OPTIMAL TRAFFIC LIGHT CONTROL

Performance Evaluation Applying A General Evaluation Methodology

A. J. Hillebrink

09/10/2018

TU Delft

OPTIMAL TRAFFIC LIGHT CONTROL

Performance Evaluation Applying A General Evaluation Methodology

by

A. J. Hillebrink

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 22 October 2018 at 11:00 hours.

Student number:	4092732	
Project duration:	1 August, 2017 – 18 August, 2018	
Thesis committee:	Prof. dr. ir. B. De Schutter,	TU Delft
	Dr. ir. A. Hegyi,	TU Delft, supervisor
	Dr. A. Czechowski	Dynniq, supervisor
	Dr. A. Dabiri	TU Delft
	Dr. M. Lu	Dynniq

An electronic version of this thesis is available at `http://repository.tudelft.nl/`.

TU Delft Delft University of Technology

dynniq energising mobility

Summary

The ongoing increase in urbanization and traffic congestion creates an urgent need to operate our transportation systems with maximum efficiency. Traffic signal control optimization is considered one of the main ways to solve traffic problems in urban networks. By optimizing the signal plans on a network wide scale, signal control efficiency can be increased, resulting in reduced vehicle delay and decreased number of stops, thus improving the traffic flow.

In publications in the field of intelligent transportation systems, a vast amount of different optimal traffic light control methods is described. With new optimization methods being developed, it is important to know their performance compared to similar methods. Such a comparison is only possible if the same performance evaluation methodology is applied to all these methods. This approach has not yet been applied in the field of intelligent transport systems. Since new or improved methods are normally evaluated using a self-composed evaluation methodology. However this increases the subjectivity in the performance evaluation, as an evaluation methodology consists of many different elements, such as: simulation software, network topology, simulation randomness, evaluation metrics, performance indicators and scenario design. As concluded in [29] a large variety of different evaluation methodologies are applied in literature, which makes them hardly comparable to one another. In addition, designed methods described in these publications are not publicly available, therefore as one wants to evaluate the potential of two methods described in different publications, it is necessary to implement both methods by themselves and perform an evaluation based on a equal methodology. A self-implementation comes with additional difficulties, as details may have been omitted by the authors or assumptions could have been made in order to simplify the traffic light control problem.

The first research objective of this thesis is to determine the design criteria of the general evaluation methodology, such that a general evaluation methodology can be designed in order to evaluate the performance of different optimal traffic light control methods objectively. In order to define these design specifications, the purpose and goal of performance evaluation has to be clear first, as this is the main focus point of the methodology. Secondly, a closer look is taken at the current design of evaluation methodologies as described in various literature publications. In order to asses the various difference in the evaluation methodology amongst them. Thirdly the different elements involved in establishing an evaluation methodology are extracted from these different evaluation methodologies. The research objective for the evaluation methodology can be summarized into the following main research question:
RQ1: What are the design specifications for a general evaluation framework, such that performance evaluation of different traffic signal optimization methods is conducted objective and unambiguous?
The following sub research questions are defined to contribute towards answering the research question 1:

- What is the purpose and goal of performance evaluation?
- How are evaluation methodologies currently designed in literature?
- What are the different elements involved in establishing an evaluation methodology?

The second objective is to evaluate the potential for practical implementation of two different mathematical traffic signal control techniques. The two techniques are chosen in this thesis as dynamic programming and Q-learning. As explained in the problem statement, an objective evaluation is only possible if both methods are evaluated using the same evaluation methodology. Therefore a methodology is designed according to the principles defined in the first research objective. Since both methods are not publicly available, they need to be implemented according to the method description described in the source publication. In addition to this research objective the various control approaches and mathematical techniques in traffic signal control that can be evaluated using the general evaluation methodology are described. This research objective can be summarized into the following main research question:
RQ2: How can the general evaluation methodology contribute towards evaluating the potential of different mathematical traffic light control techniques?
The following sub research questions are defined to contribute towards answering the research question 2:

- What are the different control approaches for traffic light control?
- What are the different mathematical techniques to improve traffic light control?

The scientific relevance of this thesis is to provide relevant arguments as to why a general evaluation methodology is of additional relevance in improving the performance evaluation of traffic signal control techniques. And how this general evaluation methodology should be formulated in general, such that it is an objective and unambiguous performance evaluation methodology. First of all, a literature study about the evaluation methodology is conducted. This literature study provides the purpose and goal of performance evaluation, which are providing a qualitative method of evaluating. Secondly it provides a comparative measure of the algorithm's performance compared to similar algorithms, assuming a similar methodology is used. Based on these requirements the evaluation methodology in the literature is evaluated. Based on this evaluation it becomes clear that different publications are hardly comparable due to variation in scenario set-ups and performance indicators. It becomes clear that the current approach, as performed in current publications in the field of intelligent transportation systems makes little sense as self-defined evaluation methodologies are used instead of a general evaluation methodology. A general evaluation methodology should also remove the subjectivity that arises in self-defined evaluation methodologies and in the assumptions made in order to simplify the traffic light control problem. In this thesis the different required components to design a general evaluation methodology are given, such as: a list of accepted simulation software packages, different network topologies, unambiguous measurement metrics, scenario descriptions and simulation set-ups. These components should be included into guidelines to formulate a general evaluation methodology in future research.

An additional literature study about traffic signal control approaches and mathematical techniques is performed. This literature study provides an overview of the different control approaches in traffic signal control and the different mathematical techniques that are designed in order to improve optimize the traffic light plans, such as: dynamic programming, fuzzy logic, reinforcement learning, heuristics-based algorithms and genetic algorithms.

The general evaluation methodology described in this thesis is applied in the evaluation by simulation of two different methods for improving traffic light control. These methods had to be self-implemented as they are not made publicly available. A self-implementation is a difficult process, as the level of detail varies among publication, critical information is not provided or the method itself has to be modified due to assumptions made by the author. If publications in the field of intelligent transport systems had been using a general evaluation methodology, a self-implementation is not necessary, since results could be compared straight away.

The research approach applied in this thesis consists of three separate parts, each with their own contribution towards answering the research objective. The literature study is the first part and is divided into two different chapters. The first chapter addresses the evaluation methodology by focussing on the purpose and goal of performance evaluation and to show the various elements involved in establishing a general evaluation methodology. The second chapter focusses on the different methods for optimal traffic light control, as described in different literature publications. This information is used to provide an overview of the various methods and approaches in addressing the optimal signal control problem that could be evaluated through a general evaluation methodology. Due to the scope of this thesis two methods are selected from this list, which are a dynamic programming and a Q-learning method. Both are described in depth, theoretically as well as in pseudo code and programmed in Python. These Python scripts are required in the third part of the methodology, in which both optimal traffic light control methods are used to optimize the traffic light control during a microscopic simulation. In the third part, the performance of both methods is evaluated by simulation according to an evaluation methodology that is designed based on the guidelines of the general evaluation methodology. Due to time constraints some of the described guidelines had to be simplified. These modifications are described first. The evaluation by simulation is carried out in the microscopic simulation road traffic simulation package SUMO. A microscopic simulation is used, as this type of model individual vehicle behaviour and interaction is modelled, as well that intersection operation could be affected by the conditions of an adjacent roadway. This cannot be captures with macroscopic models. Network modelling and simulations are performed in the microscopic simulation package SUMO, as this software is open-source and highly portable. First simulations are performed on a test-bed intersection, consisting of one isolated intersection. On this isolated intersection both methods are tested and benchmarked compared to a fixed-timing control structure. These simulations are required in order to adjust the different parameters used in both methods, as they affect the methods performance. Next performance evaluation simulations are performed on two

different three intersection networks. In addition to the performance evaluation a green wave formation and a spill-back scenario are tested in order to evaluate the method's performance with regard to these traffic phenomena.

In this thesis, an evaluation methodology was designed according to the guidelines regarding a general evaluation methodology. Due to limitations in the available time concessions had to be made in the comprehensiveness of the evaluation methodology applied in this thesis. The goal of this methodology was to provide an equal evaluation to evaluate the potential of a traffic light control method based on dynamic programming and Q-learning.

Three different networks were designed in SUMO, the first network is an isolated intersection (test-bed intersection). This network was initially used to fine-tune the different parameters, such as the interval boundaries and queue detection speed threshold. Also did this network provide insight in the method's performance on this type of network. A fixed-timing structure was also implemented to provide an additional benchmark. The lather two networks have been designed to evaluate the method's performance on two different networks containing three intersections.

On network 1 the distance between the intersections is 450 m and queue detector length is 300 m, whereas on network 2 the distance between the intersections is reduced to 250 m and queue detector length is 150 m. On both network the same adaptability test is performed, except this time the two methods are only quantitatively compared to each other, so no fixed timing structure was used.

The performance evaluation on these networks is evaluated under various traffic loads, ranging from a demand factor of 0,5 up to 1,5, where a demand factor of 1,0 described the typical volumes during a rush-hour. The metric used in this evaluation is the average cumulative delay, which is measured as the time a vehicle is present in a queue.

Additional scenarios were designed to further elaborate on the performance of both traffic light control methods considering specific traffic phenomena. The two scenarios included into the evaluation methodology are a green wave and a spill-back scenario. The goal of the green-wave formation scenario was to evaluating the ability of the method in forming a green-wave along the main corridor. Traffic demand was limited to solely the North-South and South-North route of the network, as the formation of the green-wave had to be the known optimal solution. Obtaining this known solution is considerably more complex if other traffic flows are also included into the scenario. The goal of the spill-back scenario was to evaluate the ability of the method to function under very-high traffic volumes at which the formation of spill-back on the main corridor is likely to happen. The demand factors during this scenario were 1,75 and 2,00 respectively.

Based on the results of the evaluation performed in this chapter, one cannot conclusive point one optimization method as favourable over the other, as performance compared to each other varied over the different simulations. The Q-learning method outperformed the dynamic programming method on the isolated intersection for the entire range of demand factors. However a fixed timing structure, designed on a traffic demand of 1,0, was outperforming both the dynamic programming and the Q-learning method for demand factors of 1,5. On the two multi intersection networks the results varied between the two methods. The Q-learning method (after additional network specific training) was outperforming the dynamic programming method on network 1 for traffic demand factors of 0,9 and higher, however it was outperformed by the dynamic programming method on network 2 for all demand factors. Also in the additional scenario simulations the dynamic programming method was outperforming the Q-learning method for both the green wave and spill-back scenarios.

These results show that neither of the two methods was superior to the other, as well as some remarkable outcomes. The dynamic programming method was unable to find the optimal solution, however it should do so by design. The converged policy of the Q-learning method did not result in optimal performance during high traffic demand or on the multi-intersection networks. These results could originate to the design of the method or from modifications in the required self-implementation of the method. As there are many variables, parameters, assumptions involved in the design of a method, an identical self-implementation is not possible. Also could critical information be omitted by the author. These modifications could have led to potential alterations in performance. If a general evaluation methodology was applied in the field of intelligent transport systems, a self-implementation would no longer be required as the method's performance could be looked up in the result database. Therefore would a general evaluation methodology be of additional value in qualitatively assessing the performance of traffic signal control methods and should be further researched and broadly implemented in the field of intelligent transportation systems, in order to provide a comparative measure of a method's performance.

Preface

Before you lies the thesis "Optimal traffic light control - Performance Evaluation Applying A General Evaluation Methodology", the basis of which is a performance evaluation of a dynamic programming and a Q-learning optimal traffic signal control method. It has been written to fulfil the graduation requirements of the Master degree in Civil Engineering at the Technical University of Delft.

I was engaged in researching and writing this thesis from August 2017 to August 2018. The project subject was proposed by Dynniq, where I undertook an internship. I would like to thank the committee for their feedback as received in the progress reports, and from the discussions we had. Thank you for spending your valuable time on this thesis.

Alwin Hillebrink

Dordrecht, 09/10/2018

Contents

1 Introduction

The ongoing increase in urbanization and traffic congestion creates an urgent need to operate our transportation systems with maximum efficiency. Traffic signal control optimization is considered one of the main ways to solve traffic problems in urban networks. By optimizing the signal plans on a network wide scale, signal control efficiency can be increased, resulting in reduced vehicle delay and decreased number of stops, thus improving the traffic flow.

Various methods to optimize traffic signal control have been developed and described in research papers. All of them claim to improve performance compared to a self-chosen benchmark with a simulation methodology defined by the authors themselves. This leads to the problem statement of this thesis.

1.1. Problem statement

In publications in the field of intelligent transportation systems, a vast amount of different optimal traffic light control methods is described. With new optimization methods being developed, it is important to know their performance compared to similar methods. Such a comparison is only possible if the same performance evaluation methodology is applied to all these methods. This approach has not yet been applied in the field of intelligent transport systems. Since new or improved methods are normally evaluated using a self- composed evaluation methodology. However this increases the subjectivity in the performance evaluation, as an evaluation methodology consists of many different elements, such as: simulation software, network topology, simulation randomness, evaluation metrics, performance indicators and scenario design. As concluded in [29] a large variety of different evaluation methodologies are applied in literature, which makes them hardly comparable to one another. In addition, designed methods described in these publications are not publicly available, therefore as one wants to evaluate the potential of two methods described in different publications, it is necessary to implement both methods by themselves and perform an evaluation based on a equal methodology. A self-implementation comes with additional difficulties, as details may have been omitted by the authors or assumptions could have been made in order to simplify the traffic light control problem.

1.2. Research objective

The first research objective is to determine the design criteria of the general evaluation methodology, such that a general evaluation methodology can be designed in order to evaluate the performance of different optimal traffic light control methods objectively. In order to define these design specifications, the purpose and goal of performance evaluation has to be clear first, as this is the main focus point of the methodology. Secondly, a closer look is taken at the current design of evaluation methodologies as described in various literature publications. In order to asses the various difference in the evaluation methodology amongst them. Thirdly the different elements involved in establishing an evaluation methodology are extracted from the different evaluation methodologies, as well as found in literature. The research objective for the evaluation methodology can be summarized into the following main research question:

RQ1: What are the design specifications for a general evaluation framework, such that performance evaluation of different traffic signal optimization methods is conducted objective and unambiguous?

The following sub research questions are defined to contribute towards answering the research question 1:

- What is the purpose and goal of performance evaluation?
- How are evaluation methodologies currently designed in literature?
- What are the different elements involved in establishing an evaluation methodology?

The second objective is to evaluate the potential for practical implementation of two different mathematical traffic signal control techniques. The two techniques are chosen in this thesis as dynamic programming and Q-learning. As explained in the problem statement, an objective evaluation is only possible if both methods are evaluated using the same evaluation methodology. Therefore a methodology is designed according to the principles defined in the first research objective. Since both methods are not publicly available, they need to be implemented according to the method description described in the source publication. In addition to this research objective the various control approaches and mathematical techniques in traffic signal control that can be evaluated using the general evaluation methodology are described. This research objective can be summarized into the following main research question:
RQ2: How can the general evaluation methodology contribute towards evaluating the potential of different mathematical traffic light control techniques?
The following sub research questions are defined to contribute towards answering the research question 2:

- What are the different control approaches for traffic light control?
- What are the different mathematical techniques to improve traffic light control?

1.3. Scientific relevance

In this section the scientific relevance of this thesis is described and can be found in both literature studies as well as in the evaluation by simulation based on a general evaluation methodology. In this thesis a literature study about the evaluation methodology is conducted. This literature study provides the purpose and goal of performance evaluation, which are providing a qualitative method of evaluating. Secondly it provides a comparative measure of the algorithm's performance compared to similar algorithms, assuming a similar methodology is used. Based on these requirements the evaluation methodology in the literature is evaluated. This shows different publications are hardly comparable due to different scenario set-ups and performance indicators. It becomes clear that the current approach, as performed in current publications in the field of intelligent transportation systems makes little sense as self-defined evaluation methodologies are used instead of a general evaluation methodology. A general evaluation methodology should also take away the subjectivity that arises in self-defined evaluation methodologies and in the assumptions made in order to simplify the traffic light control problem. In this thesis the different required components to design a general evaluation methodology are given, such as: a list of accepted simulation software packages, different network topologies, unambiguous measurement metrics, scenario descriptions and simulation set-ups. These components should be included into guidelines to formulate a general evaluation methodology in future research.

An additional literature study about traffic signal control approaches and mathematical techniques is performed. This literature study provides an overview of the different approaches in traffic signal control and the different mathematical techniques that are designed in order to improve the performance of traffic signal control, such as: dynamic programming, fuzzy logic, reinforcement learning, heuristics-based algorithms and genetic algorithms.

The general evaluation methodology described in this thesis is applied in the evaluation by simulation of two different methods for improving traffic light control. These methods had to be self-implemented as they are not made publicly available. A self-implementation is a difficult process, as the level of detail varies among publication, critical information is not provided or the method itself has to be modified due to assumptions made by the author. If publications in the field of intelligent transport systems had been using a general evaluation methodology, a self-implementation is not necessary, as results could be compared straight away.

1.4. Research approach

The research approach applied in this thesis consists of three separate parts, each with their own contribution towards answering the research objective.

The literature study is the first part and is divided into two different chapters. The first chapter addresses the evaluation methodology by focussing on the purpose and goal of performance evaluation and to show the various elements involved in establishing a general evaluation methodology. The second chapter focusses on

the different methods for optimal traffic light control, as described in different literature publications. This information is used to provide an overview of the various methods and approaches in addressing the optimal signal control problem that could be evaluated through a general evaluation methodology.

In the second part, two methods are selected from this list of optimal traffic light control methods. These methods are dynamic programming and Q-learning. Both are described in depth, theoretically as well as in pseudo code and programmed in Python. These Python scripts are required in the third part of the methodology, in which both optimal traffic light control methods are used to optimize the traffic light control during a microscopic simulation.

In the third part, the performance of both methods is evaluated by simulation according to a methodology that is designed based on the guidelines of the general evaluation methodology. Due to time constraints some of the described guidelines had to be simplified. These modifications are described first. The evaluation by simulation is carried out in the microscopic simulation road traffic simulation package SUMO. A microscopic simulation is used, as this type of model individual vehicle behaviour and interaction is modelled, as well that intersection operation could be affected by the conditions of an adjacent roadway. This cannot be captures with macroscopic models. Network modelling and simulations are performed in the microscopic simulation package SUMO, as this software is open-source and highly portable. First simulations are performed on a test-bed intersection, consisting of one isolated intersection. On this isolated intersection both methods are tested and benchmarked compared to a fixed-timing control structure. These simulations are required in order to adjust the different parameters used in both methods, as they affect the methods performance. Next performance evaluation simulations are performed on two different three intersection networks. In addition to the performance evaluation a green wave formation and a spill-back scenario are tested in order to evaluate the method's performance with regard to these traffic phenomena.

1.5. Thesis overview

Chapter 2 gives the goal and purpose of performance evaluation and how such evaluation is performed in the field of image recognition algorithms, as in this field a general evaluation methodology is commonly used, based on clear defined guidelines. In this chapter the literature review of how the evaluation methodology is currently performed by different publications is summarized, followed by a description of the different elements included in an evaluation methodology, which should be described by guidelines in order to propose a general evaluation methodology.

Chapter 3 summarizes the literature review of different control strategies and optimal traffic light control methods. This chapter concludes with a selection of two optimal control methods, dynamic programming and Q-learning.

Chapter 4 describes the design and implementation of a dynamic programming and Q-learning method. The different components of both methods are given such as: state, action and reward.

In chapter 5 the application of a the general evaluation methodology in evaluating the performance of both optimization methods and their results is given. The evaluation methodology is described first, followed by the set-up of the simulation environment. As well as the different simulations performed in the evaluation methodology. Starting with the test-bed intersection on which both methods are tested and variables are fine-tuned in order to improve performance. Subsequently simulations are performed on two different three intersection networks. Simulation results are presented at the end of the chapter.

Chapter 6 gives conclusion, discussion and recommendation of this thesis.

2

Evaluation methodology

When a new method or algorithm is developed in any field, it is important to know its performance compared with already developed approaches. Such comparison is only possible if the same methodology is used across all methods included into this benchmark. This evaluation can be analytical, based on mathematical proof or by simulation. As simulation software is designed to reflect real behaviour as good as possible it is chosen as the focus point for the general evaluation methodology.

An example of a field where a general evaluation methodology is applied, is in visual object recognition software research. A large visual database, called ImageNet is designed, described in[10]. This database consists of more than 1.3 million images and is regarded as the main benchmark for measuring progress in the development in image classification algorithm. Such a general evaluation methodology is not applied in the field of intelligent transport systems, therefore new or improved methods are tested in based on a self-composed simulation methodology in a self-designed simulation network with one of the available simulation methods.

In this chapter, first in section 2.1 the goal and purpose of performance evaluation is examined. Followed by how simulation methodologies are currently designed in literature in section 2.2. Next, in section 2.3 a description of the different components included in an evaluation by simulation are given and at last the conclusion of this chapter is given.

2.1. Performance evaluation

As new methods or algorithms are developed in any field, a portion of the design process is dedicated to testing. As described in [46] algorithm testing has a two-fold purpose, firstly it provides a qualitative method of evaluating. Secondly it provides a comparative measure of the algorithm against similar algorithms, assuming similar criteria are used. One of the greatest caveats is to conceive the criteria used to analyse the results of such tests. Should a criterion be designed that measures sensitivity, robustness, or accuracy?

In the broadest sense, performance testing refers to measure of some required behaviour of an algorithm or method, whether it is achievable accuracy, robustness, or adaptability. This allows the evaluation of benefits as well as the limitations, but often is such testing limited in the scope. As formal process used in performance evaluation, from the establishment of simulation method, testing scenarios, to the design of metrics is absent.

In order to evaluate the performance of different optimal traffic signal control methods objectively, the same metric definition should be used throughout different publications. Finding suitable metrics that provide an objective measure of performance in the evaluation of algorithms is regarded as difficult, due to the different objectives. But there are a few that are quite common, as concluded in [5], such as vehicle delay, number of stops, emissions. A performance metric is a meaningful and computable measure used for quantitatively evaluating the performance of any algorithm or method.

In [18] the purpose of evaluating an algorithm is given, as to understand its behaviour in dealing with different simulation scenarios. This allows the comparison of different methods under similar scenarios, in order to rank their performance and to provide guidelines for choosing certain methods on basis of application. The simulation setup itself also impacts performance, an evaluation performed on a network with a simple topology or with simple traffic demand patterns may result in better performance compared to a more

complex simulation setup. As clear guidelines for the process of performance evaluation are absent, a number of facets are considered in [47]. This research focusses on the evaluation of image recognition algorithms. Performance evaluation in these algorithms is based on how well a certain type of image can be classified. Regarding optimal traffic signal control methods, the task is to optimize the traffic signal control under different demand patterns. Thus with minor modifications allow these facets to be used in the performance evaluation of optimal traffic signal control methods as well. The following modified facets are described: testing protocol; testing regime; performance indicators; performance metrics and scenario databases.

A testing protocol describes the successive approach to perform testing. The first stage of performance evaluation involves obtaining a qualitative impression of how well an algorithm has performed based on a known result. This result is the optimal solution based on a certain test scenario. As the algorithm has passed this test, next evaluations should look at aspects of performance such as robustness, adaptability, and reliability based on different testing regimes.

A testing regime relates to the strategy used for testing the different demand patterns. Several categories can be distinguished. The first of these is *boundary value* testing, in which a subset of traffic demand patterns is tested identified as being representative. This form of testing is most commonly performed as the only testing regime in publications (table 2.1), in which a certain demand pattern is used to mimic a rush-hour or off-peak scenario. Next is a *random* demand pattern, in which traffic demand is randomly generated over certain intervals in order to randomly create different demand patterns in the network. The final test regime contains different *scenarios*, which test the performance of the algorithm regarding specific traffic phenomena, e.g. green-waves or spill back.

Performance evaluation relies on the use of indicators. Such performance indicators convey the qualities of an algorithm. They are often loose characterizations used in the specification of an algorithm, and in themselves are difficult to measure. Typical indicators are described in [47], the description is be modified to suit the evaluation of optimal traffic signal control methods:

1. accuracy: how did the algorithm perform with respect to a benchmark result? This benchmark result is based on a known solution of a specific problem or performance of a different method used as the benchmark.

2. robustness: an algorithm capacity for being applied to various network specific conditions, such as different network topologies (number of roads, streaming lanes, link lengths).

3. adaptability: how the algorithm deals with variability in traffic conditions, due to different demand levels, demand patterns and traffic behaviour.

4. reliability: the degree to which an algorithm, when a simulation is repeated with the same demand patterns, yields the same result, as this effect could occur with stochastic solution methods such as GA, simulated annealing and reinforcement learning due to their action selection policy.

5. efficiency: the practical viability of an algorithm (traffic performance, computation time and storage space)

Indicators used in the performance evaluation can be measured based on different metrics. A metric is a parameters or measure of quantitative assessment used for measurement. Also for metric definition, no rigid guidelines exist, therefore ambiguous definitions are used in literature. Also misunderstanding could occur due to using different names for the same metric. In [5] a extensive set of evaluation metrics is described, such that they can also be used for other research projects, in order to deal with both the ambiguity in the literature and the lack of clear definitions.

In this section the following different components in performance evaluation are explained: simulation approach, testing protocol, testing regime, evaluation metric definition and performance metrics. As no guidelines exist on how to address these components, they are chosen freely in researches. The next section focusses on the different simulation methodologies in literature, in order to get clear how much difference exists between different publications in the field of intelligent transport systems.

2.2. Evaluation methodology in literature

Recent developments in the world of intelligent transport systems usually evaluate the performance of a new signal control method based on the results derived from simulations. The simulation is performed based on a certain methodology, in order to evaluate the performance based on certain metrics. As a simulation

methodology consists of many different aspects, a large variety between methodologies is possible. The following aspects can be categorized: simulation software, network topology, traffic demand pattern, evaluation metrics and performance indicators.

New traffic light control methods are hardly comparable due to different scenario set-ups and performance indicators, as concluded in [29]. In this research the variation in simulation methodologies among 40 publications that evaluate new traffic light algorithms was extracted based on the following criteria:

- What kind of a scenario was used (network architecture and demand)
- How is the simulation performed (commercial software or self-developed systems)
- What measures of performance indicators were used and how were these indicators defined.

Results from this extracted information state that a high number of self-developed systems (simulation based on data representation) was used to perform the traffic simulations (10 of 40). Network topologies range from single intersection (19), corridor (11) and network (20) usage. Based on how well the simulation scenarios have been designed 89 % of single intersections could be replicated using the information given in the according paper, but only 36 % of corridors, and 20 % of the networks. The descriptions of the demand span between a single peak hour and 16 hours of an average working day.
Within the forty publications, 44 different metrics have been found to evaluate the traffic light algorithm performance. These metrics can be grouped together as “waiting time”, “queue size”, “delay”, and “travel time” used the most often. Nonetheless a variation of base metrics within these groups can be noticed. For example the metrics of "delay" was mentioned by the following names in the evaluated papers: “delay”, “average delay per vehicle per second”, “average delay per vehicle”, “delay at intersection”, “total mean delay”, “mean rate of delay“, “total delay”. Also mentioned was that only one of the examined publications gave a formula for the used measures, others just named or described the used ones, therefore it is possible that metrics that carry the same name are computed differently.
Another assumption is that new traffic light methods are often benchmarked against a fixed timing signal structure and seldom compared with other advanced methods. Based on the publications described in chapter 3, information regarding the earlier described points and benchmark methods was extracted and presented in table 2.1. Out of these publications 5 have used a fixed time signal control structure as well as other control structures and 4 have only used this control structure to benchmark the results of their described method.

From table 2.1 can be concluded that simulation methodologies greatly differ from one another. In the next sections the different simulation methodology components are further explained and the chapter is concluded with a recommendation towards a general evaluation methodology.

2.3. Simulation method

Different simulation methods exist, which can be either categorized as a data representation simulation, or simulation performed with a software package. In this section only microscopic traffic simulation models are considered in this section, macroscopic and mesoscopic models are beyond the scope of this research.

Data representation
In data representation, a simulation is conducted in a programming language e.g. python, C++ etc. Every aspect included in a micro-simulation environment should be included, like a car following model, queue dynamics and other behaviour. As including these aspects is complex, most of them are neglected or are heavily reduced in complexity. Therefore results obtained by such methods can often not be validated by microscopic traffic models, also are data representation simulations are hardly reproducible as a detailed description of the script is needed and is often not made public available. As of these points, data representation simulation should only be used for validation purposes in the first stage of the testing protocol and not for any other performance evaluation.

Microscopic traffic flow simulation software
PARAMICS, PTV VISSIM, MITSUM and SUMO are all microscopic traffic simulation software that simulate every entity (car, train and persons) individually, thereby considering all relevant properties. This also holds for the interactions between individual entities as well as travel behaviour. As different software packages are designed differently, simulation results from the same methodology could vary between them. It is unknown

Table 2.1: Different simulation components extracted from literature

Author Name	Described method	scenarios	network	performance metrics	compared method	Simulation Method
Heung [20]	Dynamic Programming	various traffic demands	two junction	single vehicle delay	fixed timing	data representation
Zang [50]	Fuzzy Control	smooth-, steady-, crowded state	simplified region	average vehicle delay	fixed timing, single fuzzy control	data representation
Jamshidnejad [24]	Fuzzy Control	dynamic demand	25 intersections	average vehicle delay	PASSER V (fixed timing)	software, PASSER V
Jamshidnejad [25]	Model Predictive Control	under-saturated, saturated, over-saturated	3 intersections	objective function, total time spend, emissions	no-control, state-feedback, optimal fixed-time	data representation
Balaji [1]	Reinforcement Learning	morning peak, morning and evening peaks and extreme scenarios of both	29 intersections	vehicle count, total mean delay current mean vehicle speed	GLIDE (modified version of SCATS)	software, PARAMICS
Zhou [52]	Reinforcement Learning	heavy/light traffic flow, on-line/off-line control	single intersection	delay time, number of waiting vehicles, intersection saturation trafficflow capacity	off-line optimized model	data representation
Dong [12]	Particle Swarm Optimization	slight-,moderate- and heavy demand	9 intersections	average vehicle delay, average stop rate	fixed timing	software, TSWAS
Hu [22]	Tabu Search	5 different demand levels, with/without signal optimization offset	56 intersections	average travel time	Genetic Algorithm	data representation
Hirulkar [21]	Particle Swarm Optimization	constant- / time-varying demands	5 intersections	average delay, average number of stops	optimization without off set	data representation
Putha [37]	Ant Colony Optimization	over-saturated conditions	20 intersections	vehicle throughput	Genetic Algorithm	data representation
Sabar [39]	Memetic Algorithm	real-world based	2-4 intersections	average vehicle delay	Genetic Algorithm, fixed-timing	software, AIM-SUN
Chin [8]	Genetic Algorith	under- /over-saturated	2 intersections	vehicle throughput, average speed	fixed timing	software, not described
Ma [30]	Artificial Fish Swarm Algorithm	real-world based	5 intersections	average vehicle delay	fixed timing	data representation
Hu [23]	Particle Swarm Optimization	real-world based (congestion, (half) free flow)	17 intersections	vehicle total travel time	Genetic Algorithm, PSO fixed-timing	software VISSIM
Wei [45]	Cellular Automata	5 different saturation levels (0.2 up to 1.0)	34 intersections	average vehicle delay	traffic responsive control	software, Visual C++
Genders [17]	Deep learning	constant demand with inverse Weibull / Burr distribution	single intersection	cumulative delay, queue length, average travel time	single layer neural network	SUMO

whether this variation is significant, but it should be minimal if the model is properly validated. A list of well validated, and thus suitable traffic simulation models should be established, such that it is known which software package to use. In this thesis SUMO [3] is chosen as the microscopic simulation package to conduct the simulation based evaluation, as it is open-source, highly portable and simulation can be accessed by Python.

In the next section the car-following model is explained, as there are many approaches to simulate individual behaviour, different approaches are used between software packages.

2.4. Car following model

A car-following model controls the driver's behaviour with respect to the preceding vehicle in the same lane. When a vehicle is constrained by a preceding vehicle it is considered as following and driving at the desired speed could lead to a collision. When a vehicle is not constrained by preceding vehicle, it is considered unconstrained and is thus able to travel at its desired speed. Car-following models typically use a large number of parameters such as desired speed, safety distances, perception thresholds and so forth, these parameters describe the car following behaviour realistically. A comparison of different car-following models used by different microsimulation software packages was performed in [34], in order to enlighten and compare the car-following behaviour of different models. The conclusion of this research, advices to use models that produce representative driving trajectories, using as few calibration parameters as possible, since models with a large number of parameters are more heavily relying on the model makers' role in the calibration work.

Table 2.2: Different simulation packages and the number of parameters used in the car following model in [34]

Software	Car-following model	Number of Parameters
AIMSUN	Gipps-model	4
MITSIM	GHR-model	10
Paramics	Fritzsche-model	10
VISSIM	Wiedemann	16
SUMO	Krauss	6

2.5. Performance indicators

To evaluate new developments in the field of intelligent transport systems, it is important that every evaluation makes use of the same set of metrics and that those metrics are calculated in the same way. Choosing the right metric for the evaluation of a certain situation is also important and not always as trivial as it might seem. In [5] was found out that many papers define their own metrics and give them a name that is logical to the author. A reader who is comparing the work of several authors has to be cautious because the same terminology in one paper might have a different meaning in another paper. Because of both the ambiguity in the literature and the lack of clear definitions, the paper describes an extensive set of performance metrics such as vehicle delay, number of stops and emissions. It is clear that metrics should be unambiguous and clear defined.

2.5.1. Aggregation

Often the performance of a new signal control method are shown by net-wide aggregated values, thus the mean over the entire simulation period over the entire network. The lower the mean travel time, or the higher the mean speed, the better the performance. Although the representation of the net-wide aggregated values is considered as sufficient, it holds some pitfalls. Because all measurements are aggregated, incidents which only occur during a part of the simulation or which are only affecting a small part of the network may get invisible, because they are affecting only a small fraction of a simulated traffic. It is also possible that effects during a period of the simulation time are cancelled out in a following period of the simulation time, e.g. travel time reduction and increase. Also, aggregation over the complete simulation time removes time-dependant changes of the metrics, for instance when using varying traffic demand. To avoid these problems, other aggregation types can be used:

- aggregating data within specific time windows.

- aggregating data only for certain origin/destination pairs or certain routes.

2.6. Traffic phenomena

Different traffic phenomena could appear during certain levels of traffic saturation, these traffic phenomena describe the occurring network conditions and should be taking into account when evaluating optimal signal control methods, because the performance related to these traffic phenomena holds relevant information of the methods' performance. The described traffic phenomena are separated for different levels of saturations and are for under saturated traffic conditions the formation of green-waves and for (over)saturated conditions spill back.

2.6.1. Degree of saturation

The degree of saturation is a ratio that indicates the proportion of available lane capacity that is used under the given cycle structure. It describes the maximum flow of vehicles that can pass for a given green time and is determined as the ratio of the arrival flow and the maximum departure flow. If the vehicle arrival flow exceeds the maximum departure flow, a residual queue will remain after the green phase has ended, this is noted as cycle failure. In [11] the following categorization of the degree of saturation based on the number of cycle failures was proposed:

- **Light traffic** Characterized by the expectation of minimized cycle failures. The objective in these conditions is to fully serve arrival queues. Cycle failures are expected to be infrequent at less than 5%.

- **Moderate traffic**: characterized by a normal operation of the intersection, in which optimization of the objectives is possible. By optimizing such objectives like, realising green waves, minimizing delay and stops, or balancing the degree of saturation on each approach, some cycle failures might occur, this does not harm the intersection operations.

- **Heavy traffic**: characterized by frequent cycle failures, but with a residual queue that ebbs and flows without growing uncontrollably. The queue is the result of stochastic arrival flows where those flows exceed capacity up to about half the time.

- **Over-saturated conditions**: characterized either by excessive residual queues that grow without control, or by queues that cause more widespread damage to the operation of a network. To prevent operation failure, spill back measures should be activated during these conditions.

The above mentioned categorization is illustrated in figure 2.1, this figure does not represent actual statistics, but rather illustrates that when average demand equals approach capacity, the percent of cycles with residual queuing will be 100%.

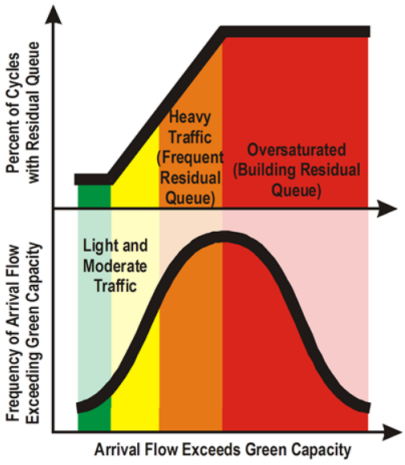

Figure 2.1: Saturation and residual queues, in [11]

2.6.2. Under-saturated traffic conditions

- **Green-wave**: when green phases of a series of intersections are coordinated to allow continuous traffic flow over several intersections in one main direction, a green wave is formed. Vehicles driving along the main route do not have to stop at these intersections, when driving along with the green wave. To evaluate this traffic phenomena the number of stops for vehicles that drive on the arterial, can be used. In [16] two different methods to facilitate the flow of vehicles in a green wave along the arterial is described, which are the uniform- and the variable bandwidth method. The latter provide significant flexibilities in the realisation of green wave propagation with a better adaptation towards the current traffic flows in the network.

2.6.3. Over-saturated traffic conditions

- **Spill back**: when the number of vehicles in the queue is increasing due to high traffic demands, exceedance of the link storage capacity is a possible consequence. When there is no storage capacity left, vehicles are unable to move onto the downstream link, thus possibly blocking other vehicles, moving into other directions, as well. This results in an increment in vehicle delay and a further propagation of spill back onto other upstream links.

2.7. Network topology

Different network topologies can be used, when simulating traffic signal control methods. The more conflicting signal groups an intersection contains, the more complex the optimization problem becomes. Different network topologies can be distinguished, each with their own use and complexity, and can be categorized as follows:

1. Isolated intersection: this network consists of a single intersection, and is useful for evaluating isolated signal control methods, or to validate a multi intersection optimal signal control method during the design phase, since fixing bugs is less complex on an isolated intersection.

2. Arterial: consist of multiple intersections connected such that a main road is formed with multiple intersections (3 or more), on which minor roads are connected. Often the demand on the main road is higher, compared to the side roads. An arterial network is used when evaluating the performance of a cooperative intersection signal control method.

3. Grid network: a grid network consists of intersections connected in a grid topology such e.g. 2×2. This network topology is useful to evaluate network-wide performance of a signal control method.

4. Real network: This network topology reflects a real world network, and is used to evaluate the performance of a signal control method on an existing case.

2.8. Simulation randomness

Most traffic simulation models use stochasticity as an important aspect of reproducing reality in a simulation scenario, this also applies to SUMO. There are multiple ways of adding stochasticity to a simulation, examples are the arrival pattern of vehicles at the nodes to the simulated network, the distribution of driver characteristics such as desired speeds, critical gaps and vehicle routes. Also decisions are simulated using random draws from certain distributions, for instance compliance to regulations, magnitude of acceleration and deceleration and lane choice decisions. Components of the car following model can also be stochastically distributed, for instance random speed variations.

Arrival process

The arrival process of vehicles at intersections could be categorized into three patterns, as described in [7]: random arrivals, grouped arrivals and mixed arrivals. The network topology has an effect on the occurring arrival pattern, since intersection influence the formation of certain arrival patterns.

In random arrivals, vehicles seem to arrive at the intersection randomly. This arrival pattern is observed when there are no other upstream intersection within 3-4 km of the observed intersection, therefore this pattern can only occur at isolated intersections. In this arrival pattern the headway between vehicles is often distributed according to the negative exponential distribution.

In grouped arrivals, vehicles arrive at the intersection in platoons with more or less a constant headway. This arrival pattern is observed when there are upstream intersections within 2 km of the observed intersection, therefore this is the most occurring traffic pattern in urban networks. A grouped arrival pattern emerges, because vehicles arriving at an upstream intersection are stopped, due to a red signal phase. When the signal phase turns green the queued vehicles are released and arrive in a platoon at a downstream intersection.

Mixed arrival patterns are a combination of random arrivals and grouped arrivals. This arrival pattern is observed when upstream intersections are located at intermediate distances, about 2-4 km. The mix of both described arrival patterns is occurring, due to the distance between the intersections being large enough for many of the released vehicles to disperse from the discharged platoon and arrive independently; yet the distance is not large enough for the entire platoon to disperse. Hence, some vehicles still arrive in a grouped formation.

2.9. Number of runs

The use of randomness is needed to represent the diversity of characteristics as they are found in reality. However, this stochasticity has a large effect on the results of the simulation, and different simulation runs can therefore produce very different results, due to a different sequence of random numbers that happen to be drawn for the mentioned processes. Therefore a minimum number of replications is needed to address this problem, as stated in [6].

Two main approaches are described in [2] to determine the number of replications: sequential- and two-step approach. In the sequential approach, one replication at a time is run until a suitable stopping criterion is met. Assuming that the outputs, Y_i, from different simulation runs are normally distributed, in [15] the following criterion was given:

$$R \geq R_i \max[2, (\frac{S_R(Y_i)\, t_{\frac{\alpha}{2}}}{d_i})^2] \tag{2.1}$$

where:

R = number of replications performed,

R_i = minimum number of replications required to estimate the mean of Y_i with tolerance d_i,

$S_R(Y_i)$ = sample standard deviation of Y_i based on R replications,

$t_{\frac{\alpha}{2}}$ = critical value of the t-distribution at significance level α

i simulation number index

In the two-step approach, an estimate of the standard deviation of Y_i is required, and is obtained by performing R_0 replications. Under the assumption that this estimate does not change significantly as the number of replications increases, the minimum number of replications required to achieve the allowable error d_i is given by:

$$R_i = [\frac{S_{R0}(Y_i)\, t_{\frac{\alpha}{2}}}{d_i}]^2 \tag{2.2}$$

In [41] the required number of replications is calculated for all evaluation metrics. The most critical (highest) value of R_i determines the number of replications required.

2.10. Demand profile

The demand profile reflects the vehicle arrival pattern on a certain link over time. Different demand profiles can be used during the simulation, and those can be divided into constant and time varying demand profiles:

1. **Constant**: A constant traffic demand profile does not change over time, it can be used to evaluate the performance of the signal controller related to a certain traffic phenomena, without the results being influenced by variations in traffic demand.

2. **Time varying**: A time varying traffic demand profile varies over time, it can be used to evaluate the signal controller performance under dynamic traffic demand conditions, in order to mimic a more realistic traffic demand profile, which does also changes over time.

2.11. Scenarios

Different scenarios can be used during the simulation process, a simulation scenario describes a traffic pattern or phenomena. The scenarios are used to test whether the optimal control method is able to handle the traffic phenomena, with the desired outcome in mind, this is called the desired behaviour of the optimal signal control method. The scenario set-up consists of values, or a range of values, for traffic demand, demand distribution and the saturation level. The following simulation scenarios could be considered: green-wave formation, spill-back prevention, rush-hour and off-peak hours.

2.11.1. Green wave

- **Desired behaviour**: formation of sequential green phases on multiple intersections (green waves) for vehicles on the main road of the arterial.
- **Demand Characteristics**: traffic demand should be kept below saturated conditions, in order to prevent other traffic phenomena from appearing. Traffic demand pattern could be either constant, to evaluate performance under a specific demand level or time varying to evaluate under more dynamic traffic demand conditions. Demand distribution should be of such ratio that a majority of the vehicles is travelling on the arterial route, since that is where the green wave should occur.

2.11.2. Spill-back

- **Desired behaviour**: To prevent spill-back, the upstream links with the lowest turn fraction towards the queued link should be prioritised. This will result in a reduced traffic inflow towards the queued link, while minimizing additional delay for vehicles with different routes. Green time on the queued link should be prioritized in order to maximising link outflow, thus reducing the queue length.
- **Demand Characteristics**: traffic demand should be in high saturated, or in over saturated conditions, otherwise queue growth and thus Spill-back will not occur. Also in this scenario the traffic demand pattern could be constant or time varying. Demand distribution should be of such ratio that a majority of the vehicles is travelling on the arterial route, since that is where Spill-back should occur.

2.11.3. Rush-hour

- **Desired behaviour**: optimize signal plan in order to achieve the lowest intersection cost, which consists of time loss and number of stops.
- **Demand Characteristics**: Traffic demand in this scenario varies over time, and increases from under saturated conditions to over saturated conditions and back to under saturated conditions. Demand distribution reflect that the majority of the vehicles drive on the arterial route, since this is the most important road in the network.

2.11.4. Off-peak hours

- **Desired behaviour**: optimize signal plan in order to achieve the lowest intersection cost, which consists of time loss and number of stops.
- **Demand Characteristics**: low traffic demand, with a more evenly distribution between the roads in the network, because lower influence of arterial road importance compared to rush-hour conditions.

2.12. Conclusion

This chapter provides the different facets of performance evaluation and gives insight in the different simulation methodologies used among publications in the field of intelligent transport systems. By describing the different components in these methodologies, it becomes clear that with this number of variabilities a general performance evaluation can only exist with rigid guidelines.

The absence of any clear guidelines on how to set up the performance evaluation of new designed optimal traffic signal control methods is considered as the major reason that individual publications are not comparable to one another. Despite useful contributions, in terms of scenario description given in [29] and metric description given in [5]. Metric interpretation remains an ambiguous task, resulting in possible implementation errors. This should be taken into account in the metric description in the remainder of this thesis.

A common standard for measuring traffic light performance using traffic simulations can only exist if a well validated traffic simulation model is used and only if every research conducts the same simulation methodology. This simulation methodology should be designed with the guidelines in mind. These guidelines should contain a list of accepted simulation software packages, different network topologies, unambiguous measurement metrics, scenario descriptions and simulation set-ups.

After these guidelines are concrete, they should also be used by new publications as their method of performance evaluation. This can be achieved by only considering publications that conducted their evaluation methodology based on these guidelines as scientific relevant. Although this is far-reaching, results in the field of image recognition algorithms show that this leads to a comparative measure between different publications. In chapter 3, the different methods for optimal traffic signal control that could be evaluated by means of the general methodology presented in this chapter are given.

3

Traffic signal control approaches and mathematical techniques

Traffic signal control optimization is considered one of the main ways to solve traffic problems in urban networks. By optimizing the traffic signal plans of an intersection, signal control efficiency can be increased, resulting in reduced vehicle time loss and decreased number of stops, thus improving the traffic flow.

Traditional signal control methods rely on mathematical traffic-flow models to attempt to solve the optimal control problem. This approach has led to many useful ideas and new methods for traffic signal control applications. Solving the optimal control problem for a single intersection is possible, however real-time solutions with dynamic traffic conditions remain a complex problem, due to many variables involved as concluded in [31]. In order to cope with this complexity, many traffic-flow models incorporated simplifications or can only be applied under specific conditions, as concluded in [35]. This has a negative effect on the accuracy of the prediction or the applicability of these models.

With the introduction of computational intelligence (CI) in the field of traffic signal control, new methodologies were developed based on the following approaches, e.g. fuzzy systems, neural networks, reinforcement learning and genetic algorithms. These methodologies made it possible to solve problems that were previously too computational complex or considered infeasible to solve, due to large computation time. As stated in [51], CI based methods are capable to obtain optimal or suboptimal solutions within the given timescale. Also do most CI methodologies not require sophisticated traffic-flow models and sometimes no model is required at all.

The goal of this chapter is to provide an overview of different traditional signal control methods, as well as describing the different computational intelligence techniques and methods described in academia using these techniques.

First, in section 3.1 an overview of the different controller types is given, this provides an outline of the characteristics of different types of control approaches and does also clarify how the methods described in section 3.2 are applied in traffic signal control. In section 3.2 an overview of the different methods designed in academia in order to improve the performance of traffic signal control is given. These methods are incorporated into the control structure in order to find the optimal or near-optimal solution of the traffic signal control problem. This problem can be drawn up in different formulations, from one publication to the other, but in general the goal is to minimize the experienced delay by travellers in a network.

Based on the overview given in this chapter two methods are selected, which are used in the microscopic simulation road traffic simulation package SUMO to evaluate their performance under different demand levels and simulation scenarios.

3.1. Overview of different control approaches

Traffic signal control strategies can be categorized based on adaptability and level of hierarchy. First the distinction in adaptability is explained, as this forms the basis of the control approach. The control approach is implemented onto a level of hierarchy, this combination result in the different control approaches.

Adaptability can be divided into fixed-timing or dynamic. In fixed-timing the cycle length is derived offline based on historical demand and turn fractions. This control strategy performs well for at the designed

demand level, but as traffic demand levels vary over time, the performance deteriorates. Also do fixed-timing tend to age fast if traffic growth is high. Opposite of fixed-timing is dynamic, in which multiple elements such as green-, red- , yellow time, cycle times and structure can be adapted based on the current traffic demand. Therefore dynamic can adapt to the current traffic demand and is therefore able to perform well at different demand levels. With the level of adaptability also evolved the desire to strive for optimal control, as mentioned in the introduction of this chapter, these control methods rely on mathematical traffic-flow models to attempt to solve the optimal control problem.

The level of hierarchy is derived from the scale on which the control approach is acting and can be divided into local or coordinated control. These two distinctions can be combined with either of the two distinctions in adaptability as mentioned above. In local control the controller only considers information obtained from the local intersection, information from neighbour intersections is thus neglected. Therefore it is possible that the followed signal plan on one intersection leads to performance deterioration on network level, due to spill-back or individual control plans that do not match with each other, resulting in multiple vehicle stops. The following control approach combinations could be considered: local fixed-timing and local dynamic

Coordinated traffic signal control also considers information from neighbouring intersections, which makes it possible to coordinated the signal control plans of multiple intersections to each other in order to form a green wave or to adjust the signal plans in general. In coordinated fixed-timing the signal plans of multiple intersections are designed in such a way that vehicles travelling within a given speed range are provided a clear passage at the intersections. This control approach holds the same disadvantages as described in the fixed-timing section. In coordinated dynamic control a coordinated control is provided, combined with the dynamic approach. As providing a coordinated control in combination with a dynamic approach already becomes a complicated task due to the numerous variabilities involved, traffic-flow models that attempt to solve the optimal control problem were developed. This models has let to many useful ideas and methods for traffic signal control applications, such as: SCOOT, SCATS and UTOPIA. Further developments has led to RHODES, PRODYN and OPAC, these methods utilise forward dynamic programming to generate optimized control strategies. Methods emerging from the advantages in computational intelligence can be applied to provide optimal or sub-optimal solutions for the signal control problem on either local or coordinated level and are further described in section 3.2 of this chapter.

3.2. Mathematical techniques

In this an overview of the different methods designed in academia in order to improve the performance of traffic signal control is given. These methods are incorporated into the control structure in order to improve the performance by finding the optimal or near-optimal solution of the traffic signal control problem. The first technique that is discussed is dynamic programming, as it is implemented in systems such as RHODES, PRODYN and OPAC. Followed by the different techniques emerged from computational intelligence.

3.2.1. Dynamic programming

Dynamic programming was introduced in [4] as a method to solve multi-stage decision problems, and is used by RHODES, PRODYN and OPAC to optimize the signal plans. Although the method requires a traffic-flow model to predict the movement of vehicles in order to achieve its optimization, an improvement in computation time could be achieved. The approach is to reduce a complex problem into simpler sub-problems, subsequently these sub-solutions are combined to result in the solution of the original problem.

In [26] an adaptive optimal planning algorithm for signal control at a single intersection using an efficient dynamic programming technique was designed. The selected objective is to minimize the total delay experienced by the vehicles passing through an intersection. The method employs forward dynamic programming in combination with two acceleration techniques that eliminate inferior states to perform an unconstrained optimal search for traffic signal plans. Also is the method applicable for decentralized coordination with an adaptive signal control scheme at each intersection.

A coordinated intersection control based on dynamic programming is proposed in [20]. This method uses a fuzzy logic controller (further described in 3.2.2) at each intersection and a dynamic programming technique to derive the green times for the phases of the signal plan. Coordination between the intersections is achieved by taking information from adjacent intersections into account. Simulation is performed on a simple two-intersection network with variations in traffic demand. Results show that the proposed method is able to reduce the delay per vehicle, particularly when the traffic demand is at the junction capacity.

3.2.2. Fuzzy logic

Fuzzy theory originated from the development of fuzzy sets, first proposed in [48]. Fuzzy logic is a form of set theory and logic in which value variation of information is taken into account. Fuzzy logic offers a formal way of handling terms like ”more”, ”less”, ”longer” by allowing a variety of degrees of applicability of predicates, rather than simply being true or false. This is in contrast to "crisp logic" where an outcome only has two possible values, i.e., true or false. Fuzzy logic mimics human perceptions in control logic, making it effective in representing human control behaviour.

Fuzzy logic controllers consists of three components: an input stage, a processing stage and an output stage. In the input stage, input values from sensors is fuzzified into fuzzy membership functions and truth values. In the processing stage all applicable rules to compute the fuzzy output functions are executed. IF-THEN rules are applied to map input or computed truth values to desired output truth values. The IF part of such rules is called the "antecedent" and the THEN part is called the "consequent". Fuzzy control systems typical consist of dozens of rules. The last stage is the output stage in which fuzzy truth values are defuzzified to obtain the variables. Since all output truth values are computed independently, the output truth values do not result in continuous variables, therefore a match with intended truth values has to be decided on. This dependence on the pre-set quantification values for the fuzzy variables is a mayor disadvantages of the controller.

Although fuzzy logic traffic signal controllers have the advantage of rules based on expert reasoning and the method does not need an accurate mathematical model. The method lack adaptability to variable traffic volumes since the fuzzy rules and membership functions are invariable. To overcome this issue, in [49] an improved fuzzy network model by incorporating a neural network algorithm (FNN) was proposed. Training based on real-time data results in better adaptation to different traffic flow and different conditions at the intersection. The proposed method shows improvement in performances and adaptability compared to traditional fuzzy control plan for traffic signal.

In [50] a method for an area coordinated control system is described. In this method the network is divided into sub-regions. And fuzzy logic control is used to discriminate traffic states and to optimize for network performance. Simulation is performed on a grid like network of the city of Changchun (China). During the simulation the following traffic states were taken into account: smooth state, steady state, crowded state. Results show that the proposed method reduce the average vehicle delay compared to single fuzzy control and fixed time control.

In [24] a method for urban traffic control using a fuzzy multi-agent system was proposed. In this method the network is divided into sub regions, each region has its own agent, using a fuzzy controller. Fuzzy rules are based on OD matrix clustering, on which an optimal cycle plan is found. Cooperation between the intersections is achieved by grouping direct up- and downstream intersections together. Simulation is executed on a network containing 25 intersections, in which each intersection is a sub region. Results show that the fuzzy multi-agent control system reduces the total average delay time of the vehicles by 19% compared with the non-fuzzy control system.

3.2.3. Neural networks

Neural Networks are inspired by the architecture of biological neural networks found in brains. Just like the neural network in the brains, a network of artificial neurons is used in neural networks. This network contains multiple interconnected neurons that learn from prior experience using algorithms, allowing the network to respond to previously unknown inputs. By employing this process an artificial network builds its own regression model based on data used during the learning process. This regression model is used to predict the outcome of new input. Neural networks are build up from multiple layers as described in [32], namely: the input, the hidden (consisting of one or multiple layers) and the output layer. The multiple layer neural network framework is displayed in figure 3.1. The input layer is the leftmost layer in the framework, the neurons within this layer are called input neurons. Input factors are described in the input layer, example of input factors are the number of vehicles in a queue or the phase time of the current phase. The output layer is the rightmost layer in the framework, the neurons in this layer are called output neurons. The output signals are produced in this layer. The layers in between are called the hidden layer, since the neurons in this layer are neither inputs nor outputs. The neurons in a neural network are connected by links, each with a corresponding weight. By using a specified activation function, ranging from linear to any other function, the activation behaviour of a neuron is described. The combination of weights and activation among the neurons in the network is used to calculate or approximate the function describing the problem. In order to obtain the desired function the synaptic weights between the layers must be adjusted. The process is called learning,

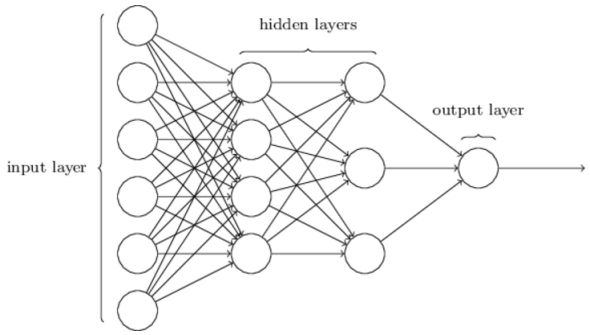

Figure 3.1: Neural network framework in [33]

this is the first important stage in a neural network. The purpose of learning process is to minimize the global error level between the models output and a desired outcome. At the start of the learning stage randomized weights are used for all neurons in a neural network. This means that the neurons do not know anything, thus they must be trained to solve the particular problem. When the global error reaches a satisfactory level the training is ended and the network uses these weights to make a decision for input data to come.

Machine learning problems can be categorized into three mayor categories, each of those categories has its own application and requirements to be used in the training stage of neural networks. The following three categories can be distinguished, and are further explained in the remainder of this section: Supervised Learning, Unsupervised Learning and Reinforcement Learning.

1. **Supervised Learning** With supervised learning the desired output is provided with the input during the learning stage. Since the desired result is known on forehand it is possible to calculate an error based on the target output and the actual output of the network. This error is used when adjusting the weights to reduce the error within accepted values. Supervised learning problems can be divided into "regression" and "classification" problems. In a regression problem, the goal is to predict results within a continuous output, which means that input variables are mapped to some continuous function. In a classification problem, the goal is to predict results in a discrete output. In other words, trying to map input variables into discrete categories.

   An example of algorithm which is using a supervised learning approach is Back-propagation, described in [19]. The classic back-propagation algorithm takes a step of a certain length at every iteration in the direction of the steepest descent. A variant to this algorithm is the resilient back-propagation (Rprop) algorithm, described in [38]. This algorithm changes the step size for individual weights as the algorithm propagates based on the derivative of the partial error. If the gradient has the same direction from one iteration to the next, the step size is increased. Conversely, if the gradient reverses direction, the step size is decreased. In [25] a model predictive controller that uses RProp is proposed. This controller uses a gradient-based optimization approach based on the RProp with the aim of finding a balanced trade-off between reduction of the total time spent by the vehicles and the total emissions. The model is simulated on a network with three controlled intersections. Three different demand profiles were used. The proposed smooth MPC controller is compared to no-control case, state-feedback, optimal fixed-timing, and non-smooth MPC-based controlled case. The simulation results have shown that the smooth MPC controller improves the performance of the network significantly, although the CPU time for the smooth controller significant decreased compared to a non-smooth controller it should be noted that the CPU time is still too high to apply the method in real-time control.

2. **Unsupervised Learning** In unsupervised learning, problems are approached with little or no idea of what the result should be. The difference to supervised learning is that due to the absence of the desired outcome it is not possible to evaluate the performance of the learning stage based on feedback on the error of the predicted results. With unsupervised learning it is possible to derive structure from data without knowing the outcome on forehand. This structure is derived by clustering the data based on relationships among the variables in the data.

   In [9] a distributed, unsupervised coordinated traffic responsive strategy in order to achieve real-time traffic control in a network is proposed. The large scale traffic signal problem is divided into multiple

sub problems each which is handled by an individual agent. The decisions made by these agents are mediated to higher level agents. Coordinated control by the agents is achieved by using a cooperative distributed problem solving approach. The proposed multi agent structure makes use of concepts and theories from neural network, evolutionary algorithm, fuzzy logic, and reinforcement learning. The proposed model is simulated on a network containing 25 intersections. Two demand scenarios were used and for benchmark purposes also the green link determining signal control system was implemented. Simulation results are based on repeated rounds of simulation and show significant improvement in the conditions of the traffic network.

3. **Reinforcement Learning** In reinforcement learning (which will be discussed in section 3.2.4) feedback is given to the agent, similar to supervised learning. Instead of providing the algorithm with a target output, a reward is given based on how well the system performed. The aim of reinforcement learning is to maximize the reward through trial and error.

3.2.4. Reinforcement learning

Reinforcement learning is a method of machine learning in which the agent learns a specific policy based on rewards and penalties. In figure 3.2 the reinforcement learning framework of an agent is displayed. The agents observes a state, which resembles a specific condition of the environment. The agent select an action to be taken next, this action selection can be performed randomly or based on a prior learned policy. As the action is taken the environment advances into a new state. If this state is positive compared to the previous one, the agent receives a reward; otherwise it receives a penalty. Based on the feedback the agent learns what action to perform according to the current. While selecting right the total reward increases, with the objective to maximise the cumulative rewards the method optimizes the decision making over time.

When projecting this framework onto the optimal traffic signal control problem, the agent is the controller that selects which stage of the intersection to become activated next, the environment is an intersection or a network of intersections, the state can contain any quantitative measurement, such as saturation level, queue length and vehicle speed and the reward is measured based on these measurements. For example, an increase in queue length results in a negative reward, where a decrease in queue length results in a positive reward.

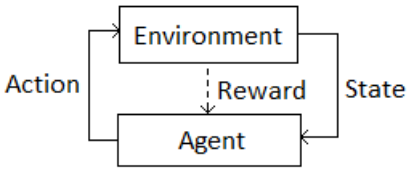

Figure 3.2: Reinforcement learning framework of an agent.

In [1] a multi agent reinforcement learning algorithm was proposed, this method was developed to optimise the green timing of an urban arterial road network in order to reduce the total travel time and delay. Data from sensors, historical patterns and communication between adjacent intersections was used to improve the network performance. The method was simulated on a network of the financial district of Singapore, consisting of 29 intersections. Four different scenarios were used in terms of different demand, the simulation results were compared to other benchmarks including GLIDE, which is the urban traffic control system currently at use. Results show that 9–15% improvement in performance was made (vehicle speed and mean delay) compared to other benchmarks in all the 20 simulation runs.

In [13] a Q-learning-based acyclic signal control system that uses a variable stage sequence is developed. In this method, a single agent is trained to achieve optimal traffic signal control at a single intersection. State definition contains the maximum queue length per stage of all stages involved at the intersection. The agent of the agent is defined as the stage that should be active next. Rewards are defined as the change in total cumulative delay of the vehicles currently present at the intersection. If the delay is reduced after taking the action, the reward is positive, otherwise the reward is negative. The designed method is evaluated on an isolated intersection on which a rush-hour with demand typical to a specific intersection in Toronto is simulated. Results show that a reduction in total delay of 36%-43% is achieved.

3.2.5. Heuristics-based algorithms

Heuristic-based algorithms are algorithms, described in [36] that either give a near optimum solution or provide a solution, but not for all instances of the problem. The algorithms makes use of technique of Meta-heuristics which are a problem-dependent technique for solving an optimization problem. Usually, the found solution is close to the optimal solution and it is found more quickly. An increase in computational speed is achieved because the algorithm ignores or even suppresses some of the problem's demands. Thus optimality, completeness, accuracy or precision is traded in for the sake of computation speed. Heuristics-based algorithms are used when classic methods are too slow or unable to find the optimal solution. Which could also be applied to solving the optimal traffic signal control problem. In [27] a survey is performed, that describes a number of well-known examples of meta-heuristic algorithms, which are: Simulated annealing, tabu search and swarm intelligence. Several methods have been designed that use these techniques to solve the optimal traffic signal control problem and are discussed below.

1. Simulated annealing

   This algorithm is used in optimization problems and is capable of achieving a reasonable approximation of a global optimum for a function with a large search space. The algorithm probabilistically decides at every iteration whether it should stay at its current state or should move towards another state. This process leads eventually to the state with the best approximation of a global optimum.

   In [12] a simulated annealing - particle swarm method for the optimization of urban traffic signal timing was developed. The algorithm is designed to address two known shortcomings of the PSO algorithm which are: getting easily trapped in local optima and difficulties to process constraints of the optimization problem. Simulation took place in a network consisting of 9 intersections and three different groups of traffic demand values were used, slight-demand, moderate-demand and heavy-demand. The proposed algorithm is compared to fixed timing and results show that it is capable of reducing the average delay per vehicle by 41% and average stops by 30.6%.

2. Tabu search

   This technique creates a dynamic set of rules (tabus). These rules prevent the system from searching around the same area. The method solves the problem of local search methods getting stuck in local sub optima.

   A randomized tabu search algorithm for network-level signal optimization problems was given in [22]. The algorithm consists of two main processes: randomized search and performance evaluation. Average traffic flow distributions in the network are used as input values to optimize signal control. The method is compared to a genetic algorithm and simulated on two networks, one test network of 6 intersections and a real-city network of 56 intersections. In the real-city network 5 different demand levels were used. Network performance is expressed in average travel time and average stopped delay. Results show that travel time can be reduced by 25% for medium and high demand levels.

3. Swarm intelligence

   Swarm Intelligence is based on decentralized agents interacting locally with one another and the environment. Several algorithms are inspired by the self-organized behaviour of animals, i.e. particle swarm optimization (PSO) algorithm, the ant colony optimization (ACO) algorithm, and artificial bee colony algorithm.

   A PSO algorithm for traffic signal timing optimization problem is described in [21]. The aim of the algorithm is to minimize the average delay and average number of stops for adjacent junctions in a network. The algorithm is simulated on a network containing five intersections. The algorithm is compared to an actuated vehicle control. Results show that delay per vehicle can be substantially reduced under both constant traffic as time-varying traffic demands.

   An ACO algorithm that is able to find optimal signal timing plans in order to reduce queues and the remove of blockages due to oversaturate traffic conditions is given in [37]. The algorithm is simulated on a network with 20 intersection and is compared to a genetic algorithm. 30 random seeds were used and traffic flow was in over saturated condition. Simulation results show that the proposed algorithm is more effective for a larger number of trials and to provide more reliable solutions compared to the genetic algorithm.

3.2.6. Genetic algorithms

Genetic Algorithms (GA) are global optimization techniques that make use of parallel and structured search strategies. The algorithm starts with an initial population of candidate solutions for an optimization problem. Based on an iterative process, similar to processes of evolution in a population, the candidates are randomly mutated or combined. Based on a fitness function the fitness of the candidates is determined. The candidates who are better at solving the problem are assigned a higher fitness value. The candidates with the highest fitness level are used in the population of the next generation and the others do not survive the process. During the evolution process information between structures of the previous generation are changed due to mutation or combination. Good properties of a candidate solution are propagated towards the next population, thereby increasing the performance of the solution. Genetic algorithms are not prone to be trapped in local optima, due to the use of parallel computing and structured search strategies. This property gives the algorithm an advantage to optimize traffic signal control in a network environment. However with increasing network size comes also the increase in dimensionality, resulting in increased convergence time. Also the convergence rate is dependent to the selected parameters and the values used. Different problems require different parameters and corresponding values.

To address this issue, an adaptive memetic algorithm for optimising signal timings in urban road networks using traffic volumes derived from induction loop detectors is described in [39]. The proposed algorithm combines the strengths of GA with the exploitation power of a local search algorithm, in an adaptive manner, so as to accelerate the search process and generate high quality solutions. The algorithm was simulated on two different real world networks of Brisbane, Australia and Plock, Poland. The results were compared to a fixed timing control structure and a genetic algorithm optimized control. 30 independent runs were used during the simulation. Results show that the proposed algorithm is performing better to alternatives in terms of reducing vehicle delay.

A multiple intersections traffic signal timing optimization with genetic algorithm to improve the traffic flow in a network was proposed by [8]. The proposed algorithm is capable of adjusting signal timing parameters during the optimization, in order to handle and manage dynamic traffic demands and thus reduce delay. The algorithm is simulated on a two intersection network and compared to fixed timing. Two demand scenarios were used, one with high demand on the main link and less demand on the side links. Another scenario has a rapid increase in demand in order to mimic a rush hour scenario. Simulation results show that the proposed algorithm was able to react on dynamic demand conditions, prevent and release queues and to achieve higher degree of performance in the traffic flow control of multiple intersections of traffic network.

A green wave traffic control system optimization based on adaptive genetic-artificial fish swarm algorithm, was proposed in [30]. This proposed algorithm introduces GA characteristics to the original artificial fish swarm, which is based on animal behaviour. By introducing a GA the defect of reduced search accuracy and convergence rate should be overcome. The proposed method is simulated on a 5 intersection urban arterial road network similar to Jianning Road in Lanzhou, China. 10 simulation periods were used with a dynamic demand pattern. Simulation results show that the proposed method achieves a better signal control program compared to the tradition method.

3.3. Conclusion

In this chapter different characteristics of control strategies are explained. Classifications can be made based on the adaptability and the level of hierarchy. With the goal of improving the traffic signal control and ultimately optimize for the best traffic signal control plan, one could state that a coordinated control strategy is favourable over a local intersection control strategy, as signal plan optimization based on a local intersection level could possibly deteriorate the performance on network level. Nevertheless in the remaining of this research the basis of the control approach is a non-coordinated control approach, as this removes the technical difficulties that arise in establishing a coordination between multiple intersections. As the implementation in Python of both methods, as isolated control strategies is already a difficult task.

Various different optimal control methods and computational intelligence techniques are described in this chapter to provide an overview. These techniques have a different extend in complexity and implementation to improve traffic signal control. In principle, every method could be implemented and evaluated in order to obtain the potential for practical implementation. However, this evaluation requires an own implementation of the method, as these methods are not publicly available. Therefore two mathematical techniques are chosen based on the level of detail in the method description of source paper and the assumed complexity of the implementation. One of these methods is based on dynamic programming and the other

is based on Q-learning. Dynamic programming is chosen, as this method is used in different traffic signal optimization strategies, like RHODES, PRODYN and OPAC. Also dynamic programming is able to achieve the optimal solution in all traffic conditions. Q-learning is chosen as reinforcement learning is a promising approach in optimal traffic signal control, as this technique has the ability to cope with the dynamics of real-time traffic flows, can be applied without any prior knowledge of the environment, no model of the system is required and the curse of dimensionality can be addressed more efficiently. These methods are in depth described in chapter 4 and are implemented in Python in order to evaluate their performance based on simulation in Sumo, as described in chapter 5.

4

Traffic signal control by dynamic programming and by reinforcement learning

In this chapter traffic signal control by dynamic programming and by reinforcement learning is given. In section 4.1 the problem description is described which is defined as follows: for a given state, what stage should be activated next such vehicles in the network experience the lowest amount of cumulative vehicle delay. In order to solve this problem with either method, the problem need to be reformulated in the following elements: stage, state, action and reward. First a general description of these elements is given. Followed by an explanation of the methods themselves, including the specific implementation of the stage, state, action and reward. At last is a step-by-step implementation of the method in pseudo code. In section 4.2 this is done for the dynamic programming and in section 4.3 for the Q-learning method.

4.1. Problem description

A number of terms used in this description are given first:

Action: The process in which the agent makes the decision of which stage to activate after the current action has ended.

Learning process: Process in which the Q-values are updating repeatedly until they reach convergence.

Incumb: incumbent, the currently known best feasible solution

Phase: time interval during which a specific light, corresponding to a specific movement has a specific colour, e.g. green, yellow and red phase.

State: Store of information extracted from the environment needed for the algorithm to perform the optimization.

Optimal Policy: State-action mapping that results in the highest accumulating rewards.

Stage: Group of non-conflicting movements of an intersection that share a simultaneous green phase.

Reward: Valuation of a performed action, in order to compare one action to the other.

The traffic signal control problem is defined as follows: for a given state s, what action a should be performed (the traffic signal stage activated next, after the current action has ended) in order to achieve the lowest amount of cumulative vehicle delay for vehicles currently in the system.

Adaptive dynamic programming and reinforcement learning are both suitable techniques to solve the optimization of the traffic signal control problem. The agent is selecting actions, thus performing the tasks of the signal controller. The environment is represented by the state of traffic. Actions performed by the agent are visualised by the traffic signals. The elements of the traffic signal control problem are further explained in the following sections, if different representations are used among the used methods, then these are described separately.

4.1.1. Stage

A stage is defined as a combination of non-conflicting movements at an intersection that share a simultaneous green period. The stage set (figure 4.1) contains the available stages. At every stage a number of non-conflicting directions at the intersection share a simultaneous green phase. As can be seen from this figure each movement can only experience a green phase at one stage only. It is also possible to combine green phase of individual directions along different stages, this is not considered in this research since due to the complex relation of this implementation in combination with a fully dynamic stage order. The stage set is equal for all intersections in the network, whereas the stage composition is normally designed for each intersection individually based on the demand, but for the sake of simplicity this is not taken into account, this could be considered in future research.

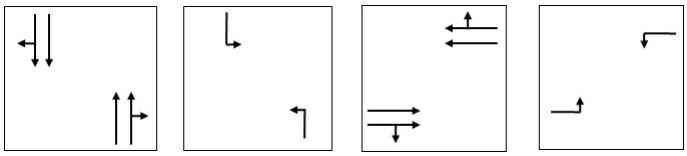

Figure 4.1: Intersection driving directions per stage.

4.1.2. State

The state of traffic at the intersection is described in the state, each decision taken by the agent leads to a new state. The state should contain sufficient information of the current state such that a particular state is unique to the condition of the environment. As the state is characteristic to the method, a further description is provided in the corresponding methods section.

4.1.3. Action

After the current state of the environment has been observed, the agent selects an action from the action set, which results in the corresponding stage to be activated next, this is referred to as an action. The action set is defined as a vector that contains all possible stages.

4.1.4. Reward

After the current state has been observed and an action has been chosen and carried out, a response from the environment is returned in the form of a new state. In order to value the action compared to the other possible actions a reward is defined. This reward is calculated based on the reward definition used by the method. The goal of the agent is to achieve the highest value of cumulative rewards, which represent the optimal action selection strategy.

4.2. Dynamic programming

In this section a forward dynamic programming, as described in [26] to solve the optimization of the traffic signal control problem is further explained. First the symbols used in this method are listed, followed by a description of the method and the different components. At last a step-by-step description of the optimization algorithm is given.

4.2.1. Symbols

k: integer that represents the current discrete time step, is used in the optimization process. Has a range of $k : 0, .., H$

k_{step}: integer that reflects the number of discrete time steps a decision is in effect and no other decisions can be made, depends on the action type, e.g. extend the current stage or end the current stage and activate any other stage

H: integer that represents the horizon, which is maximum value of k in the optimization

i: index that represents the current stage

l: index that represents a lane

$L(i)$: set of lanes that belong to each stage i, each lane is assigned to a specific stage i and could hold multiple movements (in case of the right turn direction), no movements are activated in multiple stages

N: integer that represents the number of different stages in the stage set of an intersection

S: vector that holds all relevant information to describe the state

s_i: integer, part of the state vector that corresponds with a specific stage i.

q_l: integer that represents the number of queued vehicles on lane l

Ar_l: integer that represents the number of arriving vehicles to the queue in lane l

Dep_l: integer that represents the number of departing vehicles from the queue in lane l

a: index that represents the action, it is the stage that should be in effect in the next time step

v: index that corresponds with a specific vehicle

CD^v: integer that represents the vehicle cumulative delay, is the total time spent by this vehicle v in a queue

k_{MG}: integer that represents a predefined amount of the minimum green time

k_{YR}: integer that represents a predefined amount of yellow and all red time

$k_{\triangle}$: integer that represents a predefined amount of extension time, if a current stage is maintained activated

f: integer that represents the value function, which is summation of CD of all vehicles at the intersection

$Incumb.$: integer that represents the incumbent, which is the current best feasible solution

P: set that contain the permanent states, states with $k = H$ and $f \leq Incumb$

T: set that contain the temporary states, containing all temporary states with $k < H$

4.2.2. Method description

The dynamic programming method implemented in this chapter is based on the ADPAS algorithm, described in [26]. This adaptive optimal planning algorithm solves the optimal traffic signal control problem based on a forward reaching dynamic programming technique, with the objective to minimize the cumulative delay experienced by vehicles travelling through the intersection.

Dynamic programming can be implemented as a forward or as a backward approach. The required calculations are equal in both approaches. Backward dynamic programming starts at the final state, which is the state at the end of the time horizon, thus with $k = H$ and moves backward to the initial state with $k = 0$. Forward dynamic programming starts at the initial state, with $k = 0$ and moves forward through the decision tree, until the final state, with $k = H$, at the end of the time horizon has been reached.

The backward approach requires complete prior knowledge of all the states in the decision tree, prior to solving the problem, making it computationally inefficient. As a backward dynamic approach finds the previous best state and propagates backward until it has reached the initial state of the problem. Therefore it is not possible to optimize towards the next state, as the approach does not know this state yet. A forward approach, on the other hand is generating new states as the algorithm proceeds and is able to remove inferior partial solutions from the state-space, this removes the computational burden of searching the entire state-space prior to algorithmic implementation. Thus a backward recursion approach is not computationally efficient for solving the traffic signal optimization problem.

In the optimization process, the forward dynamic programming moves forward through the decision tree, from the initial state at $k = 0$ until the final state of the decision tree has been reached at time step $k = H$. In the decision tree the algorithm moves from one decision point to the next one. The decision taken at each point is defined as: which stage will be activated until the next the next decision point. The length of decision depends of the type of action taken, different decisions lead to different values of k_{step}, based on the action type. If the action is to maintain the current stage as activated, the length of this decision is $k_{\triangle}$. On the contrary, if a different stage is selected to be activated next, the predefined amount of yellow-red time k_{YR} to clear the intersection followed by the predefined minimum green time k_{MG} for the new green phase is imposed.

$$t_{step} = \begin{cases} \triangle & if\ a = i(k) \\ k_{YR} + k_{MG} & if\ a \neq i(k) \end{cases} \tag{4.1}$$

Each decision leads to a new state, the state should incorporate sufficient information about previous decisions such that the decision tree can be completed from this point. The following information is stored in the state definition: current discrete time step in the optimization k, current activated stage a, value function f and the maximum number of vehicles of each stage $s_i, \quad i \in \{1,..,N\}$. Thus if a 4 stage intersection is assumed, the state notation at discrete time step k, $S(k)$ is:

$S(k) = (k, i(k), f, s_1(k), s_2(k), s_3(k), s_4(k))$

In figure 4.2, a schematic illustration is given on how the components s_i are determined. As there are $N = 4$ available stages at the intersection, there are also 4 components. Each of these components represents the maximum number of vehicles present on a given lane l belonging to this stage. For example: Stage 1 contains 4 lanes l in the lane set L. The queue of vehicles present on each lane is defined as: $q_1,...,q_4$ in number of vehicles. As only the maximum value of this set is taken into account in the state notation the value of s_1 is determined as follows: $s_1 = \max[q_1,...,q_4]$

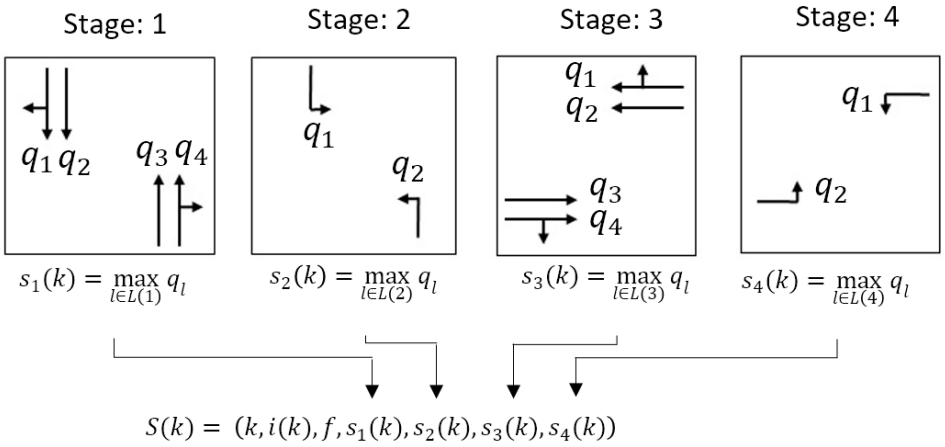

Figure 4.2: State determination for the dynamic programming method

As stated above, the objective of the algorithm is to minimize the cumulative delay experienced by vehicles travelling through the intersection. The cumulative delay is the total delay a vehicle experiences between arriving at the queue until this same vehicle departures from the queue. The value function f is the total sum of the vehicle delay induced up to the time step k. As the increment in the value function f is a direct effect of the active stage at the intersection, it can be seen as the total cost of moving to a state. As each decision gives a stage passage over the intersection, while other stages endure a red phase, during which vehicles on that stage are queued.

In figure 4.3 an illustration of the different new states obtained from the different actions is given for an intersection with three stages, A,B and C with $k_{MG} = 2$, $k_{YR} = 1$ and $k_{\triangle} = 2$. In this example the state notation is: $S(k) = (k, a, f, s_A, s_B, s_C)$, as there are three stages instead of 4. In the initial state, stage C is activated and there are no vehicles present at the intersection, thus the maximum queue length per stage is 0. At this decision point there are three different actions: switch to stage A, B or remain in stage C, thus three new states can be obtained. If the action is to terminate the stage C and switch to either stage A or B, $k_{step} = k_{YR} + k_{MG} = 3$, thus $k = 3$ for the new states obtained with this action. As can be seen in the figure, the vehicles travelling on a lane belonging to stage C during this action are stopped and account for a cost of 5 for both states. If the action is to remain in stage C, $k_{step} = k_{\triangle} = 2$, thus $k = 2$ in the new state obtained from this action. The vehicles travelling on a lane belonging to stage C can pass through the intersection as stage C is still activated. In the next optimization step the algorithm selects one of the new states and performs the same procedure as described above. This results in 9 new states. This process continues until the end of the horizon has been reached for all states. Note that the two states $(6, C, 14, 1, 3, 0)$ (displayed in red) are equal and can be aggregated into one state.

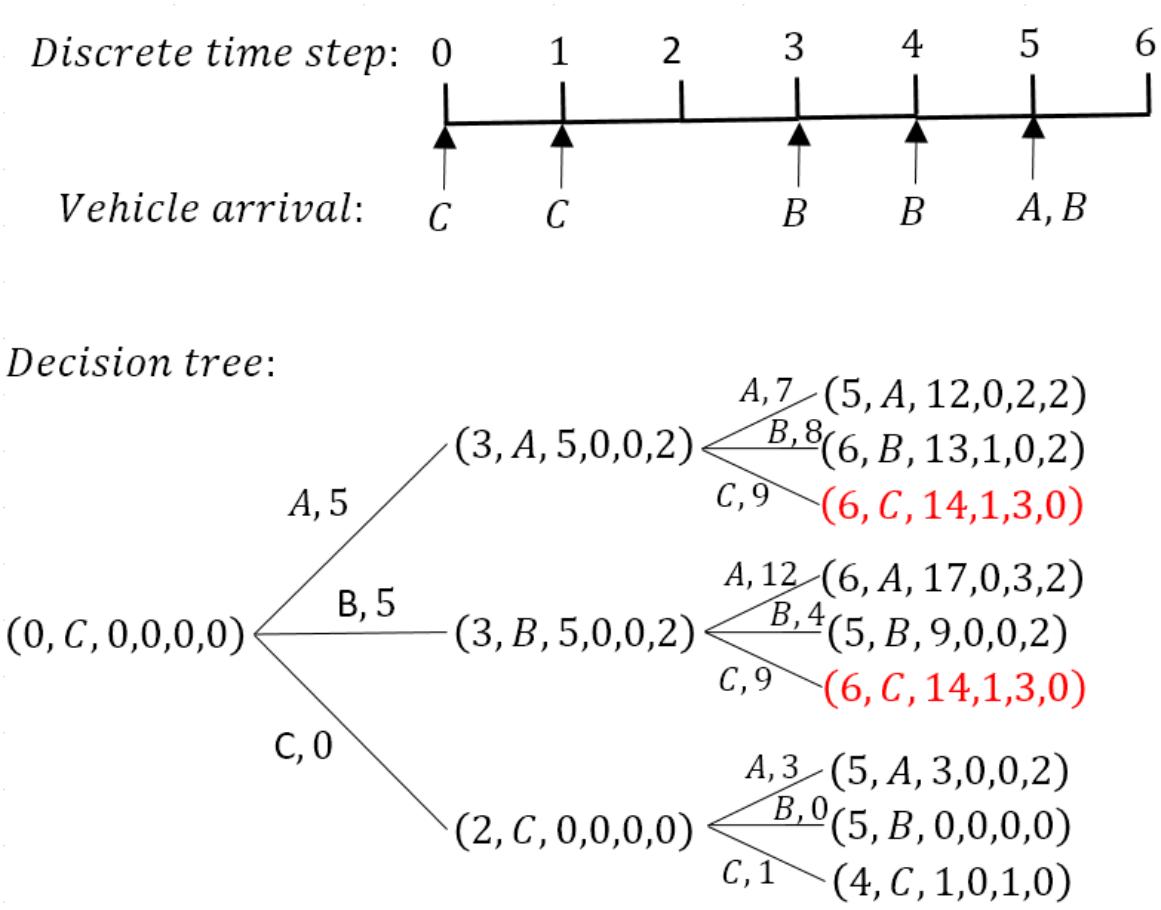

Figure 4.3: Part of the decision tree for example

In order to speed up the performance of the optimization process, and thus reducing the computation time, two acceleration techniques are included into the reaching method, to speed up performance. The first acceleration technique is based on maintaining an incumbent throughout the optimization process. This value is the minimum value of the value function of a state obtained at the end of the horizon, thus with $k = H$. The incumbent is used to eliminate any state with a value function greater than or equal to the incumbent value. As this state is unable to produce a result with a value function lower than the incumbent.

Another form of acceleration is possible by comparing two states of the same phase at the same time. If the first of the two states has a higher or equal number of maximum queued vehicles, with at the same time a higher or equal value function compared to the latter. The first state can be eliminated without a loss of optimality, since under the same circumstances, more vehicles on one phase mean more delays in the future. This way of eliminating states will accelerate the algorithm by pruning inferior states before these are investigated. These features reduce the state space and therefore reduce the required computation time.

Environment modelling

Although the decision tree moves forward from one decision point to the next with increments of k based on k_{step}, the environment itself has to be modelled for every discrete time step k (in the order of 1 second). During this modelling vehicle arrival, presence and departures from each queue for each lane at the intersection is modelled. This information is required in order to determine the cost of moving to the next stage and the value function f.

An example is given in which this modelling process is further explained. This example is illustrated in figure 4.4 and the variables involved in this representation are given in table 4.1.

In the left image the current time step is $k = 0$, a queue with a length of 4 vehicles is present on lane 1, another vehicle is approaching this queue and on lane 2, a vehicle has just entered the network. As there are currently 4 vehicles in the queue, they account for 4 seconds of total vehicle delay per time step. No vehicles

are arriving at the end of the queue as the approaching vehicle is still travelling at free flow speed, nor are there vehicles departing as the traffic signal is red.

In the middle image the time step has progressed to $k = 1$, the 4 vehicles in the queue have accounted for another four seconds of delay, which adds up to the total delay already experienced by these vehicles, thus a total of 8 seconds. As the approaching vehicle has now entered the queue, 1 vehicle has arrived into the queue. The total number of vehicles in the queue are now 5. The vehicle in the middle streaming lane is driving at free flow speed and is not in a queue and does therefore not accumulate delay.

In the right image, the time step has evolved to $k = 2$, the traffic signal has turned green, which means that queued vehicles are now accelerating. 1 vehicle has crossed the stop line, indicated by 1 departed vehicle, as there are still 4 vehicles in the queue, another 4 seconds is added up to a total of 12 seconds of total delay.

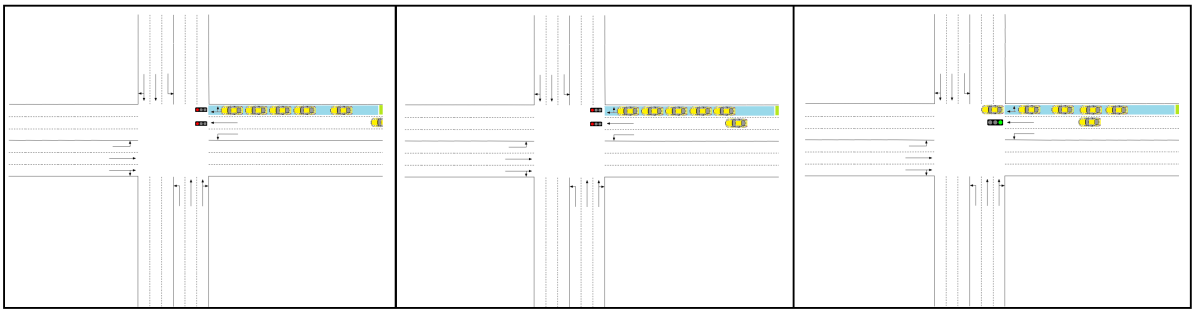

Figure 4.4: Figure of the intersection

Table 4.1: Variables complement to figure 4.4

Image	Left	Middle	Right
time step: k	0	1	2
value function: f	4	8	12
queue length lane 1: q_1	4	5	4
queue length lane 2: q_2	0	0	0
Arrival lane 1: $Ar._1$	0	1	0
Arrival lane 2: $Ar._2$	0	0	0
Departure lane 1: $Dep._1$	0	0	1
Departure lane 2: $Dep._2$	0	0	0

The main goal of the environment model is predicting how the queues per lane develop during the optimization process. In [26] several assumptions have been made regarding this modelling process. These assumption are: vehicles have an instant acceleration and deceleration, thus a vehicle is either travelling at free flow speed or it is standing still, all possible headway constraints are ignored and the entire queue of vehicles leaves the intersection as soon as the green light is turned on. These assumptions will conflict with the microscopic simulation model, for example: Headway constraints are taken into account in the car following model and vehicle characteristics, instant acceleration and deceleration conflict with normal vehicle behaviour and an instant queue dispersion will result in unrealistic vehicle flows. Therefore a different environment model had to be designed, which would function without the assumptions made in [26].

The model receives a vehicle arrival vector from the simulation environment prior to the optimization process. This vehicle arrival vector is used to determine the vehicle arrivals at any $k : 0, .., H$. As it is too complex to model vehicle deceleration precisely, it is assumed that vehicles drive at the free flow speed, until they reach the tail of the queue. As this assumption does not reflect the deceleration behaviour vehicles are considered to be earlier in the queue as they in reality are. The location of the tail is determined by the number of vehicles in the queue divided by the average vehicle length. After vehicles have arrived in the queue (reflected as Ar_l), they start to accumulate delay. If a stage is activated, queued vehicles start to depart from the queue, reflected as Dep_l, based on an assumed saturation flow value.

In this implementation a saturation flow of 1800 vehicles/hour is assumed for all directions. This is equal to 1 vehicle departure every two seconds of green time. The queue length for the current time step is determined based on equation 4.2

$$q_l(k) = q_l(k-1) + Ar_l(k) - Dep_l(k) \tag{4.2}$$

4.2.3. Modifications

As recommended by the author the end-of-horizon effect, should be further addressed when implementing the method. In this subsection the modification towards the original ADPAS method with respect to the end-of-horizon effect is explained. As the value function is determined based on the total delay experienced by all queued vehicles, it does not take remaining vehicles into account at the end of the horizon. As ignoring the remaining vehicles could lead to a sub optimal result, when the optimization horizon H is shorter than the horizon of the system dynamics, selecting the optimal solution should therefore not only be determined based on $f = Incumb.$ but also on selecting the solution with the lowest amount of the total maximum number of vehicles remaining at the intersection per stage, defined as:

$$\min \sum s_i \quad \forall i \in \{1,2,..,N\}$$

4.2.4. Algorithm

In this section the optimization algorithm as applied in the dynamic programming method is described. First the different steps of the algorithm are explained in text followed by a description of the algorithm in pseudo code.

0. **Initialization**: The initialization is the start of a new optimization process. Variables that change in value throughout the optimization are first set to their initial values, such as: $k = 0, f = 0$, $Incumb. = \infty$, $P = \emptyset$.

   The following input is received from the simulation environment: vehicle arrival vector, queue length information of all lanes at the intersection and current activated stage. Based on the input the initial state as: $S(0) = (k, i(0), f, s_1(0), s_2(0), s_3(0), s_4(0))$. The initial state is the only state present in set T at the initialisation.

1. **Stopping Criterion**: any state in the temporary set T is eliminated if the value function is equal or greater than the current incumbent value. If the temporary set T is empty, every state in the network is assigned its optimal value function or eliminated and the algorithm is terminated.

2. **State Selection**: from the temporary set T the state with smallest k is selected (if tie, smallest value function; if tie, smallest sum of maximum number of vehicles is used).

3. **Pruning by State Comparison**: the current state is eliminated if the s_i components of the state vector and the value of f are both greater than that of any state in the permanent set P, with the same time and green stage. If the current state is eliminated go to step 1 and continue with the execution of the algorithm.

4. **Reaching**: for every decision in the action set a new state is created and appended into the set T, where the increment in the time step k of the optimization horizon is dependent on whether the action was, to extend the green time of the current activated stage, or to end the green time, proceed to yellow-red and activate any of the other stages. The optimizer models the change in queue length during the interval from the current decision point to the next decision point. Vehicles that are queued on lanes belonging to the activated stage start to drive out from the queue, whereas arriving vehicles arrive to the queue of any lanes in the intersection. The value f is updated in the new phase according to adding the delay of the specific action to the value f of the current state. Next the current state is included into the permanent set P.

5. **Pruning by Incumbent**: the incumbent value of the new state is set to the new state value if this state is at the end of the planning horizon (H) and the value function is lower than the incumbent value. If the value function of the new state is greater than or equal to the incumbent value, the new state is eliminated.

6. **State Aggregation**: every new state created in step 4 is aggregated into one state with any existing state in the temporary set T with the same time, the same green phase, and the same vehicles. If the aggregated state is at the end of the planning horizon, the aggregated state is included into the permanent

set P, else the aggregated state is included into the temporary set T. When this step is completed return to step 1 and start over again.

7. **Select solution and return to simulation environment:** Once the temporary set T is empty, all possible states have either been eliminated or have reached the horizon. From the permanent set P, the solution is selected that meets the following requirements: $f = Incumb.$, in case there are multiple solutions with an equal value of f, the solution is also determined based on $\min \sum s_i \quad \forall i \in \{1,2,..,N\}$. This solution is return to the simulation environment and is executed by the signal controller.

When the algorithm is finished, the optimal solution is selected from the permanent set P, this solution has the smallest value function and shortest queue length at the horizon. The optimal solution hold the phase sequence of the signal plan, which is executed by the intersection signal controller.

Algorithm 1 Dynamic programming algorithm: reaching and pruning

Step 0: Initialization
$k = 0$, $f = 0$, $Incumb = \infty$, $P = \emptyset$, $S(0) = (k, i(0), f, s_1(0), s_2(0), s_3(0), s_4(0))$, $T = S(0)$
Step 1: Stopping criterion
for any state in set T **do**
    if $f >$ Incumb **then** delete state from set T
    EndFor
while $T \neq \emptyset$ **do** ▷ While the temporary set T is not empty
    Step 2: State Selection
    is selected from set T with lowest value of k and is called the current state, $(k, i(k), f, s_1(k), s_2(k), s_3(k), s_4(k))$.
    Step 3: Pruning by state comparison
    if $f, s_1(k), s_2(k), s_3(k), s_4(k)) \geq$ any state in P with equal value of k and i(k) **then** eliminate the current state
        return to step 1
    EndIf
    Step 4: Reaching
    For every decision of the current state → create new state
    Include the current state into P
    Step 5: Pruning by Incumbent
    for Any state generated in **Step 4 do**
        if k = H and f ≤ Incumb, Incumb ← f.
        if f > Incumb, eliminate the new state.
    EndFor
    Step 6: State Aggregation
    if in set T a state is found with the same time, the same green phase, and the same vehicles as the new state, created in **Step 4**, these states are identical and are therefore aggregated into one single state.
    if k < H, include the aggregated state into set T
    if k = H, include the aggregated state into set P
    go to **Step 1**
EndWhile
Step 7:
select state with $f = Incumb$ and $\min \sum s_i \quad \forall i \in \{1,2,..,N\}$
Return solution to simulation environment

4.3. Single agent Q-learning

In this section a single agent Q-learning as described in [13] with an ϵ-greedy action selection approach to solve the optimization of the traffic signal control problem is further explained. First the method is fully described, followed by a step-by-step description of the algorithm.

4.3.1. Symbols

s: vector that holds all relevant information to describe the state

a: index that represents the action, it is the stage that should be in effect in the next time step

s': vector that represents the new state obtained from performing action a at state s.

(s,a): combination of the state and a specific action that represents the state-action pair

$r(s,a)$: real number that represents the reward for performing action a at state s

A: set of the available actions

$Q(s,a)$: real value that represents the Q-factor which reflects the expected reward for state-action pair (s,a)

α: real value that represents the learning rate

γ: real value that represents the discount rate

ϵ: real value that represents the possibility that an action is greedy selected

$V(s,a)$: integer that represents the number of times a state-action pair has been visited.

N: integer that represents the number of stages

b: integer that represents the number of intervals in discretization

d: integer that represents the interval limit for defining interval b

4.3.2. Method description

Q-learning is the most common single agent reinforcement learning algorithm, as stated in [42]. The goal of this method is to learn a policy, which tells an agent which action to take under which circumstances. Q-learning is a model free approach, thus no model of the environment is required and instead rewards can directly be measured from the environment. The optimal policy is learned by mapping an environment's state s to the best action a based on accumulating rewards $r(s,a)$. Every state-action pair (s,a) has a value called the Q-value, that represents the expected reward for the action a performed in state s. All Q-values are stored in the Q-table, where initially these values are zero, but as the agent is observing the current state s and selecting an action a from the set of available actions A, the Q-values are updated according to the reward $r(s,a)$ and the maximum expected Q-value in the state of the next decision point s' as follows:

$$Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma \max_{a' \in A} Q(s',a')] \tag{4.3}$$

Where $\alpha, \gamma \in (0,1)$ are the learning rate and discount rate, respectively. The learning rate decreases as a state-state action pair gets visited by the agent during a simulation run, the value of α has an initial value of 1 and decreases according to $\frac{1}{V(s,a)}$, where $N(s,a)$ is the number of visits during the current simulation. As state-action pairs got visited frequently the value of $Q(s,a)$ converges towards its optimal value. The discount rate describes the influence a future reward has compared to the immediate reward, as the value of γ gets closer to 1 the Q-value $Q(s,a)$ is more heavily influenced by the reward in the future.

Actions could simply be chosen by selecting the greedy action at each decision, based on the stored Q-values. In the greedy approach, the action is selected as:

$$a = arg\max_{b \in A}[Q(s,b)] \tag{4.4}$$

However, the optimum Q-value can only be found if all the state-action pairs have been visited infinite number of times by the agent, as stated in [42]. This means that the agent cannot simply go for a greedy action selection approach all the time. At first the state-action space should be explored enough, in order to prevent a poor reward from locking in and get favoured over undiscovered actions from that point on, despite these action are not necessarily the optimal ones. Since it would be too time consuming to visit every state within the state-space an infinite number of times, the exploration of random actions and the exploitation of the best actions in Q-learning has to be balanced, this is achieved by using an ϵ-greedy action selection strategy. The different design elements of the implemented Q-learning method in terms of state, action, reward are described in the next sections.

State

In the Q-learning method the state is represented by a vector of $1+N$ components, where N is the number of stages present at the intersection. The first component is the index of the current activated stage i. The remaining N components are the maximum queue length associated with each stage.

$s_i = \max_{l \in L(i)} q_l \quad i \in \{1,2,..,N\}$

Instead of using the continuous variable for the maximum queue length associated with each stage, a discretization is applied, as proposed in [14]. The goal of this discretization is to reduce the number of values a continuous variable assumes by grouping them into b number of intervals. Each interval is bounded by one or more interval limits d. The goal of applying a discretization is to reduce the state-space, this is helpful to temper the curse of dimensionality and speed up policy convergence.

However, no further elaboration is given on how to select the number of intervals b and how to decide on the interval limits d. This is commonly known as the two key problems in association with discretization. The number of intervals and the interval limits that define them result in a state-space reduction and consequently information-loss because of this reduction. Therefore a balance need to be found otherwise the performance of the method is negatively affected. In this research the number of intervals b and their limits d are determined based on an evaluation performed in section 5.3.2.

Table 4.2: Queue length interval per discretisation

Interval b	**0**	**1**	**2**	**3**	**4**	**5**
Through - stages	$s_i = d_0$	$d_0 < s_i \leq d_1$	$d_1 < s_i \leq d_2$	$d_2 < s_i \leq d_3$	$d_3 < s_i \leq d_4$	$s_i > d_4$
Left turn - stages	$s_i = d_0$	$d_0 < s_i \leq d_1$	$d_1 < s_i \leq d_2$	$d_2 < s_i \leq d_3$	$s_i > d_3$	

Action

The action a is the stage i, selected from the set of available stages that is in effect next, as defined in equation 4.5. If the action is the same as the current activated stage, the green time for that stage is extended by 1 second. Otherwise, the green light will be switched to stage a after accounting for the yellow, all red and minimum green times. Therefore the decision point varies according to the sequence of actions taken.

$$a = i, \qquad i \in \{1,2,...,N\} \tag{4.5}$$

Note that switching to another stage does not follow any predefined order. Therefore it is possible to skip unnecessary stages, as well as extending the current stage length for as long as required. Hence, the action is the stage that should be in effect for the next interval. The length of this interval is either the extension time or the required minimum yellow-red and minimum green time.

The action selection is based on the ϵ-greedy method, described in [40], in which the probability that an action is greedy selected increases with the number of iterations performed, otherwise a random action is chosen uniformly. The value of ϵ is increasing gradually from (0.0 to 1.0) and is determined by the following equation and a plot of this function is displayed in figure 4.5

$$\epsilon = 1 - \exp^{-0.05*n} \tag{4.6}$$

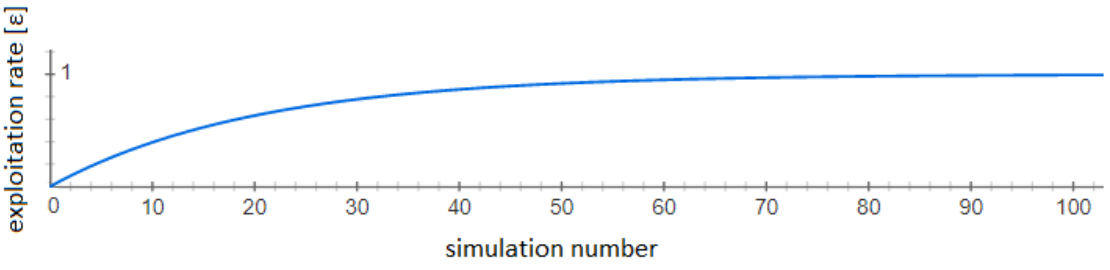

Figure 4.5: Exploitation rate function

In which n is the current simulation number. This results in more exploration at the beginning of the learning process which enables the Q-learning agent to visit the overall state-action space and gradually change the action selection to exploitation as the agent converges to the optimal policy.

Reward
The reward is defined as the change in total cumulative delay of all vehicles currently in the system between two successive decision points. The vehicle cumulative delay CD^v is the total time spent by this vehicle v in a queue. The cumulative delay for stage i is the summation of all the vehicles present on $L(i)$. The total cumulative delay CD_{total} of the intersection is summation of all the stages present at the intersection.

$$CD_i = \sum_{v \text{ travelling on l} \in L(i)} CD^v \qquad \forall i \in \{1,2,..,N\} \tag{4.7}$$

$$CD_{total} = \sum CD_i \qquad \forall i \in \{1,2,..,N\} \tag{4.8}$$

Once a vehicle fully passes the stop line, it is no longer considered to be in the system and the cumulative delay (CD^v) accounted for this vehicle is removed from the value of the total cumulative delay. If the total cumulative delay has decreased between two decision points, the rewards has a positive value, on the other hand if the cumulative delay has increased between two decision points, the reward has a negative value.

Q-table
Q-values of the entire state-action space are stored in the Q-table. Initially all values are zero, and updated according to the Q-value update rule. In table 4.3 an example Q-table is displayed. For each state N different stages could be active, reflected in the first column. For each state N different actions a can be taken, such that:

$$Q(S,A) = [Q(S,a_1),..,Q(S,a_i)] \qquad i \in \{1,2,...,N\} \tag{4.9}$$

Table 4.3: Example Q-table

State					Actions			
active stage:	s1	s2	s3	s4	a1	a2	a3	a4
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0

4.3.3. Interaction
Interactions between the agent and the simulation environment include the exchange of the environment state, the reward and the agent action. The environment state and the reward are directly measured from the simulation environment. The action is selected by the agent and exchanged to the environment by changing the traffic lights at the intersection. The interaction frequency is variable and occurs at every discrete simulation step of 1 second, as long as the current green for a signalized intersection exceeds the minimum green time. Otherwise the interaction starts after the minimum green time has passed.

4.3.4. Algorithm
In this subsection the different steps in the algorithm are explained, followed by a representation of the algorithm in pseudo code in algorithm 2.

- **Step 0: Initialisation**
  At the first simulation run the Q-table is created, the size of the Q-table equals the state-space and action-space, thus matrix size n x m, with n = set of states, m = set of actions. Initially the Q-values are set to zero, and values change during the learning process when $Q(s,a)$ is updated. At every new simulation, the state-action visits table ($V(s,a)$) in which the number of times a state-action is visited gets tracked is set to zero, the size is n x m, with n = set of states, m = set of actions

- **Step 1: Observe state** S
  The current state of the traffic conditions at the intersection is observed. This is also the state of which the Q-values are updated at the next interaction moment.

- **Step 2: Select action** a
  The agent selects an action based on the ϵ-greedy action selection strategy, thus based on either exploration or exploitation of the state-action space, from the action set and executes it for the next interaction interval.

- **Step 3: Update Q-values**
  The consequence of the action taken in step 2 is observed. The change in total cumulative delay during the interaction interval r is measured, as well as the next state s'

- **Step 4: Determine** α**-value**
  Update the value of $V(s,a)$, according to $V(s,a) = V(s,a)+1$ and determine the learning rate α based on the number of visits per state-action pair $\alpha = \frac{1}{V(s,a)}$, this value is affects in the Q-value update rule, used in step 5.

- **Step 5: Update Q-value**
  $Q(s,a)$ is updated based on the Q-value update rule, and the algorithm is repeated at step 1.

Algorithm 2 Q-learning

Input: Q-table, State
for simulation in number of simulations **do**
  Step 0: Initialisation
  if simulation number = 0 **then**
    $Q(s,a)$ is set to all zeros
  else
    Use $Q(s,a)$ from previous run
  EndIf
  $V(s,a)$ is set to all zeros
  for each interaction in simulation **do**
    Step 1: Observe state s**:**
    Step 2: Take action a **according to action selection strategy**
    Step 3: Observe r**,** s'**:**
    Step 4: Update $V(s,a)$ **and Determine** α**:**
    $V(s,a) = V(s,a)+1$
    $\alpha = \frac{1}{V(s,a)}$
    Step 5: Update Q-value:
    $Q(s,a) = (1-\alpha)Q(s,a) + \alpha[r(s,a) + \gamma \max_{a \in A} Q(s',a)]$
  EndFor
  Output: return stage that should be activated next to simulation environment
EndFor

4.4. Conclusion

In this chapter the various components included into the optimization methods in order to perform the task of traffic light control optimization are given. Both optimization methods are described and presented in pseudo code and their components are explained. The major difference between the methods in terms of implementation is that the dynamic programming method requires a model to predict the environment conditions during the optimization horizon. A new traffic flow model had to be designed in order to overcome the assumptions that have been made in [26], as these assumptions have an effect on the evaluation with a microscopic simulation model. This model is used to determine the number of vehicles in the queue of all lane on the intersection, as well as vehicle arrival and departures throughout the optimization process. In order to find the optimal solution the dynamic programming method requires a correct reflection of the queue dynamics by this traffic flow model, imperfections in this model will have an effect on the performance of the method.

The Q-learning method is a model free approach, therefore a traffic model is not required. However, the Q-learning method requires a training process in order to derive the optimal policy, this process could be time consuming if the computation power is limited. A point of concern regarding the Q-learning method is the design of discretization variables. Selecting the number of intervals b and how to decide on the interval limits d is considered a key problem in association with discretization. By choosing these variables a balance between state-space reduction (improved computation times) and information-loss should be found, also should the boundary values properly reflect the variation of the maximum queue length s_i, expected traffic demand and the link length of the network, otherwise the methods performance can be negatively be affected.

In chapter 5, the dynamic programming and Q-learning method are first simulated on a testbed intersection. This network is used to fine-tune the variables used in the methods in order to improve performance. After which an evaluation by simulation is performed on two different three intersection networks.

5

Evaluation by simulation based on a general evaluation methodology

In this chapter, first the applied simulation methodology is explained. Followed by a description of the simulation environment and the connection between the environment and the different optimizers. The simulation environment is the network designed in SUMO and the optimizer is the client, which interacts with the simulation environment by sending commands and receiving information. Next the test-bed intersection network is explained, in which the designed methods from chapter 4 are compared to each other and a fixed-timing signal control structure. As the methods performance relies on several parameters, some require tuning during multiple simulation reruns in order to find the right setting and to improve performance.

Following simulations have been performed on two different three intersection networks, with varying distances between the intersections. A number of simulations are performed, from a performance evaluation to the simulation of two different traffic phenomena, which are green-wave formation and spill-back prevention. At last the conclusion of this chapter is presented.

5.1. Evaluation methodology

The purpose of evaluation by simulation conducted in this chapter is to obtain the potential of both the dynamic programming as the Q-learning method. As stated before, this is only possible if the same evaluation methodology is applied. The simulation methodology applied in this chapter is designed based on the information obtained from the literature study, described in chapter 2 and is an example of a general evaluation methodology to evaluate the performance of different traffic signal optimization methods.

When designing the simulation methodology in this research, a balance had to be found between a comprehensive evaluation and the time available. Therefore some components, described in chapter 2 had to be simplified, including: the simulation randomness, number of simulation runs, metrics and different types of scenarios. The simulation randomness is reduced to one vehicle type with the same vehicle properties for all vehicles inserted into the network. Vehicles are uniformly inserted into the network given the specific route demand (vehicles/hour). As the simulation randomness is taken out, the stochasticity does no longer exists. Therefore the pre described minimum number of simulations to address this stochasticity is no longer needed, as every simulation run results in the same outcome if the actions taking during this simulation run are exactly the same. However this cannot be applied to the Q-learning method, as found during the simulations, results could vary between them. Therefore ten simulation runs are performed for all Q-learning involved simulations.

As the Q-learning method requires a real-time obtained metric to measure the effect of its action in the environment and in this way train the agent, a metric had to be used that fits this requirement. At the same time, this metric has to unambiguous, clear and can be extracted from the simulation environment without difficult code (explained in 5.2.1). For this reason the queue length detectors are used to measure the time spend by each vehicle on the detector when in a queue. If the driving speed of a vehicle is lower than, or equal to the queue speed threshold a vehicle is considered in the queue.

The number of different simulation scenarios are reduced to a green wave and a spill-back scenario, as rush-hour and off peak conditions are already captured in the adaptability test. During the adaptability test

a range of demand factors is used, ranging from 0,5 up to 1.5 times the original demand.

The testbed intersection is used, in which the performance with respect to fixed-timing as the reference is measured. The performance is evaluated under different traffic loads, expressed by the demand factor. The multi-intersection network is used to evaluate the performance in different network topologies, also under the same different demand factors. To evaluate the performance, with respect to specific traffic phenomena a green wave and a spill-back scenario are included.

5.2. Simulation environment

The simulation environment are the networks, designed in the continuous road traffic simulation package SUMO, described in [28]. The interaction between the client and the SUMO environment is implemented through the Traffic Control Interface (TRaCI), proposed in [44], as displayed in 5.1. TraCI gives access to a running road traffic simulation, it allows to retrieve values of simulated objects, e.g. intersections or vehicles.

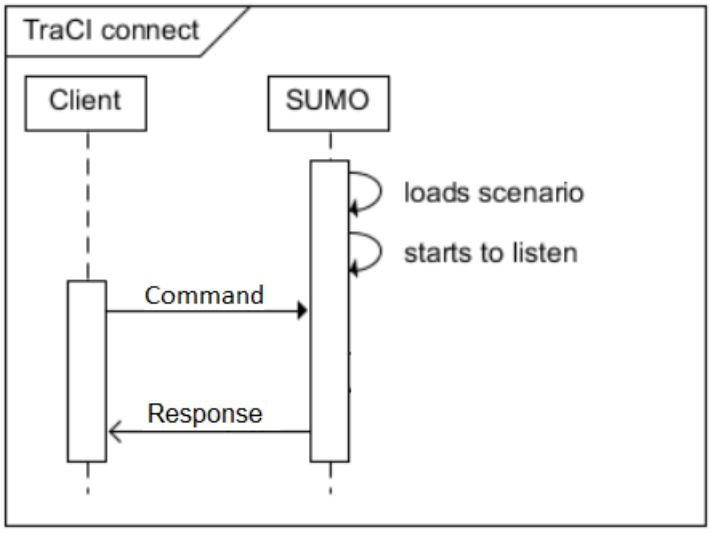

Figure 5.1: TraCI: establishing a connection to SUMO in [44]

The client application sends commands to SUMO to control the simulation run, such as to control the signalized intersections, retrieve environmental details or to advance to the next simulation step. In response SUMO answers to each command and returns information that depend on the given command. Within the client one of the methods described in chapter 4 is executed in order to determine the traffic light control commands, based on the data returned by SUMO.

Detectors are located in the network and retrieve information at every simulation step (0,1 second), such as: queue length, vehicles present in the queue and detection of vehicle arrival and system departure. The following different detectors types are:

- queue detectors: these detectors are located longitudinal on the steaming lane and provide queue information such as queue length and vehicle presence, this information is required in the current state of the environment.

- vehicle arrival detectors: these detectors are located at some distance from the stop line and are required to establish the vehicle arrival vector, which predicts the arrival process of vehicles during the optimization horizon. This information is required by the dynamic programming method modelling process.

- stop line detectors: these detectors are located at the stop line of a streaming lane and provide information of vehicles leaving the system (intersection), this information is required to update the cumulative delay of the system. This information is required to determine the reward of the Q-learning method and is also used in the cumulative delay metric that evaluates the performance of all methods.

5.2.1. Cumulative delay module

The cumulative delay is used as the metric to quantitatively describe and evaluate the methods performance, the cumulative delay is actively used to reward the action of the Q-learning method during the training process. Also is the cumulative delay used as the metric to evaluate the performance of all methods in this chapter, as described in section 5.1. The cumulative delay of all vehicles currently in the system is derived from the data obtained from the queue length detectors. These detectors receive the number of vehicles and their corresponding ids, for each queue on every lane within the intersection.

It should be noted that the time a vehicle is spending in a queue does not fully reflect the total delay as experienced by a particular vehicle. As the deceleration and acceleration phase, or interaction with other vehicles, also influences the vehicle travel speed and subsequently the experiencing additional delay. By increasing the speed threshold of the queue detectors, vehicles are earlier detected, as their travel speed becomes lower than the speed threshold at an earlier stage and remain detected while they accelerate, as long as their travel speed is below the speed threshold. Thus vehicles are also captured while decelerating and accelerating, which improves capturing range of the time spend in the queue.

The cumulative delay based on queue detector data can be extracted from the simulation environment without the need of complex calculations. This makes this metric a clear and unambiguous metric. Nevertheless, one should take notion that the length of the queue detector is valuable in the process of queue detection, as queues can only be detected if they are located on the detector itself. If queues are exceeding the queue length detector, vehicles that fall outside of the detector are not taken into account in the queue and vehicle data. Therefore queue length detectors should be of sufficient length to prevent the queue from exceeding the detectors range.

Queue detection speed threshold influence on method performance

As stated in the previous subsection, the moment on which a vehicle is detected by the queue detector is based on the speed threshold value. In this subsection a closer look is taken into the effect of this variable on the performance of Q-learning method, as this method does not make use of the arriving vehicle vector. In figure 5.2 the difference in detection distance is illustrated for a speed threshold of 10 m/s and 5 m/s respectively. A vehicle is approaching the stop line, at which a traffic light is red, thus the vehicle is decelerating, as soon the travelling speed of this vehicle becomes smaller than the speed threshold the vehicle is detected and becomes part of the state as a queued vehicle. From this figure it becomes clear that by increasing the speed threshold, a vehicle is earlier detected and therefore the controller can earlier react to the presence of this vehicle. Therefore the traffic light can turn green while the vehicle is 20 m away from the stop line at a speed threshold of 10 m/s instead of 10 m at a speed threshold of 5 m/s. Thus by changing the speed threshold value, vehicles are earlier detected by the queue detectors, the effect on the performance of the method are evaluated next.

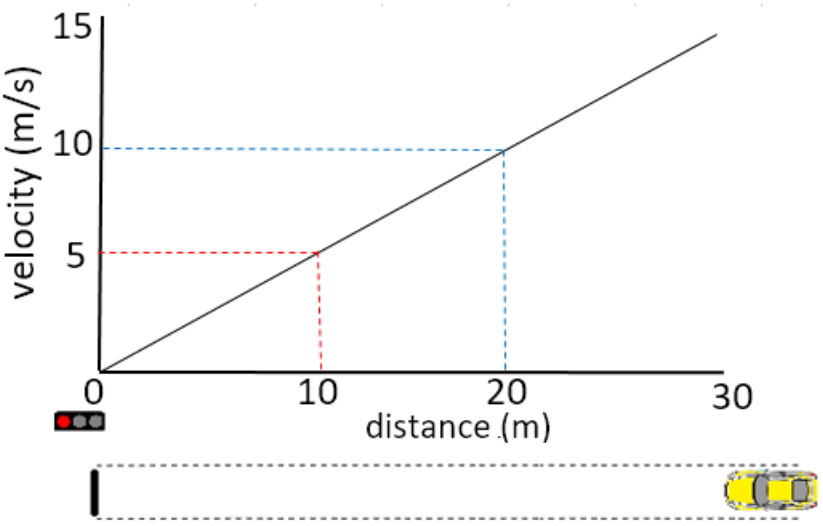

Figure 5.2: Detection distance at different speed thresholds

In figure 5.3 the effect of the queue detection speed threshold value with respect to the average cumulative delay per vehicle, for both the fixed timing as the Q-learning algorithm is displayed. The dynamic programming method predicts the arriving vehicles on its own, without making use of the queue length detectors and is therefore not included in this figure. Since the Q-learning method relies on the queue length detectors for estimating the queue lengths of the different stages, it is interesting to examine the influence of the queue detection speed threshold value on the performance of the method. Fixed timing is used as the benchmark method, as its phase structure is not influenced by the queue length detectors.

In figure 5.3 it can be seen that by increasing the queue detection speed threshold value, the average cumulative delay per vehicle increases when using a fixed timing traffic light control structure. As earlier states, vehicles are earlier detected and therefore will accumulated more delay, thus this increase is as expected. However an increase in average cumulative delay per vehicle is not noticed at the Q-learning method, as these values are even decreasing with increasing speed thresholds. As decelerating vehicles can be earlier detected by the queue detectors, these vehicles do also become part of the state information. As can be seen from the results presented in figure 5.3 this effect has a positive outcome on the performance of the method.

The jump in average cumulative delay per vehicle at a threshold speed of 15 m/s is due to the fact that this value is equal to the driving speed of the vehicles in the network. Therefore all vehicles are considered to be in the queue during the entire time they are present on the detector. Thus the queue detection speed threshold value results in optimal performance at a value of 10 m/s and should always be lower than the traffic free flow speed.

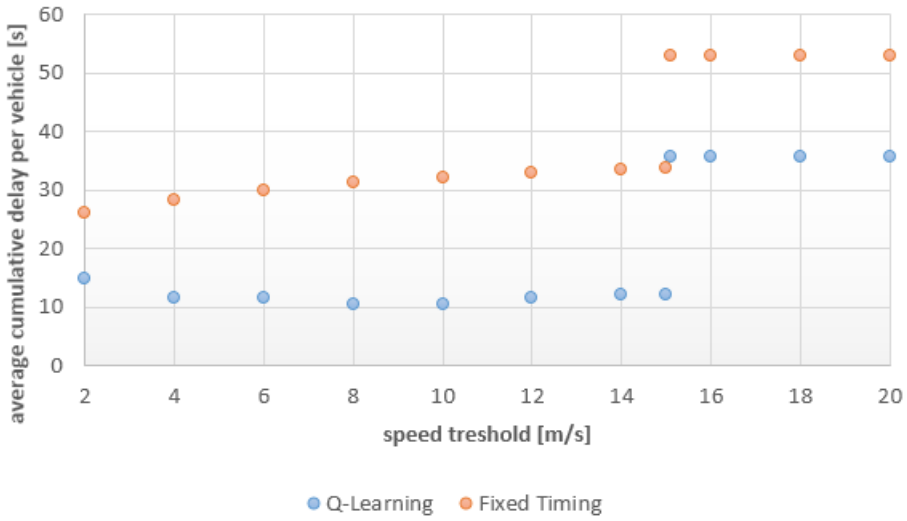

Figure 5.3: Queue length detection speed threshold

Cumulative delay processing

The cumulative delay is processed by the cumulative delay module (3), which is included into the client. Queue length and vehicle information is measured by queue length detectors in the simulation environment and is send to the client. This information is received as a list of vehicle ids by the cumulative delay module. This module checks if a vehicle id is already present in the cumulative delay data, which means that the vehicle was already present in the previous time step. If true, the delay for this vehicle id is increased by 1 second. If a particular vehicle has just arrived in in the queue, its vehicle id does not exist in the cumulative delay data, thus a new vehicle id entry, with a delay value of 0, has to be created. As the cumulative delay should only reflect the sum of vehicle delay of all vehicles currently in the system, departing vehicles should be removed from the cumulative delay data. This action is performed in the final step of the cumulative delay module. As vehicles cross the stop line, they are no longer present in the system. The corresponding vehicle ids are appended into the outflow vehicle data and all vehicle ids are removed from the cumulative delay data.

Algorithm 3 Cumulative delay module

```
for vehicle id list in vehicle IDS do
    Check if vehicle id exists in cumulative delay data
    if True then
        increase vehicle id delay by 1 second.
    else
        create vehicle id entry
for vehicle id in outflow vehicle data do
    remove vehicle id from cumulative delay data
```

5.3. Testbed intersection

The first simulation network is an isolated intersection called the testbed intersection. The purpose of this network is to evaluate the performance of both methods when applied on a single intersection. The performance of both methods is also compared to the results of a fixed-timing structure, which is used as a benchmark. The testbed intersection is also used to verify that both methods are working correctly with the environment and the simulation network is functioning as required. This verification is based on visual observations and retrieved data from both methods. It is less complicated to debug the method's operation on a single intersection compared to a network of intersections.

Testbed intersection topology

The testbed intersection is displayed in figure5.4, this intersection consist of 4-approaches with 3-lanes including an exclusive left turn streaming lane. Right turn movements do not have an exclusive lane and is therefore combined with the trough movement on the right hand lane of the approach. This intersection is chosen as an example of a multi-phase intersection with 4 different stages. The stage set of this intersection is given in figure 4.1, this set equals the possible actions of the controller.

Queue length, vehicle arrival and vehicle departure detectors are located on every lane of all approaching roads at the intersection. Queue length detectors have a length of 300 m each and are located from the stop line reaching upstream, vehicle arrival detectors have a length of 0,5 m and are located 315 m upstream of the stop line, vehicle departure detectors have a length of 0,5 m and are located on the stop line.

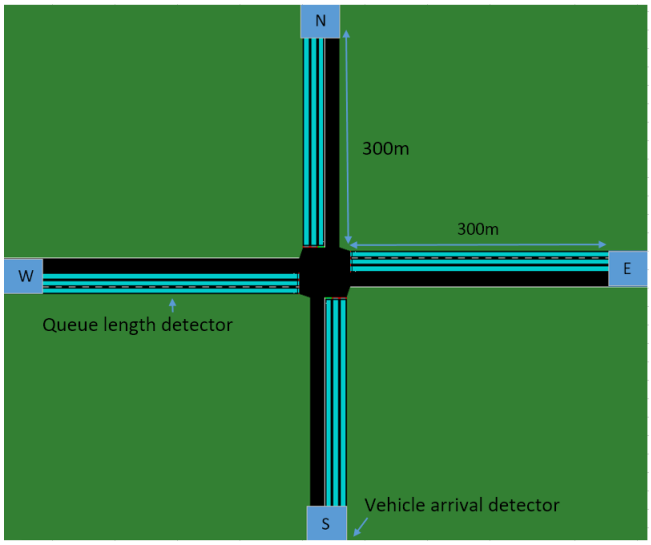

Figure 5.4: Testbed network

The following data is extracted from each detector at every time step (0,1 second), queue length detectors measure the number of vehicles and their corresponding vehicle IDs in the queue, vehicle are considered to be in the queue if their speed is below a certain speed threshold as defined in the configuration files of the simulation.

Vehicle arrival detectors detect vehicles approaching the stop line, in order to estimate a future arrival pattern. This estimation is based on the assumption of a continuous motion at the free flow speed until the

back of the queue has been reached, due to vehicles slowing down the estimated position of the vehicle can be off a few metres. Vehicle departure detectors detect the vehicle IDs that have left the corresponding queues.

The traffic flows (vehicle/hour) of each Origin-Destination pair are given in table 5.1, in the form of an Origin-Destination (OD) matrix. These values are described in [13] and represent the morning rush-hour of a major intersection in Downtown Toronto. The value of each Origin-Destination pair is the number of vehicles that is evenly inserted in the network during the simulation period. As this would result in a uniform vehicle arrival pattern, 4 periods within the 1 hour simulation are defined. During these periods the actual vehicle flows

The traffic flow pattern has a variable profile, this is achieved by using 4 periods of 15 minutes during which the arrival rate is determined based on a variation around the mean arrival rate.

For example: if 100 vehicles / hour are considered, this means that 25 vehicles / 15 minutes would arrive. This could be reflected by the following number of vehicle per period: 20, 25, 30 and 25.

This results in a vehicle flow with a variation for each 15 minute time period, which is more realistic compared to a uniform profile. The variation of each 15 minute time period is generated by a random seed number, which is the same for all simulations for reproducibility reasons and to save on simulation time.

Table 5.1: Trafic volume (Origin-Destination matrix)

	E	**S**	**N**	**W**	Total
E	0	50	50	250	350
S	100	0	750	90	940
N	125	375	0	90	590
W	430	100	100	0	630
Total	655	525	900	430	2510

5.3.1. Dynamic programming

The model used within the dynamic programming method needs to be modified according to the intersection. The total number of lanes used in this intersection is 12. There are four different stages, as given in 4.1. As stated in chapter 4, vehicles depart from the queue with an assumed flow rate of 1800 vehicles/hour. The travel time between the vehicle arrival detector is determined at 20 seconds, thus the vehicle arrival vector holds the vehicle arrivals for the next 20 seconds and this is also the value of the time horizon, $H = 20$. The required computation time to optimize the traffic light control plan over a horizon of 20 seconds is given in table 5.2.

Table 5.2: Computation time

Horizon length [s]	**Min [s]**	**Max [s]**	**Mean [s]**
20	0,0	0,48	0,17

5.3.2. Single agent Q-learning

The single agent Q-learning method is a model-free approach and therefore does not require any modelling to the intersection layout. On the other hand a learning process is required, in order to converge the action selection strategy towards the optimal policy.

Queue length discretization intervals

As a proper function of the Q-learning method relies on the suitability of the chosen queue length intervals. If the chosen boundary values (table 5.3) do not match the encountered queue lengths, the Q-learning method is unable to deal with these conditions adequately.

For instance, if the encountered queue length is always lower than the bottom boundary of one or more discretization intervals, this will lead to these states from ever being used, and at the same time resulting in a larger state-space. Other questions regarding the discretization intervals are, what should be the span between the top and bottom boundary, what value should be used as the top boundary in the last interval and how many intervals should be used. In order to examine the effect of different queue length discretization intervals on the performance, a small evaluation has been performed. In this evaluation different queue length interval compositions have been evaluated. For each composition the Q-learning method learning

process was repeated for 100 simulation runs, a demand factor of 1.0 was used and traffic demand was similar to table 5.1. Results from this evaluation are displayed in table 5.4, as can be seen from these results the first evaluation resulted in an average cumulative vehicle delay of 16 seconds. During the simulation it was noted that the number of vehicles in the queue that could be dissipated during a green phase was far lower than the interval span used in this evaluation, therefore it was decided to decrease the interval span in the next evaluation. By decreasing the interval span in the second evaluation, the average cumulative delay decreased to 13.8 seconds. During this simulation it was noted that the number of vehicles in the queue exceeded the value of 8 vehicles by some margin, thus in evaluation 3 the last boundary was increased, which resulted in an average cumulative delay of 10,6 seconds. This is also the interval composition used in the next simulations.

Table 5.3: Queue length interval

Interval b	**0**	**1**	**2**	**3**	**4**	**5**
Through - stages	$s_i = d_0$	$d_0 < s_i \leq d_1$	$d_1 < s_i \leq d_2$	$d_2 < s_i \leq d_3$	$d_3 < s_i \leq d_4$	$s_i > d_4$
Left turn - stages	$s_i = d_0$	$d_0 < s_i \leq d_1$	$d_1 < s_i \leq d_2$	$d_2 < s_i \leq d_3$	$s_i > d_3$	

Table 5.4: Different interval boundary values with result.

d_0	**d_1**	**d_2**	**d_3**	**d_4**	**average cumulative delay**
0	5	10	20	30	16
0	2	4	6	8	13.8
0	2	4	8	16	10,6

Policy convergence

To proof that the Q-learning method is capable of converging towards the optimum policy, a number of 1 hour simulations is performed in which traffic demand is solely travelling from one OD-pair per simulation and one Q-table has been used throughout the entire series of simulations. A demand of 500 vehicles / hour has been used per OD-pair. A modified epsilon function has been used, since the optimisation of this traffic demand pattern is rather simple and therefore convergence can be achieved much faster without the risk of the agent getting trapped in a sub optimal solution. The following epsilon function has been used:

$$\epsilon = 1 - \exp^{-0,2*n} \tag{5.1}$$

The following OD-pairs are used, OD0: North-South and South-North, OD1: East-West and West-East, OD2: North-East and South-West and OD3: East-South and West-North. In figure 5.5 the results from these simulations are displayed. As can be seen in this figure, the first simulation show higher cumulative vehicle delay as action the agent is exploring, thus selecting actions randomly. Cumulative vehicle delay gradually decreases until the optimum policy for all OD-pairs have been found around simulation number 10, at which cumulative vehicle delay has been reduced to 0,

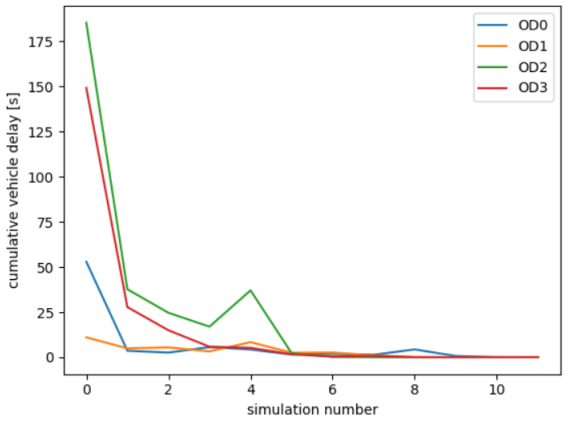

Figure 5.5: Policy convergence with specific OD-pairs

Training process
The system used for performing the learning process is a computer running Windows 10, with 16GB of RAM memory and an i7-7700HQ CPU at 2,8GHz. The learning process consists of 150 simulation runs of 3600 simulation seconds, performed in the gui-mode of SUMO. In total the learning process took 28.123 seconds, thus on average 187,5 seconds per simulation run. The first number of simulations take longer due to the exploration behaviour of the agent, as the behaviour is becoming more exploitation driven the simulation time drops. For every simulation run the OD-matrix described in table 5.1 is used, with a demand factor of 1.0 and with vehicle arrival rates as described in section 5.2.

In figure 5.6 the convergence of the Q-values towards the optimum policy is demonstrated. As can be seen from this figure the average cumulative vehicle delay decreases until a limit has been reached, which resembles the optimum policy. At the start of the learning process the agent explores the state-space as can be seen by the high value of average vehicle delay, as the action selection strategy becomes more exploitative, the average cumulative vehicle delay decrease, until optimum policy convergence has been achieved after simulation run 110, after which deviation in average vehicle delay is minimal. The learning process is terminated after 150 simulation runs.

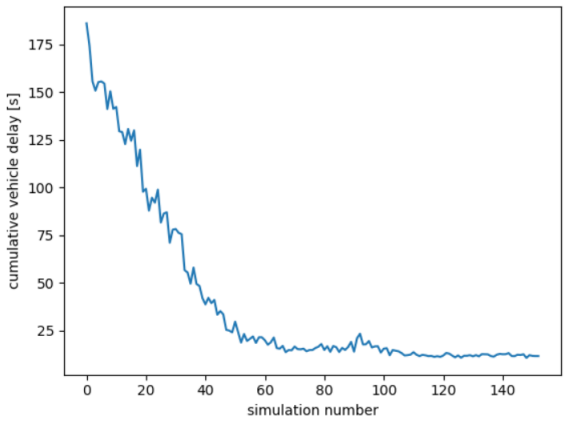

Figure 5.6: Average cumulative vehicle delay with simulation runs

5.3.3. Fixed-timing
A fixed time control plan is used to benchmark the performance of both dynamic programming and single agent Q-learning. A control plan based on a fixed-timing structure is known to perform well at its design demand level. As this control plan lacks adaptability performance will deteriorate for other demand levels. The signal control plan is optimised for the described traffic demand, based on the Webster method [43], using the following parameters: Saturation Flow S=1800 vehicle/hour/lane and a combined Yellow+Red Time of 3 seconds. This results in the following green times: EW: 26 seconds, EW-left turn: 11 seconds, NS: 47 seconds, NS-left turn: 14 seconds.

5.3.4. Performance evaluation
To test the methods general performance, a series of simulations has been carried out. The goal of these simulations is to evaluate how well a method has performed compared with the results of a fixed time structure. Traffic volumes are as described in table 5.1 and to test the adaptability of the different methods towards different demand levels, a factorisation of the base demand is applied. This factorisation ranges from 0,5 up to 1,5 and is used to evaluate the robustness of both methods. As a demand factor of 1,0 resemble rush-hour conditions, the range of demand factors from 0,5 up to 1,0 are considered to be more realistic when considering both off-peak as rush-hour demand levels.

Performance is reflected by the average cumulative delay per vehicle, as this can unambiguously describe the performance of the method.

Average cumulative delay per vehicle is given in equation 5.2 and is calculated as the sum of the cumulative delay per vehicle (CD^v) of all vehicles that have been in the system divided by the total number of vehicles that have been inserted into the system ($N_{Veh.}$), this information is given by the simulation environment. As described in section 5.3, the traffic flows per Origin-Destination pair are kept constant over all the

simulations performed, thus the total number of vehicles are equal for each simulation run, otherwise a fair comparison could not be conducted.

$$\text{average cumulative delay per vehicle} = \frac{\sum CD^v}{N_{Veh.}} \tag{5.2}$$

Results
Simulation results are presented in table 5.5 and figure 5.7. Results are described for each (optimal) signal control method individually.

The results of the fixed-timing have little fluctuation with increasing traffic demand factors. As this signal control strategy is designed at a traffic demand at a demand factor of 1,0, it is expected to perform sub-optimal for other demand factors. Furthermore, as this method is not able to adapt to the current traffic demand, poor performance is expected compared to both dynamic programming and Q-learning. Although it should be noted that the level of influence of this effect upon the performance reduces as an adaptive signal control tends to behave as a fixed-timing plan at high traffic demand.

The results of the dynamic programming optimal control method are quite surprising, as this method is expected to find the optimal solution in all traffic conditions. The difference in performance compared to the Q-learning method increases rapidly at demand factors of 1,0 and higher. Explanations for these results can be sought at different causes, one of the most obvious reasons are several assumptions made in[26], such as instantaneous queue dissolvement and instant stop and go behaviour of vehicles. As these assumptions are not incorporated in this research the functioning of the method itself could be negatively impacted. Solutions designed to overcome these assumptions or in implementing the method in general could contain coding errors or imperfections that could have led to this result.

The results of the Q-learning method are satisfactory, although it should be noted that the average cumulative delay per vehicle increases rapidly for demand factors of 1,10 and higher. An explanation for this behaviour can likely be sought in the queue length discretisation intervals. As the interval range increases from 2 up to 8 vehicles, as can be seen in table 5.4.

Table 5.5: Average cumulative delay per vehicle, in seconds, for different demand factors

Demand Factor	**0,50**	**0,75**	**0,90**	**1,00**	**1,10**	**1,25**	**1,50**
Dynamic Programming	6,9	8,5	11,6	20	29,3	49,5	67,4
Q-Learning	5,2	6,7	9,2	10,4	13,7	22,2	44,1
Fixed-Timing	23,6	25	26,2	26,7	27,2	28,3	31,8

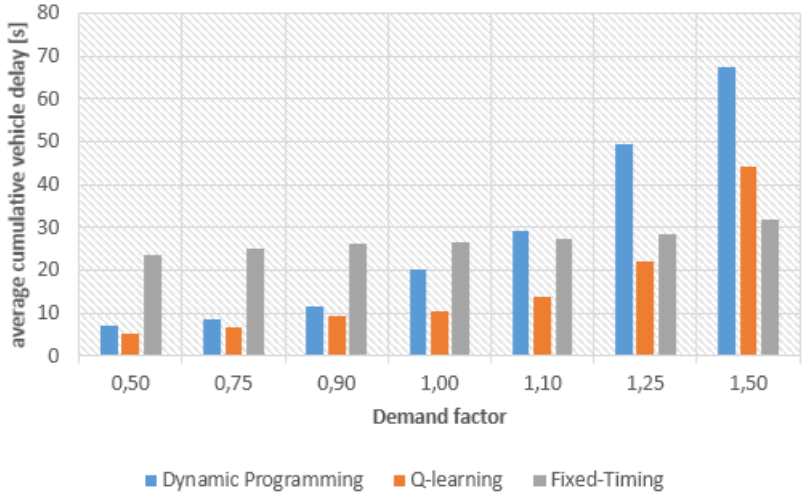

Figure 5.7: Average cumulative delay per vehicle, in seconds, for different demand factors

5.4. Multi-intersection networks

Network 1
The first multi-intersection network, displayed in figure 5.8 consists of three identical intersections, with a similar layout as described in section 5.3. The outflow road to the South and North of the intersection is a two lane road for a length of 50 metres, after which the road has three lanes for a length of 400 metres until the stop line of the intersection. The side roads, East and West of the intersection, approach the intersection with three lanes for the entire length of 400m and the outflow road consist of a two lane road for the entire length of 400m. The arterial consist of the roads that connect each other to the South and North of the intersection. The entire arterial route is the route that connects the South origin to the North destination, and vice versa. Traffic that is travelling from the South/North to the East/West of any intersection is driving partially on the arterial route until they turn towards their destination on the East/West. Due to the distributor function, the arterial route serves more traffic compared to the East-West and West-East bound routes at the intersections.

Detector type and specifications are similar as described in section 5.2. Each intersection consists of a total of 12 approaching lanes, thus 12 detectors per intersection. In total there are 36 detectors in the network. Intersections are controlled by their own agent and no information is shared between the individual agents, thus agents can only optimize the traffic light control plan based on the information extracted from the local detectors.

Network 2
The second multi-intersection also consists of three identical intersections, as displayed in figure 5.9. The only difference compared to network 1 is the reduced distance between the intersections and consequently a shorter length of the detectors within the network. The distance between the intersections consists of a 50m two lane section and a 200m three lane section and roads from the East and West are both 200m in length. The length of queue-length detectors is reduced to 150m.

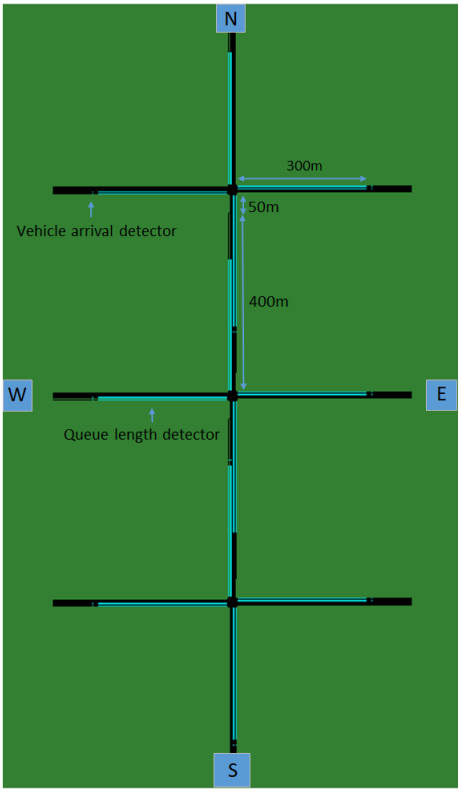

Figure 5.8: Network 1

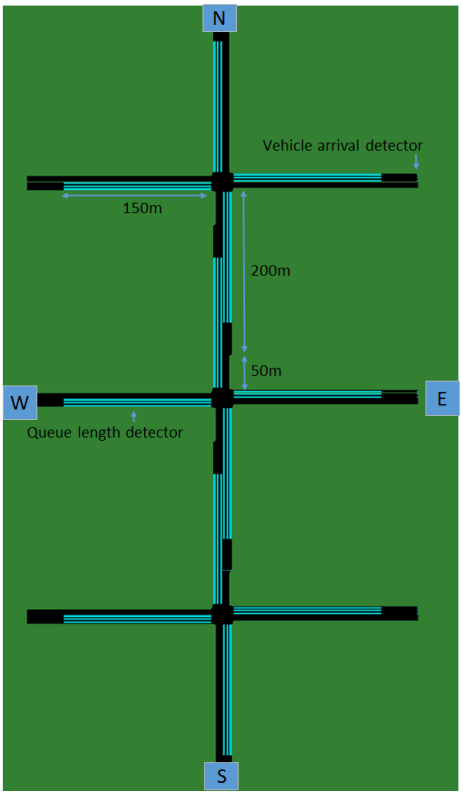

Figure 5.9: Network 2

5.4.1. Dynamic programming
In the multi-intersection network simulation, every intersection is equipped with its own instance of the optimization algorithm, this provides the optimal signal plan for the intersection based on local arrival data, derived from the detectors. A local optimization approach is chosen over a global optimization, since the length of the optimal signal plans differs from optimization to optimization epoch. This can better be handled

by a local optimizer compared to a global optimizer. With the local optimization approach every intersection can determine when a current optimized phase sequence is ending, and a new optimization run has to be performed. This reduces the number of optimizations executed in the network over time.

Network 1 consists of three intersections with similar dimensions as the test-bed intersection, therefore no modifications are required to run the dynamic programming method within this network and a similar optimization horizon length of 20 seconds is maintained. For network 2, on the other hand the dimensions are considerably shorter compared to the test-bed intersection. As approaching vehicles can only be detected on the direct approach links towards the intersection and as this information is required in order to solve the optimization problem, the optimization horizon length could only last 10 seconds.

In table 5.6 the computation time needed to perform a single optimization for different horizon lengths is displayed. The values presented in this table are the time required by the dynamic programming method in order to perform a single optimization for all intersections in the network. The difference between the minimum and maximum time required to perform an optimization could be explained by the fact that an optimization is only performed if the agent is at a decision point, where the minimum values are reached when only one agent performs an optimization and maximum values are likely to happen when all agents perform their optimization.

Table 5.6: Computation time

Horizon length [s]	**Min [s]**	**Max [s]**	**Mean [s]**
10	0,0	0,27	0,03
20	0,0	0,93	0,14

5.4.2. Q-learning

With the learning process already performed for the test-bed intersection and similar intersection layout is used throughout the intersections in the network, one could state that the same Q-table could be used. However as it could be possible that states occur that were not encountered earlier, due to the size of the state-space and limitations in the performed training process it could be possible that the performance of the Q-learning method is negatively affected. Therefore an additional learning process is performed in order to examine this effect. In this learning process the earlier obtained Q-table is used as the initial Q-table, which makes it possible to skip the first stage of the training process in which the agent action selection behaviour is mostly exploration oriented. In the network based simulation every intersection has its own agent, however all agents use the same Q-table, which makes it possible to update the same Q-table three times per evaluation, instead of once when every agent had its own Q-table. It should be noted that usage of the same Q-table is only possible if all of the intersections in the network share the same topology, otherwise the Q-table does not fit the intersection and a unique Q-table had to be used.

The additional learning process was started at simulation run 50, such that some exploration was still performed by the agent, but not at such extend that performance would drop considerably. In total 50 simulation runs of 3600 simulation seconds each were performed, which took 31.600 seconds in total, with an average time per simulation of 632 seconds, this partial learning process took far longer compared to the one conducted for the test-bed intersection. Possibly explained by the fact that free flow travel time is longer in the network compared to the test-bed intersection and a larger number of vehicles travelled through the network.

5.5. Performance evaluation

A similar performance evaluation has been performed as described in section 5.2.4. Traffic volumes as described in table 5.1 are applied to every intersection, this results in the following OD-matrix as displayed in table 5.7. The intersections are numbered, with the top intersection as number 1, the middle intersection as number 2 and the bottom intersection as number 3. In total 5330 vehicles travel through the network, at a demand factor of 1.0 and 2510 vehicles travel through each intersection separately.

5.6. Simulation scenarios

In the following subsections different simulation scenarios are described. Each scenario is designed in order to test the methods performance given a specific traffic condition. The following scenarios are described:

Table 5.7: Network traffic volume (Origin-Destination matrix)

	E-1	**W-1**	**N-1**	**E-2**	**W-2**	**E-3**	**W-3**	**S-3**	Total
E-1	0	250	100	0	0	0	0	50	400
W-1	430	0	100	0	0	0	0	100	630
N-1	125	90	0	125	90	125	90	375	1020
E-2	0	0	50	0	250	0	0	50	350
W-2	0	0	100	430	0	0	0	100	630
E-3	0	0	50	0	0	0	250	50	350
W-3	0	0	100	0	0	430	0	100	630
S-3	100	90	750	100	90	100	90	0	1320
Total	655	430	1250	655	430	655	430	825	5330

green wave, spill-back, rush-hour and off-peak conditions. The first two scenarios are designed to evaluate the performance given a specific traffic phenomena. The other scenarios are designed to evaluate the performance under different and variable demand profiles, to reflect the adaptability of the methods. Of the described scenarios only the first two are simulated as well, since these scenarios aim at specific traffic phenomena, where the latter two are already covered in the adaptability test, described in section 5.4 and results from simulating a rush-hour or an off-peak hour would not differ much compared to a demand factor of 1-1,5 or 0,5.

5.6.1. Green wave

In this scenario the methods ability to form a stage activation sequence, at different intersections, such that traffic driving along the arterial does not encounter any red traffic light lights, further referred to as a 'green wave' is examined. When a green wave is formed, traffic driving along the arterial does not need to stop at a consequence of stopping all traffic entering the intersection from the other approaches. A green-wave can be broken if any of the conflicting movements at the intersection have a green phase, when the North-South and the South-North route experience a green wave, these conflicting movements are the left turn and all movements from either West or East.

A green wave will only be formed, if activating the stage that includes the arterial route as one of the intersection movements is regarded as the optimal solution, at each intersection separately. Thus to be clear this scenario has no intention to form a green wave along the intersections in the network, if this does not result in the optimal signal control solution at the intersection.

Demand profile

In order to test the performance of a method with respect to the scenario goal of forming a green wave along the intersections of the arterial route, a specific traffic demand profile has to be used of which it is known that the formation of a green wave is the optimal solution. An example of such origin-destination pattern is to only include traffic that travels the entire arterial route (vehicles driving from south to north and vice versa), and to exclude traffic on all other routes. The desired behaviour of the signal controller is to activate stage 1 only (figure 4.1). For other origin-destination patterns it is more complex to determine if the formation of a green-wave is the optimal solution and are therefore not considered in this scenario.

5.6.2. Spill-back

Spill-back occurs when the queue length on a road approaching the intersection exceeds the available space on the lane. This occurs when the number of vehicles arriving at the queue is much higher than the number of vehicles leaving the queue. As the number of vehicles in the queue increases, so does the queue length, potentially blocking any upstream intersections causing a total gridlock of the network. Spill-back occurrence is related to the length of a streaming lane, therefore it is more likely to occur if the intersections are fairly close to each other, as less cars are required to exceed the available space. With the described network in mind, a number of 60 vehicles (total road length divided by the average vehicle length + average distance between vehicles) is needed on each lane prior to the queue of vehicles exceeding the streaming lane of an intersection.

The goal of this scenario is to test whether a method is capable of serving traffic under exceptional high traffic demand. During such demand levels spill-back is likely to happen, as queued vehicles take up all

available streaming space. The desired behaviour of the controller during such conditions is to optimize the traffic light control plan such that the queue outflow is maximised and at the same time the queue inflow is reduced. This can be achieved by increasing the amount of green time for stages with long queue lengths and at the same time reduce the inflow of stages with lower queue lengths.

Both methods optimize the traffic light control, based on the local queue length information obtained at intersection level such that stages that result in higher cumulative vehicle delay, which do also have the longest number of queued vehicles are prioritised over other stages. And as these stages are prioritised, at the same time inflow from the side roads is reduced.

A more optimal approach would be to reduce queue inflow by prioritising stages that hold the lowest turn fraction towards the road that potentially suffers from a spill-back situation. Other stages that hold a high turn fraction should be hold back from given a green phase, for as long as there is enough space available to store the number of approaching vehicles. But as this approach require additional information (turn fractions), and neither method is designed to specifically counter spill-back this is not taken into account.

Demand profile
In order to achieve a potential spill-back situation, the overall traffic demand should be in over saturated conditions. A demand factor of 1.75 and 2.00 is used to achieve a high traffic demand in which spill-back is likely to appear. With this demand factor traffic from the side roads (East and West from the intersection) towards the arterial (North and South) is increased, in order to grow the queue length on the arterial links in the network. Any traffic distribution can be used as long as the traffic demand is high enough that queues cannot be fully dissipated during a green phase.

5.6.3. Rush-hour
A rush-hour simulation scenario is characterised by high vehicle demand, in which queues are not able to fully dissolve during a green phase. This demand can be constant over time or time-varying, in which demand is increasing towards a top and afterwards decreases. The majority of the traffic demand is towards the arterial route of the network, therefore a chance of spill-back occurrence is possible, but is not the intention of this simulation scenario. The goal of this scenario is to examine the performance of the method under high traffic demand conditions.

5.6.4. Off peak conditions
Off peak condition are characterised by low traffic demand in which queues are likely to fully dissolve during a green phase and green waves could occur. Traffic distribution is likely to be more evenly distributed between the arterial road and the side roads during the off-peak since the distributor functionality of the arterial is not that prominent as compared to rush hour conditions. The goal of this scenario is to evaluate the methods performance under a more evenly distributed traffic pattern with lower demand.

5.7. Results
In the following section the results from the performance evaluation, green-wave and spill-back simulation are presented. First the results from the performance evaluation are discussed, followed by the results of the green-wave and spill-back scenario simulation. Both the Dynamic-Programming as the Q-learning optimization method are considered with these simulations, a Fixed-timing based signal control structure is not considered as this control structure is not affected by current demand levels and results are therefore not meaningful as a benchmark.

5.7.1. Adaptability test
The results of the adaptability test are separated for both network 1 and network 2. First the results of network 1 are discussed, followed by a discussion of the results of network 2. The results from the Q-learning method are divided based on the Q-table used by the agent during the simulation. The regular Q-learning uses the Q-table as obtained during the test-bed learning process, whereas Q-learning(NT) uses the Q-table as obtained during the additional network learning process. Such that performance differences between both learning processes can be pointed out.

Network 1
The results of simulating the adaptability test on the three intersection network 1 are given in table 5.8 and figure 5.10. First the results of the Q-learning method without additional training on the specific network are

discussed. The results with a demand factor of 0,5 and 0,75 respectively are broadly similar compared to the Q-learning method with additional training on the specific network (Q-learning(NT)), however this cannot be said for the other demand factors. With the implementation of the Q-learning method, it is expected that once a policy is obtained through training the agent, this policy should be applied to other similar problems as well. As the performance is significantly improved by additional training on the network, questions arise whether the policy was sufficient converged for the entire state-space. Consequently this could imply that the initial training process conducted in the testbed intersection simulations was not sufficient.

Results of the Q-learning method with additional training on the specific network show similarities compared to the testbed intersection simulations, except for demand factors of 0,5 and 0,75, as the dynamic programming method is outperforming the Q-learning(NT) method. Also it should be noted that the differences between both methods are much smaller in the network simulation, compared to the test-bed simulation. A possible explanation for this, cannot easily be given, however a possible explanation could be that the Q-learning method is less efficient as arriving vehicles are not taken into account into the state. Since this would result in a dramatic increase of the state-space, due to the many possible combinations of this vector. In [17] the increment in state-space by using the arrival vector as part of the state is countered by using an ANN instead of a Q-table, this however is a whole different implementation of a reinforcement learning approach.

Results of the dynamic programming method are quite as expected based on the earlier results of the testbed intersection. However in the network 1 simulations, the performance compared to the Q-learning(NT) method turned out to be relatively better, as differences between both methods are smaller. As already stated, this could be related to the use of arriving vehicle information into the state definition.

Table 5.8: Network 1: Average cumulative vehicle delay with different demand factors

	0,5	**0,75**	**0,9**	**1,0**	**1,10**	**1,25**	**1,50**
Dynamic-Programming	6,7	21,6	35,8	45,2	60,6	85	120
Q-Learning	13,7	30,9	66,3	142	173,5	208,1	254
Q-Learning(NT)	13,7	28	35,2	37,3	42,8	69,5	100,3

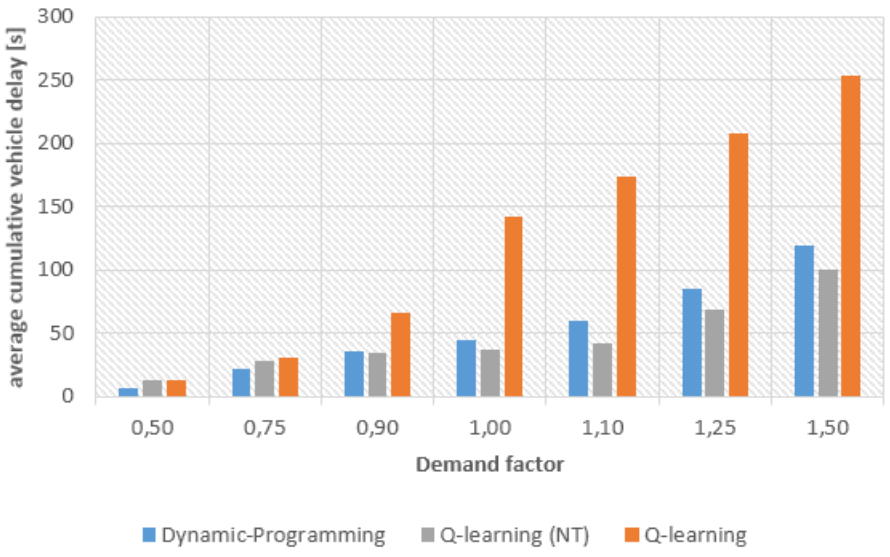

Figure 5.10: Network 1: Average cumulative vehicle delay with different demand factors

Network 2

The results of simulating the adaptability test on the three intersection network 2 are given in table 5.9 and figure 5.11. The following differences compared to the simulation results with network 1 can be noticed. The Q-learning(NT) method did not experience the same performance gain, compared to the Q-learning method without additional training on the specific network. As the same set-up has been used across both networks, the origin of this effect should be looked for in other non-constant variables related to the network topology, such as the distance between the intersections, travel time etc. It is likely that as vehicle travel faster between

intersections, queues at the next intersection grow faster as well. Therefore it could be possible that the used queue length discretisation intervals are not applicable within the current network topology. As lower queue length values do barely occur. This effect was already mentioned in section 5.3.2, and could be considered one of the downsides of using state with discrete intervals.

The dynamic programming method achieved an improved performance compared to the Q-learning(NT) method, for all demand factors used in the adaptability test. This outcome has not been noticed before on either the testbed intersection, nor on network 1. Variables that changed in the dynamic programming method are also related to the reduced distance between the intersections, such as the reduction of the planning horizon. The length of the planning horizon is decreased from 20 to 10 seconds, consequently possible errors that could arise during the optimization, due to the modelling process within the planning horizon, have less impact on the results. This could also be the reason why the average cumulative delay had a smaller increase over the range of demand factors on network 2 compared to network 1.

Table 5.9: Network 2: Average cumulative vehicle delay with different demand factors

	0,5	**0,75**	**0,9**	**1,0**	**1,10**	**1,25**	**1,50**
Dynamic-Programming	5,8	16,8	29,7	44,8	58,1	67	82,6
Q-Learning	15	36,3	76	105	110,4	123,7	132,6
Q-Learning(NT)	15,1	34,5	55,7	69,7	82,7	87,1	96,4

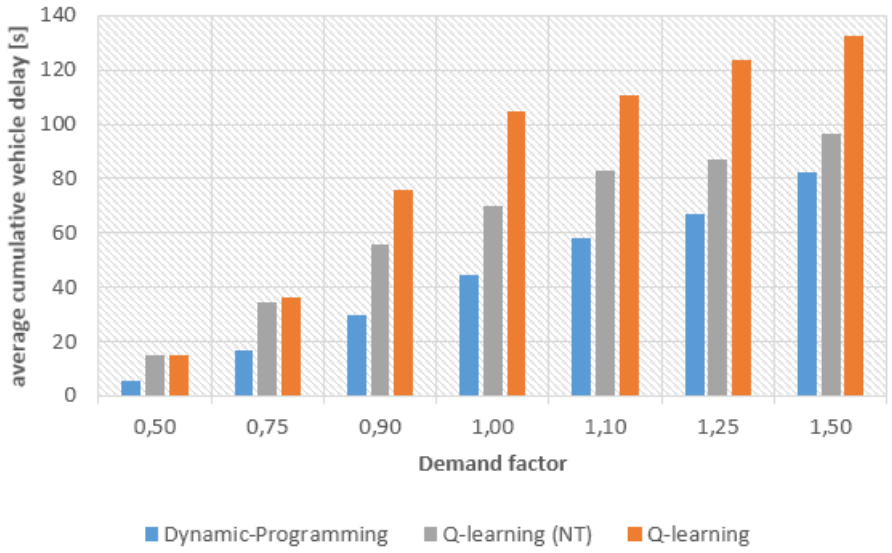

Figure 5.11: Network 2: Average cumulative vehicle delay with different demand factors

5.7.2. Green wave

The results of simulating the green wave formation scenario are given in table 5.10. The results of network 1 do not differ much between Q-learning(NT) and Dynamic-Programming, as both methods achieve an average cumulative delay of near zero. On network 2, the results are slightly better for the dynamic programming method, as an average cumulative delay of zero is achieved. Whereas the Q-learning(NT) method resulted in an increased average cumulative delay. This results is in line with previous results of both networks, as Q-learning(NT) was also performing worse compared to its performance on network 1 in the adaptability test.

Nevertheless the average cumulative delay is almost non-existent for both methods on each intersection individually, therefore one could conclude that both methods are capable of optimization of the individual signal plans such that a green wave occurs, with Dynamic-Programming performing better than Q-learning(NT).

Table 5.10: Average cumulative vehicle delay, per intersection and in total

Network	Method	I	II	III	Total
I	DP	0,06	0,03	0,14	0,22
	QL(NT)	0,11	0,03	0,23	0,38
II	DP	0	0	0	0
	QL(NT)	0,4	0,3	0,3	1

5.7.3. Spill-back

The results of simulating the spill-back scenario are given in table 5.11. For both networks a traffic demand of 1,75 was used. During these simulation runs, visual observations noted that queue lengths did not increase in such levels that the occurrence of spill-back was noted on network 1. Streaming lanes in this network have such a high capacity, spill-back is prevented from occurring. Therefore a demand factor of 2,0 is only simulated on network 2, as streaming lanes on this network are much smaller.

A remarkable result from the spill-back scenario simulations is that simulations with a demand factor of 1,75, achieved lower levels of average cumulative vehicle delay on network 1, compared with results from the adaptability test under lower demand factors at the same network. An explanation for this effect could be that larger platoons are moved through the network in one go, as this is more effective than breaking this platoon up in smaller sections.

A similar effect did not occur on network 2, although the increase in average cumulative delay was only minor for the dynamic programming method compared to the adaptability test demand factor of 1,5. This could be explained by the fact that streaming lane capacity in network 2 is far lower compared to network 1, as the distance between the intersections is reduced by 200m. As spill-back occurs, vehicles are not able to depart from the queue, as this will result in a blockage of the intersection. Therefore the green phase of a stage is not or only partly effective used.

As neither of the two methods have any spill-back prevention measures included into the optimization process, it is of no surprise spill-back occurred. However, there is still a difference between both methods performance when a high traffic demand is simulated. As the Q-learning method makes use of a state with queue length discretisation, the action selection strategy becomes less effective as queue lengths are way beyond the maximum values within the intervals. This has a negative effect on the performance of the Q-learning method.

Dynamic programming on the other hand is more flexible in dealing with extreme traffic demands, as no predefined intervals are used. However its performance is negatively affected by the fact that the current implementation cannot effective track the cumulative delay of the individual vehicles in its optimisation.

Table 5.11: Average cumulative vehicle delay, per intersection and total

Network	Demand Factor	Method	I	II	III	Total
I	1,75	DP	31,40	29,29	26,01	86,7
		QL(NT)	25,48	24,82	23,76	74,1
II	1,75	DP	30,70	29,85	24,89	85,4
		QL(NT)	37,97	33,85	42,71	114,5
II	2,00	DP	40,63	32,18	26,65	99,45
		QL(NT)	46,6	39,15	35,13	120,9

5.8. Conclusion

In this chapter first the simulation methodology is explained. This simulation methodology serves as an example of a general evaluation methodology as described in chapter 2. Due to time constraints, a balance had to be found between a comprehensive evaluation and the required time. Therefore some earlier described components had to be simplified, including the simulation randomness, number of simulation runs, metrics and different type of scenarios. The methodology that remained had the goal evaluating the two optimization methods, as accurate and effective as possible.

Simulations are performed on the microscopic and continuous road traffic simulation package SUMO. A connection between the Python scripts and the SUMO environment is established with the Traffic Control

Interface. This connection is a vital component, as it allows the traffic lights in the SUMO environment to be controlled outside of SUMO, by the optimization method.

Three different networks are designed in SUMO, the first network serves as a testbed intersection and the latter two are used to evaluate the performance on a network containing three intersections. Cumulative delay is used as the performance metric for all simulations conducted in this chapter, as this metric is clear and unambiguous and could be easily extracted from the simulation environment by the cumulative delay module.

The testbed intersection was used to perform the validation of the different optimization methods. During these simulations it became clear that several parameters, like interval boundaries and queue detection speed threshold, had to be fine-tuned to allow for improved performance, mainly in the Q-learning optimization method.

To evaluate the performance of both optimization methods a simulation with different demand factors ranging from 0,5 up to 1,5 is performed, with a fixed timing control being benchmark.

The results of the dynamic programming optimal control method were quite surprising, as this method is expected to find the optimal solution in all traffic conditions. The difference in performance compared to the Q-learning method increased rapidly at demand factors of 1,0 and higher. Explanations for these results can be sought at different causes, one of the most obvious reasons are several assumptions made in[26], such as instantaneous queue dissolvement and instant stop and go behaviour of vehicles. As these assumptions are incorporated in this research the functioning of the method itself could be negatively impacted. Another cause could be found in the solutions designed to implement the method, without the assumptions made in [26] or in the coding in general.

The results of the Q-learning method are satisfactory, although it should be noted that the average cumulative delay per vehicle increases rapidly for demand factors of 1,10 and higher. An explanation for this behaviour can likely be sought in the queue length discretisation intervals, as these are tuned towards shorter vehicle queues.

Following simulations were performed on two different three intersection networks. On network 1 the distance between the intersections is 450m and queue detector length is 300m. First the results of the Q-learning method without additional training on the specific network are discussed. The results with a demand factor of 0,5 and 0,75 respectively are broadly similar compared to the Q-learning method with additional training on the specific network (Q-learning(NT)), however this cannot be said for the other demand factors. With the implementation of the Q-learning method, it is expected that once a policy is obtained through training the agent, this policy should be applied to other similar problems as well. As the performance is significantly improved by additional training on the network, questions arise whether the policy was sufficient converged for the entire state-space. Consequently this could imply that the initial training process conducted in the testbed intersection simulations was not sufficient.

Results of the Q-learning method with additional training on the specific network show similarities compared to the testbed intersection simulations, except for demand factors of 0,5 and 0,75, as the dynamic programming method is outperforming the Q-learning(NT) method. Also it should be noted that the differences between both methods are much smaller in the network simulation, compared to the test-bed simulation. A possible explanation for this, cannot easily be given, however a possible explanation could be that the Q-learning method is less efficient as arriving vehicles are not taken into account into the state, although this would result in a dramatic increase of the state-space, due to the many possible combinations of this vector.

Results of the dynamic programming method are quite as expected based on the earlier results of the testbed intersection. However in the network 1 simulations, the performance compared to the Q-learning(NT) method turned out to be relatively better, as differences between both methods are smaller. As already stated, this might be related to the use of arriving vehicle information into the state definition.

On network 2 the distance between the intersections is reduced to 250m and queue detector length is 150m. The following differences compared to the simulation results with network 1 were noticed. The Q-learning(NT) method did not experience the same performance gain, compared to the Q-learning method without additional training on the specific network. As the same set-up has been used across both networks, the origin of this effect should be looked for in other non-constant variables related to the network topology, such as the distance between the intersections, travel time etc. It is likely that as vehicle travel faster between intersections, queues at the next intersection grow faster as well. Therefore it could be possible that the used queue length discretisation intervals are not applicable within the current network topology. As lower queue length values do barely occur. This effect was already mentioned in section 5.3.2, and could be considered one of the downsides of using state with discrete intervals.

The dynamic programming method achieved an improved performance compared to the Q-learning(NT) method, for all demand factors used in the adaptability test. This outcome has not been noticed before on either the testbed intersection, nor on network 1. Variables that changed in the dynamic programming method are also related to the reduced distance between the intersections, such as the reduction of the planning horizon. The length of the planning horizon is decreased from 20 to 10 seconds, consequently possible errors that could arise during the optimization, due to the modelling process within the planning horizon, have less impact on the results. This could also be the reason why the average cumulative delay had a smaller increase over the range of demand factors on network 2 compared to network 1.

Additional scenarios were designed to further elaborate on the performance of both optimal control methods and two of those scenarios were simulated on both multi intersection networks. These are a green wave and a spill-back scenario.

The green wave formation scenario had the goal of evaluating the ability of the method in forming a green wave, with traffic demand solely on the North-South and South-North route of the network. The results of network 1 do not differ much between Q-learning(NT) and Dynamic-Programming, as both methods achieve an average cumulative delay of near zero. On network 2, the results are slightly better for the dynamic programming method, as an average cumulative delay of zero is achieved. Whereas the Q-learning(NT) method resulted in an increased average cumulative delay. This results is in line with previous results of both networks, as Q-learning(NT) was also performing worse compared to its performance on network 1 in the adaptability test.

Nevertheless the average cumulative delay is almost non-existent for both methods on each intersection individually, therefore one could conclude that both methods are capable of optimization of the individual signal plans such that a green wave occurs, with Dynamic-Programming performing better than Q-learning(NT).

The spill-back scenario had the goal of creating spill-back within the network by loading a high traffic demand. For both networks a traffic demand of 1,75 was used. During these simulation runs, visual observations noted that queue lengths did not increase in such levels that the occurrence of spill-back was noted on network 1. Streaming lanes in this network have such a high capacity, spill-back is prevented from occurring. Therefore a demand factor of 2,0 is only simulated on network 2, as streaming lanes on this network are much smaller.

A remarkable result from the spill-back scenario simulations is that simulations with a demand factor of 1,75, achieved lower levels of average cumulative vehicle delay on network 1, compared with results from the adaptability test under lower demand factors at the same network. An explanation for this effect could be that larger platoons are moved through the network in one go, as this is more effective than breaking this platoon up in smaller sections.

A similar effect did not occur on network 2, although the increase in average cumulative delay was only minor for the dynamic programming method compared to the adaptability test demand factor of 1,5. This could be explained by the fact that streaming lane capacity in network 2 is far lower compared to network 1, as the distance between the intersections is reduced by 200m. As spill-back occurs, vehicles are not able to depart from the queue, as this will result in a blockage of the intersection. Therefore the green phase of a stage is not or only partly effective used.

As neither of the two methods have any spill-back prevention measures included into the optimization process, it is of no surprise spill-back occurred. However, there is still a difference between both methods performance when a high traffic demand is simulated. As the Q-learning method makes use of a state with queue length discretisation, the action selection strategy becomes less effective as queue lengths are way beyond the maximum values within the intervals. This has a negative effect on the performance of the Q-learning method.

Dynamic programming on the other hand is more flexible in dealing with extreme traffic demands, as no predefined intervals are used. However its performance is negatively affected by the fact that the current implementation cannot effective track the cumulative delay of the individual vehicles in its optimisation.

Based on the results of the evaluation performed in this chapter, one cannot conclusive point one optimization method as favourable over the other. The Q-learning method was outperforming dynamic programming on the isolated intersection and on network 1, whereas dynamic programming outperformed the Q-learning method on all networks when a low traffic demand was used and on network 2 for all levels of traffic demand. Also was the dynamic programming method favourable in the green wave and spill-back scenarios over the Q-learning method.

These results show that neither of the two methods was superior to the other, as well as some remarkable

outcomes. The dynamic programming method was unable to find the optimal solution, however it should do so by design. The converged policy of the Q-learning method did not result in optimal performance during high traffic demand or on the multi-intersection networks. These results could originate to the design of the method or from modifications in the required self-implementation of the method. As there are many variables, parameters, assumptions involved in the design of a method, an identical self-implementation is not possible. Also could critical information be omitted by the author. These modifications could have led to potential alterations in performance. If a general evaluation methodology was applied in the field of intelligent transport systems, a self-implementation would no longer be required as the method's performance could be looked up in the result database. Therefore would a general evaluation methodology be of additional value in qualitatively assessing the performance of traffic signal control methods and should be further researched and broadly implemented in the field of intelligent transportation systems, in order to provide a comparative measure of a method's performance. In the next chapter the overall conclusion of this thesis is given, as well as the discussion and recommendations for further research.

6

Conclusion, discussion and recommendation

In this chapter the conclusion, discussion and recommendation of the research are given. First the conclusion is given, in which the main research questions are answered. Followed by the discussion, in which limitations, the choices made and emerged questions of the research are discussed. At last is the recommendation section in which recommendations for further research are given in order to improve the research conducted in this thesis and the general evaluation methodology presented in this thesis.

6.1. Conclusion

Increase in urbanization and traffic congestion creates an urgent need to operate our transportation systems with maximum efficiency. Traffic light control optimization is considered one of the main ways to solve traffic problems in urban networks. By optimizing the signal plans on a network wide scale, signal control efficiency can be increased, resulting in reduced vehicle delay and decreased number of stops, thus improving the traffic flow.

In publications in the field of intelligent transportation systems, a vast amount of different optimal traffic light control methods is described. With new optimization methods being developed, it is important to know the performance compared to other methods. Such comparison is only possible if the same evaluation methodology is used across all methods included into this benchmark. A general evaluation methodology is not applied in the field of intelligent transport systems, therefore publications evaluate their findings based on a self-defined evaluation methodology. Based on findings in [29] and additional observations described in chapter 2, an evaluation methodology consists of many different elements, such as: network topology, simulation randomness, evaluation metrics, performance indicators, scenario design and simulation software packages. Self-defined evaluation methodologies consist of different interpretations of these elements, therefore a great difference amongst the various methodologies is possible. Consequently, the performance described in one publication is only comparable with findings in another publications, if both evaluation methodologies are exactly equal, which is rarely the case. In addition, designed methods described in these publications are not publicly available, therefore as one wants to evaluate the potential of two methods described in different publications, it is necessary to implement both methods by themselves and perform an evaluation based on a equal methodology. In order to improve the performance evaluation and to obtain the potential for practical implementation a general evaluation methodology is required.

The first research objective is to define the required design specifications, such that a general evaluation methodology can be designed in order to evaluate the performance of different optimal traffic light control methods objectively. In order to define these design specifications, the purpose and goal of performance evaluation have to be clear first, as this is the main focus point of the methodology. Secondly, a closer look is taken at how current evaluation methodologies are designed and carried out by different publications in order to point out the various difference in the evaluation methodology amongst them. Thirdly the different elements involved in establishing an evaluation methodology are extracted from the different evaluation methodologies, as well as found in literature.

As described in [46] the purpose and goal of performance evaluation is two-fold purpose, firstly it provides

a qualitative method of evaluating. Secondly it provides a comparative measure of the algorithm against similar algorithms, assuming similar criteria are used. In the field of intelligent transport systems many different methods are described to improve traffic signal control. As concluded in [29] the evaluation methodology described in these publications greatly differ to one another, as a consequence the results from these publications are hardly comparable due to different scenario set-ups and performance indicators. This conclusion is also confirmed by an additional evaluation performed in this research among 16 publications. It becomes clear that the first purpose of performance evaluation is met, however no comparative measure is possible as no equal evaluation methodology is applied.

By means of a general evaluation methodology a methods performance can be objectively evaluated and compared to other methods. This general evaluation methodology should be designed based on rigid guidelines. These guidelines should contain accepted simulation software, different simulation network topologies, unambiguous measurement metrics, scenario descriptions for specific traffic phenomena and simulation set-up criteria. Such that it is no longer possible to evaluate a methods performance based on a self-defined and thus possibly subjective evaluation methodology. Furthermore, by using the same simulation methodology across different publications enables the comparison of their performance to one another.

In this thesis, an evaluation methodology was designed according to the guidelines regarding a general evaluation methodology. Due to limitations in the available time concessions had to be made in the comprehensiveness of the evaluation methodology applied in this thesis. The goal of this methodology was to provide an equal evaluation to evaluate the potential of a traffic light control method based on dynamic programming and Q-learning.

Three different networks were designed in SUMO, the first network is an isolated intersection (test-bed intersection). This network was initially used to fine-tune the different parameters, such as the interval boundaries and queue detection speed threshold. Also did this network provide insight in the method's performance on this type of network. A fixed-timing structure was also implemented to provide an additional benchmark. The lather two networks have been designed to evaluate the method's performance on two different networks containing three intersections.

On network 1 the distance between the intersections is 450 m and queue detector length is 300 m, whereas on network 2 the distance between the intersections is reduced to 250 m and queue detector length is 150 m. On both network the same adaptability test is performed, except this time the two methods are only quantitatively compared to each other, so no fixed timing structure was used.

The performance evaluation on these networks is evaluated under various traffic loads, ranging from a demand factor of 0,5 up to 1,5, where a demand factor of 1,0 described the typical volumes during a rush-hour. The metric used in this evaluation is the average cumulative delay, which is measured as the time a vehicle is present in a queue.

Additional scenarios were designed to further elaborate on the performance of both traffic light control methods considering specific traffic phenomena. The two scenarios included into the evaluation methodology are a green wave and a spill-back scenario. The goal of the green-wave formation scenario was to evaluating the ability of the method in forming a green-wave along the main corridor. Traffic demand was limited to solely the North-South and South-North route of the network, as the formation of the green-wave had to be the known optimal solution. Obtaining this known solution is considerably more complex if other traffic flows are also included into the scenario. The goal of the spill-back scenario was to evaluate the ability of the method to function under very-high traffic volumes at which the formation of spill-back on the main corridor is likely to happen. The demand factors during this scenario were 1,75 and 2,00 respectively.

The second objective is to obtain the potential for practical implementation of two different mathematical traffic light control techniques, these two techniques are in this thesis chosen as dynamic programming and Q-learning. As explained in the problem statement, an objective evaluation is only possible if both methods are evaluated using the same evaluation methodology, which is the methodology described in the first research objective. As both methods are not publicly available, both had to be implemented according to the method description in the source publication. In addition to this research objective the various control approaches and mathematical techniques in traffic light control that can be evaluated using the general evaluation methodology are described.

Classifications in control approaches can be made based on the adaptability (fixed-timing versus dynamic) and the level of hierarchy (local control versus coordinated control). With the goal of improving the traffic light control and ultimately optimize for the best traffic light control plan, one could state that a coordinated control strategy is favourable over a local intersection control strategy, as local optimization could

possibly deteriorate the performance on network level. Due to limitations in the implementation a local optimization is chosen over a coordinated control, as this removes the technical difficulties that arise in establishing a coordination between multiple intersections. Since both methods require a self-implementation in Python.

Various mathematical techniques to improve traffic signal control have been described in literature such as: dynamic programming, fuzzy logic, artificial neural networks, reinforcement learning, heuristics-based algorithms and genetic algorithms. As these techniques differ in complexity and applicability, an overview is given in this thesis. In principle could every method be implemented and evaluated in order to obtain the potential for practical application. However, as these methods are not publicly available the evaluation requires an self-implementation of the method. Therefore two mathematical techniques are chosen based on the level of detail in the method description of source paper and the assumed complexity of the implementation. One of these methods is based on dynamic programming and the other is based on Q-learning. Dynamic programming is chosen, as this method is used in different traffic light optimization strategies, like RHODES, PRODYN and OPAC. Also dynamic programming should be able to achieve the optimal solution in all traffic conditions. Q-learning is chosen as reinforcement learning is a promising approach in optimal traffic light control, as this technique has the ability to cope with the dynamics of real-time traffic flows, can be applied without any prior knowledge of the environment, no model of the system is required and the curse of dimensionality can be addressed more efficiently.

Due to different characteristics of both methods, the dynamic programming method required a model of the environment and the Q-learning method required a training process in order to obtain the control policy. The model of the environment is required in order to simulate the arrival and departure of vehicles at the respective queues in the network. Based on these queue dynamics the cost of activating a specific stage is determined. The training process is a number of repeated simulations in which the agent is selecting actions randomly. If an action is reducing the cumulative delay of vehicles in the system the action is rewarded and otherwise the action is penalised. The training process is performed on both the isolated intersection as on the multi intersection network with a demand factor of 1,0.

Based on the results of the evaluation performed in this chapter, one cannot conclusive point one optimization method as favourable over the other, as performance compared to each other varied over the different simulations. The Q-learning method outperformed the dynamic programming method on the isolated intersection for the entire range of demand factors. However a fixed timing structure, designed on a traffic demand of 1,0, was outperforming both the dynamic programming and the Q-learning method for demand factors of 1,5. On the two multi intersection networks the results varied between the two methods. The Q-learning method (after additional network specific training) was outperforming the dynamic programming method on network 1 for traffic demand factors of 0,9 and higher, however it was outperformed by the dynamic programming method on network 2 for all demand factors. Also in the additional scenario simulations the dynamic programming method was outperforming the Q-learning method for both the green wave and spill-back scenarios.

These results show that neither of the two methods was superior to the other, as well as some remarkable outcomes. The dynamic programming method was unable to find the optimal solution, however it should do so by design. The converged policy of the Q-learning method did not result in optimal performance during high traffic demand or on the multi-intersection networks. These results could originate to the design of the method or from modifications in the required self-implementation of the method. As there are many variables, parameters, assumptions involved in the design of a method, an identical self-implementation is not possible. Also could critical information be omitted by the author. These modifications could have led to potential alterations in performance. If a general evaluation methodology was applied in the field of intelligent transport systems, a self-implementation would no longer be required as the method's performance could be looked up in the result database. Therefore would a general evaluation methodology be of additional value in qualitatively assessing the performance of traffic signal control methods and should be further researched and broadly implemented in the field of intelligent transportation systems, in order to provide a comparative measure of a method's performance.

6.2. Discussion

In this section the various choices made in this thesis and remarkable results obtained from the implementation of both methods, the general evaluation methodology and the conducted evaluation by simulation are discussed in separate sections.

6.2.1. Implementation of the methods

The dynamic programming method described in [26], was in fact qualitatively assessed on an over-simplified simulation environment. Assumptions that queues would dissolve immediately at the start of a green phase, exact knowledge of the vehicle arrivals and no interaction between the vehicles have led to many required modifications in order to implement the method for this thesis. Although best efforts were given to overcome the assumptions made is the source publication and to model the traffic flow as precisely as possible, these elements are likely the reason why the variation in performance was obtained from the various simulations conducted in the evaluation. This is most likely the reason why the implemented dynamic programming method in this thesis did not succeed in obtaining the optimal signal plan despite dynamic programming should always be able to find the optimal solution despite the conditions. If the various components involved in the optimization, ranging from vehicle arrivals predicted, from queue dynamics and traffic flow modelling are not fully reflecting the actual conditions in the simulation network an error occurs between what the method finds as the optimal solution and what would actually be the optimal solution given the actual traffic conditions in the environment.

The Q-learning method described in [13] contained various variables regarding the discretization and training process. These variables have an effect on the performance as was found out in this thesis. If the number of intervals b in the discretization is chosen too low the policy is likely to be too general as the number of states is limited. On the other hand if the value of b is set too high, the number of states becomes too large, which will result in policy convergence problems. Even if the value b result in a sufficient number of states, poor interval limits d could still negatively affect the performance. If the boundary values are too far apart from each other variation in the value of s_i can hardly be captured in the discretization. Also if the discretization interval does not reflect the traffic demand or the link length, no distinction can be made between the different state-actions. A small interval between the discretization boundary results in a higher number of discretizations required in order to fit the deviation in maximum queue length, this voids the reduction in the curse of dimensionality. Variables involved in the training process have likely caused convergence to the optimum policy problems if a value of more than three seconds was used for the yellow-red time i.e. the action selection strategy was unable to select the best action at a given state. This resulted in a higher cumulative delay compared to the both the dynamic programming method, as the fixed-timing benchmark. As the training process is time consuming a solution to this problem was not found in this thesis. Likely does the problem originates from the number of simulation runs performed in the training process, however as computational power is limited during this research, extending the agents training process was not possible, as it would take considerable amounts of time. For this reason a yellow-red time of 3 seconds is used, instead of more realistic values based on the minimum yellow time plus the clearance times due to mutual conflict points.

6.2.2. General evaluation methodology

The following limitations had an effect on conducting the thesis and the results that are found and are discussed in this section. The first limitation is that little documentation could be found on the requirements of a well designed evaluation methodology, therefore a literature study had to be performed on how publications had designed their own evaluation methodology. Parts that fit the purpose and requirements of a general evaluation methodology were combined, resulting in the proposed evaluation methodology described in this research. It remains questionable if the designed methodology is fully objective, foremost because there is no clear definition of an objective evaluation methodology, but as the evaluation methodology is designed without any of the particular optimization methods in mind, it could be considered a useful contributions towards a general evaluation methodology.

The general evaluation methodology described in this thesis had to be simplified in the evaluation by simulation conducted in this thesis, due to time limitation constraints. A balance had to be found between a comprehensive evaluation and the time available. Therefore some components had to be simplified, including: the simulation randomness, number of simulation runs, metrics and different types of scenarios. By means of this simplification it was possible to conduct an extensive evaluation by simulation within the available time frame.

6.2.3. Conducted evaluation by simulation

The stage set used in the simulation of this thesis did not contain any overlapping green of a direction over multiple stages. Therefore each direction had a specific stage in which it could be activated. Normally the green phase of a specific direction could span over multiple stages if this direction does not form a conflict

with either of the activated stages. As this is more complex to code in both SUMO as in Python it is chosen to limit the green-phase of a direction to one stage only.

In chapter 3 of this thesis, various optimal signal control methods described in publications in the field of intelligent traffic control are given. Due to time limitations in this thesis only two of these methods could be implemented and evaluated in order to obtained their potential for practical implementation. This choice has been made based on the perceived complexity of the implementation of the method and the level of detail of the method description, as provided in the source publication. As the code of these methods is are not public available, it was necessary to self-implement the methods and suit them to the simulation environment, which was not possible if the method was either too complex or method description was lacking important details for a proper implementation.

6.3. Recommendation

In this section the recommendations for further research are given and can be either oriented towards the research carried out in this thesis, or towards improving the general evaluation methodology.

6.3.1. Research in this thesis

Although both publications state that their designed method improved the traffic conditions, based on their results. The implementation in this thesis showed a number of remarkable results. Further research on the model design in the dynamic programming method and variable design in the Q-learning method could improve the results obtained from the evaluation methodology as performed in this thesis. This could possible alter the potential of both methods as obtained in this thesis. Furthermore as both implementations are proven to be correct the traffic light control problem itself could be expanded by including additional transportation modes, such as pedestrians, bicycles and public transport. This would improve the realism of the simulation and this simulation would provide results that describe the methods performance in everyday practice, instead of the more theoretical results often obtained from research publications.

As stated in the discussion, the set with the available stages could be expanded, such that green-phases of a number of directions could span over multiple stages. This is improves the realism of the traffic light control as this approach is commonly applied in reality and it also improves the throughput of the intersection which improves the quality of the simulation under higher traffic demand.

Also could a coordinated control strategy be considered instead of the isolated strategy utilised in this thesis. A coordinated control has to deal with more data and it also increases the complexity of the optimization problem. It would be interesting to know how an implementation of both methods, based coordinated control would perform compared to the results obtained in this thesis.

6.3.2. General evaluation methodology

The simulation methodology described in this thesis is an example of how a general evaluation methodology could be implemented to evaluate the performance of optimal traffic light control methods. The methodology described in this thesis could be improved by taken more randomization into account when micro simulating, as this would increase the reality of the simulation. Consequently the number of simulation runs should be increased too, in order to properly reflect a methods performance. Also could the number of traffic demand patterns and vehicle arrival patterns be increased to improve the results obtained from the adaptability test and the number of network topologies could be increased in order to improve a methods robustness evaluation.

In order to establish a general evaluation methodology concrete guidelines and motivations why an evaluation should be performed derived from these guidelines must be drawn up. Information regarding evaluation methodologies provided by this thesis could be useful in doing so. Also can expert opinions be taken into account when designing these guidelines, as their opinion can contribute and further improve these guidelines.

As found out while implementing both methods for traffic light control in this thesis. Assumptions made in the source publication of these methods could lead to unexpected results when self-implementing the method, as these assumptions interfere with an objective evaluation methodology. Therefore should the general evaluation methodology be extended with a description of the simulation process framework, this process should describe how the various elements involved in the simulation process should be designed, such that no assumptions are made that simplify the traffic light control problem.

The general evaluation methodology reveres to a list of accepted micro-simulation software packages. In

this thesis the micro-simulations were performed in SUMO, however multiple simulation software packages are available. It would be interesting to know if various simulation software packages are capable of utilising traffic light control as described in this thesis and how the results would vary between them. Based on these results a list of accepted micro-simulation software packages could be established which can benefit the application of the general evaluation methodology.

Bibliography

[1] PG Balaji, X German, and D Srinivasan. Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems*, 4(3):177–188, 2010.

[2] Jerry Banks. *Handbook of simulation : principles, methodology, advances, applications, and practice*. Wiley Co-published by Engineering & Management Press, New York Norcross, Ga, 1998. ISBN 978-0-471-13403-9.

[3] Michael Behrisch, Laura Bieker, Jakob Erdmann, and Daniel Krajzewicz. Sumo–simulation of urban mobility: an overview. In *Proceedings of SIMUL 2011, The Third International Conference on Advances in System Simulation*. ThinkMind, 2011.

[4] Richard Bellman. The theory of dynamic programming. Technical report, RAND CORP SANTA MONICA CA, 1954.

[5] Robbin J Blokpoel, Daniel Krajzewicz, and Ronald Nippold. Unambiguous metrics for evaluation of traffic networks. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 1277–1282. IEEE, 2010.

[6] Wilco Burghout. A note on the number of replication runs in stochastic traffic simulation models. *Unpublished report, Stockholm: Centre for Traffic Research*, 2004.

[7] Partha Chakroborty. *Principles of transportation engineering*. PHI Learning Private Limited, Delhi, 2017. ISBN 9788120353459.

[8] Yit Kwong Chin, KC Yong, Nurmin Bolong, Soo Siang Yang, and Kenneth Tze Kin Teo. Multiple intersections traffic signal timing optimization with genetic algorithm. In *Control System, Computing and Engineering (ICCSCE), 2011 IEEE International Conference on*, pages 454–459. IEEE, 2011.

[9] Min Chee Choy, Dipti Srinivasan, and Ruey Long Cheu. Cooperative, hybrid agent architecture for real-time traffic signal control. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: systems and humans*, 33(5):597–607, 2003.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE, 2009.

[11] Richard W Denney, Larry Head, and Kevin Spencer. Signal timing under saturated conditions. 2008.

[12] Chaojun Dong, Shiqing Huang, and Xiankun Liu. Urban area traffic signal timing optimization based on sa-pso. In *Artificial Intelligence and Computational Intelligence (AICI), 2010 International Conference on*, volume 3, pages 80–84. IEEE, 2010.

[13] Samah El-Tantawy and Baher Abdulhai. An agent-based learning towards decentralized and coordinated traffic signal control. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 665–670. IEEE, 2010.

[14] Samah El-Tantawy, Baher Abdulhai, and Hossam 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*, 14(3):1140–1150, 2013.

[15] George S Fishman. Principles of discrete event simulation.[book review]. 1978.

[16] Nathan H Gartner and Chronis Stamatiadis. Arterial-based control of traffic flow in urban grid networks. *Mathematical and computer modelling*, 35(5-6):657–671, 2002.

[17] Wade Genders and Saiedeh Razavi. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*, 2016.

[18] Michael D Heath, Sudeep Sarkar, Thomas Sanocki, and Kevin W Bowyer. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(12):1338–1359, 1997.

[19] Robert Hecht-Nielsen et al. Theory of the backpropagation neural network. *Neural Networks*, 1 (Supplement-1):445–448, 1988.

[20] Tsin Hing Heung, Tin Kin Ho, and Yu Fai Fung. Coordinated road-junction traffic control by dynamic programming. *IEEE Transactions on Intelligent Transportation Systems*, 6(3):341–350, 2005.

[21] Parag Hirulkar, Rahul M. Deshpande, and Preeti Bajaj. Optimization of traffic flow through signalized intersections using pso. 2011.

[22] Ta-Yin Hu and Li-Wen Chen. Traffic signal optimization with greedy randomized tabu search algorithm. *Journal of Transportation Engineering*, 138(8):1040–1050, 2012.

[23] Wenbin Hu, Huan Wang, Liping Yan, and Bo Du. A swarm intelligent method for traffic light scheduling: application to real urban traffic networks. *Applied Intelligence*, 44(1):208–231, 2016.

[24] Anahita Jamshidnejad, Bart De Schutter, and Mohammad J Mahjoob. Urban traffic control using a fuzzy multi-agent system. In *Control Conference (ECC), 2015 European*, pages 3041–3046. IEEE, 2015.

[25] Anahita Jamshidnejad, Ioannis Papamichail, Markos Papageorgiou, and Bart De Schutter. Sustainable model-predictive control in urban traffic networks: Efficient solution based on general smoothening methods. *IEEE Transactions on Control Systems Technology*, 2017.

[26] Chang Ouk Kim, Yunsun Park, and Jun-Geol Baek. Optimal signal control using adaptive dynamic programming. In *International Conference on Computational Science and Its Applications*, pages 148–160. Springer, 2005.

[27] Natallia Kokash. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, pages 1–8, 2005.

[28] Daniel Krajzewicz, Jakob Erdmann, Michael Behrisch, and Laura Bieker. Recent development and applications of SUMO - Simulation of Urban MObility. *International Journal On Advances in Systems and Measurements*, 5(3&4):128–138, December 2012.

[29] Daniel Krajzewicz, Robbin Blokpoel, Wolfgang Niebel, Cornelia Hebenstreit, Jérôme Härri, Michela Milano, Anna-Chiara Bellini, and Thomas Stützle. Towards a unified evaluation of traffic light algorithms. 2014.

[30] Changxi Ma and Ruichun He. Green wave traffic control system optimization based on adaptive genetic-artificial fish swarm algorithm. *Neural Computing and Applications*, pages 1–11, 2015.

[31] Magdi S Mahmoud, Faisal A Al-Nasser, and Fouad M Al-Sunni. Network-based strategies for signalised traffic intersections. *International Journal of Systems, Control and Communications*, 5(1):15–48, 2013.

[32] Takashi Nakatsuji and Terutoshi Kaku. Development of a self-organizing traffic control system using neural network models. *Transportation Research Record*, 1324(4), 1991.

[33] Michael Nielsen. Using neural nets to recognize handwritten digits, 2017. URL `http://neuralnetworksanddeeplearning.com/chap1.html`.

[34] Johan Janson Olstam and Andreas Tapani. Comparison of car-following models. Technical report, 2004.

[35] Markos Papageorgiou, Christina Diakaki, Vaya Dinopoulou, Apostolos Kotsialos, and Yibing Wang. Review of road traffic control strategies. *Proceedings of the IEEE*, 91(12):2043–2067, 2003.

[36] Judea Pearl. *Heuristics: intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc., Reading, MA, 1984. ISBN 0-201-05594-5.

[37] Rahul Putha, Luca Quadrifoglio, and Emily Zechman. Comparing ant colony optimization and genetic algorithm approaches for solving traffic signal coordination under oversaturation conditions. *Computer-Aided Civil and Infrastructure Engineering*, 27(1):14–28, 2012.

[38] Martin Riedmiller and Heinrich Braun. A direct adaptive method for faster backpropagation learning: The rprop algorithm. In *Neural Networks, 1993., IEEE International Conference on*, pages 586–591. IEEE, 1993.

[39] Nasser R. Sabar, Le Minh Kieu, Edward Chung, Takahiro Tsubota, and Paulo Eduardo Maciel de Almeida. A memetic algorithm for real world multi-intersection traffic signal optimisation problems. *Engineering Applications of Artificial Intelligence*, 63:45 – 53, 2017. ISSN 0952-1976. doi: http://dx.doi.org/10.1016/j.engappai.2017.04.021. URL `http://www.sciencedirect.com/science/article/pii/S0952197617300854`.

[40] Richard Sutton. *Reinforcement learning : an introduction*. MIT Press, Cambridge, Mass, 1998. ISBN 9780262193986.

[41] Tomer Toledo and Haris Koutsopoulos. Statistical validation of traffic simulation models. 1876:142–150, 2004.

[42] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

[43] FV Webster. Traffic signal settings, road research technical paper no. 39. *Road Research Laboratory*, 1958.

[44] Axel Wegener, Michał Piórkowski, Maxim Raya, Horst Hellbrück, Stefan Fischer, and Jean-Pierre Hubaux. Traci: an interface for coupling road traffic and network simulators. In *Proceedings of the 11th communications and networking simulation symposium*, pages 155–163. ACM, 2008.

[45] Junhua Wei, Anlin Wang, and Nianci Du. Study of self-organizing control of traffic signals in an urban network based on cellular automata. *IEEE transactions on vehicular technology*, 54(2):744–748, 2005.

[46] Michael Wirth, Matteo Fraschini, Martin Masek, and Michel Bruynooghe. Performance evaluation in image processing. *EURASIP journal on Applied signal processing*, 2006:211–211, 2006.

[47] Michael A Wirth. Performance evaluation of image processing algorithms in cade. *Technology in cancer research & treatment*, 4(2):159–172, 2005.

[48] Lotfi A Zadeh. Fuzzy sets. *Information and control*, 8(3):338–353, 1965.

[49] Lilin Zang, Lei Jia, and Yonggang Luo. An intelligent control method for urban traffic signal based on fuzzy neural network. In *Intelligent Control and Automation, 2006. WCICA 2006. The Sixth World Congress on*, volume 1, pages 3430–3434. IEEE, 2006.

[50] Limin Zhang, Guifen Chen, and Tianyu Zhang. Study on area coordination control system based on traffic state discrimination. In *Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), International Congress on*, pages 1248–1252. IEEE, 2016.

[51] Dongbin Zhao, Yujie Dai, and Zhen Zhang. Computational intelligence in urban traffic signal control: A survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4): 485–494, 2012.

[52] Xiaoke Zhou, Fei Zhu, Quan Liu, Yuchen Fu, and Wei Huang. A sarsa (λ)-based control model for real-time traffic light coordination. *The Scientific World Journal*, 2014, 2014.