drawn from a normal distribution with the deterministic dwell time as the mean and a standard deviation of 1. The maximum of the stochastic dwell time is set to twice the duration of what the deterministic dwell time was. These values are assumed to be realistic, yet they are not based on research. The last stochastic element in this scenario are the traffic lights. The traffic light have a fixed cycle which takes 75 seconds. This cycle length is chosen since an optimal cycle length should be between 60 and 90 seconds according to the National Association of City Transportation Officials [3]. The first 40 seconds of the cycle the busses will have a red light, the last 35 seconds they have a green light. This represents an intersection where the busses drive on the main artery. The experiment in the 'Stochastic' scenario gives an idea of how well the MARL algorithm can handle stochasticity.

The third scenario, called 'Stochastic - Traffic Light control' (TL), introduces agents for the traffic lights. This means the MARL algorithm has some control over the traffic lights. The cycle with the red and green phase is still in place, but when a bus arrives, the agent controlling the traffic light can decide on increasing the red

or green phase with 10 seconds. In this way the traffic lights can have some influence on the headways between the busses, yet their influence is bounded by the predefined cycle of the red and green phase. Aside from the traffic light agents, all properties of this scenario are identical to scenario 2 (ST). The experiment in this scenario gives insight into the performance gain when the traffic lights are also part of the MARL control algorithm.

While these three scenarios vary on several aspects, they have one thing in common: they are homogeneous in nature. Meaning that along the route the properties of all the route sections and stops are the same. This is not the case in the fourth scenario: 'Deterministic - Heterogeneous' (DH). In this scenario the edges have different lengths, i.e. the travel time along the different route sections is not equal. The route lengths are shown in figure 6. These lengths are chosen to replicate a bus route with a rural section (the long route sections) and a section through a more densily populated area (the short route sections). Additionally, the stops have uneven demand, with the same values as in the work of Wang et al. [56], which is shown in figure 7. Apart from the heterogeneity, all the properties of this scenario are equal to that of the idealised scenario. In this scenario the performance of the agents will depend on their ability to deal with the changing conditions along the route.

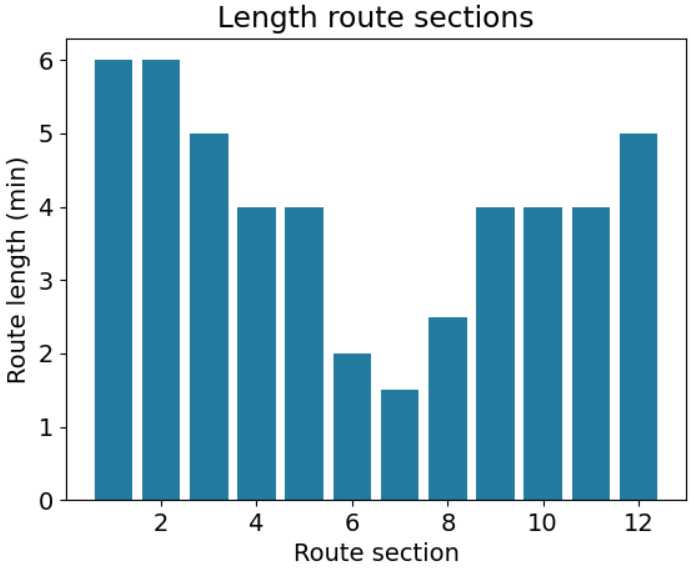

Figure 6: The route lengths in minutes for all the route sections.

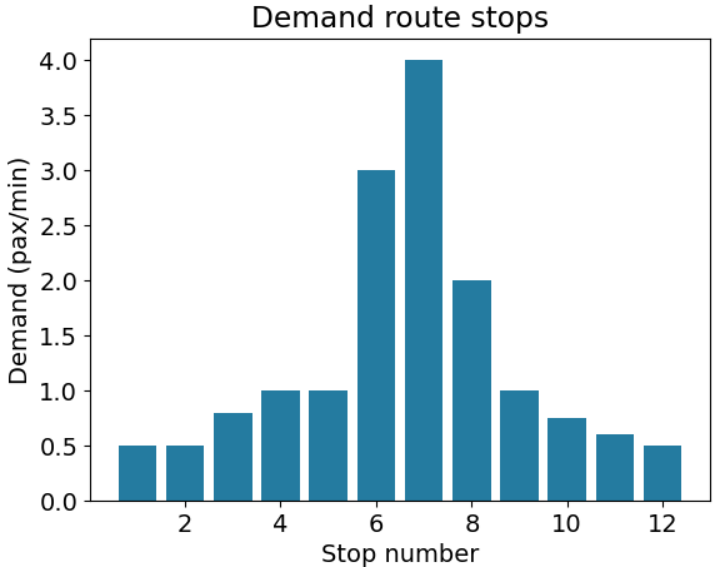

Figure 7: The average demand per stop in passengers per minute for every stop.

In the last scenario everything comes together. The bus route in this scenario will be both heterogeneous and will have stochastic processes. The demand per stop and the length of the route sections is the same as in the 'Deterministic - Heterogeneous' scenario. Furthermore, the traffic lights in this scenario are controlled by the MARL algorithm. This should represent the most realistic scenario possible with the simulation model used, therefore it is titled 'Realistic' (RE). This scenario gives an idea of the performance of a MARL control algorithm in daily operations.

Table 7: An overview of the scenarios used for the experiments.

Name	**Travel time**	**Route sections**	**Demand stops**	**Dwell time**	**Traffic lights**
Idealised (ID)	Deterministic	Identical	Identical	Deterministic	No
Stochastic (ST)	Stochastic	Identical	Identical	Stochastic	Yes - Uncontrolled
Stochastic -Traffic Light Control (TL)	Stochastic	Identical	Identical	Stochastic	Yes - Controlled
Deterministic - Heterogeneous (DH)	Deterministic	Different	Different	Deterministic	No
Realistic (RE)	Stochastic	Different	Different	Stochastic	Yes - Controlled

4.4 Application

The model used in this study is created to reflect the real world operations of busses on a high frequent line. However, due to time and computation power restrictions, several assumptions are made in the model. Furthermore, the scenarios created are synthetic to be able to test the performance of the MARL algorithm under specific conditions. This means it is easy to create a simulation scenario that does not reflect the reality. To make sure the scenarios are as realistic as possible it is paramount to simulate the vital processes of the operations in a realistic manner. If this is successful, then the simulation model can appropriately simulate any bus line, given the right settings.

This means that the stochasticity in the processes of the model should be represented correctly. As seen in table 7, there are several stochastic processes in the simulation model. Firstly, the dwell time can be set as a stochastic function of the number of passengers. As described in section 4.3, this stochasticity is created by

drawing the dwell time from a normal distribution around the deterministic dwell time. This is implemented this way due to the natural nature of this process and it is similar to the implementation in VISSIM (a popular simulation tool) [35]. The second stochastic process is the passenger arrival at the stops. This process is commonly modelled by a Poisson distribution [45] and this is also used in the simulation model of this study. The second stochastic component of the simulation model is the traffic light implementation. Modelling traffic lights realistic is a difficult feat, especially without modelling all the traffic on the road. Another difficulty is the diversity in the settings for traffic lights; they may differ between main and secondary roads and between different regions. Therefore, in this simulation model the assumption is made to use predefined cycles for the traffic lights, as explained in section 4.3. The last stochastic process is the travel time along the route sections. This process is the biggest stochastic factor in most scenarios. The distribution that best represents the travel times can alter between route sections since each section can have disparate properties (chance on congestion, tidal traffic etc.). Still, in this study it is chosen to take the same distribution for all route sections for the sake of simplicity. The distribution chosen is the log normal distribution, similar as in the work of Uno et al. [54], Srinivasan et al. [50] and Li et al. [34]. This is done for two reasons: the first is that a log normal distribution has a finite left tail, meaning it is possible to make sure that negative travel times are not modelled. The second reason is that the log normal distribution fits the observed data quite well. The match between the log normal distribution and observed data is shown in the coming sections.

4.5 Case Study Data

For this research a data set was made avaible of 6 months of data from one bus line, from September first up until the first of April. Since the data set is from the Netherlands, this means the summer holiday is not included in the data. The data used for this purpose comes from a high frequent bus line in the Netherlands; line 300. This is a bi-directional bus line between Haarlem station and Amsterdam Bijlmer ArenA, with a frequency of 10 times per hour. The bus line has several properties that make it interesting for this study. The first is the high frequency, which is essential for bunching studies. The second is the combination of several low demand stops with some high demand stops around Schiphol Airport and the end stops of the route. The third is that some route sections are on public road, while others have dedicated infrastructure. The route of the bus line is shown in figure 8.

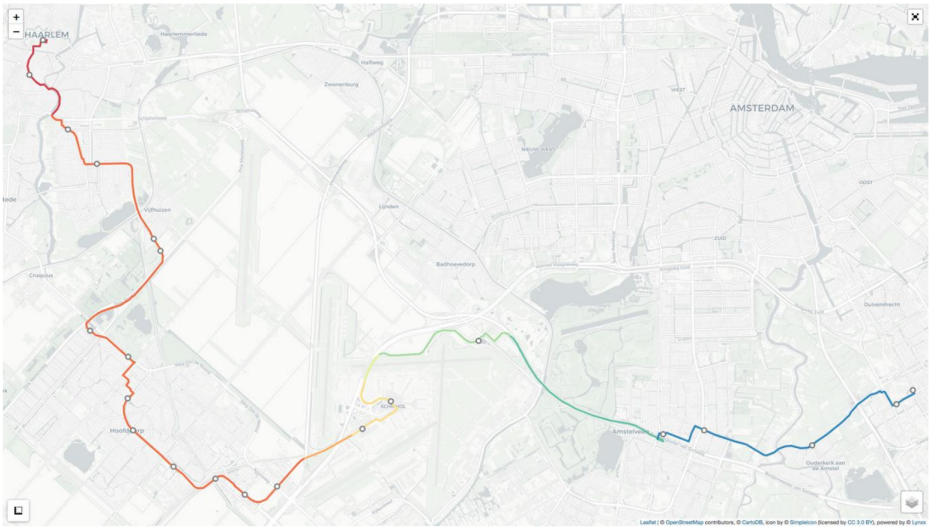

Figure 8: Route of bus line 300 [37]

The data used from bus line 300 is the planned and the realised arrival and departure time for each stop.

The data is processed to be able to visualise and analyse it. Firstly, the planned and realised driving times between stops are calculated by using the planned and realised departure and arrival times. Next, for each route section the results are aggregated. This yields the distribution of the driving times for each route section. However, in this distribution there are still some outliers. These outliers could be generated by a disruption in

the system or by an error in the data generation. All these outliers are filtered out of the data, irrespective of the cause of the outlier. This is done because disruptions are not meant to be modelled in the simulation used in this study and thus they should not be represented in the used distributions. The threshold for the outlier filter is three standard deviations from the median of the data. This is calculated per route section.

The last filter that is applied to the data is based on the time of day. The conditions on a route section can differ throughout the day, e.g. around 11:00 the conditions on the road are usually different from 8:00. This means the driving time distributions shift throughout the day. In this study only static distributions are used, i.e. either a peak conditions are modelled or off-peak conditions. In the analysis the afternoon peak is selected since peak conditions are expected to be more stochastic and the afternoon peak is usually more spread out than the morning peak. A more spread out peak means, more hours can be selected that are expected to fall within the same distribution and more data means a better estimate of the true distribution. The data selected are the driving times between 16:00 and 19:00.

4.6 Case Study Specifications

As discussed in section 4.5 the driving time distribution during the afternoon peak is analysed for every route section of line 300. In appendix B all the results can be found. One of these distributions is plotted in figure 9. As is visible in this figure, the distribution of the observed driving times for this route section is plotted along with the corresponding log normal fit. Lastly, the planned driving time of the route section by the operator is plotted in the same figure.

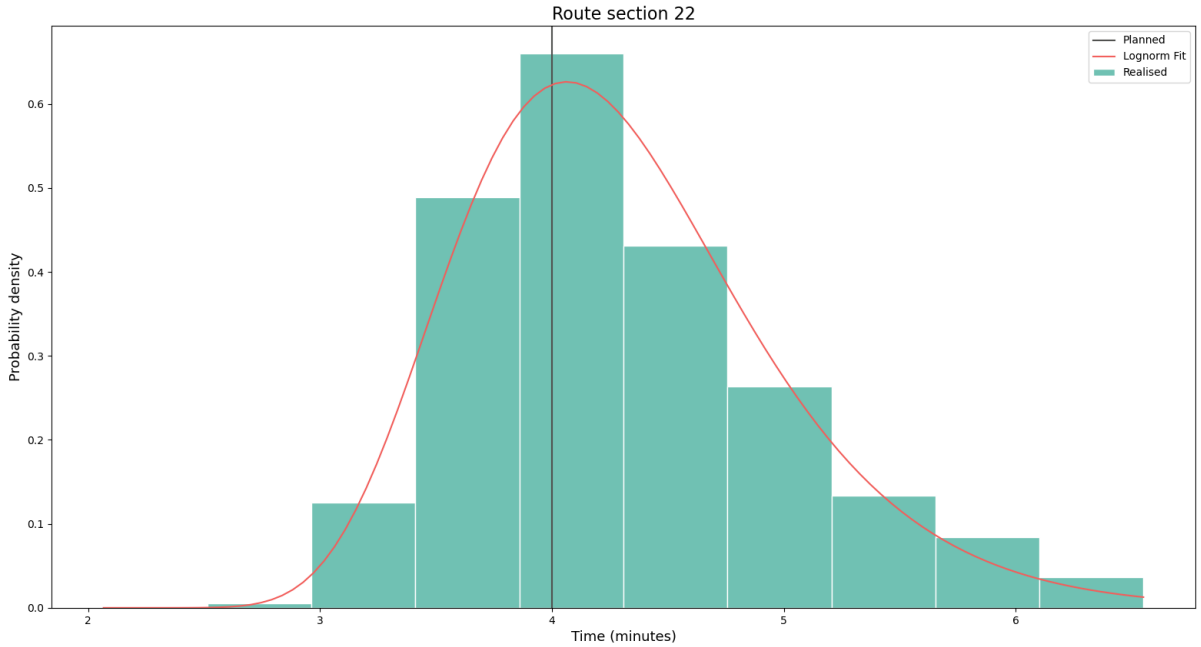

Figure 9: The driving time distribution of the 22nd route section during the afternoon peak. Besides the observed distribution, a log normal fit and the planned driving time are plotted.

The observed distribution from figure 9 fits the log normal distribution well, this also holds for many of the other route sections. This indicates that a log normal distribution is an appropriate distribution to model the driving times in the simulation. Additionally, it provides input for realistic standard deviations of the log normal distributions used in the simulation environment. Yet, there are some route sections where the observed distribution resembles the log normal distribution less. An example is given in figure 10.

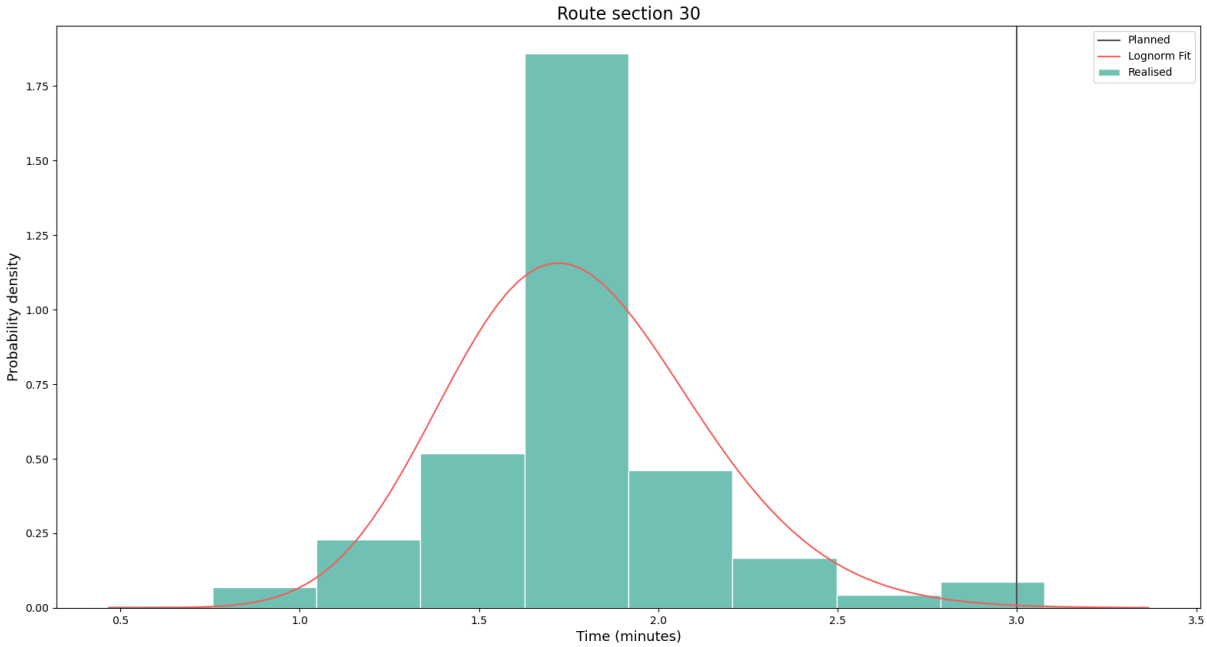

Figure 10: The driving time distribution of the 30th route section during the afternoon peak. Besides the observed distribution, a log normal fit and the planned driving time are plotted.

On this route section the driving time distribution is rather narrow, yet with long tails. This is less well approximated with a log normal distribution. To find the cause for this different shape, the route section and its characteristics should be analysed.

Even though some of the of the observed driving time distributions may be better approximated by another distribution than log normal, the log normal distribution appears to be a good choice for most of the observed distributions, as can be seen in appendix B.

Another interesting point to take away from this data analysis is that the observed driving times are often quite far from what was planned. In this study there is no planned timetable, therefore a discrepancy between the planned and observed driving times is not an issue. However, when MARL is applied in a timetable setting, it could become a problem. Yet, an RL agent will adjust to this discrepancy automatically when it is trained on such a route.

5 Results

Following the main research question (1.4.2), the most important aspect of this study is to investigate the performance of the selected MARL algorithm under different circumstances. The selected MARL algorithm and the selected circumstances are described in section 4.1 and 4.3 respectively.

As is evident from the preceding sections, the algorithm, the observation space, action space and reward function are not thoroughly optimised to get the best performance out of the MARL algorithm. This is not the point of this study. The most interesting point is the difference of performance of the MARL algorithm under different circumstances. This is also the reason that in this section, not only the results of the best performing policy is shown, but the results of the best five policies. This gives insight in the variance among the learned policies in a given scenario, and how easy it is to learn a good policy in the concerning scenario. The best performing policies give an estimate of the performance that could be obtained with a MARL algorithm if it were to be optimised for a given scenario.

In the coming sections the results of the experiments are discussed. There are three aspects to look at in the results: the learning process, the performance and the behaviour of the MARL policies. In section 5.1, the learning process is discussed. This entails whether or not the MARL algorithm converged to a reasonably good policy, and if so, how much time it needed. The performance of the MARL algorithm and how it compares with the benchmarks is discussed in section 5.2. Lastly, the behaviour of the learned policies is discussed in section 5.3.

5.1 Learning Process

There are five learning trajectories per scenario, as discussed in section 4.2. Each of these trajectories has a unique seed and therefore a unique learning curve. The learning curves of all these runs are analysed, as described in section 4.2. These learning curves show how quickly the policies improved and how stable the learning process was. The reward function is based on the average travel time of the passengers, meaning that as the reward increases, the average travel time decreases.

Evidently, every learning curve is unique, yet some patterns are visible. The learning curves for all scenarios are shown in figure 11.

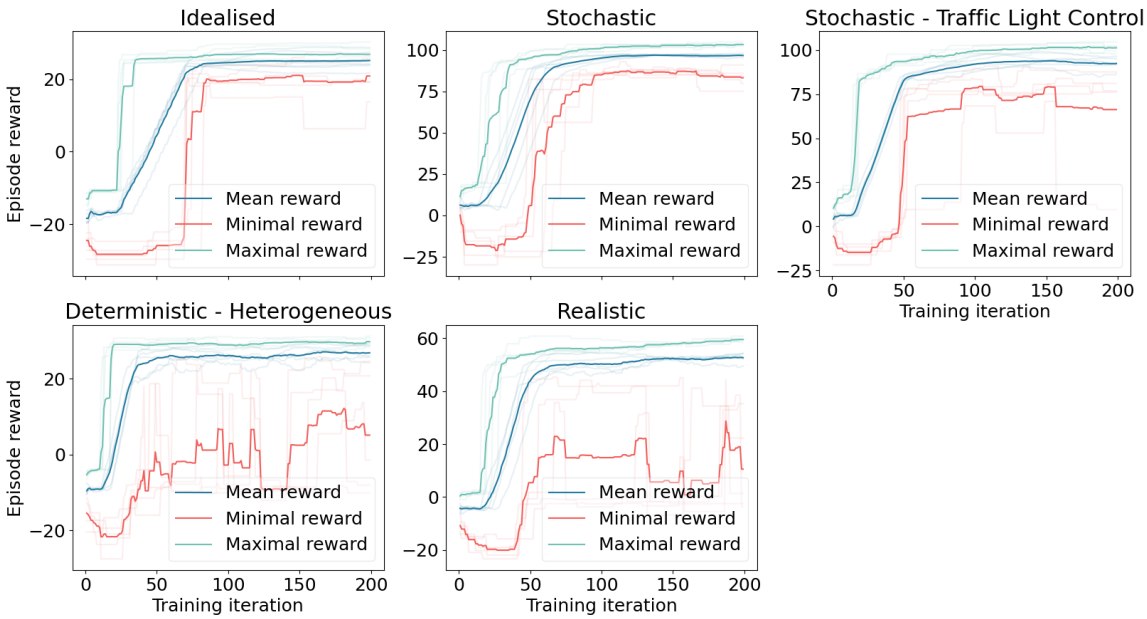

Figure 11: An overview of the learning curves of the training process in all five scenarios. The minimal (red), the mean (blue) and the maximal (green) total reward throughout the learning process are shown. For each scenario there are five selected policies. The clear lines are the average minimal, mean and maximal rewards of these five policies. The transparent lines are the individual minimal, mean and maximal rewards of the five different policies and give an idea of the spread in the rewards.

From figure 11 it is evident that the learning curves in all scenarios roughly have the same shape; a flat start, followed by a clear linear increase in reward and then topping of at its maximum. The flat start of the learning curve is explained by the warming-up phase (4.1) of the algorithm; it is collecting data, but the learning has not started yet. Then the algorithm starts to learn and converges to its maximum reward. In figure 11 the total reward is displayed, this is the sum of the individual rewards of all the bus and traffic light agents. The maximal total reward reached differs between the scenarios since the number of traffic light agents is not the same in every scenario. As discussed in section 3.1.3 the bus and traffic light agents share the same reward function and the reward is based on the global state of the system, the individual learning curves are thus identical to the total learning curve.

Interestingly, the gap between the average maximum and minimum reward is not of the same proportion in every scenario. The scenarios 'Deterministic - Heterogeneous' and 'Realistic' show a larger gap between the maximum and the minimum reward than the other scenarios. Specifically, the minimum reward is quite a bit lower than the mean and maximum reward. This indicates that in these scenarios it is harder for the MARL algorithm to learn a policy that always performs well. This is also reflected by the number of converged policies. From the ten different seeds used, 10, 8, 7, 7 and 6 of them converged for the scenario 1 to 5 respectively. Judging from these numbers, the stochasticity, the traffic light agents and the heterogeneity all increase the difficulty of the learning process for the algorithm.

The training is done on a Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz with usage around 40% and training one single policy takes around 25 minutes with this hardware.

5.2 Performance

In this section the performance of the learned MARL policies is evaluated based on the metrics explained in section 4.2. Additionally, the performance is compared with the benchmarks as described in section 4.2.

First, a performance comparison is given between the best policy of the MARL algorithm and the holding benchmark for every scenario. Afterwards, the performance per scenario is discussed and compared with the benchmarks.

5.2.1 Performance Overview

As mentioned in section 1.4.1, the objective of this study is to investigate the potential of MARL for more realistic settings than the idealised one, which is tested by Wang et al. [56]. In this idealised setting Wang et al. succeed in outperforming all their rule based benchmarks, but it remains unclear if such a performance would also be achievable in a more realistic (and thus more complex) setting. Moreover, it is unclear which additional complexities are difficult to handle for a MARL algorithm and which are easily incorporated in a policy.

The first of the five scenarios in this study is very similar to the simulation environment used by Wang et al. and the performance in this scenario is compared with the performance in the other four scenarios to see if the MARL algorithm has the same potential in these scenarios. All of this is done with an 'off the shelve' MARL algorithm and with a simple observation space, which most likely leads to sub optimal absolute performance. Although the absolute performance of the MARL algorithm may be sub optimal, the performance difference between the scenarios can be used to assess the potential of such an algorithm in each scenario.

The performance of the best policy is compared with the performance of the benchmark for every scenario in figure 12. In figure 13 the average holding time per stop of the best MARL policy is compared with the average holding time per stop of the benchmark. Figure 12 shows that the average total travel time, experienced crowding and excess waiting time are lower under the MARL algorithm than under the benchmark. The only exception to this is the deterministic - heterogeneous scenario. In the deterministic - heterogeneous scenario, the MARL algorithm scores very well on the coefficient of variation. Arguably this policy is too focused on the headway equalisation since it does not lead to shorter travel times or a lower experienced crowding.

In the other scenarios the MARL algorithm manages to outperform the benchmark on the average waiting time as well as the average in vehicle time. How this improvement is achieved differs between the scenarios. In the idealised scenario the MARL algorithm is able to realise better equalised headways than the benchmark with less holding, see figure 13. This leads to a small improvement in the experienced crowding as well and a large improvement in the average excess waiting time due to the higher bus speeds. For the stochastic and the stochastic - traffic light control scenario, the coefficient of variation is higher than in the benchmark. This, in combination with the shorter average holding time (13) means the MARL algorithm opts for a 'looser' bus management here. This leads to higher bus speeds which in turn leads to lower experienced crowding, shorter waiting times, shorter in vehicle times and a big drop in the excess waiting time. In these scenarios the performance gain is thus achieved by shorter headways instead of more reliable headways. Interestingly, the added control over the traffic lights in the TL scenario is mostly used by the MARL algorithm to improve the headway equalisation, not to increase the bus speeds. This means the traffic light agents add value to the system by improving the headway equalisation. The fifth scenario is a combination of the stochastic - traffic light control and the deterministic - heterogeneous scenario in terms of characteristics and this translates to the results. The MARL algorithm is still able to outperform the benchmark, similar to scenario three. However, the improvement in average holding time and excess waiting time is not as big as in the first three scenarios.

These results indicate that stochasticity is not a problem for the MARL algorithm, but that the heterogeneity does increase the difficulty. In the heterogeneous scenarios the agents have to learn the best behaviour per section; i.e. they have to combine the headway observation with the location observation to make the best decision. This added difficulty is visible in the results. Furthermore, it is clear that the traffic light agents do add value to the system. What stands out from the coefficient of variation is that the MARL algorithm equalised the headways very strict in the deterministic scenarios, but much looser in the stochastic scenarios.

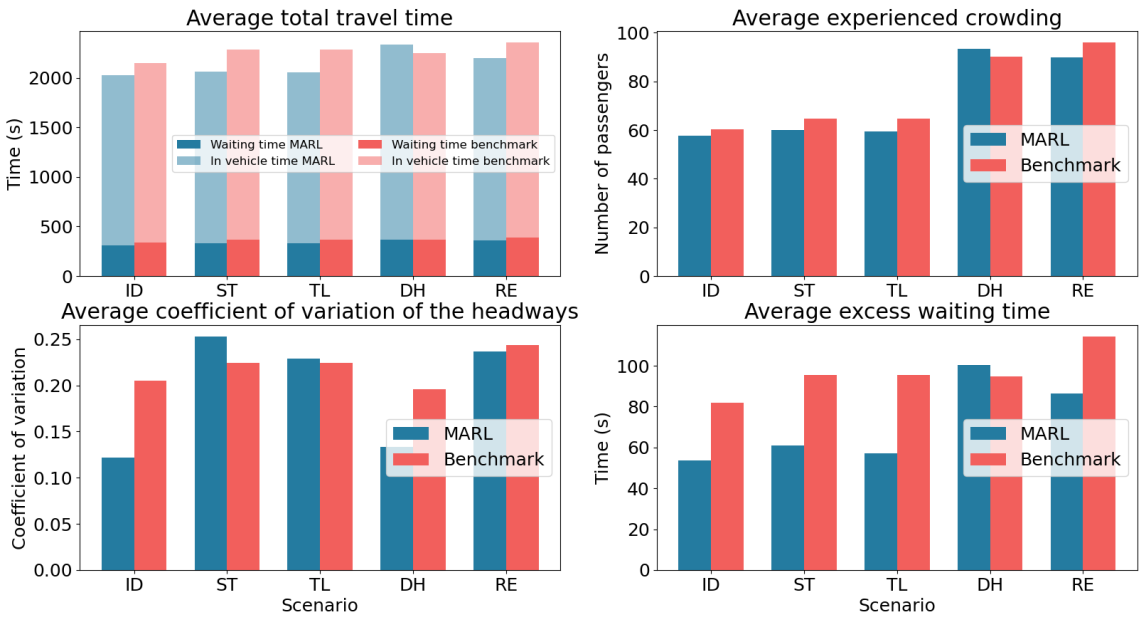

Figure 12: The average metric scores in all five scenarios of the best MARL policy (blue) and the benchmark (red). The total travel time consists of waiting time (blue / red) and in vehicle time (green / yellow).

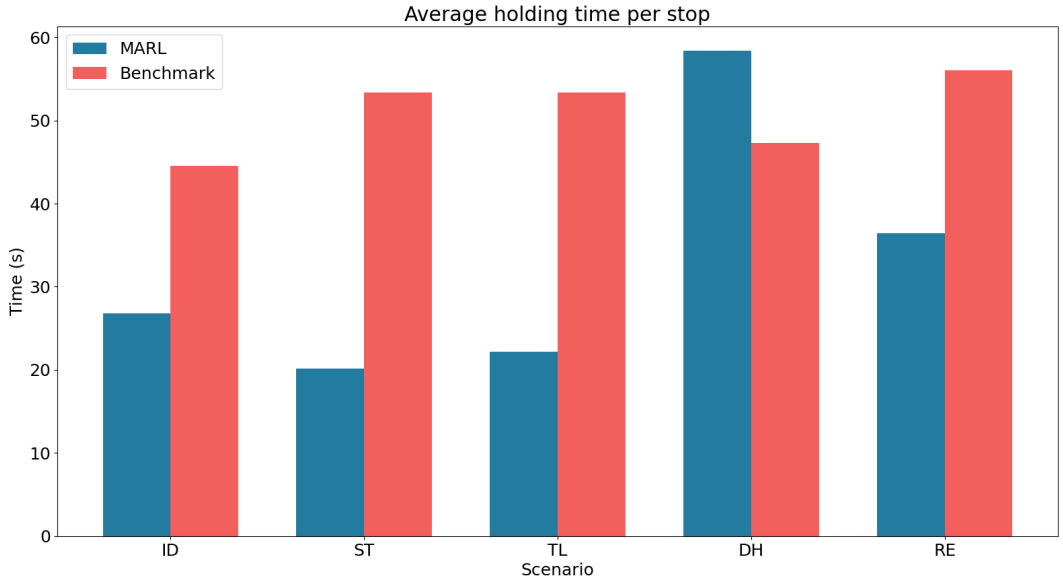

Figure 13: The average holding time per stop in all five scenarios. The scores are displayed for the best MARL policy as well as the holding benchmark.

Note that in some cases the holding benchmark or the MARL policies use a lot of holding time, as apparent from figure 13. In this study this is allowed since there is no predefined schedule or headway a bus has to stick to. The pros and cons of this decision are discussed in section 6.

5.2.2 Idealised Scenario

For every scenario ten different policies are trained. From these ten the best five are selected for comparison with the benchmark, as explained in section 4.2. In this section the five best policies of the idealised scenario are compared with both the holding and no control benchmark.

In figure 14 the average holding times per stop are shown. Only two of the five policies manage to improve on the benchmark performance and the relative differences between the policies are rather large. From figure 15 it is clear that the differences in performance between the policies are also significant. The third policy is best overall, with the best scores on all metrics except the coefficient of variation of the headways. Whereas the first policy performs poorly on all metrics and uses a lot of holding too.

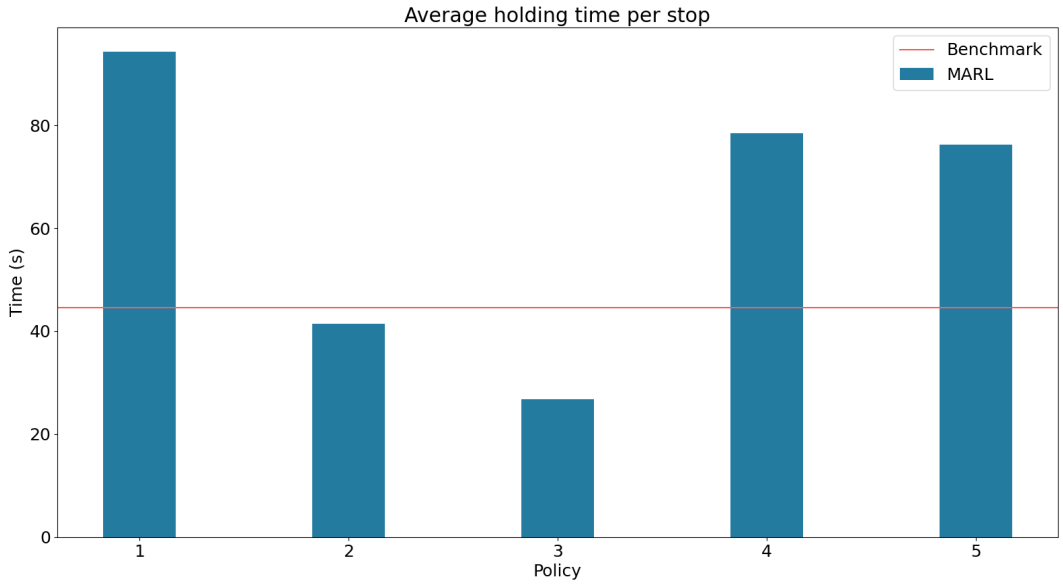

Figure 14: The average holding times per stop of the five best policies of the MARL algorithm compared with the holding benchmark (horizontal line). (idealised scenario)

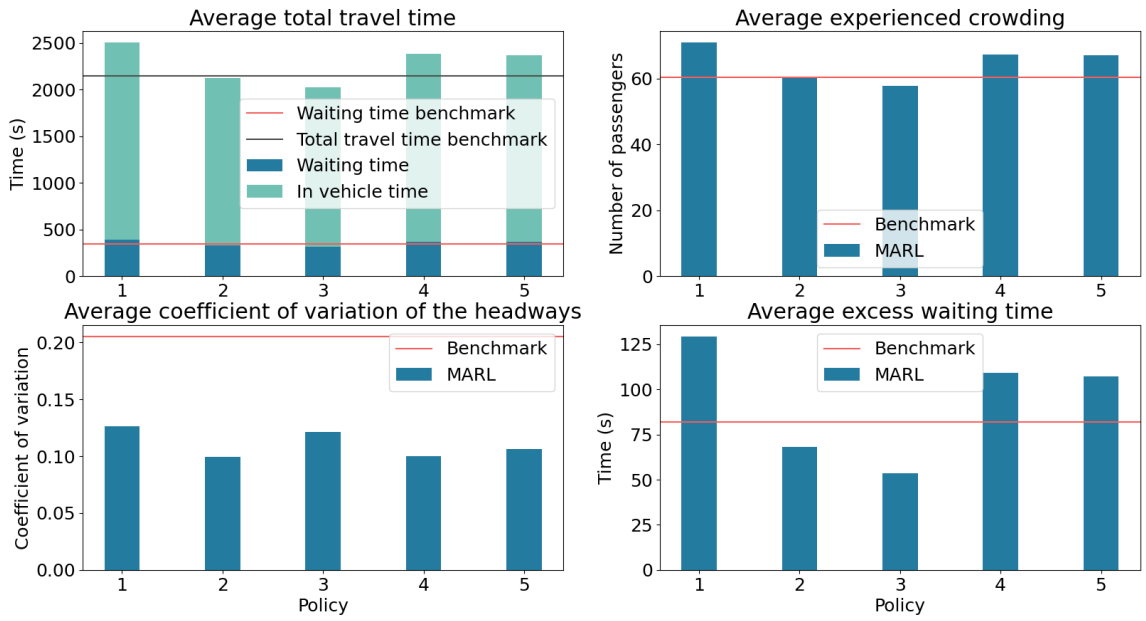

Figure 15: The average metric scores of the five best MARL policies and the holding benchmark (horizontal lines). (idealised scenario)

In table 8 the scores of the best MARL policy, the holding benchmark and the no control benchmark are shown. From this figure it is clear that the MARL policy performs far better than the no control benchmark. The MARL algorithm has thus succeeded to learn a useful policy.

Table 8: An overview of the performance in the idealised scenario of the best MARL policy and the two benchmarks for several metrics.

Metric	**MARL**	**Holding Benchmark**	**No Control Benchmark**
Average holding time	26	44	0
Average total travel time	2024	2146	5133
Average waiting time	312	340	2367
Average in vehicle time	1712	1805	2766
Experienced crowding	58	60	418
Coefficient of variation headways	0.12	0.21	2.11
Excess waiting time	53	82	2145

5.2.3 Stochastic Scenario

In the stochastic scenario all the MARL policies use less holding than the holding benchmark. The difference between the two is substantial as is visible in figure 16. This is a different pattern than the one visible in the idealised scenario.

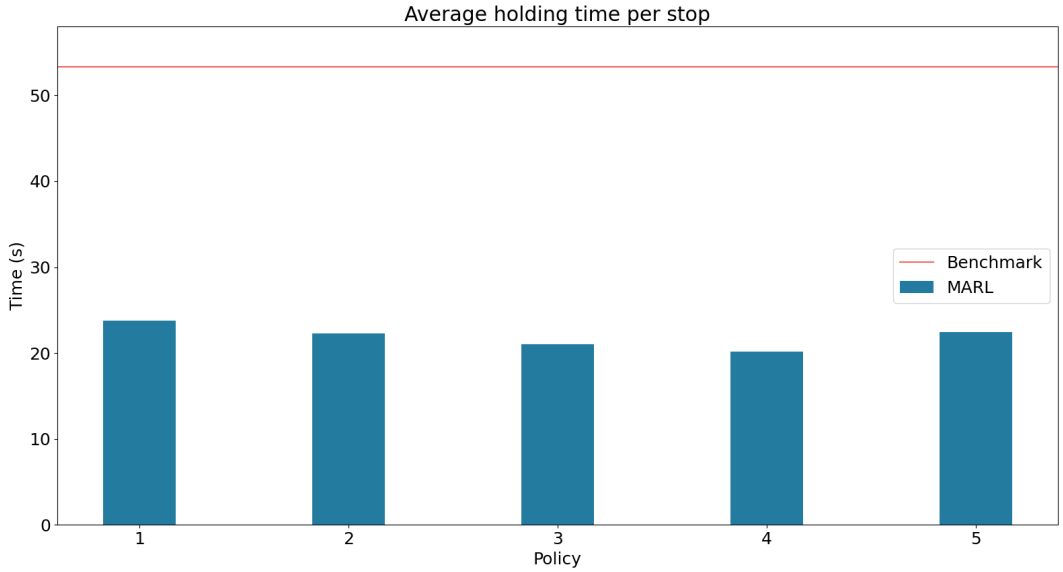

Figure 16: The average holding times per stop of the five best MARL policies and the holding benchmark (horizontal line). (stochastic scenario)

The performance on all metrics is also more constant than in the idealised scenario, see figure 17. It outperforms the benchmark on all metrics except the coefficient of variation. As discussed, this indicates a 'looser' control strategy than the benchmark.

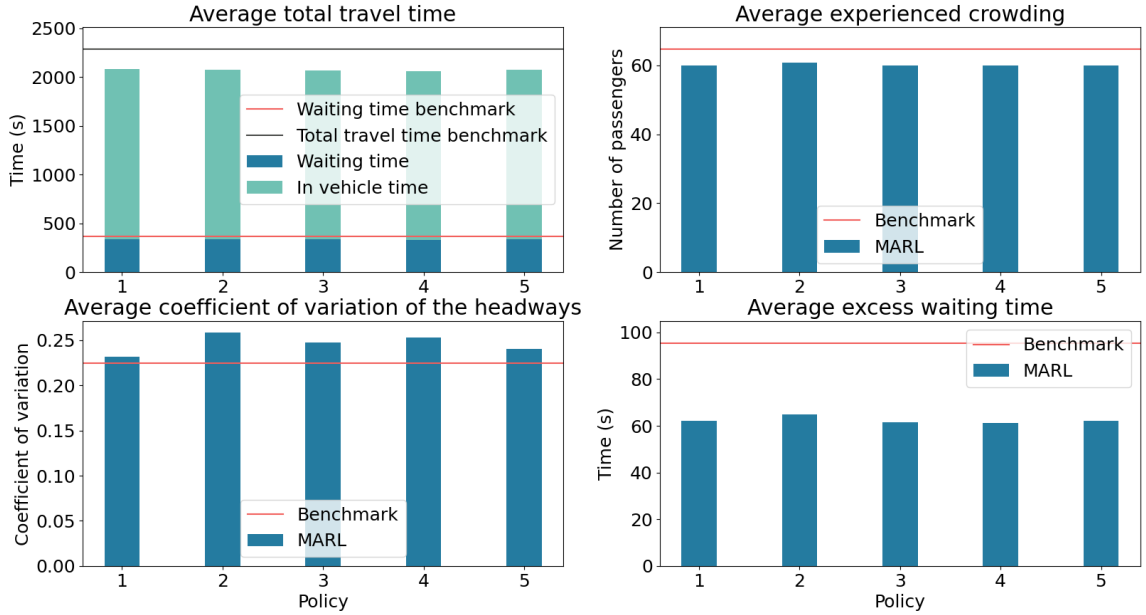

Figure 17: The average metric scores of the five best MARL policies and the holding benchmark (horizontal line). (stochastic scenario)

Similar to the idealised scenario the MARL policies have learned a useful policy that easily outperforms the no holding control benchmark, as can be seen in table 9.

Table 9: An overview of the performance in the stochastic scenario of the best MARL policy and the two benchmarks for several metrics.

Metric	MARL	Holding Benchmark	No Control Benchmark
Average holding time	20	53	0
Average total travel time	2061	2285	5161
Average waiting time	332	366	2360
Average in vehicle time	1728	1919	2800
Experienced crowding	60	65	416
Coefficient of variation headways	0.25	0.22	2.08
Excess waiting time	61	95	2140

5.2.4 Stochastic Scenario - Traffic Light Control

Like the stochastic scenario the MARL policies in the this scenario use less holding time than the holding benchmark, see figure 18. Additionally, the policies have roughly the same score, analogous to scenario two.

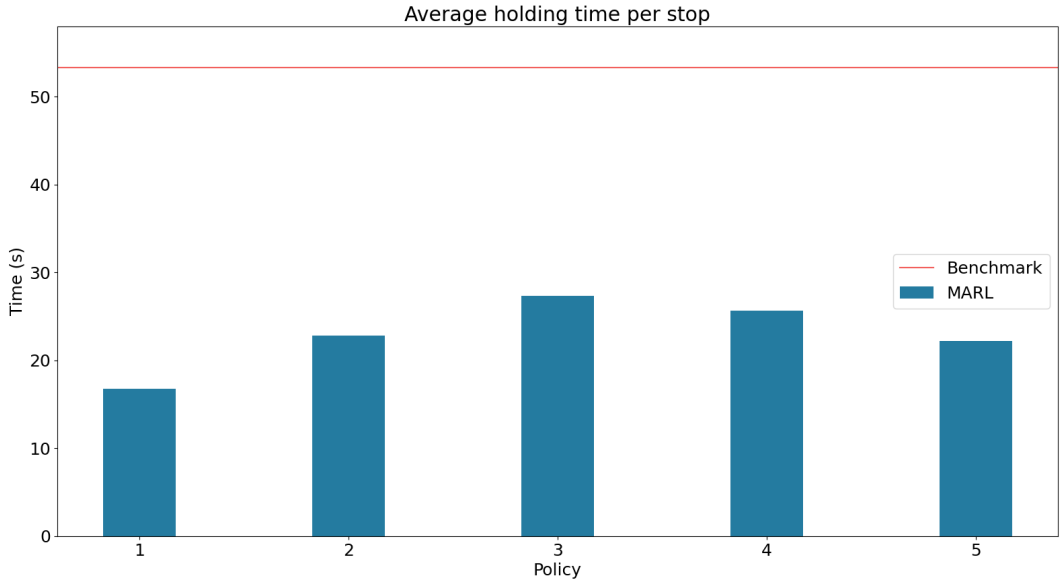

Figure 18: The average holding times per stop of the five best MARL policies and the holding benchmark (horizontal line). (stochastic traffic light control scenario)

Judging from figure 19 and 17, the only policy in this scenario that really manages to reap the benefits of the traffic light agents is policy five. The other policies score more or less the same as the policies in scenario 2 (ST).

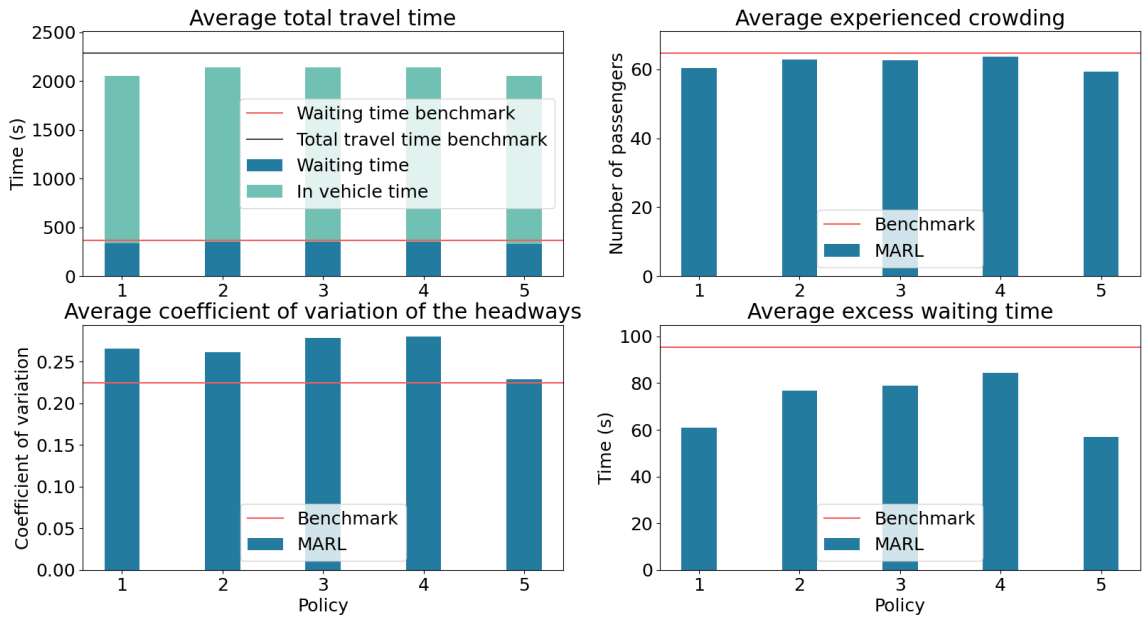

Figure 19: The average metric scores of the five best MARL policies and the holding benchmark (horizontal line). (stochastic traffic light control scenario)

Lastly, table 10 shows the scores of the best MARL policy, the holding benchmark and the no control benchmark. From this figure it is evident that the MARL policies have learned something useful.

Table 10: An overview of the performance in the stochastic - traffic light control scenario of the best MARL policy and the two benchmarks for several metrics.

Metric	**MARL**	**Holding Benchmark**	**No Control Benchmark**
Average holding time	22	53	0
Average total travel time	2053	2285	5161
Average waiting time	328	366	2360
Average in vehicle time	1725	1919	2800
Experienced crowding	59	65	416
Coefficient of variation headways	0.23	0.22	2.08
Excess waiting time	57	95	2140

5.2.5 Deterministic - Heterogeneous Scenario

When looking at the holding time used by the MARL policies in this scenario (figure 20), it stands out that they all use more holding time than the holding benchmark. Interestingly, the amount of holding used does not correlate with the coefficient of variation of the policy, see figure 21. This shows that some policies (1, 2 and 4) did not manage to use the holding actions effectively.

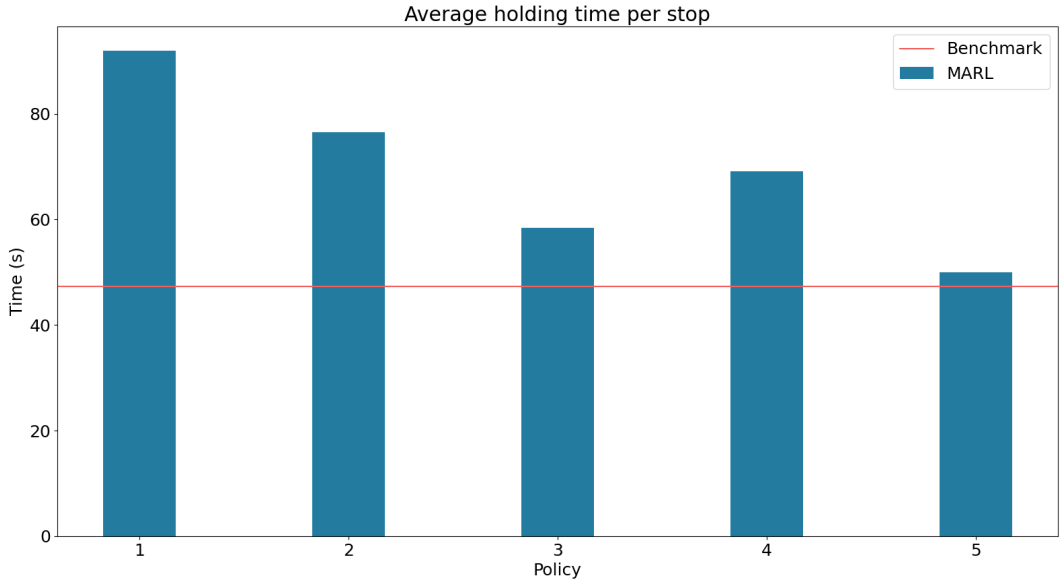

Figure 20: The average holding times per stop of the five best MARL policies and the holding benchmark (horizontal line). (deterministic - heterogeneous scenario)

Figure 21 also shows that the MARL algorithm does not find its way to an effective policy in this scenario as easy as in scenario two and three. Similar to the idealised scenario, the policies differ significantly in their performance characteristics. It is possible that the fact that both scenarios are deterministic plays a role in this.

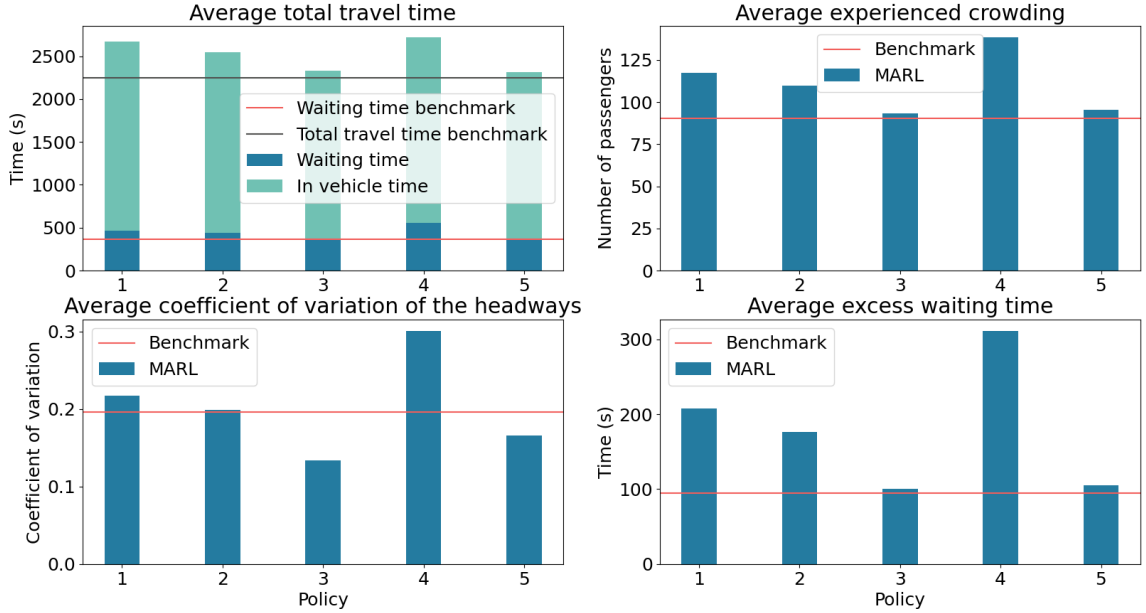

Figure 21: The average metric scores of the five best MARL policies and the holding benchmark (horizontal line). (deterministic - heterogeneous scenario)

The exact scores on all the metrics of the best MARL policy and both benchmarks are shown in table 11.

Clearly the MARL policy is better than no control.

Table 11: An overview of the performance in the deterministic - heterogeneous scenario of the best MARL policy and the two benchmarks for several metrics.

Metric	MARL	Holding Benchmark	No Control Benchmark
Average holding time	58	47	0
Average total travel time	2333	2246	6949
Average waiting time	367	363	2985
Average in vehicle time	1966	1882	3964
Experienced crowding	93	90	721
Coefficient of variation headways	0.13	0.20	2.21
Excess waiting time	100	95	3095

5.2.6 Realistic Scenario

Figure 22 shows a pattern that is similar to that of scenario two and three, the other stochastic scenarios. All the policies use less holding time than the holding benchmark. Interestingly, this does not necessarily lead to worse headway equalisation as seen in figure 23.

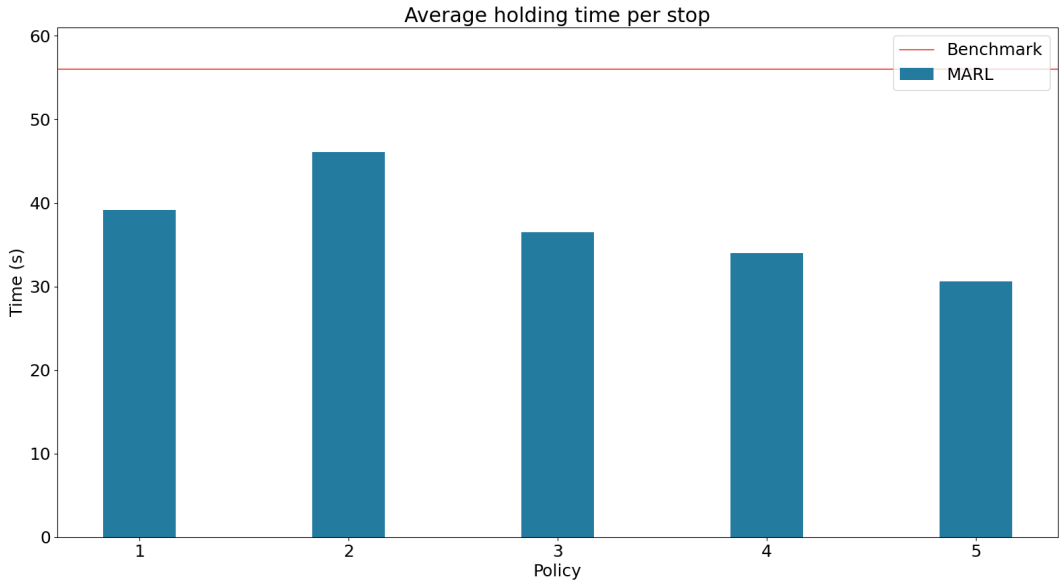

Figure 22: The average holding times per stop of the five best MARL policies and the holding benchmark (horizontal line). (realistic scenario)

The mix of characteristics of this scenario appears to be visible in the performance as well. All the policies are reasonably good, similar to the pattern in the other stochastic scenarios. However, there is no clear difference with the benchmark on a specific metric, the average coefficient of variation of the headways is not clearly larger as in the other stochastic scenarios and the average excess waiting time is not that much better than the benchmark. What is clear from these results is that the addition of the stochasticity leads to a more consistent learning process between the different policies. Furthermore, the added traffic lights create more opportunity for the MARL to reach a better performance, making it easier to outperform the benchmark than

in the deterministic - heterogeneous scenario.

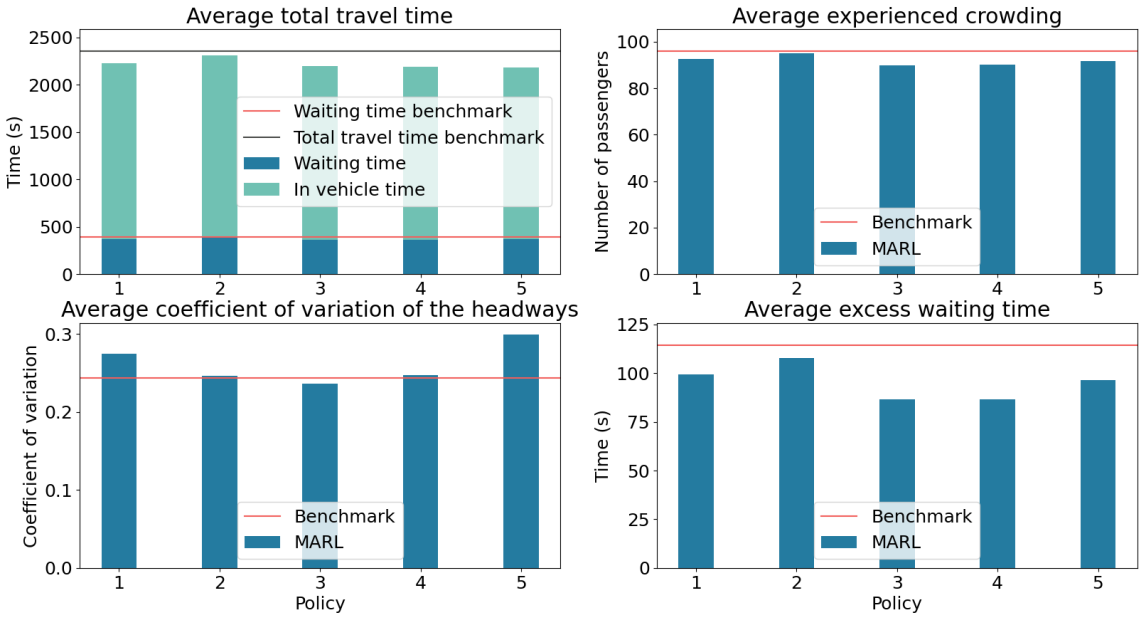

Figure 23: The average metric scores of the five best MARL policies and the holding benchmark (horizontal line). (realistic scenario)

Lastly, table 12 compares the performance of both the best MARL policy and the holding benchmark with the no control benchmark. Once again the no control benchmark yields results that are far worse than those of the MARL policy and the holding benchmark. This shows that even in more complicated scenarios like this one, the MARL algorithm is able to learn a well performing policy.

Table 12: An overview of the performance in the realistic scenario of the best MARL policy and the two benchmarks for several metrics.

Metric	**MARL**	**Holding Benchmark**	**No Control Benchmark**
Average holding time	36	56	0
Average total travel time	2194	2353	7192
Average waiting time	359	388	3115
Average in vehicle time	1835	1965	4077
Experienced crowding	90	96	750
Coefficient of variation headways	0.24	0.24	2.22
Excess waiting time	87	114	3235

5.3 Policy Analysis

To get an outline of the policies learned by the MARL algorithm, the observation-action pairs of the simulation are stored during the evaluation runs. This means that a data set is stored with all the received observations the bus agents got when they took a certain action. This data is displayed in figure 24, which shows the median observation for each of the possible action for the bus agents. Around the median the 15^{th} and 85^{th} percentile

values are shown. This bandwidth shows the range of observations that led to a specific action. This gives insight into the policy learned by the MARL agents. Note that the MARL algorithm does not necessarily use all available actions, which can lead to a larger bandwidth than observed in the holding benchmark. Naturally, the shortest and the longest holding action can have a bigger bandwidth and the median can be different than expected since these are 'end points' of the action space. For example, when the forward headway is 200 or 300 seconds shorter than the backward headway, the bus will probably hold for 180 seconds, yielding a large bandwidth for this action.

The y-axis here is the backward headway minus the forward headway in seconds. A positive value on the y-axis thus means that the concerning bus is closer to the bus in front of him than the one behind. For every scenario a positive correlation between the action taken and the observation can be observed, which is unsurprising.

One observation that can be made from this figure is that the MARL lines are above the benchmark line for the longer holding actions. This holds for all scenarios but the deterministic heterogeneous scenario and the outlier in the stochastic - traffic light control scenario. This means the MARL policies allow a larger gap between the forward and backward headway before they use a long holding action. Furthermore, figure 24 shows that the MARL policies never use the 180 seconds holding action.

The outlier in the stochastic - traffic light control scenario shows the need for checking the learned policies on unexpected behaviour before such a policy can be deployed. Even though this policy is effective, as seen in 5.2.1, a holding action of 90 seconds is most probably undesired when the forward headway is already greater than the backward headway.

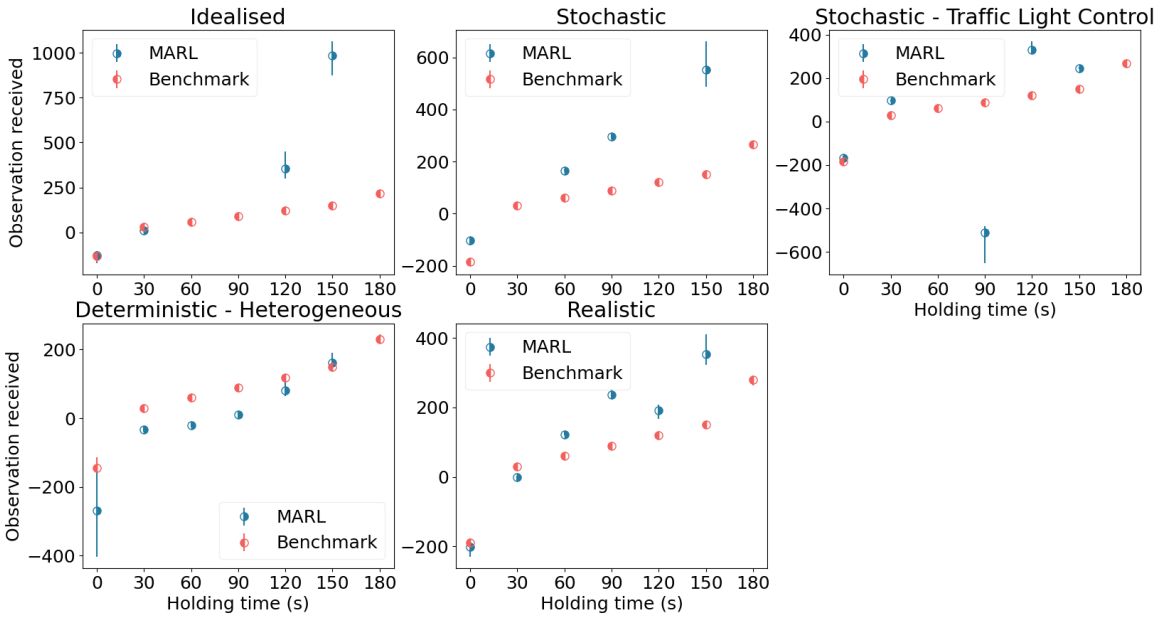

Figure 24: A visualisation of the policies learned by the best MARL policy per scenarios. For each action the median headway observation is shown along with a bandwidth between the 15^{th} and 85^{th} percentiles. Note that for the scenarios DH and RE the observation received by the agents also contains the current position of the agent on the route, this observation is not displayed in this visualisation.

Besides the bus agents, scenario 'Stochastic - Traffic Light Control' and 'Realistic' also have traffic light agents. These agents have three choices; do nothing, reduces or increase the delay of the bus by 10 seconds. Figure 25 shows how often each of these actions are selected by these agents.

It is immediately clear that the agents controlling the traffic lights do not simply decrease the delay of the busses to aid their objective of minimal travel time. Strikingly, the -10 seconds delay action is not used at all

in the stochastic - traffic light control scenario and barely used in the realistic scenario. This seems to indicate that in these policies the busses overestimate the speeds of the bus in front of them, which, if the preceding bus is indeed slower than hoped, can be neutralised by the traffic lights. This is speculation, and it is very well possible that other seeds may have lead to a different strategy.

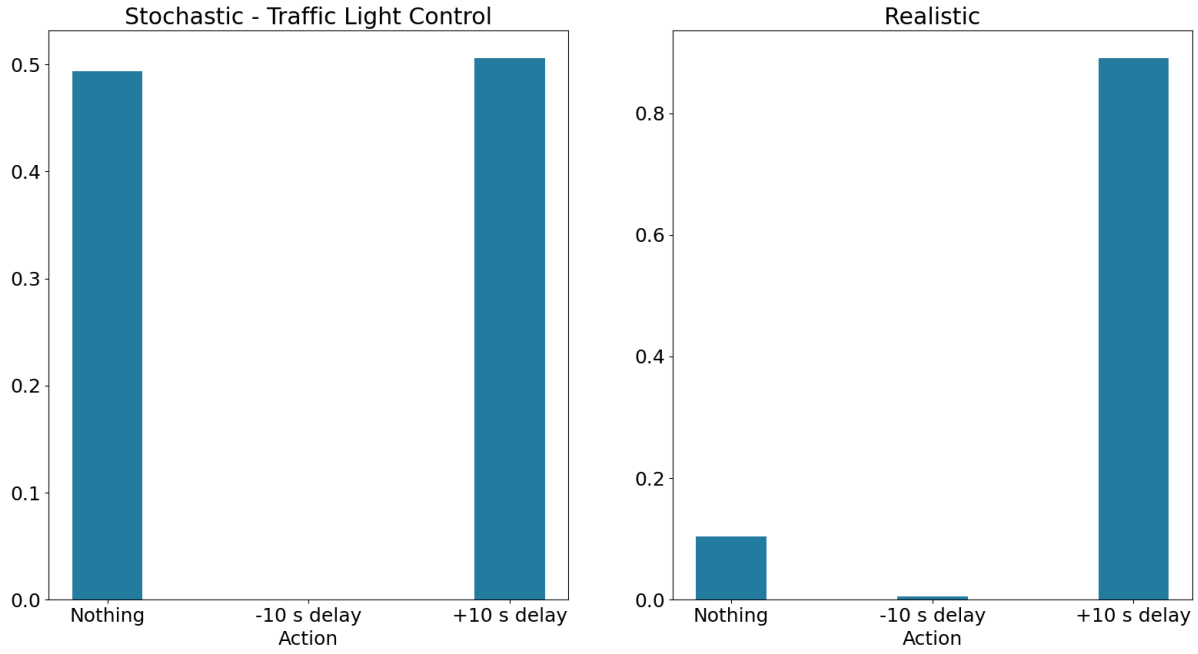

Figure 25: The fractions of the actions chosen by the traffic light agents of the best MARL policy for both scenarios with traffic lights.

This sums up the analysis of the MARL algorithm. In section 5.1 the learning process is analysed, showing the reward obtained during the many learning episodes. In sections 5.2 and 5.3 the best performing policy of the learning process is executed greedily (i.e. without learning) in a number of simulation runs to assess the performance.

6 Conclusions

The goal of this study is to investigate the potential of a MARL algorithm for real world bus management. As a first step to achieve this goal, a literature review is executed. With the information from the literature review, a simulation model is developed which is connected to the RL framework RLlib. Within this simulation model, five different scenarios are designed for experiments. Lastly, the learning process, the performance and the behaviour of the MARL algorithm is analysed. In the coming sections the key findings, contributions and limitations of this study are discussed along with several recommendations for future work.

6.1 Key Findings

This study aims to investigate the performance and potential of a MARL algorithm under different types of (combinations of) environmental and control complexity, as posed in the main research question in section 1.4. In this section the research sub questions are put forward to support the answer to the main research question. An answer is provided to all the research sub questions and the main research.

The first research sub question is: *What is the current state-of-the-art of applying MARL systems to both headway management and intersection control?*

For intersection control there are numerous studies done with the use of MARL and the performance of these implementations promises to outperform the current control algorithms. The use of MARL for headway management is not as mature as for intersection control. Wang et al. [56] successfully implemented a MARL algorithm for headway management in an idealised setting. The results of this study are promising, but there are still several hurdles to take before such an implementation can be implemented in practice. A combination of both intersection control and headway management in an MARL is not found in the literature.

The second research sub question is: *What are valuable extensions to the current state-of-the-art in terms of simulation environment and control complexity?*

The starting point for this question is again the work done by Wang et al. [56]. Several possible extensions to their simulation environment are identified in section 1.4.1. All the identified route extensions (simulation environment) are implemented in the simulation used in this study. For the control extensions the traffic light control is implemented, but the speed control is not. The traffic lights are selected because controlling traffic lights on route seems widely applicable whereas speed control is usually only possible on routes with dedicated infrastructure. They were not both selected to limit the scope of the study.

The third research sub question is: *For which (combinations) of the above mentioned extensions is it feasible to implement them in a simulation environment?*

This question combines with the previous question. All identified extensions can be implemented in the simulation due to the freedom of the selected simulation framework. Yet, to limit the scope of the study and the time needed for building the simulation framework, not all extensions are implemented.

The fourth research sub question is: *What is the performance of a MARL algorithm under the above mentioned environment complexities?*

The answer to this question is built upon the work done by Wang et al. [56], where they manage to outperform their benchmarks with their MARL algorithm. The extensions to their work are introduced in section 1.4.1 and the final experiment scenarios are described in 4.3. The absolute performance of the MARL algorithm in these scenarios is not optimised thoroughly. Therefore a scenario similar to the configuration from Wang et al. [56] is implemented to have a point of reference in terms of performance relative to the used benchmarks.

The expectation before the experiments was that the performance of the MARL algorithm would be the best in the idealised scenario. For the stochastic scenario the expectation was that the learning could be less stable, but that the MARL algorithm may be able to find a better balance between headway equalisation and operational speed than the holding benchmark. When the traffic light agents are introduced, like in the traffic light control scenario, the expectation was that the learning process becomes less stable again. In fact, the expectation was that converging to good behaviour could be hard in this scenario, since the bus and traffic light agents have

to cooperate. This same expectation holds for the last two scenarios; deterministic - heterogeneous and realistic.

Some of these expectations turned out to be true, yet some results turned out different than expected. In the idealised scenario all of the ten seeds converge, this number then decreases with each scenario and in the realistic scenario only six of the ten policies converge. This confirms the expectation that the scenarios are increasingly complex for the MARL algorithm. However, this does not reflect the performance of the converged policies.

In the idealised reference scenario the performance of the MARL algorithm outperforms the holding benchmark in terms of travel time, as shown in figure 12. This is also the case for the stochastic, the stochastic - traffic light control and the realistic scenario. Meaning that these scenarios are not too complex to be handled by a MARL algorithm. The performance in the stochastic and the stochastic - traffic light control scenario indicates that stochasticity is not a problem at all for the MARL algorithm since the performance gap with the benchmark increases compared with the idealised scenario. On the contrary, all five policies in the stochastic and the stochastic - traffic light control scenario perform well, which is not the case in the idealised scenario. Another aspect that stands out in the stochastic and the stochastic - traffic light control scenario is that the MARL algorithm exhibits a 'looser' headway equalisation strategy; the average coefficient of variation is higher than the one of the benchmark. In the deterministic scenarios, i.e. the idealised and the deterministic - heterogeneous, the MARL algorithm does the opposite: the average coefficient of variation is lower than the one of the benchmark. When looking at all scenarios together it is clear that on average the MARL algorithm uses less holding than the benchmark. Meaning that the performance gain largely stems from shorter headways, instead of more reliable headways. This pattern implies that in a stochastic setting a 'looser' headway management strategy can be beneficial for the average total travel time of the passengers. This is also reflected by the analysis of the policies in figure 24, where the policies allow for a bigger difference between the forward and backward headway before they use a long holding action.

From a comparison between the stochastic and the stochastic - traffic light control scenario it stands out that the traffic light agents are able to add value to the system. Even though it seems to be hard to reach a good cooporation between the busses and the traffic lights (only one of the five policies manages to achieve this), the result is a lower average coefficient of variation with similar performance on the other metrics, see figure 12.

The deterministic - heterogeneous scenario is the only one where the MARL algorithm was not able to outperform the benchmark, which implies that it is significantly harder to learn a good policy for a heterogeneous route than for a homogeneous one. When both the heterogeneity and the stochasticity are combined (scenario five), the MARL algorithm was able to outperform the benchmark. One possible cause of the improved performance in the realistic scenario is the stochasticity. It is possible that the stochasticity yields a more robust learning process since the feedback from the environment is not exactly the same every time for the same observation-action pair. Yet, the difficulty of the heterogeneity in the realistic scenario is still reflected by the lower minimal reward of the learning curve, see figure 11.

These results are very promising and it is likely that if more effort is spent on the optimisation of the algorithm in these scenarios, that they will be able to perform even better. This means there is potential for MARL algorithms in the field of bus management.

The fifth research sub question is: *What is the benefit of using a MARL system for the problem of bus management compared with rule based management?*

By definition MARL has several advantages over rule based bus management. The possibility to learn constantly and update the policy is one of these benefits. Another benefit is that a MARL policy can theoretically use a lot of information in its decision making, which can lead to better performance. However, this comes with the practical challenge of designing and tuning a stable learning process. Lastly, a benefit of a MARL algorithm is that the objective of the systems can be the objective of the algorithm. This eliminates the need for heuristics and possibly the tuning of their parameters. In this study the average passenger travel time is chosen as the objective of the algorithm. This is also the objective of the bus systems and eliminates the need to find the right balance between headway equalisation and driving speed.

Yet, it also has some disadvantages; one should check a trained policy for unexpected behaviour. An ex-

ample of such behaviour is shown in figure 24, where the policy in the stochastic - traffic light control scenario learned to hold for 90 seconds when the forward headway is already bigger than the backward headway. Another disadvantage is that currently the implementation of a MARL bus management system will likely take more effort than a rule based system since it is a novel technology.

The sixth research sub question is: *To what extent is the used implementation transferable to other situations?*

With the simulation model created in this study it is possible to recreate bus routes with various lengths and characteristics. Additionally, the MARL training setup used can be applied in all possible configurations of the simulation environment. Therefore, the used implementation is transferable to many different situations. However, if a scenario is created with very different characteristics than the ones used in this study it may be beneficial to use a different observation space and different hyperparameter settings. In scenarios with different characteristics the performance may also be different. Furthermore, it is not possible to simulate a bus network containing multiple bus lines with the used simulation model.

The last research sub question is: *To what extent can the used implementation be translated from a model environment to the real world?*

The answer to this question depends on the specific situation to which 'real world' refers. There are many technical issues to be addressed when a MARL bus management system is implemented, but they are not addressed in this study. However, several choices made in the training setup are made with real world deployment in mind. These are the following characteristics:

- The step size in the training setup is 15 seconds. This allows for some data latency since the bus agents have 15 seconds to receive their action after sending their observation.
- The 15 second step size also means that the observation frequency is limited which is less likely to lead to a data overload.
- The action space of the bus agents consists of realistic holding actions that are simple enough to implement.
- The used information for the observation (the headways) is often available in bus systems.

In terms of realism of the simulation environment, the used simulation model captures some of the most important dynamics of the bus management problem and uses stochastic processes for many of these. This can reflect a real world scenario well, but a simulation model always lacks some details and interactions found in the real world.

Therefore, a reasonable performance is expected if the used implementation is implemented in the real world. However, the technical details of the implementation are not investigated yet. There are many possible restrictions that one can encounter when implementing such a MARL solution in a real world scenario. One of the biggest ones could be an underlying timetable the busses have to stick to, this would restrict their action freedom. A similar issue could be restrictions in the action selection for the traffic lights due to constraints from the authorities. Furthermore, other (technical) issues could arise like a delay in the data (observations) connection, a full data outage or drivers that do not follow the prescribed actions. Such issues need solutions that are tailored to the specific situation since the constraints can vary between different real world implementations. A more general topic that is paramount for real world implementation is that the used policies should be thoroughly checked for unexpected behaviour. These points indicate that a real world implementation of a MARL bus management system still needs additional research.

6.2 Contributions & Limitations

As mentioned before, the work from Wang et al. [56] can be seen as the starting point for this study. The main contributions from this study are expanding the simulation environment in which the MARL algorithm is trained to a more realistic environment. This is done by making several of the processes in the simulation stochastic and creating a heterogeneous bus route. As discussed in the previous section, the behaviour of the MARL algorithm is notably different in the stochastic scenarios compared with the deterministic scenarios.

This indicates that the conclusions drawn from deterministic simulation settings have limited value.

Further, as this is an exploratory study the performance of multiple seeds is shown per scenario. The difference between the seeds shows the relevance of investigating multiple seeds. Also, the difference between the performance of the policies from the different seeds was not of the same magnitude in all scenarios; in the idealised and the deterministic - heterogeneous the differences are substantial, while in the stochastic and the stochastic - traffic light control they are not. The differences of the realistic scenario are somewhere in between.

Another contribution is the fact that the agents of the MARL algorithm are not rewarded for their headway equalisation, but for the average total travel time of the passengers. By choosing a simulation length around the value of twice the average passenger travel time, this reward function becomes effective, as discussed in section 3.1.3. With a shorter simulation length it is possible that the MARL agents will optimise their actions for the passengers with short journeys only, giving results with limited value. With longer simulation lengths, it becomes increasingly difficult to dissect the good behaviour from the bad behaviour in a run, since only one reward is given. This may lead to slower learning or not learning at all. By selecting this reward function, the problem is no longer headway management, since the headways are not a direct objective, but a broader concept which is called bus management in this study. The added traffic light control contributes to this broader concept as well.

The potential for performance gain in bus management is greater than in headway management only, since headway management is a subset of bus management. Additionally, the results show that the more complex scenarios are not too difficult for a MARL algorithm, showing that a MARL algorithm has potential outside idealised scenarios as well. The absolute performance in the experiment scenarios is probably still sub optimal due to a lack of thorough optimisation. However, this remains uncertain until the needed optimisation is done and the corresponding results are analysed. This marks the biggest limitation of this study; it is exploratory and does not execute an in depth analysis per scenario to optimise the performance.

A different limitation of this study is that the scenarios in this study do not use a timetable or a predefined headway, instead the headway emerges. This is not the standard in many regions around the world. If the simulation environment would have to simulate a scenario with a timetable, the bus agents will have less freedom in choosing their holding actions. Nonetheless, the experiments done in this study may provide useful input for creating timetables since it shows the optimal holding actions when there are no constraints.

Finally, the choices made in the design of the scenarios have impact on the results. One of these choices is the capacity of the busses, which is infinite in the created scenarios. This affects the potential for bunching in the system since the difference in occupancy between the busses can get larger than with a limited capacity. Therefore, this choice affects the performance of the no control benchmark, but its effect on the holding benchmark and the MARL control is probably limited.

6.3 Recommendations for Future Work

This study shows that there is potential for MARL in the field of bus management. Nevertheless, there is still a lot that should be investigated before this can become a reality.

One of the most essential topics for future research would be assessing the final learned policy. It is important that a policy that is used outside a simulation environment does not exhibit unexpected behaviour in certain situations. Another option is to detect unexpected behaviour and create a safety system that takes over in these situations and executes default behaviour. The more influence the actions of the MARL agents have, the more important this becomes. Another aspect of real world deployment could be retraining the MARL agents when the conditions have changed. Retraining can be done from scratch, i.e. change the configuration of the simulation model and train a new policy. Another option is to start with the legacy policy and train it for a number of episodes in a simulation model with a different configuration. Evidently, data from the bus network is needed to be able to configure the simulation model correctly. The best practices for retraining are an interesting and essential topic for future research.

As indicated in section 6.2, one of the big limitations of this study is that it does not optimise for each scenario to obtain the maximal performance achievable. It is interesting to see if a bigger observation space and

a more extensive hyperparameter optimisation can lead to better performance. From the results of this study it is clear that this is especially the case for the deterministic - heterogeneous and the realistic scenario due to their heterogeneity. In these scenarios it can be an advantageous for the bus agents to have information about the amount of passengers at stops, the amount of passengers in the busses and the actions of the other busses. The more these factors fluctuate along the route and with time, the more useful this information becomes. Therefore, a larger observation space becomes more interesting in complex scenarios. Yet, this also adds challenges; the learning process becomes increasingly complex and the availability of the used information in a real world setting can become a problem. Since this type of research is highly scenario dependent, it is probably only worth while with a specific case study in mind. Such a study could be done with a similar configurable simulation environment as used in this study, this allows for adjustments to new input data our a change in the route.

One of the benefits of MARL is the automated decision process. In this study this benefit is not fully utilised since the actions are only taken on a stop level. It could be beneficial for the performance of the MARL algorithm to add more decision moments to the environment. This approach could lead to a near continuous observation and action cycle. Even though this could improve the performance, the challenge with such an algorithm will be to translate it to a setup that can be used in the real world; continuous actions and observations do not align well with the limitations of bus drivers and data connections.

As mentioned in section 6.2, the main contribution of this study is the added realism to the simulation environment. Even though big steps are made from the idealised scenario towards more realistic scenarios, there are still some possible additions that could be investigated in future research. One of these is to introduce changing conditions of the route as a function of the time of day. Another one could be to introduce route sections where the busses have dedicated infrastructure. This could combine well with introducing a speed up / slow down action for the bus agents. Similar to the previous topic, such a study would probably make more sense with a specific case study in mind.

A less essential, but still interesting, topic would be to investigate the behaviour of the agents in the system further when the observation space is extended. If the average travel time is used as the objective function, new insights may be found when the behaviour of the agents is examined in specific situations.

Another topic that is not directly essential, but that may yield interesting insights in both the MARL training process and the best bus management strategies, is cross evaluation. Cross evaluation in this case would mean to evaluate a policy on a different scenario than on which it is trained. It is unclear what this will yield in terms of performance, but the analysis may provide new ideas for bus management strategies and the MARL training process.

Another possible future addition, besides extending the observation space, can be to add more actors to the system. This can be especially interesting when a whole network is modelled. In this study traffic lights are added, but, with an ITS in mind, more actors could be included. These actors could be on-route actors like traffic lights, but they could also be demand steering actors, like a mobile travel application.

When MARL is successfully used for bus lines, it is interesting to expand the simulation from one line to a complete network. This would introduce a new set of interactions in the simulation model, which in turn introduces new challenges to the learning process. This topic has potential in terms of system optimisation, but the state-of-the-art is currently not yet sufficiently developed. If such a study is to be executed, using a global variable like the average passenger travel time seems a good fit for finding the system optimum. Yet, some boundary conditions may have to be introduced to make sure each line gets the priority it should have. Additionally, intermediate rewards may be necessary since there is a chance that some lines in such a network have trips that are too long to reward at the end of the simulation only. Also, the observation function should be reconsidered in this case since the agents should take more information into account. To limit the complexity for the agents an actor critic algorithm may be the best fit since this could work with a simple observation space for the actors and a critic who can assess the global state of the system. Moreover, using an exploratory setup like this study may prove to be infeasible since the number of possible configurations of a network are larger than on a single line. An option could be to recreate an existing network in a simulation. Again, for such a simulation model a similar configurable setup could be used as in this study to be able to try many configurations of the network.

Lastly, in section 1.4.1 the topics of resilience and robustness are introduced. These topics are not a focus

of this study since the focus is on the day-to-day operations on a bus line, but they are interesting to investigate. Disturbances of the bus system are handled well by a MARL algorithm, but it is appealing to investigate whether a MARL algorithm could handle major disruptions as well. These studies are vital when a MARL algorithm is used continuously, but when a backup system is still in place this is not the case. In that case they should be viewed as exploratory studies for wider application of MARL. Using a MARL algorithm to solve disruptions of the system could save time that is needed to come up with contingency plans for all possible disruption scenarios. Instead, the MARL algorithm could learn what to do in each of these scenarios through a simulation setup. Yet, keeping a human in the loop seems to be a wise option here, since it is hard to check such a policy on unexpected behaviour. In terms of the simulation setup for robustness and resilience studies, a disrupted simulation start state can be used with a negative reward for every time step it is in a disrupted state. The goal of the MARL algorithm would then be to 'solve' the disruption as quick as possible and get the system back to its normal state. Variants of such a reward function could be explored as well, for example taking the impact of the used actions into account. New disruptions have to be added to the simulation environment to make sure it is realistic and that it covers more than delays only, i.e. a malfunctioning bus etc. Proper input data is needed to model disruptions like this.

Bibliography

[1] https://docs.ray.io/en/master/rllib.html.

[2] https://gitlab.com/team-simpy/simpy.

[3] https://nacto.org/publication/urban-street-design-guide/intersection-design-elements/traffic-signals/signal-cycle-lengths/.

[4] H.Z. Aashtiani and H. Iravani. Application of dwell time functions in transit assignment model. *Transportation Research Record*, 2002.

[5] B. Abdulhai and L. Kattan. Reinforcement learning: Introduction to theory and potential for transport applications. *Canadian Journal of Civil Engineering*, 2003.

[6] F. Alesiani and K. Gkiotsalitis. Reinforcement learning-based bus holding for high-frequency services. 2018.

[7] K. Ampountolas and M. Kring. Mitigating bunching with bus-following models and bus-to-bus cooperation. 2015.

[8] J.J. Bartholdi and D.D. Eisenstein. A self-coördinating bus route to resist bus bunching. *Transportation Research Part B: Methodological*, 2012.

[9] S.J. Berrebi, E. Hans, N. Chiabaut, J.A. Laval, L. Leclercq, and K.E. Watkins. Comparing bus holding methods with and without real-time predictions. *Transportation Research Part C: Emerging Technologies*, 2018.

[10] O. Cats, A. Larijani, H.N. Koutsopoulos, and W. Burghout. Impacts of holding control strategies on transit performance: Bus simulation model analysis. *Transportation Research Record*, 2011.

[11] H. Ceylan and M.G.H. Bell. Traffic signal timing optimisation based on genetic algorithm approach, including drivers' routing. *Transportation Research Part B: Methodological*, 2004.

[12] P. Chandrasekar, R.L. Cheu, and H.C. Chin. Simulation evaluation of route-based control of bus operations. *Journal of Transportation Engineering*, 2002.

[13] W. Chen, K. Zhou, and C. Chen. Real-time bus holding control on a transit corridor based on multi-agent reinforcement learning. 2016.

[14] A.H.F. Chow, S. Li, and R. Zhong. Multi-objective optimal control formulations for bus service reliability with traffic signals. *Transportation Research Part B: Methodological*, 2017.

[15] F.L. Da Silva and A.H. Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 2019.

[16] C.F. Daganzo. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological*, 2009.

[17] C.F. Daganzo and J. Pilachowski. Reducing bunching with bus-to-bus cooperation. *Transportation Research Part B: Methodological*, 2011.

[18] P. Dai, K. Liu, Q. Zhuge, E.H.-M. Sha, V.C.S. Lee, and S.H. Son. Quality-of-experience-oriented autonomous intersection control in vehicular networks. *IEEE Transactions on Intelligent Transportation Systems*, 2016.

[19] F. Delgado, J.C. Munoz, and R. Giesen. How much can holding and/or limiting boarding improve transit performance? *Transportation Research Part B: Methodological*, 2012.

[20] Y.-J. Deng, X.-H. Liu, X. Hu, and M. Zhang. Reduce bus bunching with a real-time speed control algorithm considering heterogeneous roadway conditions and intersection delays. *Journal of Transportation Engineering Part A: Systems*, 2020.

[21] K. Dresner and P. Stone. A multiagent approach to autonomous intersection management. *Journal of Artificial Intelligence Research*, 2008.

[22] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): Methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 2013.

[23] M. Estrada, J. Mensión, J.M. Aymamí, and L. Torres. Bus control strategies in corridors with signalized intersections. *Transportation Research Part C: Emerging Technologies*, 2016.

[24] W. Fan and R.B. Machemehl. Characterizing bus transit passenger waiting times. 2002.

[25] Q. Gao, S. Zhang, G. Chen, and Y. Du. Two-way cooperative priority control of bus transit with stop capacity constraint. *Sustainability (Switzerland)*, 2020.

[26] R. Gao, Z. Liu, J. Li, and Q. Yuan. Cooperative traffic signal control based on multi-agent reinforcement learning. *Communications in Computer and Information Science*, 2020.

[27] I. Grondman, L. Busoniu, G.A.D. Lopes, and R. Babuška. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews*, 2012.

[28] E. Hans, N. Chiabaut, and L. Leclercq. Investigating the irregularity of bus routes: Highlighting how underlying assumptions of bus models impact the regularity results. *Journal of Advanced Transportation*, 2015.

[29] J.B. Ingvardson, O.A. Nielsen, S. Raveau, and B.F. Nielsen. Passenger arrival and waiting time distributions dependent on train service frequency and station characteristics: A smart card data analysis. *Transportation Research Part C: Emerging Technologies*, 2018.

[30] Y. Jiang, S. Li, and D.E. Shamo. A platoon-based traffic signal timing algorithm for major-minor intersection types. *Transportation Research Part B: Methodological*, 2006.

[31] L.M. Kieu and C. Cai. Stochastic collective model of public transport passenger arrival process. *IET Intelligent Transport Systems*, 2018.

[32] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis. Multiagent reinforcement learning for urban traffic control using coordination graphs. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2008.

[33] W. Leong, K. Goh, S. Hess, and P. Murphy. Improving bus service reliability: The singapore experience. *Research in Transportation Economics*, 2016.

[34] M. Li, X. Zhou, and N.M. Rouphail. Quantifying travel time variability at a single bottleneck based on stochastic capacity and demand distributions. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 2017.

[35] M.-T. Li, F. Zhao, L.-F. Chow, H. Zhang, and S.-C. Li. Simulation model for estimating bus dwell time by simultaneously considering numbers of disembarking and boarding passengers. *Transportation Research Record*, 2006.

[36] H Liu, A Skabardonis, and W Zhang. A Dynamic model for adaptive bus signal priority. *TRB 2003 Annual Meeting*, 2003.

[37] Lynxx. PvA Lynxx Schiphol. Technical report.

[38] V. Mnih, K. Kavukcuoglu, D. Silver, A.A. Rusu, J. Veness, M.G. Bellemare, A. Graves, M. Riedmiller, A.K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[39] J. Nzouonta, N. Rajgure, G. Wang, and C. Borcea. Vanet routing on city roads using real-time vehicular traffic information. *IEEE Transactions on Vehicular Technology*, 2009.

[40] A OroojlooyJadid and D Hajinezhad. A review of cooperative multi-agent deep reinforcement learning, 2019.

[41] L. Panait and S. Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 2005.

[42] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev. Time limits in reinforcement learning. 2018.

[43] T. Rashid, M. Samvelyan, C.S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. 2018.

[44] Y. Rizk, M. Awad, and E.W. Tunstel. Decision making in multiagent systems: A survey. *IEEE Transactions on Cognitive and Developmental Systems*, 2018.

[45] H.R. Sayarshad and J.Y.J. Chow. Survey and empirical evaluation of nonhomogeneous arrival process models with taxi data. *Journal of Advanced Transportation*, 2016.

[46] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *Proximal Policy Optimization Algorithms*, 2017.

[47] D. Silver, A. Huang, C.J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

[48] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. Van Den Driessche, T. Graepel, and D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 2017.

[49] I.I. Sirmatel and N. Geroliminis. Mixed logical dynamical modeling and hybrid model predictive control of public transport operations. *Transportation Research Part B: Methodological*, 2018.

[50] K.K. Srinivasan, A.A. Prakash, and R. Seshadri. Finding most reliable paths on networks with correlated and shifted log-normal travel times. *Transportation Research Part B: Methodological*, 2014.

[51] L. Sun, Q. Meng, and Z. Liu. Transit assignment model incorporating bus dwell time. *Transportation Research Record*, 2013.

[52] P. Sunehag, G. Lever, A. Gruslys, W.M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J.Z. Leibo, K. Tuyls, and T. Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. 2018.

[53] Richard S. Sutton and Andrew G. Barto. Reinforcement Learning, Second Edition: An Introduction - Complete Draft. *The MIT Press*, 2018.

[54] N. Uno, F. Kurauchi, H. Tamura, and Y. Iida. Using bus probe data for aanalysis of travel time variability. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, 2009.

[55] N. van Oort, J.W. Boterman, and R. van Nes. The impact of scheduling on service reliability: Trip-time determination and holding points in long-headway services. *Public Transport*, 2012.

[56] J. Wang and L. Sun. Dynamic holding control to avoid bus bunching: A multi-agent deep reinforcement learning framework. *Transportation Research Part C: Emerging Technologies*, 2020.

[57] W. Xia, H. Li, and B. Li. A control strategy of autonomous vehicles based on deep reinforcement learning. 2016.

[58] Y. Xuan, J. Argote, and C.F. Daganzo. Dynamic bus holding strategies for schedule reliability: Optimal linear control and performance analysis. *Transportation Research Part B: Methodological*, 2011.

[59] B. Yang and C. Monterola. Efficient intersection control for minimally guided vehicles: A self-organised and decentralised approach. *Transportation Research Part C: Emerging Technologies*, 2016.

[60] H. Yang and S. Yagar. Traffic assignment and signal control in saturated road networks. *Transportation Research Part A*, 1995.

Appendices

A Scenario Setups

A.1 Scenario 1 - Idealised

Table 13: An overview of the settings used for scenario 1 (ID).

Property	**Value**
Number of bus agents	6
Capacity busses	Infinite
Capacity stops	1
Edges with traffic lights	None
Arrival function nodes	Poisson
Destination distribution nodes	10% for the coming 10 stops
Passenger arrival rate	1 passenger per minute for all stops
Boarding rate	3 seconds per passenger
Alighting rate	1.8 seconds per passenger
Dwell time function nodes	Maximum boarding and alighting time Deterministic
Travel time edges	Type: Deterministic Length: 240 seconds
Observation space bus agents	Size: 1 Value: [backward headway - forward headway]
Observation space traffic light agents	Not present
Action space bus agents	Size: 7 Affects: Holding time Value: [0, 30, 60, 90, 120, 150, 180] seconds
Action space traffic light agents	Not present
End trigger	t = 4000 seconds
Reward function	Moment: End of simulation Based on: Average travel time

A.2 Scenario 2 - Stochastic

Table 14: An overview of the settings used for scenario 2 (ST).

Property	**Value**
Number of bus agents	6
Capacity busses	Infinite
Capacity stops	1
Edges with traffic lights	All
Arrival function nodes	Poisson
Destination distribution nodes	10% for the coming 10 stops
Passenger arrival rate	1 passenger per minute for all stops
Boarding rate	3 seconds per passenger
Alighting rate	1.8 seconds per passenger
Dwell time function nodes	Maximum boarding and alighting time Stochastic
Travel time edges	Type: Lognormal Min: 120 seconds Mean: 240 seconds Max: 480 seconds
Observation space bus agents	Size: 1 Value: [backward headway - forward headway]
Observation space traffic light agents	Size:1 Value: [backward headway - forward headway]
Action space bus agents	Size: 7 Affects: Holding time Value: [0, 30, 60, 90, 120, 150, 180] seconds
Action space traffic light agents	Size: 1 Affects: Delay bus at traffic lights Value: [+0] seconds
End trigger	t = 4000 seconds
Reward function	Moment: End of simulation Based on: Average travel time

A.3 Scenario 3 - Stochastic - Traffic Light Control

Table 15: An overview of the settings used for scenario 3 (TL).

Property	Value
Number of bus agents	6
Capacity busses	Infinite
Capacity stops	1
Edges with traffic lights	All
Arrival function nodes	Poisson
Destination distribution nodes	10% for the coming 10 stops
Passenger arrival rate	1 passenger per minute for all stops
Boarding rate	3 seconds per passenger
Alighting rate	1.8 seconds per passenger
Dwell time function nodes	Maximum boarding and alighting time Stochastic
Travel time edges	Type: Lognormal Min: 120 seconds Mean: 240 seconds Max: 480 seconds
Observation space bus agents	Size: 1 Value: [backward headway - forward headway]
Observation space traffic light agents	Size:1 Value: [backward headway - forward headway]
Action space bus agents	Size: 7 Affects: Holding time Value: [0, 30, 60, 90, 120, 150, 180] seconds
Action space traffic light agents	Size: 1 Affects: Delay bus at traffic lights Value: [-10, +0, +10] seconds
End trigger	t = 4000 seconds
Reward function	Moment: End of simulation Based on: Average travel time

A.4 Scenario 4 - Deterministic - Heterogeneous

Table 16: An overview of the settings used for scenario 4 (DH).

Property	Value
Number of bus agents	6
Capacity busses	Infinite
Capacity stops	1
Edges with traffic lights	None
Arrival function nodes	Poisson
Destination distribution nodes	10% for the coming 10 stops
Passenger arrival rate stops	Unit: Passengers per minute [0.5, 0.5, 0.8, 1, 1, 3, 4, 2, 1, 0.75, 0.6, 0.5]
Boarding rate	3 seconds per passenger
Alighting rate	1.8 seconds per passenger
Dwell time function nodes	Maximum boarding and alighting time Deterministic
Travel time edges	Type: Deterministic Length: [360, 360, 300, 240, 240, 120, 90, 150, 240, 240, 240, 300] seconds
Observation space bus agents	Size: 2 Value: [backward headway - forward headway, position]
Observation space traffic light agents	Not present
Action space bus agents	Size: 7 Affects: Holding time Value: [0, 30, 60, 90, 120, 150, 180] seconds
Action space traffic light agents	Not present
End trigger	t = 4000 seconds
Reward function	Moment: End of simulation Based on: Average travel time

A.5 Scenario 5 - Realistic

Table 17: An overview of the settings used for scenario 5 (RE).

Property	Value
Number of bus agents	6
Capacity busses	Infinite
Capacity stops	1
Edges with traffic lights	[0, 0, 1, 0, 0, 1, 1, 1, 0, 1, 0, 0]
Arrival function nodes	Poisson
Destination distribution nodes	10% for the coming 10 stops
Passenger arrival rate stops	Unit: Passengers per minute [0.5, 0.5, 0.8, 1, 1, 3, 4, 2, 1, 0.75, 0.6, 0.5]
Boarding rate	3 seconds per passenger
Alighting rate	1.8 seconds per passenger
Dwell time function nodes	Maximum boarding and alighting time Stochastic
Travel time edges	Type: Lognormal Min: [180, 180, 150, 120, 120, 60, 45, 75, 120, 120, 120, 150] seconds Mean: [360, 360, 300, 240, 240, 120, 90, 150, 240, 240, 240, 300] seconds Max: [720, 720, 600, 480, 480, 240, 180, 300, 480, 480, 480, 600] seconds
Observation space bus agents	Size: 2 Value: [backward headway - forward headway, position]
Observation space traffic light agents	Size:1 Value: [backward headway - forward headway]
Action space bus agents	Size: 7 Affects: Holding time Value: [0, 30, 60, 90, 120, 150, 180] seconds
Action space traffic light agents	Size: 1 Affects: Delay bus at traffic lights Value: [-10, +0, +10] seconds
End trigger	t = 4000 seconds
Reward function	Moment: End of simulation Based on: Average travel time

B Data Analysis

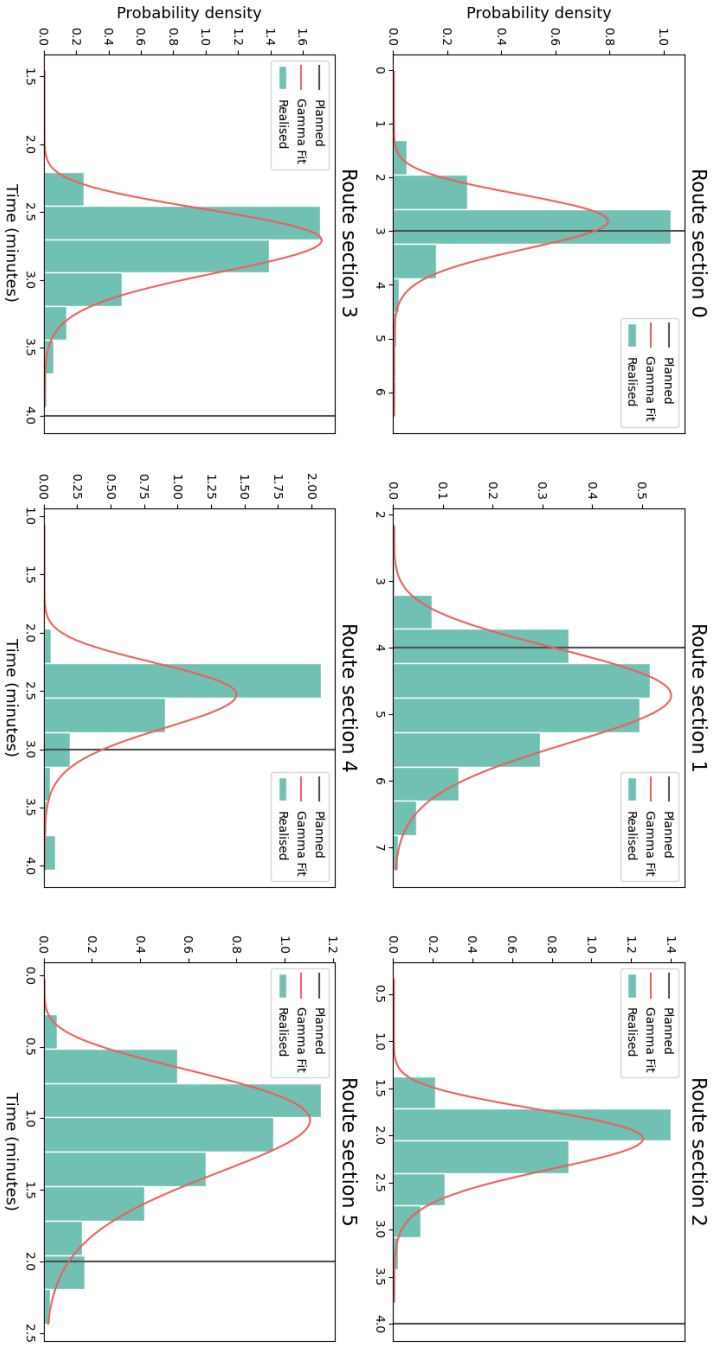

Figure 26: The observed driving time distribution on route segments 0 to 5 plotted with a log normal fit and the planned driving time by the operator.

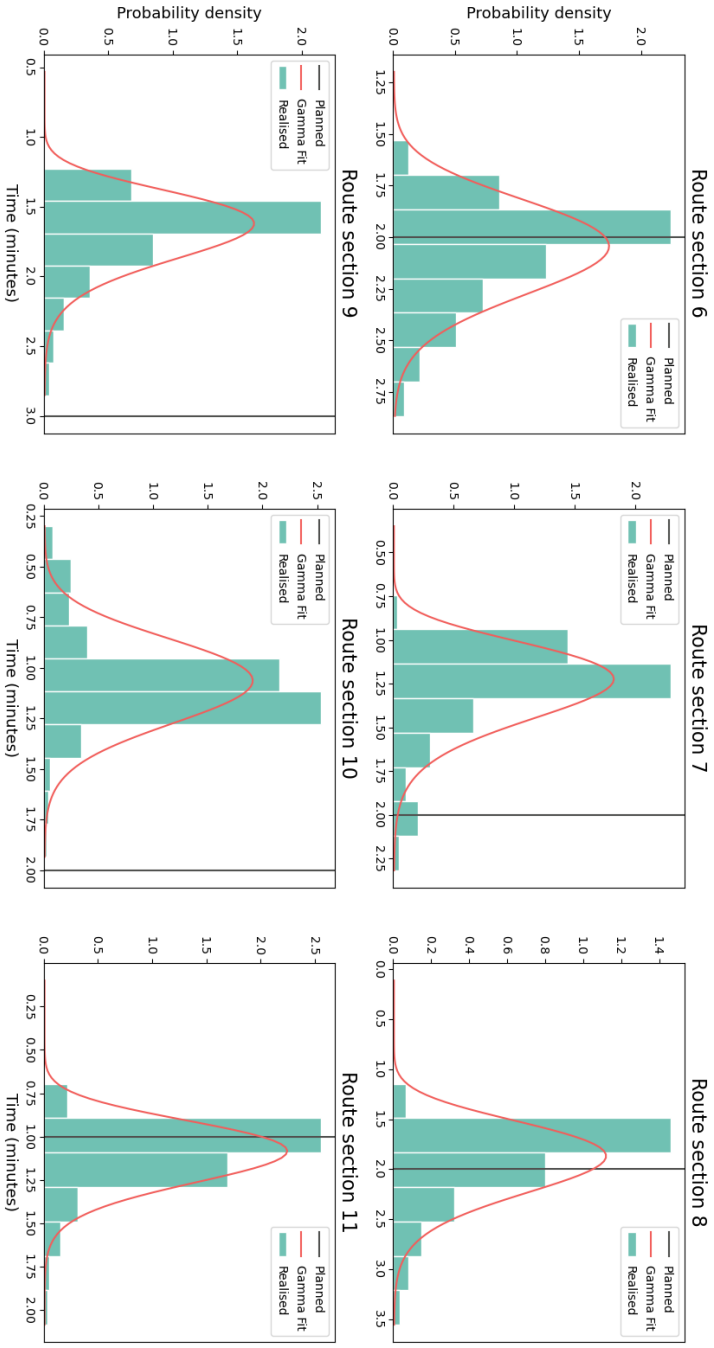

Figure 27: The observed driving time distribution on route segments 6 to 11 plotted with a log normal fit and the planned driving time by the operator.

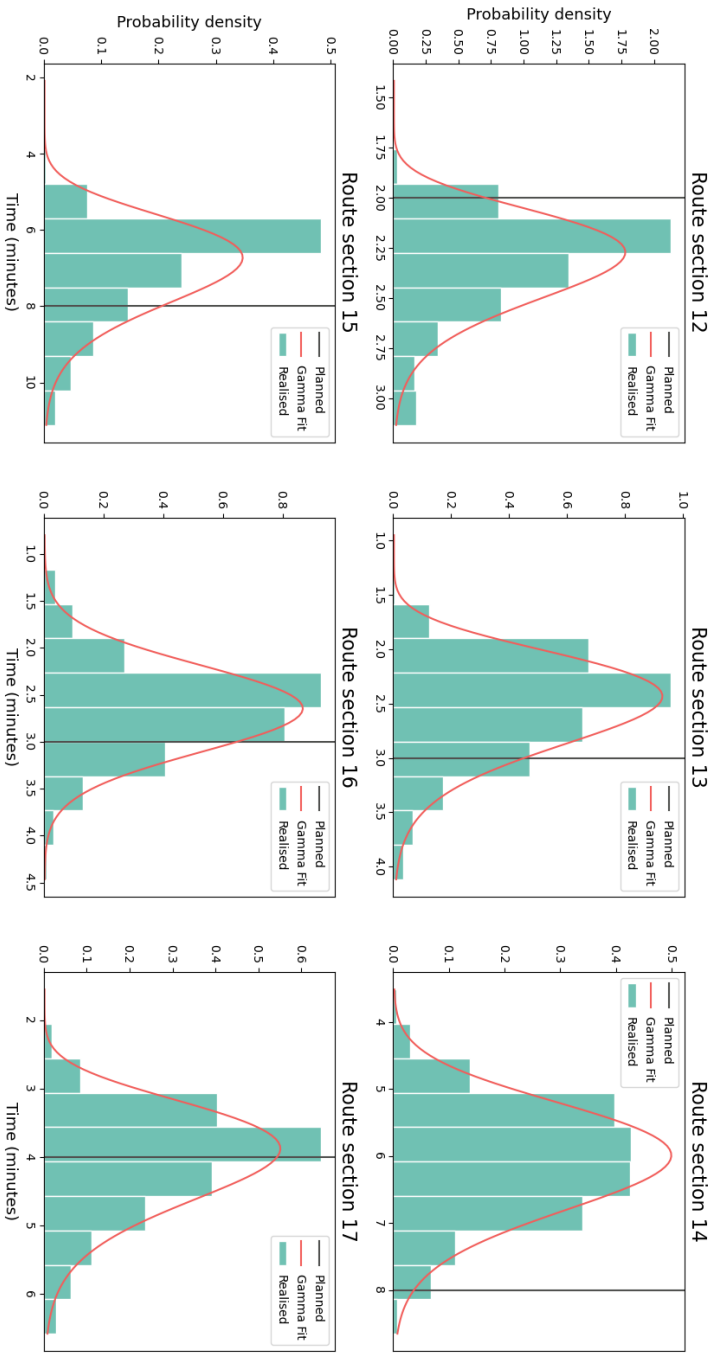

Figure 28: The observed driving time distribution on route segments 12 to 17 plotted with a log normal fit and the planned driving time by the operator.

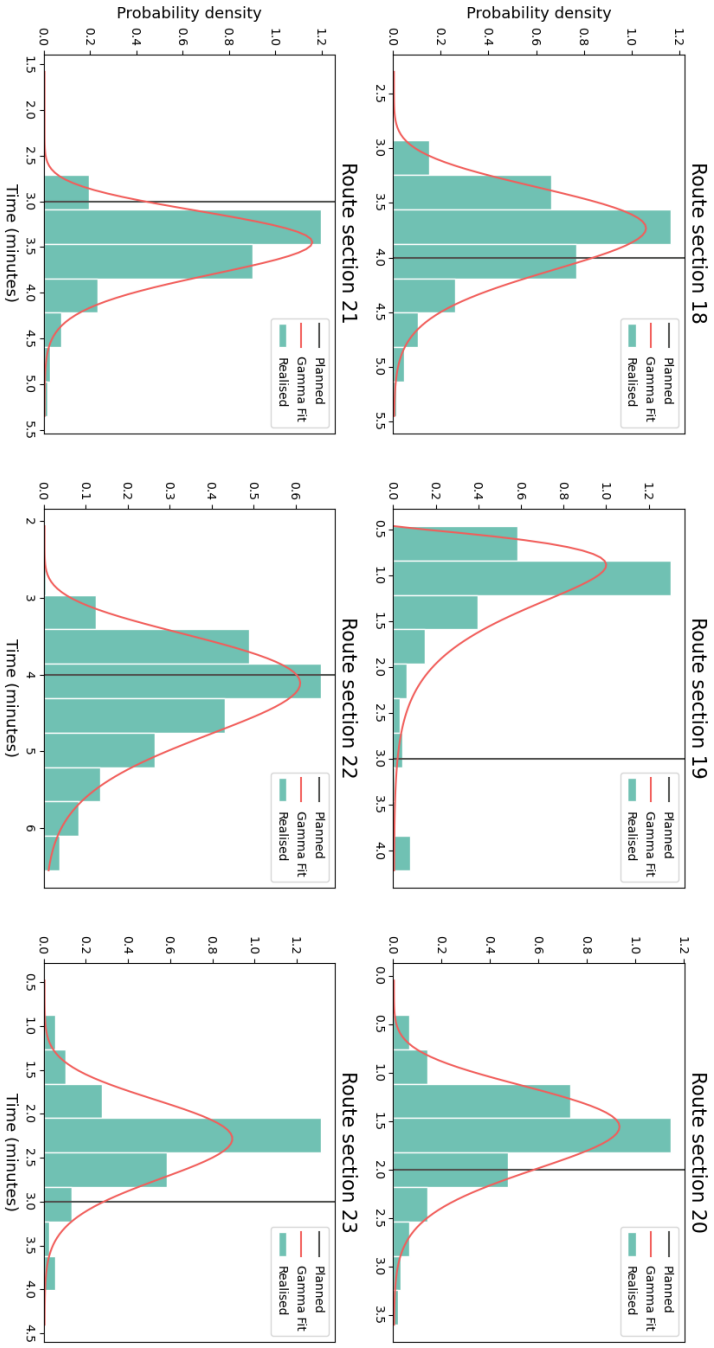

Figure 29: The observed driving time distribution on route segments 18 to 23 plotted with a log normal fit and the planned driving time by the operator.

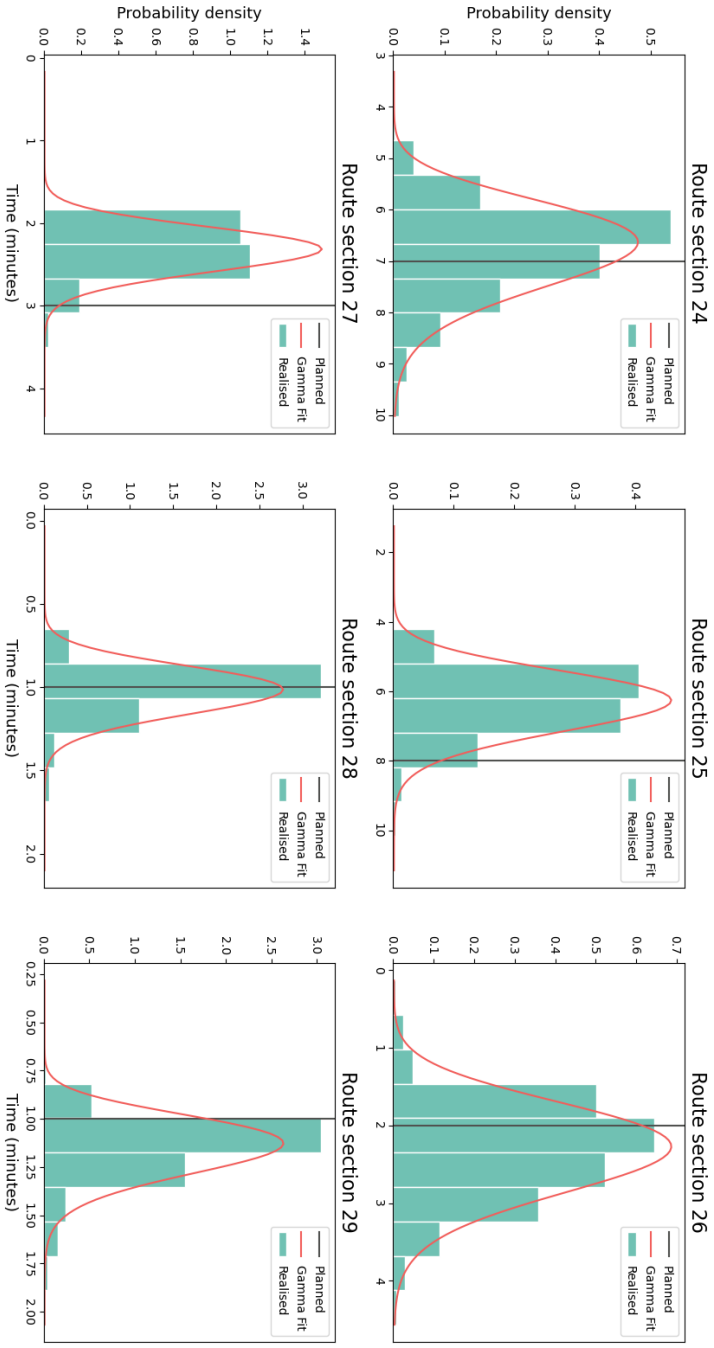

Figure 30: The observed driving time distribution on route segments 24 to 29 plotted with a log normal fit and the planned driving time by the operator.

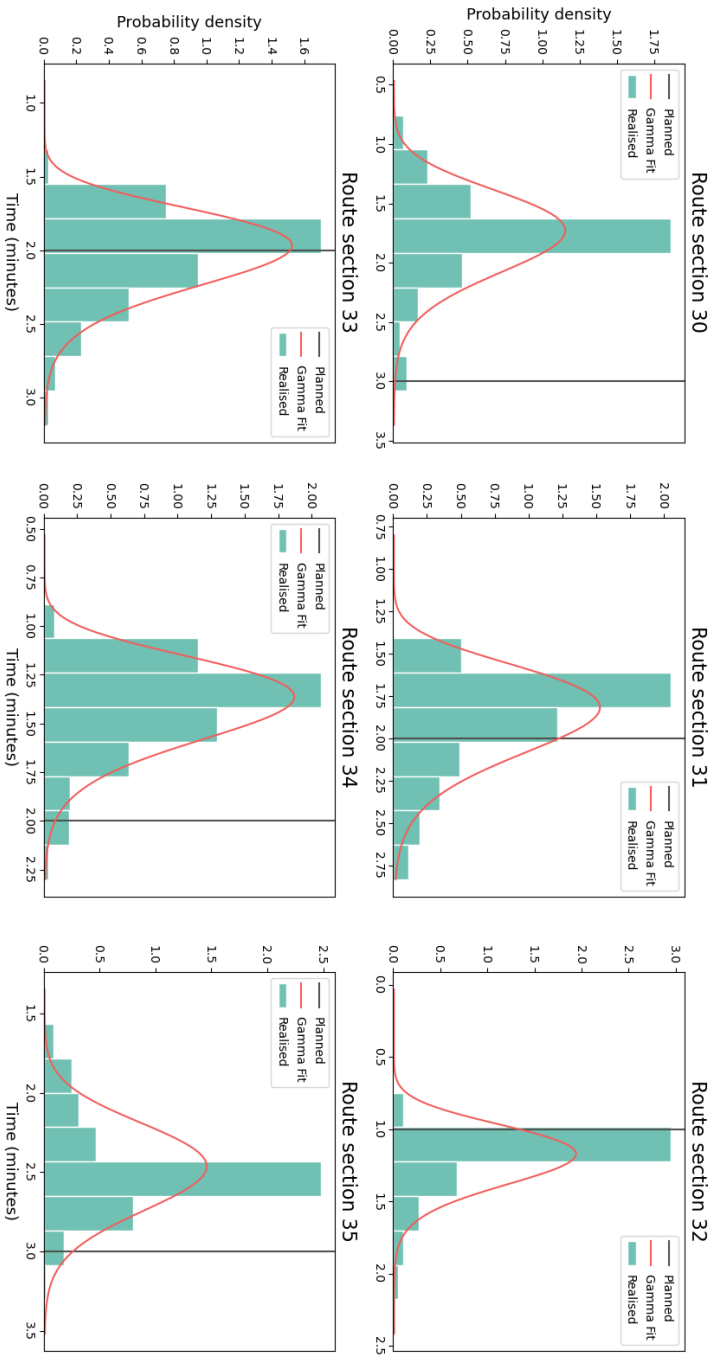

Figure 31: The observed driving time distribution on route segments 30 to 35 plotted with a log normal fit and the planned driving time by the operator.