Towards Autonomous and Distributed Traffic Signal Control

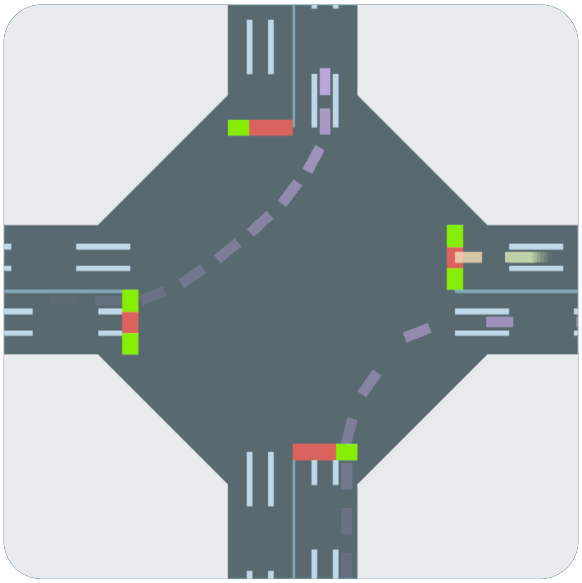
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Automatic Control Laboratory

Master Thesis

Towards Autonomous and Distributed Traffic Signal Control

Vinzenz Tütsch
August 28, 2023



Advisors
Dr. Kenan Zhang
Zhiyu He

Professors
Prof. Dr. Florian Dörfler
Prof. Dr. John Lygeros

Acknowledgements

First and foremost I am extremely grateful to my two supervisors, Dr. Kenan Zhang and Zhiyu He for their invaluable advice, continuous support, motivation, and patience during my master thesis. I would also like to acknowledge Prof. Dr. Florian Dörfler and Prof. Dr. John Lygeros for providing the opportunity to conduct this project. Lastly, I want to express my sincere appreciation to my family and friends for their wholehearted support and encouragement throughout this endeavor.

Zürich, August 2023

Vinzenz Tütsch

Abstract

The challenge of optimizing traffic signal control has significant implications for individual well-being, economics, and the environment. To tackle this, our thesis presents two core strategies. Firstly, we introduce a centralized approach rooted in the principles of Model Predictive Control (MPC). This centralized method utilizes a predictive traffic dynamics model to effectively establish a baseline performance for comparisons. In contrast, our primary focus is on the second strategy, which involves a distributed algorithm chosen for its scalability benefits in extensive traffic networks. A prominent instance of such a distributed algorithm is the Max Pressure (MP) algorithm, renowned for its theoretical maximization of throughput under specific assumptions - such as infinite queue lengths. To overcome this limitation, we incorporate a technique commonly employed in communication and queuing theory, referred to as the Lyapunov Drift-Plus-Penalty (LDPP) framework. While upholding that same assumption, the LDPP framework facilitates the integration of an additional penalty function designed to specifically address this constraint. Beyond mitigating the limitations of this assumption, this function also fosters collaboration among neighboring intersections by incorporating their actions. By adopting this approach, we successfully transform the original traffic signal control problem into a consensus challenge among neighboring intersections, departing from the assumption of independent intersection actions as seen in the MP formulation. This collaborative methodology strives for more globally optimal solutions. Moreover, we substantiate our approach with theoretical stability guarantees that limit the congestion size.

To solve this resulting consensus problem, two algorithms are employed. The first is a consensus-based Alternating Direction Method of Multipliers (ADMM) formulation, while the second is a custom Greedy algorithm. Both algorithms are executed in a fully distributed manner. While ADMM approaches close-to-optimal solutions, the Greedy algorithm approximates the optimal solution with a significantly reduced computational cost. Simulation results validate the effectiveness of our proposed method, demonstrating significant enhancements of travel time by up to 30% compared to the MP algorithm. Additionally, average queue lengths experience a congestion reduction of 40%, particularly in networks with shorter inter-lane distances. Furthermore, we illustrate that our algorithm showcases performance similar to the centralized formulation.

Contents

Acknowledgements **i**

Abstract **iii**

List of Figures **vii**

List of Tables **ix**

Acronyms **xi**

List of Symbols **xiii**

1 Introduction **1**
1.1 Background 1
1.2 State-of-the-Art 2
1.3 Problem Formulation 3
1.4 Contributions 3
1.5 Thesis Outline 4

2 Problem Definition and Preliminaries **5**
2.1 Problem Setting 5
2.2 Maximum Pressure Algorithm 7

3 Centralized Formulation **11**
3.1 Mixed Integer Linear Program 11
3.2 Modifications and Heuristic Methods 13

4 Distributed Formulation **17**
4.1 Theoretical Framework 17
4.2 Consensus 29

5 Performance Analysis **37**
5.1 Simulation Environment 37
5.2 Evaluation Metrics 39
5.3 Simulation Results 40

6 Conclusion and Future Work **57**
6.1 Conclusion 57
6.2 Future Work 58

Bibliography **58**

A Proof of Theorem 4.1.1 **63**

B Proof of Theorem 4.1.2 **65**

C Proof of Theorem 4.1.5 **67**

D Queue Model Validation (Fine Network) **69**

E Simulation Parameters **71**

List of Figures

2.1 Overview of the overall network structure. 6
2.2 Phase and traffic movements in an intersection. As an example, the dotted lines show phase ϕ_2 without including any right turns. 7
2.3 Illustration of the max pressure control assuming we only have 2 possible movements, namely North-South and East-West, we can depict the pressure for each phase. In the current example, we see that the pressure for N-S is higher and thus would get activated in the next update. Here we assumed that $c_{l,m} = 1$ for all lanes. 8

3.1 Illustration of expanding the time horizon be the introduction of the new parameter K. In this example we set $K = 5$, $\Delta = 2$, $T = 3$. The time points at which we approximate the model are denoted by the orange circles. 15

4.1 A 2x2 intersection network illustrating the relationships between various variables. The red dashed ellipse signifies the vector encompassing all local $\boldsymbol{x}$ vectors. . . . 31

5.1 The fine traffic network with a shorter block length 150 meters. 38
5.2 The coarse traffic network with a larger block length of 300 meters. 38
5.3 A depiction of the large traffic network featuring a block length of 150 meters and an arrangement of intersections in a 10×10 grid layout. 39
5.4 The mean and standard deviation of the prediction error of simulations with two different Horizon lengths simulated in the coarse network. More prediction steps give a lower mean error. 41
5.5 The mean and standard deviation of the prediction error of simulations with two different capacity and exogenous flows simulated in the coarse network. A lower flow increases the accuracy of the model overall. 42
5.6 The average and standard deviation of prediction errors from simulations, comparing scenarios with and without the inclusion of entry lanes in the coarse network. Notably, the error decreases when entry lanes are omitted. 43
5.7 The mean and standard deviation of prediction errors in simulations in the coarse network, both when utilizing the queue model approximation and when using the original model. Notably, the approximation yields slightly inferior results compared to the original model. 43
5.8 Comparison of travel time and throughput of all algorithms in the fine network . 45
5.9 Comparison of car speeds for different algorithms in the fine network. The red dotted line indicates the timestamp at which the last car enters the network. . . . 46
5.10 Comparison of congestion metrics and waiting time metrics within the fine network. The red dotted line indicates the timestamp at which the last car enters the network. 47
5.11 Comparison of different optimality gaps in the fine network. ADMM is formulated as a minimization problem, whereas Greedy is a maximization problem 48

5.12 Comparison of computation times for one simulation time step in the fine network 49
5.13 Comparative analysis of the algorithmic phase transition probabilities in the fine network 49
5.14 Comparison of travel time and throughput of all algorithms in the coarse network 51
5.15 Comparison of car speeds for different algorithms in the coarse network. The red dotted line indicates the timestamp at which the last car enters the network . . . 51
5.16 Comparison of congestion metrics and waiting time metrics within the coarse network 52
5.17 Comparison of different optimality gaps in the coarse network. ADMM is formulated as a minimization problem, whereas Greedy is a maximization problem . . 53
5.18 Comparison of computation times for one simulation time step in the coarse network. 54
5.19 Comparative analysis of the algorithmic phase transition probabilities in the coarse network 54
5.20 Comparison of travel time and throughput of all algorithms in the large network. 56

D.1 The mean and standard deviation of the prediction error of simulations with two different capacity and exogenous flows simulated in the fine network. A lower flow increases the accuracy of the model overall. 69
D.2 The mean and standard deviation of the prediction error of simulations with and without considering entry lanes in the fine network. The error decreases as we do not consider the entry lanes. 70
D.3 The mean and standard deviation of the prediction error of simulations with and without approximating the queue model in the fine network. The approximation is slightly worse than the original model. 70

List of Tables

5.1 All recorded metrics and units for comparison. 40

E.1 The simulation parameters for the centralized controller. 71
E.2 The simulation parameter for ADMM with the binary penalty. 71
E.3 The simulation parameter for ADMM with the continuous penalty. 71
E.4 The simulation parameter for the Greedy algorithm with the continuous penalty. 71

Acronyms

ADMM Alternating Direction Method of Multipliers.

CTM Cell Transmission Model.

DRL Deep Reinforcement Learning.

LD Lyapunov Drift.

LDPP Lyapunov Drift-Plus-Penalty.

m.r.s mean rate stable.

MILP Mixed Integer Linear Program.

MP Max Pressure.

MPC Model Predictive Control.

RL Reinforcement Learning.

List of Symbols

Traffic Model

$\mathcal{G}$	Set of all intersections in the network	
$\mathcal{L}$	Set of all links/lanes in the network	
$\mathcal{L}_{entry}$	Set of all entry links/lanes that enter the network	
$\mathcal{L}_{exit}$	Set of all exit links/lanes that exit the network	
(l, m)	Movement from lane l to lane m	
$\mathcal{M}$	Set of all possible movements (l, m) in a network	
$r_{l,m}$	Proportion of cars in lane l intending to flow into lane m	
d_k	Inflow of cars that enter the network	[cars/sec]
$s_{l,m}$	Binary variable representing the green light of movement (l, m)	
ϕ_k	Set of movements that can receive the right of way simultaneously without collisions called phase k	
Φ_i	Set of phases that are defined in intersection i	
$q_{l,m}$	Number of cars in the queue (l, m)	[cars]
C / $c_{l,m}$	Saturation flow rate	[cars/sec]
$\mathcal{U}_l$	Set of upstream lanes of lane l	
$\mathcal{D}_l$	Set of downstream lanes of lane l	

Maximum Pressure Algorithm

$w_{l,m}$	Weight of movement (l, m) as defined by the Maximum Pressure algorithm	
P_{i,ϕ_k}	Pressure in intersection i for phase ϕ_k as defind by the Maximum Pressure algorithm	
ϕ_i^*	Optimal phase determined by the Maximum Pressure algorithm for intersection i	
D	Feasible set of demand vectors	

Centralized Formulation

$f_i(\Phi)$	Local objective for intersection i	

α	Discount factor	
$y_{l,m}$	Maximal outflow of lane m (not considering the downstream lane)	[cars/sec]
$\tilde{y}_{l,m}$	Maximal outflow of lane m (considering the downstream lane)	[cars/sec]
q_{max}	Maximal capacity of all lanes	[cars]
K	Interval between two subsequent traffic predictions	[cars]

Distributed Formulation

$Q_{l,m}$	Real-valued random variable representing the queue of movement (l, m)	
$\mathbf{Q}$	Vector of random variables $Q_{l,m}$	
$a_{l,m}$	Maximal inflow into queue (l, m)	[cars/sec]
$b_{l,m}$	Maximal outflow of queue (l, m)	[cars/sec]
$L(\mathbf{Q}(t))$	Lyapunov function	
$\Delta(\mathbf{Q}(t))$	One-slot conditional Lyapunov drift	
$p(t)$	Penalty function	
q_{thresh}	Threshold value used in the binary penalty	[cars]
$\tilde{P}_{i,\phi_k}$	Modified pressure definition incorporating the penalty function	
$\hat{P}_i$	Local objective function of intersection i	
O	Global objective function	
$\boldsymbol{x}_i$	Local binary decision matrix for the ADMM and Greedy algorithm	
$\boldsymbol{z}$	Global binary decision matrix for the ADMM algorithm	
$\tilde{\boldsymbol{z}}$	Global copy of the local binary decision matrix $\boldsymbol{x}_i$ for the ADMM algorithm	
$\boldsymbol{\lambda}$	Dual variable for the ADMM algorithm	
ρ	Dual update step size	
$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda})$	Lagrangian	
DET_i	Termination Flag for intersection i	
$\boldsymbol{X}_i$	Tensor incorporating decision of intersection i and its neighbors	
$\mathcal{N}_i$	Set of neighboring intersection of intersection i	
$\tilde{\mathcal{N}}_i$	Set of neighboring intersection of intersection i including intersection i	

Chapter 1

Introduction

1.1 Background

Traffic is an omnipresent problem that most people deal with on a daily basis. Many cities are experiencing an increase in traffic. Only last year, the UK Department for Transport published a forecast for future traffic volumes [39]. Road traffic in England and Wales is expected to increase by up to 54% between 2025 and 2060, which would be a huge increase. Not only will traffic bring more cars and therefore more waiting times, it will also have a negative impact in several areas. On the one hand, the exhaust fumes from the vehicles pollute the environment, and on the other hand, the increase in traffic also harms the economy. Sri Lanka, for example, is estimated to have lost up to 40 billion rupees due to traffic congestion in 2012 [17]. This is equivalent to about 1.5% of Sri Lanka's gross domestic production. This is compounded by the problem of individual well-being and health, such as excessive fatigue and even problems related to the cardiovascular systems [30].

In Switzerland, over 95 % of the traffic signals utilize adaptive control techniques that dynamically respond to traffic situations [31]. This is a significant change from using fixed time signals, which lack dependence on traffic and are currently reserved only for situations where hardware failure has occurred and the safe routing of vehicles is paramount. Nevertheless, the historical development of traffic management systems has shown the early emergence of intelligent transport systems in the 20th century, as demonstrated in the literature [28]. Fixed time control strategies remain a persistent element among the pioneering methods, and are still integrated into the urban landscape of many cities today.

However, a significant limitation of fixed timed control strategies becomes apparent when considering their inability to adapt to daily variations in traffic flow, resulting in longer waiting times. A comprehensive study [18] found that fixed time control methods contribute to over 10 % of global traffic delays. As a result, a substantial amount of research has focused on developing adaptive control strategies utilizing data from various sources such as induction loops, infrared sensors, and video analysis [38]. These endeavors have resulted in numerous remarkable algorithms such as SCOOT [16], SCATS [23], and PRODYN [15], all of which function as central control systems [38].

Even though these advancements have taken place, the financial cost of implementing and maintaining such adaptive systems is substantial. Therefore, numerous cities implement hybrid systems that blend multiple different types of systems to balance effectiveness and cost efficiency. Toronto provides an illustrative example. It has a primary traffic signal system but still employs SCOOT in combination for approximately 15% of its intersections. In essence, though fixed time

control strategies are historically important, modern traffic management practices emphasize adaptability and responsiveness as prerequisites.

1.2 State-of-the-Art

In recent years, the problem of traffic signal control has attracted approaches from many research fields, including evolutionary algorithms, linear programming, fuzzy logic, swarm intelligence, image processing, and even artificial intelligence [38]. In linear programming, a popular choice is what is known as Cell Transmission Model (CTM), which was introduced in 1995 by Daganzo [8] to solve the kinematic wave equation. CTM predicts future traffic by estimation of traffic flow and density at finite approximation points. Thus, both time and space are discretized. Roads are divided into smaller segments called cells, and the model can capture both queue dynamics and the traffic phenomenon known as shock waves [22]. Several methods have been proposed using this formulation, incorporating additional traffic light constraints and using this model to formulate a mixed-integer linear program to be optimized [21, 27]. For example, Timotheou et al. [37] presents two online distributed strategies based on the CTM for the optimization of the travel time. They optimize over a neighborhood of intersections to encourage the intersections to collaborate on the problem rather than to behave in a selfish way. This year, a new publication extended the original CTM by simultaneously optimizing signal timing, speed limits and lane allocation. There is still more potential to CTM to improve the formulation further. Thus, CTM remains a subject of research in the future.

Another pivotal aspect of research involves the transition from a centralized approach to a distributed framework for addressing the issue, as already introduced by certain algorithms discussed earlier. Notably, in 2013, Pravin Varaiya presented a paper on a distributed methodology known as the Max Pressure (MP) algorithm, which can be seamlessly applied to individual intersections without necessitating any communication [40]. This algorithm also provides theoretical insights into how to achieve maximum throughput under certain assumptions. Its simple formulation has subsequently provided the basis for much subsequent research. These span from simulation-based investigations [4, 33] to further refinements aimed at addressing practical challenges stemming from the assumptions inherent in the MP algorithm [13, 44]. A detailed examination of some of these challenges arising in the context of MP is presented in Chapter 2.

Conventional methods for managing traffic signals, like the Max-pressure (MP) algorithm, commonly rely on specific assumptions, such as the notion of infinitely lengthy lanes, within the traffic model. These assumptions are utilized to simplify the complexity of the problem and make it more feasible to address. However, these assumptions do not always entirely reflect reality. This is where Reinforcement Learning (RL) comes in, a promising area of research that is still emerging. One of the main reasons for this is that it does not depend on an underlying model or heuristic assumptions [42]. It simply observes states, acts, and learns to incrementally improve the signalling process. Traditional RL has had the limitation that the state space can't be very large, due to the *Curse of Dimensionality* [5], as the state-action pairs grow exponentially with the dimension of the states and actions. This problem has largely been resolved with the advent of DRL. Deep learning helps to approximate the Q-function, which improves the scalability and reduces the size of the state space. Since then, numerous agent formulations have been examined in the literature. The description of the state space ranges from simple discretization [3] to large state spaces that even use images as input to their network [10, 20]. However, as noted by Zheng et al. [47], a larger state space description does not necessarily lead to improved behavior, and they suggested using smaller state spaces instead. The same applies to the design of

reward functions, where the introduction of complex reward functions results in highly sensitive outcomes. Even a minor difference can result in completely different outcomes in a simulation. Thus, Wei et al. [41] proposes combining RL with theoretical traffic signal control strategies. Specifically, they combined the framework from MP and utilized it in their reward function to facilitate the learning process and conceivably reduce the system's sensitivity. Nevertheless, it's important to highlight that in addition to the steep learning expenses, Reinforcement Learning (RL) involves acquiring knowledge through trial-and-error. Implementing a real-world RL model that continuously updates online can be both risky and expensive.

1.3 Problem Formulation

As indicated previously, a diverse range of solution strategies are available for addressing the traffic signal control problem, which involves optimizing travel time by intelligently managing traffic lights. Each strategy has its specific areas of focus. In this context, our goal is to establish a framework that overcomes the limitations often associated with Deep Reinforcement Learning, including challenges related to convergence, interpretability, and generalization as well as robustness issues [25]. Consequently, we intend to develop a lightweight policy - one that is straightforward to implement and doesn't necessitate solving complex linear programs, a requirement seen in certain algorithms within the Cell Transmission Model. Additionally, we are working toward a distributed algorithm that can scale effectively to encompass an entire city's traffic. While MP already represents a fully distributed approach, with each intersection independently controlling traffic lights, our objective is to expand this framework. We aim to accomplish this by introducing incentives for neighboring intersections to collaborate, leading to the attainment of more optimal global solutions.

1.4 Contributions

Following a similar approach as presented by [41], which combined DRL with MP, our objective is to merge the MP technique with the concept of rewarding or penalizing each intersection based on its choices and those of its neighboring intersections. However, instead of applying this method within a DRL framework, we utilize MP as the foundation of our approach, extending its formulation to foster effective cooperation and thereby reshape the traffic signal problem into a consensus problem.

1. Our contribution entails the proposal of a novel controller type for addressing the traffic signal control problem. We pivot our approach around MP, while simultaneously extending its formulation to foster collaboration between intersections. We achieve this by adapting the Lyapunov Drift-Plus-Penalty (LDPP) theory from communication and queuing systems, thereby modifying its context from communication networks to the realm of traffic signal control. Through the integration of two novel penalty functions, the modified formulation deviates from the isolated optimization of intersections as observed in MP. Instead, it incentivizes intersections to work collaboratively, resulting in collective travel time reduction. This advancement results in a scalable, completely distributed algorithm. Furthermore, we showcase that our algorithm can achieve the theoretical maximum throughput similar to MP, or can be adapted to resolve challenges such as infinite queue lengths commonly encountered in MP.

2. Our solution effectively transforms the traffic signal control problem into a consensus problem. To address this transformation, we introduce two distinct consensus algorithms. The first is the Alternating Direction Method of Multipliers (ADMM), and the second is a

custom-designed Greedy algorithm that approximates the ADMM solution with enhanced computational efficiency. Both algorithms are inherently decentralized, relying solely on communication among neighboring components.

3. Within a simulated environment, we validate our approach and present a comparative analysis against the MP formulation, highlighting its superior performance. As a baseline reference, we additionally construct a centralized formulation that employs predictive horizons to anticipate future traffic conditions. We illustrate that our algorithm consistently achieves outcomes akin to those of the centralized formulation, reaffirming its efficacy.

1.5 Thesis Outline

This thesis is structured into the following sections: Chapter 2 gives a detailed introduction to the problem formulation and an explanation of the original MP formulation. Chapter 3 introduces the centralized problem formulation, as well as possible modifications to it. Chapter 4 begins with a complete analysis of the theoretical aspects of MP and LDPP, including a proof of their stability. Subsequently, both the ADMM and the Greedy optimal consensus algorithm are introduced. Chapter 5 verifies the used traffic model and shows comparisons of different metrics in a simulation for all aforementioned algorithms. Finally, in Chapter 6, the thesis is concluded and an outlook of possible future extensions of this work is provided.

Chapter 2

Problem Definition and Preliminaries

2.1 Problem Setting

This section outlines the precise traffic model and its related terminology. A directed network comprises links denoted as $\mathcal{L}$ and nodes as $\mathcal{G}$. Links are essentially just separate lanes on a road. The network has three types of lanes: internal links that connect an intersection to its neighbor, entry lanes $\mathcal{L}_{entry}$ that have no start node, and exit lanes $\mathcal{L}_{exit}$ that have no end node. Moreover, we define the exogenous inflow or demand into an entry lane $k \in \mathcal{L}_{entry}$ as d_k in vehicles per second. In this thesis, we will use the terms exogenous flow and demand interchangeably. A traffic movement through an intersection denotes the traffic flow from an upstream lane l to a downstream lane m, which is represented by (l, m). The whole set of possible movements in the network is recognized as $\mathcal{M}$, which is precisely equal to the entire series of queues. The turn ratio of a traffic movement is designated as $r_{l,m}$, and it describes the proportion of vehicles on lane l that would flow into the downstream lane m.

In this part, we will introduce a more detailed view of an intersection. An intersection is a set of upstream and downstream lanes that are connected as traffic movements. In our setting, we have one lane for each turning direction defined, and every turn has its traffic light, designated by the variable $s_{l,m}$. When a car waits in a specific queue, the turning direction (and thus the road) is determined, but the exact downstream lane is unknown, as it depends on the direction in which the car wants to proceed further. The binary variable $s_{l,m}$ is defined as 1 if the movement (l, m) has the right of way and as 0 if it receives a red light. A phase $\phi_k = \{(l, m)|s_{l,m} = 1\},\ l \in \mathcal{L} \setminus \mathcal{L}_{exit},\ m \in \mathcal{D}_l$ can simply be defined as a combination of traffic movements in which cars can simultaneously take their right of way without the risk of collision. We represent the set of achievable phases as $\Phi_i, ; i \in \mathcal{G}$, and when a phase is in operation (denoted by $\phi_k = 1$), it signifies that all movements associated with that phase receive a green signal concurrently. Many research papers [41, 46] only consider four different phases. However, as introduced by Zheng et al. [48], we use eight phases to provide the algorithm with more freedom to choose the best option independently. It is also important to note that we assume that right turns are always allowed, as it is common in literature.

The dynamics in our network are described using the widely-used store-and-forward model [2]. The model states refer to queues, where a queue $q_{l,m}$ exists for each traffic movement that indicates the number of cars on lane l planning to turn on to lane m. The dynamics can be summarized as follows:

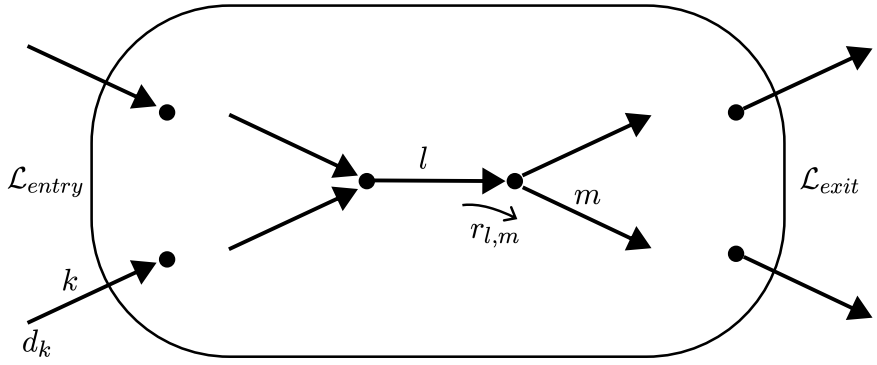


Fig. 2.1: Overview of the overall network structure.

$$q_{l,m}(t+1) = q_{l,m}(t) - y_{l,m}(t)s_{l,m}(t) + \sum_{k \in \mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)r_{l,m}(t) \qquad l \in \mathcal{L} \setminus \mathcal{L}_{entry} \qquad (2.1)$$

$$q_{l,m}(t+1) = q_{l,m}(t) - y_{l,m}(t)s_{l,m}(t) + d_l(t)r_{l,m}(t) \qquad l \in \mathcal{L}_{entry} \qquad (2.2)$$

$$q_{l,m}(t+1) = q_{l,m}(t) - y_{l,m}(t) + \sum_{k \in \mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)r_{l,m}(t) \qquad l \in \mathcal{L}_{exit} \qquad (2.3)$$

$$y_{l,m}(t) = \min\{q_{l,m}(t), c_{l,m}(t)\} \qquad l \in \mathcal{L} \qquad (2.4)$$

where $\mathcal{U}_l$ is defined as the upstream lanes of lane l and $c_{l,m}(t)$ are the saturation flow rates. This refers to the number of vehicles that can depart if the appropriate traffic light is activated in one time step. We also assume that the network is homogeneous [37], which means that we define $c_{l,m}(t) = C,\ l \in \mathcal{L}, m \in \mathcal{D}_l$ as a constant for the whole network. Exit lanes lack traffic lights; hence, we assume a steady outflow from these lanes whenever they contain vehicles.

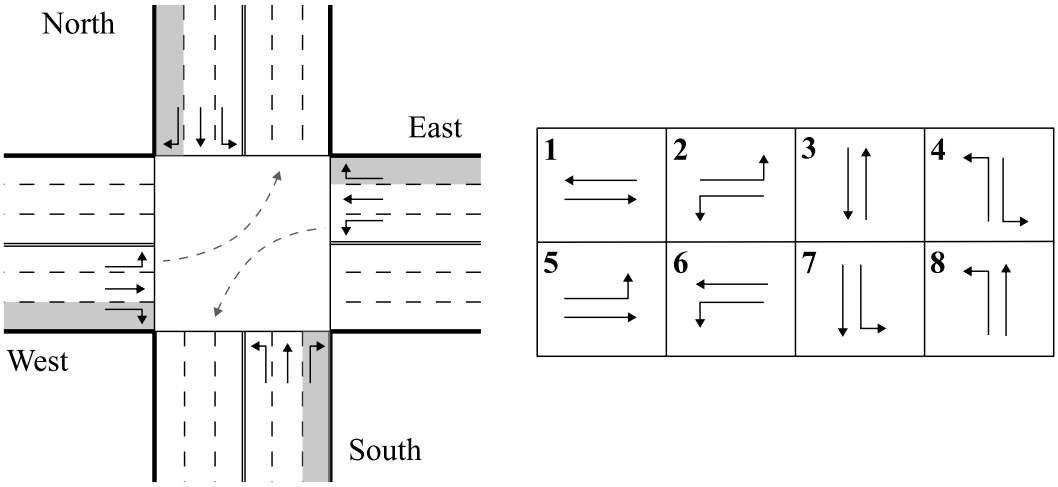


Fig. 2.2: Phase and traffic movements in an intersection. As an example, the dotted lines show phase ϕ_2 without including any right turns.

2.2 Maximum Pressure Algorithm

Tassiulas et al. [35] first developed a maximum stability control algorithm, called backpressure or MP control, for a queuing network for interdependent servers in 1992. It schedules activations for different servers to ensure that the network remains maximally stable. Subsequently, Varaiya [40] and Wongpiromsarn et al. [43] demonstrated maximum stability in the design of a traffic signal controller using the same decentralized algorithm, independently from one another. They utilized the model described in (2.1) and proved that, under specific assumptions, the algorithm guarantees maximum throughput. The critical assumptions are the need for separate queues for each turning movement, the unboundedness of queue capacities, and the actuation method, which periodically decides which phase is active during a fixed time step [24]. Regarding their algorithm, turn ratios and saturation flow rates information are sufficient and, unlike other model predictive control approaches, no knowledge of actual or expected traffic demand is required. Afterward, several modifications to this algorithm have arisen. For example, Gregoire et al. [13] has demonstrated that the MP algorithm is not work conserving and has the potential to cause congestion propagation situations. They adapted the algorithm to take queue capacities into consideration and showed that for high-demand situations, their algorithm outperformed MP. Xiao et al. [44] proposed another modified algorithm that explicitly proved stability for finite queues, without assuming infinite queue capacities. Most MP algorithms are acyclic, meaning they do not guarantee sequential activations for each phase. In reality, this may lead to complications as there is no guarantee that a single car has to wait infinitely long to get the right of way and may thus lead to confusion and anger among individuals. However, there exist some cyclic MP algorithms that show stability for a traffic network, such as Le et al. [19]. Nevertheless, in the following, we present the original acyclic maximum pressure algorithm formulated by Varaiya [40] and Wongpiromsarn et al. [43].

The actual MP formalism, as outlined by Varaiya [40], will now be introduced. The algorithm defines a weight per traffic flow to determine the urgency of a lane to obtain a green light as:

$$w_{l,m}(t) = q_{l,m}(t) - \sum_{p \in \mathcal{D}_m} r_{m,p}(t) q_{m,p}(t) \tag{2.5}$$

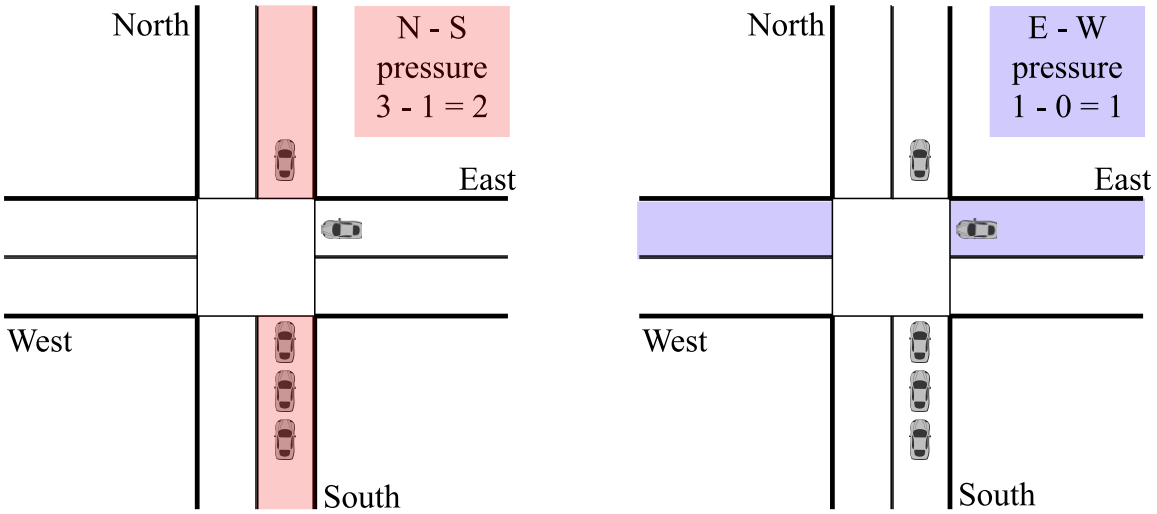


Fig. 2.3: Illustration of the max pressure control assuming we only have 2 possible movements, namely North-South and East-West, we can depict the pressure for each phase. In the current example, we see that the pressure for N-S is higher and thus would get activated in the next update. Here we assumed that $c_{l,m} = 1$ for all lanes.

where $\mathcal{D}_m$ is the set of downstream lanes of lane m. If we define $\bar{q}_{m,p} = \sum_{p \in \mathcal{D}_m} r_{m,p} q_{m,p}$ as the average downstream queue length, then the weight is simply the difference between upstream and downstream queue length. This weight is then used to define the so-called pressure $P_{i,k}(t)$ as:

$$P_{i,\phi_k}(t) = \sum_{(l,m) \in \phi_k} w_{l,m}(t) c_{l,m}(t) s_{l,m}(t) \tag{2.6}$$

$$= \sum_{\substack{(l,m) \in \phi_k \\ s_{l,m}=1}} w_{l,m}(t) c_{l,m}(t) \tag{2.7}$$

The actual MP policy is now very straightforward. It just selects the phase with the greatest pressure and activates it during the next period.

$$\phi_i^*(t) = \arg\max\{P_{i,\phi_k}(t) | \phi_k \in \Phi\} \tag{2.8}$$

Due to the fact that each intersection can independently perform the algorithm without requiring communication between them, this policy makes it remarkably straightforward to decentralize implementation. Further, Varaiya demonstrated in his work [40] that when the demand d_l belongs to a feasible set D, it ensures that MP achieves maximal throughput. Conversely, if the demand falls outside this set ($d_l \notin D$), then there exists no controller, which is capable of stabilizing the system. The feasible set is characterized as the highest achievable demand that permits the existence of a theoretical controller limiting the average of each mean lane length.

The algorithm further defines a maximum pressure time step $\Delta \in \mathbb{R}$ that defines the interval between two subsequent traffic light updates. Each intersection chooses its best phase during

Algorithm 1 Max Pressure (Code for intersection i)

if $t \mod \Delta = 0$ **then** ▷ t is the simulation time
  for $(l, m) \in \mathcal{L}_i$ **do**
    Calculate weight $w_{l,m}(t) \leftarrow q_{l,m}(t) - \sum_{p \in \mathcal{D}_m} r_{m,p}(t) q_{m,p}(t)$
  end for

  for $\phi_s \in \Phi$ **do**
    Calculate pressure $P_{i,k}(t) \leftarrow \sum_{\substack{(l,m) \in \phi_k \\ s_{l,m}=1}} w_{l,m}(t) c_{l,m}(t)$
  end for

  Determine $\phi_i^*(t) \leftarrow \arg\max\{P_{i,\phi_k}(t) | \phi_k \in \Phi\}$ and apply it
end if

the update in accordance with (2.8) and uses that phase for the following Δ seconds, at which point another update will take place. Thus, the algorithm can be summarized as:

Chapter 3

Centralized Formulation

The subsequent centralized formulation establishes a reference performance against which the remaining algorithms will be evaluated. By incorporating a predictive horizon and leveraging information from the entire network, this formulation is set up to outperform the other algorithms. At its core, the algorithm draws inspiration from the principles of the Max Pressure formulation. The ultimate objective of the controller is to minimize travel time, a metric influenced by various factors such as traffic signals and movements. However, this metric doesn't offer immediate feedback on the effectiveness of our real-time optimization; rather, it presents a delayed reward [41]. Consequently, directly minimizing travel time becomes a complex challenge, often requiring the introduction of a surrogate reward functions that provides instantaneous feedback. We tackle this issue by combining the MP formulation with our traffic model and formulated it as a mixed integer linear program.

3.1 Mixed Integer Linear Program

Max Pressure only focuses on the instantaneous time step, trying to optimize traffic throughput. Though theoretically this delivers the maximum throughput, the assumption of infinite queue lengths is flawed, which may lead to suboptimal solutions. While maintaining the crucial component of the algorithm, we intend to address this problem. As is typical for an MPC controller, our algorithm provides a prediction horizon T that anticipates the traffic circumstances in the future. Then, instead of making a decision based simply on the knowledge available right now, we integrate it with future forecasts to create our best possible approach. Explicitly, we specify the following as our goal function:

$$\max \sum_{i \in \mathcal{G}} f_i(\Phi) = \max \sum_{i \in \mathcal{G}} \sum_{\tau=0}^{T} \sum_{\phi_k \in \Phi_i} \alpha^\tau P_{i,\phi_k}(\tau) \tag{3.1}$$

where $\alpha \in [0, 1]$ is a tunable discount factor and setting $\alpha = 0$, results in the original maximum pressure objective function. The pressure $P_s(t)$ and weight $w_{l,m}(t)$ are defined the same way as in (2.6) and (2.5) respectively:

$$w_{l,m}(t) = q_{l,m}(t) - \sum_{p \in \mathcal{D}_m} r_{m,p}(t) q_{m,p}(t) \qquad \begin{matrix} \forall l \in \mathcal{L} \setminus \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.2}$$

$$P_{i,k}(t) = \sum_{\substack{(l,m) \in \phi_k \\ s_{l,m}=1}} w_{l,m}(t) c_{l,m}(t) \qquad \begin{matrix} \forall i \in \mathcal{G} \\ \forall k \in \Phi_i \end{matrix} \tag{3.3}$$

In order to ensure passenger safety and prevent collisions, the system requires the activation of only one phase at any given time. This entails the imposition of an additional set of constraints, outlined as follows:

$$\sum_{\phi_k \in \Phi_i} \phi_k = 1 \qquad \begin{matrix} \forall i \in \mathcal{G} \\ \forall k \in |\Phi| \end{matrix} \tag{3.4}$$

Similarly to the MP approach, the construction of the linear program relies on the traffic model elucidated in Equation (2.1). This model serves as the foundation for predicting the future states of each lane.

$$\begin{aligned} q_{l,m}(t+1) = q_{l,m}(t) &- \sum_{\phi_{i,k} \in \Phi_i} \sum_{s_{l,m} \in \phi_{i,k}} \tilde{y}_{l,m}(t) s_{l,m}(t) \\ &+ \sum_{k \in \mathcal{U}_l} y_{k,l}(t) s_{k,l}(t) r_{l,m}(t) \end{aligned} \qquad \begin{matrix} \forall l \in \mathcal{L} \setminus \{\mathcal{L}_{entry} \cup \mathcal{L}_{exit}\} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.5}$$

$$q_{l,m}(t+1) = q_{l,m}(t) - \sum_{\phi_{i,k} \in \Phi_i} \sum_{s_{l,m} \in \phi_{i,k}} \tilde{y}_{l,m}(t) s_{l,m}(t) + d_l(t) r_{l,m}(t) \qquad \begin{matrix} \forall l \in \mathcal{L}_{entry} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.6}$$

$$q_{l,m}(t+1) = q_{l,m}(t) - \sum_{\phi_{i,k} \in \Phi_i} \sum_{s_{l,m} \in \phi_{i,k}} y_{l,m}(t) s_{l,m}(t) + \sum_{k \in \mathcal{U}_l} y_{k,l}(t) s_{k,l}(t) r_{l,m}(t) \qquad \begin{matrix} \forall l \in \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.7}$$

We made a slight adjustment to the lanes' outflow so that they now account for the capacity of the downstream lanes and limit the flow to prevent queue spillover. Specifically, we define the outflow and inflow as:

$$y_{l,m}(t) = \min\{q_{l,m}(t), C\} \qquad \begin{matrix} \forall l \in \mathcal{L} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.8}$$

$$\tilde{y}_{l,m}(t) = \min\left\{q_{l,m}(t), C, \min_{p \in \mathcal{D}_m}\{q_{max} - q_{m,p}\}\right\} \qquad \begin{matrix} \forall l \in \mathcal{L} \setminus \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{3.9}$$

Equation (3.9) restricts the outflow of a lane, considering the most congested downstream lane. Our assumption here is that all lanes share an identical maximal capacity, denoted as q_{max}. While

this perspective is cautiously conservative, it is plausible that other downstream lanes could potentially accommodate more vehicles. However, given the uncertainty about the downstream lane a vehicle will enter - as intersections are unaware of the cars' routes - this approach effectively minimizes the potential for queue overflow. Given that the min function isn't linear, we can transform (3.8) and (3.9) with a straightforward conversion:

$$y_{l,m}(t) \leq q_{l,m}(t) \quad \wedge \quad y_{l,m}(t) \leq C \qquad \begin{array}{l} \forall l \in \mathcal{L} \\ \forall m \in \mathcal{D}_l \end{array} \tag{3.10}$$

$$\tilde{y}_{l,m}(t) \leq q_{l,m}(t) \quad \wedge \quad \tilde{y}_{l,m}(t) \leq C \quad \wedge \quad \tilde{y}_{l,m}(t) \leq e_m \qquad \begin{array}{l} \forall l \in \mathcal{L} \setminus \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{array} \tag{3.11}$$

$$e_m \leq q_{max} - q_{m,p} \qquad \begin{array}{l} \forall p \in \mathcal{D}_m \\ \forall l \in \mathcal{L} \setminus \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{array} \tag{3.12}$$

The original MP algorithm necessitates the availability of precise turn ratios, or as proposed by Varaiya [40], these ratios could be approximated, potentially utilizing historical data. However, we opt for a straightforward approach, grounded in our assumption of uniform flow. Given this assumption, we consider a uniform distribution of traffic across all lanes. Consequently, we define the turn ratio as follows:

$$r_{l,m}(t) = \frac{1}{|\mathcal{D}_l|} \qquad \begin{array}{l} \forall l \in \mathcal{L} \setminus \mathcal{L}_{exit} \\ \forall m \in \mathcal{D}_l \end{array} \tag{3.13}$$

With all constraints now established, we can consolidate the comprehensive Mixed Integer Linear Program (MILP) as follows:

$$\max \sum_{i \in \mathcal{G}} \sum_{\tau=0}^{T} \sum_{\phi_s \in \Phi} \alpha^{\tau} P_{i,\phi_k}(\tau) \tag{3.14}$$

s.t.: - constraints: (3.2) – (3.4) & (3.10) – (3.13)

- model dynamics: (3.5) – (3.7)

- initial conditions

The algorithm follows the same sequence as the MP approach. In other words, at time step $t = \tau$, we compute the optimal actions for each traffic light and implement these actions for the subsequent Δ seconds, after which we reoptimize at time $t = \tau + \Delta$.

3.2 Modifications and Heuristic Methods

The MILP formulation (3.14) performs effectively and serves as the aforementioned baseline. However, a notable challenge of the current formulation is the substantial number of constraints

introduced for each lane, leading to slower optimization processes for extended prediction horizons. In the subsequent sections, we introduce two modifications aimed at enhancing its practical utility.

Approximations

Equations (3.10) to (3.12) introduce a substantial number of variables and constraints into the optimization problem, thereby causing a notable slowdown in the optimization process. Although the Gurobi optimization toolbox [36] offers an option to directly model min functions, it does not support the inclusion of summations within the min function. For instance, expressions like $\min_{p\in\mathcal{D}_m} q_{max} - q_{m,p}$ require the introduction of additional variables to first model the sum, thereby augmenting the overall count of variables and constraints once more. Here, we try to reduce this amount of using an approximation of the model. In an attempt to address these challenges, we aim to alleviate this issue by utilizing an approximation of the model.

Assuming that the traffic network experiences traffic flow on nearly every lane, or at least a majority of them, we can attempt to approximate these equations using the following approach:

$$y_{l,m}(t) = \min\{q_{l,m}(t), C\} \approx C \tag{3.15}$$

$$\tilde{y}_{l,m}(t) = \min\left\{q_{l,m}(t), C, \min_{p\in\mathcal{D}_m}\{q_{max} - q_{m,p}\}\right\} \approx C \tag{3.16}$$

The constant C takes on a relatively conservative value, serving to restrict the number of cars capable of transitioning from one lane to another within a single time step, which corresponds to one second within the simulation. Consequently, this approximation maintains a reasonable degree of accuracy when lanes are not lacking vehicles. Further substantiation of the validity of this approximation will be provided in Chapter 5.

Expanding the Time Horizon

Although it's feasible to set $T < \Delta$, it's generally preferable to have $T > \Delta$ to incorporate multiple optimization steps into the future. However, as T increases, it introduces an additional extensive set of constraints to the problem formulation, consequently resulting in substantial slowdowns. Furthermore, predicting the traffic situation at each time step, i.e., every second, might not be necessary. Instead, we can accommodate larger prediction intervals, thereby forecasting the future state only at intervals of K simulation time steps. This approach effectively extends the effective prediction horizon without introducing a higher number of constraints. In Figure 3.1, we illustrate a scenario in which we choose $K = 5$, $\Delta = 2$, and set the time horizon to $T = 3$. This adjustment effectively extends our original horizon from just 3 sec to a total of 15 sec. Simultaneously, a traffic light update occurs at 10 sec. In the subsequent sections of this thesis, we will employ two distinct notations. When we use Δ without units, it indicates the number of prediction time steps between successive traffic light updates. On the other hand, when we employ the unit sec with Δ, it signifies the time interval in seconds before the occurrence of the next traffic light update.

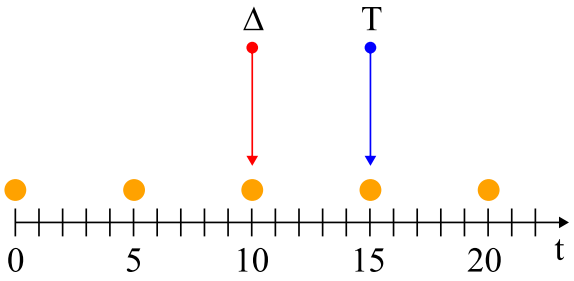


Fig. 3.1: Illustration of expanding the time horizon be the introduction of the new parameter K. In this example we set $K = 5$, $\Delta = 2$, $T = 3$. The time points at which we approximate the model are denoted by the orange circles.

Chapter 4

Distributed Formulation

A pivotal aspect of the centralized formulation lies in the observation that as the network size increases, both the number of optimization variables and constraints grow exponentially. This characteristic renders the approach impractical for scaling up to the size of an actual city, where traffic light adjustments must occur promptly to prevent extended waiting times for vehicles. To address this, we endeavor to decompose the problem into smaller, scalable subproblems, specifically addressing individual intersections. By fostering collaboration among the intersections, a coherent solution can be achieved. This approach offers a notable advantage: the problem can be disintegrated and distributed, facilitating faster and more scalable solutions.

Conversely, the Max Pressure method is inherently distributed; however, it lacks a significant aspect. The existing definition of pressure fails to incorporate the impact of neighboring intersections, resulting in a lack of coordination among them. Consequently, individual intersections operate independently, making decisions driven solely by self-interest, without regard for collaboration with neighboring counterparts. This absence of coordination fosters a self-centric approach, where each intersection prioritizes its own interests without accounting for the repercussions on neighboring intersections. In this section, our goal is to develop an innovative control policy that encourages consensus and cooperative behavior among neighboring intersections, ultimately enhancing overall system performance. Moreover, we will present a rigorous mathematical proof to substantiate our assertions.

We initiate by presenting the theoretical foundation, beginning with the Max Pressure algorithm, followed by an exploration of an extended iteration known as Lyapunov Drift-Plus-Penalty (LDPP). This extended approach introduces a penalty mechanism aimed at incentivizing intersections to collaborate. Finally, we illustrate the practical application of this theory within the problem environment.

4.1 Theoretical Framework

In the realm of communication and queuing systems, a prevalent strategy for network control is the employment of the Lyapunov Drift (LD) methodology. Neely [26] has extensively contributed to this subject, with numerous articles dedicated to it. In the forthcoming section, we adopt and adapt his analysis, applying it to our specific problem context. Prior to demonstrating the problem's stability, we introduce the relevant notation, followed by an explanation of stability definitions.

4.1.1 Queue Model

Let $Q_{l,m}(t)$ be a non-negative real-valued random variable representing the queue of movement (l, m) as previously defined. We denote $\mathbf{Q}(t) \triangleq [Q_{l',m'}(t), \ldots, Q_{l,m}(t)]$ as the random vector encompassing the individual queues. Further, we introduce some shorthand notation, for our previously introduced queue model (2.1).

$$Q_{l,m}(t+1) = Q_{l,m}(t) - y_{l,m}(t)s_{l,m}(t) + \sum_{k \in \mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)r_{l,m}(t) + d_{l,m}(t) \tag{4.1}$$

$$= Q_{l,m}(t) - b_{l,m}(t) + a_{l,m}(t) \tag{4.2}$$

Here, the stochastic processes $a_{l,m}(t)_{t=0}^{\infty}$ and $bl, m(t)_{t=0}^{\infty}$ consist of sequences of real-valued random variables defined across time slots $t \in 0, 1, 2, ...$, encapsulating the complete inflow and outflow characteristics. Additionally, for all lanes that do not serve as entry lanes, we establish $d_{l,m}(t) = 0$, denoted by $l \notin \mathcal{L}_{entry}$. Furthermore, our assumption is that each demand belongs to the feasible set D as defined in Section 2.2. This is crucial because, as demonstrated by Varaiya [40], if this condition is not met, there is no controller capable of achieving stability for such a demand.

4.1.2 Stability Properties

The fundamental stability criterion we aim to achieve revolves around demonstrating that the cumulative queue size does not experience unbounded growth but remains within well-defined limits. This rationale is the basis for introducing the following stability definitions:

Definition 4.1.1 (Mean rate stable (m.r.s))**.** *A discrete time process $Q(t)$ is mean rate stable if:*

$$\lim_{t \to \infty} \frac{\mathbb{E}\{|Q(t)|\}}{t} = 0 \tag{4.3}$$

Definition 4.1.2 (strongly stable)**.** *A discrete time process $Q(t)$ is strongly stable if:*

$$\limsup_{t \to \infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{|Q(\tau)|\} < \infty \tag{4.4}$$

We will demonstrate that our algorithm is mean rate stable (m.r.s), and under certain conditions, it also attains strong stability.

4.1.3 Lyapunov Drift Optimization

In order to express the extent of congestion in the network as a scalar value, we establish a quadratic Lyapunov function $L(\mathbf{Q}(t))$ for this purpose:

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} Q_{l,m}(t)^2 \tag{4.5}$$

This function has three main properties:

- $L(\mathbf{Q}(t)) \geq 0$ for all networks and time $t \in \{0, 1, 2, \ldots\}$ and is only zero if all queues are empty

- If $L(\mathbf{Q}(t))$ is small, then all queues are small
- If $L(\mathbf{Q}(t))$ is large, then at least one queue is large too

Further, we define the so-called one-slot conditional Lyapunov drift as

$$\Delta(\mathbf{Q}(t)) \triangleq \mathbb{E}\Big\{ L\Big(\mathbf{Q}(t+1)\Big) - L\Big(\mathbf{Q}(t)\Big) \;\Big|\; \mathbf{Q}(t) \Big\} \tag{4.6}$$

With the requisite definitions now established, we are equipped to present our initial theorem.

Theorem 4.1.1 (Lyapunov Drift (See Theorem 4.1 in [26]))**.** *Consider the quadratic Lyapunov function* (4.5)*, and assume* $\mathbb{E}\{L(\mathbf{Q}(0))\} < \infty$*. Suppose there are constants* $B > 0$*,* $\epsilon \geq 0$ *such that the following drift conditions holds for all slots* $\tau \in \{0, 1, 2, \dots\}$ *and all possible* $\mathbf{Q}(\tau)$*:*

$$\Delta(\mathbf{Q}(\tau)) \leq B - \epsilon \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} |Q_{l,m}(\tau)| \tag{4.7}$$

Then:

(a) If $\epsilon \geq 0$*, then all queues* $Q_{l,m}(t)$ *are* **mean rate stable**

(b) if $\epsilon > 0$*, then all queues are* **strongly stable** *and :*

$$\limsup_{t \to \infty} \sum_{\tau=0}^{t-1} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\{|Q_{l,m}(\tau)|\} \leq \frac{B}{\epsilon} \tag{4.8}$$

The proof follows the description of Neely [26].

Proof. See Appendix A □

This proof holds significance within the broader context of Lyapunov optimization. With its establishment, we can validate a segment of the original proof proposed by Varaiya [40] for the MP algorithm. In fact, minimizing $\Delta(\mathbf{Q}(t))$ in every time step, results in the MP algorithm. However, let's consider a scenario where, in addition to the queues $\mathbf{Q}$ we aim to stabilize, there exists a corresponding stochastic penalty process denoted as $p(t)$. Our objective is to minimize the time average of this penalty process, or at the very least, reduce it below a target threshold denoted as p^*. To achieve this, we extend the current drift formulation through the introduction of a penalty function.

4.1.4 Lyapunov Drift Plus Penalty Optimization

General Result

In communications and queuing theory, the penalty often is a measure of used power, for example to send network packages over links. Therefore, it is a natural goal to minimize the power while still ensuring the network does not get congested by too many packets. Since in our setting, we do not have a notion of power, we will abuse the theory to our liking to give an incentive to intersections to work together. The exact penalty function we are going to use will be introduced at a later time.

We start with a general result of the Lyapunov Drift-Plus-Penalty theory and continue later with the actual proof for our algorithm. We assume that the expected penalty function has a finite lower bound p_{min} so that for all t and all possible control actions, we have:

$$\mathbb{E}\{p(t)\} \geq p_{min} \tag{4.9}$$

Theorem 4.1.2 (Lyapunov plus penalty optimization (See Theorem 4.2 in [26]))**.** *Suppose $L(\mathbf{Q}(t))$ and p_{min} are defined by (4.5) and (4.9) and that $\mathbb{E}\{L(\mathbf{Q}(0))\} < \infty$. Suppose there are constants $B \geq 0$, $V \geq 0$, $\epsilon \geq 0$, and p^* such that for all slots $\tau \in \{0, 1, 2, \dots\}$ and all possible values of $\mathbf{Q}(\tau)$, we have*

$$\Delta(\mathbf{Q}(\tau)) + V\mathbb{E}\{p(\tau)|\mathbf{Q}(\tau)\} \leq B + Vp^* - \epsilon \sum_{\substack{l\in\mathcal{L} \\ m\in\mathcal{D}_l}} |Q_{l,m}(\tau)| \tag{4.10}$$

Then all queues $\mathbf{Q}(t)$ are mean rate stable. Further, if $V > 0$ and $\epsilon > 0$ then time average expected penalty and queue backlog satisfy:

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{p(\tau)\} \leq p^* + \frac{B}{V} \tag{4.11}$$

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{\substack{l\in\mathcal{L} \\ m\in\mathcal{D}_l}} \mathbb{E}\{|Q_{l,m}(\tau)|\} \leq \frac{B + V(p^* - p_{min})}{\epsilon} \tag{4.12}$$

Finally, if $V = 0$, then (4.12) still holds, and if $\epsilon = 0$ then (4.11) still holds

Proof. See Appendix B □

We have demonstrated that achieving the inequality (4.10) guarantees a stable system. Now, our focus shifts towards elucidating the derivation of this bound. To achieve this, our first step involves proving that an alternative stochastic policy, known as the ω-only policy, achieves mean rate stability. Subsequently, we will establish that our policy consistently makes the most optimal instantaneous decisions within the framework of our specifically designed problem, and is therefore better than any ω-only policy.

ω-only Policy

We define the random event $\omega(t)$, which is a random event observed on slot t (such as new car arrivals or different traffic conditions) as follows:

Definition 4.1.3 (ω-only policies)**.** *A ω-only policy is characterized as a stationary and random policy. In this policy, at each time step t, it observes the random event $\omega(t)$ and subsequently selects a control action $s(t) \in \mathcal{A}_{\omega(t)}$. The policy is determined as a pure, and possibly randomized, function of the observed $\omega(t)$. We denote this action as s^* throughout the time horizon $t \in 0, 1, 2, \dots$. Here, $\mathcal{A}_{\omega(t)}$ represents the entire possible action space associated with the event $\omega(t)$.*

Assumption 1. *$\omega(t)$ is a stationary process with a stationary probability distribution $\pi(\omega)$.*

$$Pr[\omega(t) = \omega] = \pi(\omega) \qquad \forall t \in \{0, 1, 2, \dots\} \tag{4.13}$$

The inflow $a_{l,m}(t)$, as well as the outflow $b_{l,m}(t)$ defined in (4.1), are both formulated as general functions of the random event $\omega(t)$ and a control action $\phi(t)$. This can be expressed as $a_{l,m}(t) = \hat{a}_{l,m}(s(t), \omega(t))$ and $b_{l,m}(t) = \hat{b}_{l,m}(s(t), \omega(t))$. Similarly, we define $p(t)$ as a general function of the random event $\omega(t)$ and the control action $s(t)$, which can be represented as $p(t) = \hat{p}(s(t), \omega(t))$.

Assumption 2. *The arrival function $\hat{a}_k(s(t), \omega(t))$ and the service function $\hat{b}_k(s(t), \omega(t))$, in conjunction with the stationary probabilities $\pi(\omega)$, adhere to the following boundedness properties: For all instances t and all control decisions $s(t) \in \mathcal{A}_{\omega(t)}$, the following conditions hold:*

$$\mathbb{E}\{\hat{a}_{l,m}(s(t), \omega(t))^2\} \leq \sigma^2 \qquad \begin{matrix} \forall l \in \mathcal{L} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{4.14}$$

$$\mathbb{E}\{\hat{b}_{l,m}(s(t), \omega(t))^2\} \leq \sigma^2 \qquad \begin{matrix} \forall l \in \mathcal{L} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{4.15}$$

for some finite constant $\sigma^2 > 0$. Moreover, for all instances t and all actions $s(t) \in \mathcal{A}_{\omega(t)}$, we necessitate that the expectation of $p(t)$ remains bounded by finite constants p_{min} and p_{max}.

$$p_{min} = 0 \leq \mathbb{E}\{\hat{p}(s(t), \omega(t))\} \leq p_{max} \tag{4.16}$$

Taking an alternative perspective, we can now reevaluate the problem by aiming to minimize the average time-weighted expectation of the penalty, denoted as $\bar{p} \triangleq \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{p(\tau)\}$. In doing so, our focus shifts to minimizing the penalty while maintaining network stability, rather than directly pursuing the minimization of travel time or the maximization of pressure.

$$\min \quad \bar{p} \tag{4.17}$$

$$s.t. \quad Q_{l,m}(t) \text{ are mean rate stable } \forall l \in \mathcal{L},\ m \in \mathcal{D}_l \tag{4.18}$$

$$s(t) \in \mathcal{A}_{\omega(t)}\ \forall t \tag{4.19}$$

This formulation now encapsulates the current problem we aim to solve. It follows that if we can design a control algorithm that chooses $\alpha(t) \in \mathcal{A}_{\omega(t)}$ for all t, makes all actual queues $\mathbf{Q}(t)$ mean rate stable, and yields a time average expectation of p(t) that is equal to our target p^*, then we have solved the problem (4.17) - (4.19) and with it, and with it, we will have successfully tackled our original challenge formulation. The upcoming theorem delineates a crucial aspect of the proof and is, in fact, a crucial aspect of every feedback control law [40].

Theorem 4.1.3 (Optimality over ω-only Policies (See Theorem 4.5 in [26]))**.** *Suppose $\omega(t)$ is a stationary process with distribution $\pi(\omega)$, and the system satisfies the boundedness assumptions* (4.14) *and* (4.15)*. If the problem* (4.18) - (4.19) *is feasible, then for any $\delta > 0$ there is an ω-only policy $s^*(t)$ that satisfies $s^*(t) \in \mathcal{A}_{\omega(t)}$ for all t, and:*

$$\mathbb{E}\{\hat{a}_k(s^*(t), \omega(t))\} \leq \mathbb{E}\{\hat{b}_k(s^*(t), \omega(t))\} + \delta \quad \forall k \in \{1, \ldots, K\} \tag{4.20}$$

Proof. The proof is done in Neely [26]. □

Min Drift Plus Penalty Algorithm

We now start to apply the theory onto our problem, specifically our queue model. The following lemma yields directly our control strategy and explains how the controller comes into existence.

Lemma 4.1.4. *[Modified from Lemma 4.6 in [26]] Suppose $\omega(t)$ is i.i.d. over slots. Under* **any** *control algorithm, the drift-plus-penalty expression has the following upper bound for all t, all possible values of $\mathbf{Q}(t)$, and all parameters $V \geq 0$:*

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(t)|\mathbf{\Theta}(t)\} \leq B_{max} + V\mathbb{E}\{p(t)|\mathbf{\Theta}(t)\} + \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} Q_{l,m}(t)\mathbb{E}\{a_{l,m}(t) - b_{l,m}(t)|\mathbf{Q}(t)\} \tag{4.21}$$

where B_{max} represents a maximal possible value and will be explicitly derived in (4.37) *for any control policy.*

Proof. The proof is shown in the derivation of (4.39). □

Rather than directly minimizing the drift plus penalty expression in every time slot t, our strategy aims to minimize the bound provided on the right-hand side of equation (4.21). This is done via the framework of opportunistically minimizing a (conditional) expectation [26], which basically states that for minimizing the expectation of a function, we can instead minimize the function itself first. Hence, the min drift-plus-penalty algorithm can be summarized as

$$\min \quad \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} Q_{l,m}(t)\Big(a_{l,m}(t) - b_{l,m}(t)\Big) + Vp(t) \tag{4.22}$$

$$\omega(t) \in \mathcal{A}_{\omega(t)} \tag{4.23}$$

and this algorithm will solve the problem formulation (4.17) - (4.19). This makes the framework so powerful that it does not need to know the probability $\pi(\omega)$ to evaluate the policy, but just observes it and is able to make a decision and the above bound holds. The stability of the controller introduced in equation (4.22) can be established by leveraging Lemma 4.1.4 in conjunction with Theorem 4.1.3. However, the forthcoming two theorems, accompanied by their respective proofs, will provide a comprehensive demonstration of the stability. Firstly, the stability of the ω-only policy will be established, followed by the proof of the min drift-plus-penalty algorithm.

Theorem 4.1.5 (Performance of Min Drift-Plus-Penalty Algorithm (Modified from Theorem 4.8 in [26]))**.** *Suppose $\omega(t)$ is i.i.d over slots with probabilities $\pi(\omega)$, the problem* (4.17)-(4.19) *is feasible, and that $\mathbb{E}\{L(\mathbf{Q}(0))\} < \infty$. Then we have:*

(a) Time average expected cost satisfies:

$$\limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\{p(\tau)\} \propto \frac{B}{V} \tag{4.24}$$

where B is defined in (4.37).

(b) Queues $\mathbf{Q}(t)$ *are mean rate stable*

(c) Suppose we define a function $\Psi(\epsilon)$ *that is bounded by* $p_{min} \leq \Psi(\epsilon) \leq p_{max}$ *and that there are constants* $\epsilon > 0$ *and for which the Slater condition:*

$$\mathbb{E}\Big\{\hat{p}(\phi^*(t), \omega(t))\Big\} = \psi(\epsilon) \tag{4.25}$$

$$\mathbb{E}\Big\{\hat{a}_{l,m}(s^*(t), \omega(t))\Big\} \leq \mathbb{E}\Big\{\hat{b}_{l,m}(s^*(t), \omega(t))\Big\} - \epsilon \qquad \begin{matrix} \forall l \in \mathcal{L} \\ \forall m \in \mathcal{D}_l \end{matrix} \tag{4.26}$$

hold, then all queues $\mathbf{Q}(t)$ *are strongly stable, and we have:*

$$\limsup_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \sum_{\substack{l\in\mathcal{L} \\ m\in\mathcal{D}_l}} \mathbb{E}\{Q_{l,m}(\tau)\} \leq \frac{B + V\Psi(\epsilon)}{\epsilon} \tag{4.27}$$

where p_{max}, p_{min} *are defined in* (4.35).

Proof. See Appendix C □

The relations (4.24) and (4.27) nicely show the tradeoff between the average queue backlog and the minimal possible penalty bound. Choosing a larger V will force the penalty function to become smaller at the cost of possibly longer queues, while making V larger returns the opposite. Therefore, we get the tradeoff $[\mathcal{O}(1/V), \mathcal{O}(V)]$. In its usual setting, this tradeoff can be intuitively used to tradeoff used power, for example in a communication network, versus queue backlog. In our setting, however, it is not really straightforward what consequences this tradeoff will bring in simulation, as it is more a tradeoff between selfish behavior and collaborative work among neighbors. The following theorem now summarizes the shown result and gives the explanation on why our algorithm in (4.22) gives the optimal result.

Theorem 4.1.6 (Stability of Min Drift-Plus-Penalty Algorithm)**.** *Lemma 4.1.4 showed that the relation* (4.21) *holds for any controller, since we picked the largest possible* B_{max}*. Theorem 4.1.5 then showed that using this bound, there exists an* ω*-only policy that satisfies the stability criterion and its performance expectation. It follows that if on every slot* t*, there exists a particular control action that satisfies this drift requirement, then the drift-plus-penalty minimizing policy* (4.22) *must also satisfy this drift requirement, which thus proves stability of the min drift-plus-penalty algorithm.*

Proof. With Lemma 4.1.4 and Theorem 4.1.5 we can show that

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(s(t))|\mathbf{Q}\} \leq B_{max} + V\mathbb{E}\{p(s(t)_{min}, \omega(t))|\mathbf{Q}\} \tag{4.28}$$

$$+ \sum_{\substack{l\in\mathcal{L} \\ m\in\mathcal{D}_l}} q_{l,m}(t)\mathbb{E}\Big\{a_{l,m}(s(t)_{min}, \omega(t)) - b_{l,m}(s(t)^{min}, \omega(t))\Big\}$$

$$\leq B_{max} + V\mathbb{E}\{p(s^*(t), \omega(t))|\mathbf{Q}\} \tag{4.29}$$

$$+ \sum_{k=1}^{K} Q_{l,m}(t)\mathbb{E}\Big\{a_k(s^*(t), \omega(t)) - b_k(s^*(t), \omega(t))\Big\}$$

where $s(t)_{min}$ denotes the min drift-plus-penalty control input s^* the ω-only policy and B_{max} is chosen as the largest upper bound for any control algorithm. Thus, we showed that our proposed control policy yields mean rate stability or, if the Slater conditions (4.25) - (4.26) hold, even strong stability. □

4.1.5 Traffic Signal Control Problem

Penalty Definition

As already mentioned earlier, our goal is to incite neighboring intersections to work together. We try to do this by introducing a penalty function that forces intersections to consider the traffic situation of neighbors. For that, we will use once more our queue model (2.1). Namely, each intersection i has the incentive to keep the traffic amount of all lanes that are part of intersection i and its neighbor $\mathcal{N}_i$ below a threshold value q_{thresh}. This value could be set as the maximal capacity of each lane, or it could be set lower to give an incentive to each intersection to keep the lanes overall smaller and hence more balanced. This indicates that each intersection devises a signal plan not only for itself but also for its neighboring intersections.

One significant limitation of both the original MP formulation is the disregard for lanes that never receive a green light, since not many cars are waiting on these lanes. However, such behavior is deemed unacceptable in real-world scenarios, necessitating the establishment of a fairer distribution of green times. One possible solution involves implementing a fixed green cycle time, wherein all green lights are granted the right of way for a minimum specified duration as done in Sun et al. [34]. Another alternative would be to apply penalties to the algorithm whenever it does not allocate a green signal to phases that have remained inactive for a considerable duration. We combine now both of these ideas into one penalty function as

$$p_i(t) = \sum_{\substack{l \in \mathcal{L}_{i,in} \\ m \in \mathcal{D}_l}} \left(h^{(1)}_{l,m}(t) + \sum_{p \in \mathcal{D}_m} h^{(2)}_{l,m,p}(t) \right) + h^{(3)}_{l,m}(t) \tag{4.30}$$

where

$$h^{(1)}_{l,m}(t) = \begin{cases} 1, & q_{l,m}(t) - y_{l,m}(t)s_{l,m}(t) + \sum_{k \in \mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)r_{l,m} > q_{thresh} \\ 0, & \text{otherwise} \end{cases} \tag{4.31}$$

$$h^{(2)}_{l,m,p}(t) = \begin{cases} 1, & q_{m,p}(t) - y_{m,p}(t)s_{m,p}(t) + y_{l,m}(t)s_{l,m}(t) > q_{thresh} \\ 0, & \text{otherwise} \end{cases} \tag{4.32}$$

$$h^{(3)}_{l,m}(t) = \sum_{k \in \mathcal{S}_{l,m}} \phi_k(t) \Big(\sum_{\tau = t-L}^{t} \phi_k(\tau) \Big) \tag{4.33}$$

where L is a constant design parameter. $\mathcal{L}_{i,in}$ is the set of incoming lanes of an intersection i and $\mathcal{S}_{l,m}$ is the set of phases that movement (l,m) is part of, since it is possible that movement (l,m) receives green light in different phases. $h^{(1)}_{l,m}(t)$ penalizes the overflow of any incoming lane into intersection i. Hereby, it optimizes over the possible traffic light choices $s_{k,l}$ of the upstream intersection that would be optimal. On the other hand, $h^{(2)}_{l,m,p}(t)$ gives a penalty if any of the outgoing lanes of intersection i exceeds the maximal value q_{thresh}. The downstream lane. For the last term in $h^{(2)}_{l,m,p}(t)$, we assume the inflow of (m,p) is proportional to the outflow of (l,m). $h^{(3)}_{l,m}(t)$ counts how many times lane (l,m) has had green in the last $L \in \mathbb{Z}$ time steps. This gives

an incentive to give green to lanes that had not had the right of way for a longer time.

Similar as for the drift function (4.6), we define the penalty $p(t)$ as the sum over all lanes to be able to apply later Theorem 4.1.2.

$$p(t) = \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \left(h^{(1)}_{l,m}(t) + \sum_{p \in \mathcal{D}_m} h^{(2)}_{l,m,p}(t) + h^{(3)}_{l,m}(t) \right) \tag{4.34}$$

Theoretical Derivation

As assumed in Assumption 2, we require a lower and upper bound for our penalty function. Using the penalty (4.34), we get the following bounds:

$$p_{min} = 0 \leq \mathbb{E}\{\hat{p}(\phi(t), \omega(t))\} \leq k \cdot \mathcal{L} + h = p_{max} \tag{4.35}$$

where k is a constant that depends on the specific road network and h depends on the past chosen actions and corresponds to the penalty term $h^{(3)}_{l,m}(t)$ in (4.33).

Further, we will explicitly show how we can arrive at the bound of (4.21), which thus leads us to our specific min-drift-plus penalty policy. We will first derive an expression for the drift without the penalty to simplify notation, and we will add it back in a later step.

$$\begin{aligned}
\Delta(\mathbf{Q}(t)) &= \frac{1}{2} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ Q_{l,m}(t+1)^2 - Q_{l,m}(t)^2 \Big| \mathbf{Q} \Big\} \\
&= \frac{1}{2} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ Q_{l,m}(t)^2 - 2 \cdot Q_{l,m}(t)\Big(b_{l,m}(t) - a_{l,m}(t)\Big) + \Big(a_{l,m}(t) - b_{l,m}(t)\Big)^2 \Big| \mathbf{Q} \Big\} \\
&\quad - \frac{1}{2} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ Q_{l,m}(t)^2 \Big| \mathbf{Q} \Big\} \\
&= \frac{1}{2} \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ \Big(a_{l,m}(t) - b_{l,m}(t)\Big)^2 \Big| \mathbf{Q} \Big\} - \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ Q_{l,m}(t)\Big(b_{l,m}(t) - a_{l,m}(t)\Big) \Big| \mathbf{Q} \Big\} \\
&= \alpha + \beta
\end{aligned} \tag{4.36}$$

Using (4.1) α can be upper bounded as follows:

$$
\begin{aligned}
\alpha &= \frac{1}{2}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{\Big(a_{l,m}(t)-b_{l,m}(t)\Big)^2\Big|\mathbf{Q}\Big\} \qquad (4.37)\\
&= \frac{1}{2}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{\Big(R_{l,m}\sum_{k\in\mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)+d_{l,m}(t)-y_{l,m}(t)s_{l,m}(t)\Big)^2\Big|\mathbf{Q}\Big\}\\
&\le \frac{1}{2}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{\Big(y_{l,m}(t)s_{l,m}(t)\Big)^2\Big|\mathbf{Q}\Big\}+\frac{1}{2}\mathbb{E}\Big\{\Big(r_{l,m}\sum_{k\in\mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)+d_{l,m}(t)\Big)^2\Big|\mathbf{Q}\Big\}\\
&\le \frac{1}{2}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} c_{l,m}^2+\frac{1}{2}r_{l,m}\Big(\sum_{k\in\mathcal{U}_l} c_{k,l}+d_{max}\Big)^2 = B_{max}
\end{aligned}
$$

d_{max} is the max exogenous inflow rate on each lane. This proves Lemma 4.1.4 and shows the bound for B_{max}. The important thing here is that the bound on α holds for any control strategy, since it represents the worst-case scenario. Having defined the bound, we proceed to reconfigure β in order to derive the precise controller that we will employ. This derivation also sheds light on the development of the Max Pressure (MP) algorithm.

Further, we can rewrite β as:

$$
\begin{aligned}
\beta &= -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)\Big(b_{l,m}(t)-a_{l,m}(t)\Big)\Big|\mathbf{Q}\Big\}\\
&= -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)\Big(s_{l,m}(t)y_{l,m}(t)-r_{l,m}\sum_{k\in\mathcal{U}_l} s_{k,l}(t)y_{k,l}(t)-d_{l,m}\Big)\Big|\mathbf{Q}\Big\}\\
&= -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)s_{l,m}(t)y_{l,m}(t)\Big|\mathbf{Q}\Big\}+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)r_{l,m}\sum_{k\in\mathcal{U}_l} y_{k,l}(t)s_{k,l}(t)\Big|\mathbf{Q}\Big\}\\
&\quad+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)d_{l,m}(t)\Big|\mathbf{Q}\Big\}\\
&\overset{(a)}{=} -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)s_{l,m}(t)y_{l,m}(t)\Big|\mathbf{Q}\Big\}+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{y_{l,m}(t)s_{l,m}(t)\sum_{p\in\mathcal{D}_m} r_{m,p}(t)y_{m,p}(t)\Big|\mathbf{Q}\Big\}\\
&\quad+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)d_{l,m}(t)\Big|\mathbf{Q}\Big\}\\
&= -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{s_{l,m}(t)y_{l,m}(t)\Big(Q_{l,m}(t)-\sum_{p\in\mathcal{D}_m} r_{m,p}Q_{m,p}(t)\Big)\Big|\mathbf{Q}\Big\}+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)d_{l,m}(t)\Big|\mathbf{Q}\Big\}\\
&= -\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{s_{l,m}(t)y_{l,m}(t)w_{l,m}(t)\Big|\mathbf{Q}\Big\}+\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\Big\{Q_{l,m}(t)d_{l,m}(t)\Big|\mathbf{Q}\Big\}
\end{aligned}
$$

where we defined

$$w_{l,m}(t) = Q_{l,m}(t) - \sum_{p \in \mathcal{D}_m} r_{m,p}(t) Q_{m,p}(t) \tag{4.38}$$

which is the same expression as the weight from (3.2) as it was done by Varaiya [40]. In the second term in (a) we switched from summing over all upstream lanes to summing over all downstream lanes.

We can now use these results together with the penalty defined before and plug it into the drift plus penalty expression:

$$\begin{aligned} \Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(t)|\mathbf{Q}\} \leq B_{max} - \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ s_{l,m}(t) y_{l,m}(t) w_{l,m}(t) \Big| \mathbf{Q} \Big\} \\ + \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ Q_{l,m}(t) d_{l,m}(t) \Big| \mathbf{Q} \Big\} + V\mathbb{E}\{p(t)|\mathbf{Q}\} \end{aligned} \tag{4.39}$$

As previously elucidated, rather than directly minimizing the drift plus penalty expression in each time slot t, our approach is geared towards minimizing the bound presented on the right-hand side of equation (4.39). Consequently, we try to optimize for:

$$\min \quad \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} \mathbb{E}\Big\{ - s_{l,m}(t) y_{l,m}(t) w_{l,m}(t) + V p(t) \Big| \mathbf{Q} \Big\} \tag{4.40}$$

where we have omitted terms that remain constant or are unrelated to the control action $s_{l,m}(t)$. In the proof of Theorem 2 from Varaiya [40], it was demonstrated how the second term within the expectation can be additionally manipulated to align with the initial formulation of the maximum pressure concept [40]. Consequently, the algorithm for minimizing drift-plus-penalty can be concisely formulated as follows, using once more the framework of opportunistically minimizing an expectation:

$$\min \quad \sum_{\substack{l \in \mathcal{L} \\ m \in \mathcal{D}_l}} -s_{l,m}(t) c_{l,m}(t) w_{l,m}(t) + V p(t) \tag{4.41}$$

This resulting expression now corresponds to the one in (4.22), tailored to address our specific problem. If we neglect the penalty term, we explicitly get back the MP policy as shown in (2.8). By combining the proof outlined in (4.37) for Lemma 4.1.4 with the aforementioned steps, we can progress towards demonstrating the stability of our custom controller. This progression leads us to the conclusions drawn in both Theorem 4.1.5 and 4.1.6, collectively affirming the stability of our designed controller.

Implementation

Now that we have proved the stability of our method, we go ahead and show the actual implementation per intersection. Within an intersection, it is feasible to allocate the right of way to multiple movements simultaneously, referred to as the phase, as explained in the Chapter 2.

Similarly, we will modify the previous definition of pressure by incorporating the penalty function. For each movement (l, m), we define the weight as in (2.5) and define the modified pressure $\tilde{P}_{i,k}(t)$ of each phase according to:

$$\tilde{P}_{i,\phi_k}(t, \phi_k) = \sum_{(l,m)\in\phi_k} -w_{l,m}(t)c_{l,m}(t)s_{l,m}(t) + Vp_i(t) \tag{4.42}$$

$$= -P_{i,\phi_k}(t) + V \sum_{\substack{l\in\mathcal{L}_{i,in} \\ m\in\mathcal{D}_l}} \left(h^{(1)}_{l,m}(t) + \sum_{p\in\mathcal{D}_m} h^{(2)}_{l,m,p}(t) + h^{(3)}_{l,m}(t)\right) \tag{4.43}$$

where $h^{(1)}_{l,m}(t)$, $h^{(2)}_{l,m}(t)$ and $h^{(3)}_{l,m}(t)$ are defined as in (4.31) - (4.33). We now try to make the pressure as small as possible, contrary to the MP formulation. This is however not a contradiction, as we add the weight negatively, but then on the other hand add the penalty positively, since we want to make the penalty as small as possible.

If we were to implement the penalty solely in this manner for each intersection, each intersection would simply attempt to select the traffic light actions of its neighbors to minimize its own penalty, disregarding the neighbor's traffic pressure. This would lead to a situation where the optimal set of chosen phases, encompassing the intersection's own phases $\phi_{i,k}$ and the neighboring phases $\phi_{i',k}$, $\forall i' \in \mathcal{N}_i$, would exhibit significant disparities. Such discrepancies would complicate the task of reaching a consensus among neighbors. However, since our primary focus remains on an algorithm primarily rooted in the Max Pressure (MP) approach, we augment the objective function of each intersection with the neighbors' traffic pressure. Specifically, we can formulate the local objective per intersection as follows:

$$\hat{P}_i(t, \Phi_i) = -\sum_{i'\in\{\mathcal{N}_i\cup\{i\}\}} \sum_{\phi_k\in\Phi_i} P_{i',\phi_k}(t) + Vp_i(t) \tag{4.44}$$

and we will refer to this penalty as the binary penalty function. The penalty weight, denoted as V, doesn't need to be uniform across all penalty terms. In our simulation, we employ distinct weights for each penalty term, denoted as V_1, V_2, and V_3.

Continuous Penalty Function

Due to the fact that we defined our penalty function in (4.34) binary valued, we were able to upper bound it by a simple constant as it was done in (4.35). However, this penalty function punishes any unwanted behavior uniformly, meaning that overshooting a lane by 20 cars will be punished the same as a minor violation of only 1 car. In this section, we introduce another penalty function that directly reflects the degree of violation. More explicitly, we encourage the intersections to work together not by punishing, but by defining a reward, whenever an intersection adapts its decision for the benefit of a neighbor. Namely, assume intersection i gives green light to the movement (l, m). Then, in the event that the downstream lane m is granted a green light by the neighboring intersection i', a reward will be assigned. The magnitude of this reward is contingent upon the utility associated with that particular decision. More specifically, we define the penalty, or more specifically utility, function as:

$$p_i(t) = \sum_{\substack{l \in \mathcal{L}_{i,in} \\ m \in \mathcal{D}_l}} \sum_{p \in \mathcal{D}_m} a_{l,m,p}(t) \tag{4.45}$$

$$a_{l,m,p}(t) = \begin{cases} -\gamma q_{m,p}(t), & s_{l,m}(t) = 1 \wedge s_{m,p}(t) = 1 \\ 0, & \text{else} \end{cases} \tag{4.46}$$

γ is hereby a tuning parameter that will influence how big the reward is and thus how selfish intersections will make their decisions. The presence of the negative sign arises because our intention is to create motivation for maximizing this utility function. This approach stems from the fact that we've framed the entire issue as a minimization problem, as illustrated in eq. (4.41). We will call this penalty function the continuous penalty for the remainder of the thesis.

A drawback of this penalty is that we do not have theoretical guarantees that this formulation will yield stability, as we do not have a guaranteed upper bound for it. Nevertheless, we will confirm its performance in Chapter 5.

4.2 Consensus

Up until this point, we have formulated the theory for computing pressure and penalty values for each intersection, enabling them to autonomously ascertain their optimal strategies for both themselves and their neighboring counterparts. However, a challenge remains in achieving consensus for these decisions among neighboring intersections. In the forthcoming section, we will explore two algorithms tailored to tackle this consensus problem among adjacent intersections. These algorithms are geared towards cultivating a cooperative approach, where intersections cease to operate in isolation and instead collaborate towards attaining a global optimum. This global optimum, denoted as O, is defined as the cumulative sum of all pressures and penalties:

$$O = \sum_{i \in \mathcal{G}} \hat{P}_i(t) \tag{4.47}$$

with the penalty function being either the binary or the continuous penalty function.

4.2.1 ADMM Consensus

Introduced in 1974 [12], the Alternating Direction Method of Multipliers (ADMM) has found applications across diverse fields, spanning from machine learning and image processing to control systems, optimization, finance, and economics. This widely utilized convex optimization algorithm offers a compelling feature: its optimization process can be distributed. Moreover, it has proven highly practical as it, under certain assumptions, converges to the true optimal solution and often requires only a few tens of iterations for numerous problems.

Another advantageous trait of ADMM is its adaptability to address consensus problems [6]. By reformulating, it can effectively tackle consensus challenges. The standard formulation for N nodes attempting to achieve consensus takes the following form:

$$\min_{x_1,\ldots,x_N,z} \sum_{i=1}^{N} f_i(x_i) + g(z) \tag{4.48}$$

$$\text{s.t. } x_i - z = 0,\ i = 1,\ldots,N \tag{4.49}$$

where $f_i : \mathbb{R}^z \to (\mathbb{R} \cup \infty)$, $g : \mathbb{R}^z \to (\mathbb{R} \cup \infty)$ represent two continuous, convex functions. Here, x_i denotes the local variables, and z represents the global, shared variable that all nodes collaborate to achieve consensus on. However, in our case, we deal with discrete functions involving binary decision variables. Consequently, the theoretical convergence guarantees are not directly applicable. Nonetheless, heuristically, we still anticipate the algorithm to perform effectively.

In our context, the local objective $f_i(x_i)$ corresponds to the local objective as defined in equation (4.44). The matrix $\boldsymbol{x_i} \in [0,1]^{\tilde{\mathcal{N}_i} \times \Phi_i}$ encapsulates the local binary decision matrix containing local phase decisions. Here, $\tilde{\mathcal{N}_i} = \mathcal{N}_i \cup i$. In contrast, the global shared matrix is denoted as $\boldsymbol{z} \in [0,1]^{\mathcal{G} \times \Phi_i}$.

An essential aspect of our formulation is that each intersection does not seek consensus for the entire $\boldsymbol{z}$ matrix, but solely for those elements that hold relevance - namely, the elements determining the phase for itself and its neighbors. As such, $\boldsymbol{x_i}$ constitutes a subvector of $\boldsymbol{z}$. To alleviate the complexity of notation, we will stack the entries of $\boldsymbol{x_i}$ and $\boldsymbol{z}$ to create vectors $\boldsymbol{x_i} \in [0,1]^{n_i}$ and $\boldsymbol{z} \in [0,1]^{\tilde{\mathcal{N}_i} \cdot \Phi_i \times 1}$, respectively. In this context, we define $n_i = |\tilde{\mathcal{N}_i}| \cdot |\Phi_i|$, representing the number of entries in the vector $\boldsymbol{x_i}$. This notation is consistent with that introduced by Boyd et al. [6].

For clarity, we denote the g-th entry of the global variable as $\boldsymbol{z}_g$. Additionally, let $\mathcal{M}(i,j) = g$ denote the mapping between the global variable $\boldsymbol{z}_g$ and the local $(\boldsymbol{x_i})_j$, where $(\boldsymbol{x_i})_j$ signifies the j-th entry of $\boldsymbol{x_i}$. To provide further clarity, refer to Figure 4.1 for a visual representation of the relationships between the various variables. Consequently, we formulate the consensus constraint as follows:

$$(\boldsymbol{x_i})_j = \boldsymbol{z}_{\mathcal{M}(i,j)}, \quad i \in \mathcal{G}, \quad j = 1,\ldots,n_i \tag{4.50}$$

We will simplify the notation further by introducing a duplicate of the global variable, denoted as $\tilde{\boldsymbol{z}}_i \in [0,1]^{n_i}$, where $(\tilde{\boldsymbol{z}}_i)j = \boldsymbol{z}_{\mathcal{M}(i,j)}$. This equivalency makes $\tilde{\boldsymbol{z}}_i$ the global counterpart of the locally defined variable $\boldsymbol{x_i}$. By combining this concept with our local objective function from equation (4.44), we can encompass the entire problem in the following formulation:

$$\min_{\boldsymbol{x}_1,\ldots,\boldsymbol{x}_{|\mathcal{G}|},\boldsymbol{z}} \quad \sum_{i=1}^{N} \hat{P}_i(t, \boldsymbol{x_i}) \tag{4.51}$$

$$\text{s.t.} \quad \boldsymbol{x_i} - \tilde{\boldsymbol{z}}_i = 0, \quad i = 1,\ldots,|\mathcal{G}| \tag{4.52}$$

Employing the conventional approach, we introduce the dual variable $\boldsymbol{\lambda} \in [0,1]^{n_i}$ and proceed to formulate the augmented Lagrangian as follows:

$$L(\boldsymbol{x}, \boldsymbol{z}, \boldsymbol{\lambda}) = \sum_{i=1}^{N} \left(f_i(\boldsymbol{x_i}) + \boldsymbol{\lambda}_i^T (\boldsymbol{x_i} - \tilde{\boldsymbol{z}}_i) + \frac{\rho}{2} ||\boldsymbol{x_i} - \tilde{\boldsymbol{z}}_i||_2^2 \right) \tag{4.53}$$

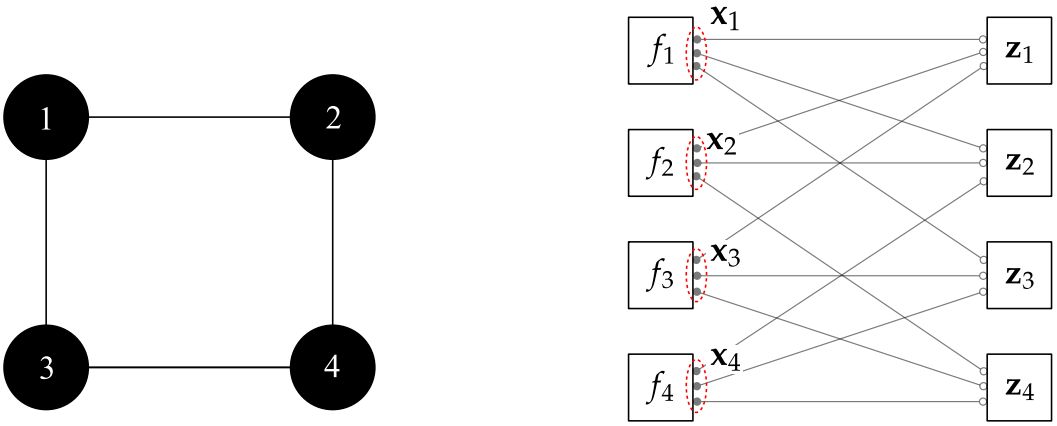


Fig. 4.1: A 2x2 intersection network illustrating the relationships between various variables. The red dashed ellipse signifies the vector encompassing all local $\boldsymbol{x}$ vectors.

The Alternating Direction Method of Multipliers (ADMM) algorithm comprises three primary iteration steps, which are iteratively executed multiple times until convergence is achieved or a stopping criterion is reached.

$$\boldsymbol{x}_i^{k+1} = \arg\min_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \left[\boldsymbol{\lambda}_i^k\right]^T \boldsymbol{x}_i + \frac{\rho}{2} ||\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i^k||_2^2 \right) \tag{4.54}$$

$$\boldsymbol{z}^{k+1} = \arg\min_{\boldsymbol{z}} \sum_{i=1}^{N} \left(- \left[\boldsymbol{\lambda}_i^k\right]^T \tilde{\boldsymbol{z}}_i + \frac{\rho}{2} ||\boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i||_2^2 \right) \tag{4.55}$$

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \rho(\boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i^{k+1}) \tag{4.56}$$

The updates for $\boldsymbol{x}$ and $\boldsymbol{\lambda}$ are decoupled and can be managed by each intersection independently, relying solely on communication with their respective neighbors. However, the update for $\boldsymbol{z}$ necessitates information from all intersections, indicating the requirement for a central computational unit to handle this aspect. In the subsequent discussion, we will illustrate how we can disentangle (4.55) into local updates that exclusively demand local communication. For the sake of the forthcoming derivation, we will assume that $\boldsymbol{z}$ can take on continuous values rather than solely binary ones. Subsequently, we will extend the outcome to encompass the binary scenario. This transition will enable us to determine the minimizer of $\boldsymbol{z}_g$ by calculating the partial derivative of (4.55) and setting it equal to zero.

$$\frac{\partial}{\partial \boldsymbol{z}_g}\boldsymbol{z}^{k+1} = \frac{\partial}{\partial \boldsymbol{z}_g}\sum_{i=1}^{N}\Big(-\Big[\boldsymbol{\lambda}_i^k\Big]^T\tilde{\boldsymbol{z}}_i + \frac{\rho}{2}||\boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i||_2^2\Big) \tag{4.57}$$

$$= \frac{\partial}{\partial \boldsymbol{z}_g}\sum_{i=1}^{N} -\begin{bmatrix}(\boldsymbol{\lambda}_i^k)_1,\dots,(\boldsymbol{\lambda}_i^k)_{n_i}\end{bmatrix}\begin{bmatrix}\boldsymbol{z}_{\mathcal{M}(i,1)}\\ \vdots \\ \boldsymbol{z}_{\mathcal{M}(i,n_i)}\end{bmatrix} + \frac{\rho}{2}\left|\left|\begin{bmatrix}(\boldsymbol{x}_i^{k+1})_1\\ \vdots \\ (\boldsymbol{x}_i^{k+1})_{n_i}\end{bmatrix} - \begin{bmatrix}\boldsymbol{z}_{\mathcal{M}(i,1)}\\ \vdots \\ \boldsymbol{z}_{\mathcal{M}(i,n_i)}\end{bmatrix}\right|\right|_2^2$$

$$= \sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{z}_g} - \Big((\boldsymbol{\lambda}_i^k)_1\cdot z_{\mathcal{M}(i,1)} + \dots + (\boldsymbol{\lambda}_i^k)_{n_i}\cdot \boldsymbol{z}_{\mathcal{M}(i,n_i)}\Big)$$

$$+ \sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{z}_g}\frac{\rho}{2}\Big(||\boldsymbol{x}_i^{k+1}||_2^2 - 2\Big((\boldsymbol{x}_i^{k+1})_1\cdot \boldsymbol{z}_{\mathcal{M}(i,1)} + \dots + (\boldsymbol{x}_i^{k+1})_{n_i}\cdot \boldsymbol{z}_{\mathcal{M}(i,n_i)}\Big)\Big)$$

$$+ \sum_{i=1}^{N}\frac{\partial}{\partial \boldsymbol{z}_g}\frac{\rho}{2}\Big(\boldsymbol{z}_{\mathcal{M}(i,1)}^2 + \dots + z_{\mathcal{M}(i,n_i)}^2\Big)$$

$$= \sum_{\mathcal{M}(i,j)=g} -(\boldsymbol{\lambda}_i^k)_j + \frac{\rho}{2}\Big(-2\cdot(\boldsymbol{x}_i^{k+1})_j + 2\cdot \boldsymbol{z}_{\mathcal{M}(i,j)}\Big) \stackrel{!}{=} 0$$

We can solve for $\boldsymbol{z}_{\mathcal{M}(i,j)} = \boldsymbol{z}_g$ as follows:

$$\sum_{\mathcal{M}(i,j)=g}(\boldsymbol{\lambda}_i^k)_j + \rho\cdot(\boldsymbol{x}_i^{k+1})_j = \sum_{\mathcal{M}(i,j)=g}\rho\cdot\boldsymbol{z}_{\mathcal{M}(i,j)} \tag{4.58}$$

$$= \boldsymbol{z}_g\cdot\rho\sum_{\mathcal{M}(i,j)=g}1 \tag{4.59}$$

and thus we get the update equation:

$$\boldsymbol{z}_g^{k+1} = \frac{\sum_{\mathcal{M}(i,j)=g}\Big((\boldsymbol{x}_i^{k+1})_j + \frac{1}{\rho}(\boldsymbol{\lambda}_i^k)_j\Big)}{\sum_{\mathcal{M}(i,j)=g}1} \tag{4.60}$$

$$= \bar{\boldsymbol{x}}_g^{k+1} + \frac{1}{\rho}\bar{\boldsymbol{\lambda}}_g^k \tag{4.61}$$

Here, $\bar{\boldsymbol{x}}_g$ and $\bar{\boldsymbol{\lambda}}_g$ represent the averages across all (i,j) where $\mathcal{M}(i,j) = g$. This equation can be further simplified. By averaging the $\boldsymbol{\lambda}$-update and substituting in (4.60), we arrive at:

$$\bar{\boldsymbol{\lambda}}_g^{k+1} = \bar{\boldsymbol{\lambda}}_g^k + \rho\Big(\bar{\boldsymbol{x}}_g^{k+1} - \boldsymbol{z}_g^{k+1}\Big) \tag{4.62}$$

$$= \bar{\boldsymbol{\lambda}}_g^k + \rho\Big(\bar{\boldsymbol{x}}_g^{k+1} - \bar{\boldsymbol{x}}_g^{k+1} - \frac{1}{\rho}\bar{\boldsymbol{\lambda}}_g^k\Big)$$

$$= 0 \tag{4.63}$$

This means that after the first iteration, the dual variable has an average value of zero, which further simplifies (4.60) to:

$$\boldsymbol{z}_g^{k+1} = \bar{\boldsymbol{x}}_g^{k+1} = \frac{\sum_{\mathcal{M}(i,j)=g} (\boldsymbol{x}_i^{k+1})_j}{\sum_{\mathcal{M}(i,j)=g} 1} \tag{4.64}$$

This adjustment effectively converts the $\boldsymbol{z}$ update into a localized averaging procedure rather than a global one. In essence, only the processing elements that hold a stake in a given feature $\boldsymbol{z}_g$ contribute to the vote for $\boldsymbol{z}_g$.

Returning to the scenario where $\boldsymbol{z}$ is binary-valued, the approach of taking derivatives and setting them to zero is no longer applicable. Nevertheless, we can still decentralize the optimization steps for $\boldsymbol{z}$ locally.

$$\begin{aligned}
\boldsymbol{z}^{k+1} &= \underset{\boldsymbol{z}\in[0,1]}{\arg\min} \sum_{i\in\mathcal{G}} -\left[\boldsymbol{\lambda}_i^k\right]^T \tilde{\boldsymbol{z}}_i + \frac{\rho}{2} \sum_i \left|\left| \boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i \right|\right|_2^2 \\
&= \underset{\boldsymbol{z}\in[0,1]}{\arg\min} \sum_{i\in\mathcal{G}} -\begin{bmatrix} (\boldsymbol{\lambda}_i^k)_1, \dots, (\boldsymbol{\lambda}_i^k)_{n_i} \end{bmatrix} \begin{bmatrix} \boldsymbol{z}_{\mathcal{M}(i,1)} \\ \vdots \\ \boldsymbol{z}_{\mathcal{M}(i,n_i)} \end{bmatrix} + \frac{\rho}{2} \left|\left| \begin{bmatrix} (\boldsymbol{x}_i^{k+1})_1 \\ \vdots \\ (\boldsymbol{x}_i^{k+1})_{n_i} \end{bmatrix} - \begin{bmatrix} \boldsymbol{z}_{\mathcal{M}(i,1)} \\ \vdots \\ \boldsymbol{z}_{\mathcal{M}(i,n_i)} \end{bmatrix} \right|\right|_2^2 \\
&= \underset{\boldsymbol{z}\in[0,1]}{\arg\min} \sum_{i\in\mathcal{G}} -\Big((\boldsymbol{\lambda}^k)_1 \cdot \boldsymbol{z}_{\mathcal{M}(i,1)} + \cdots + (\boldsymbol{\lambda}^k)_{n_i} \cdot \boldsymbol{z}_{\mathcal{M}(i,n_i)} \Big) \\
&\quad + \sum_{i\in\mathcal{G}} \frac{\rho}{2} \Big(||\boldsymbol{x}_i^{k+1}||_2^2 - 2\Big((\boldsymbol{x}_i^{k+1})_1 \cdot \boldsymbol{z}_{\mathcal{M}(i,1)} + \cdots + (\boldsymbol{x}_i^{k+1})_{n_i} \cdot \boldsymbol{z}_{\mathcal{M}(i,n_i)} \Big) \Big) \\
&\quad + \sum_{i\in\mathcal{G}} \frac{\rho}{2} \Big(\boldsymbol{z}_{\mathcal{M}(i,1)}^2 + \cdots + z_{\mathcal{M}(i,n_i)}^2 \Big) \\
&= \underset{\boldsymbol{z}\in[0,1]}{\arg\min} \sum_{i\in\mathcal{G}} \sum_{\mathcal{M}(i,j)=g} -(\boldsymbol{\lambda}^k)_j \cdot \boldsymbol{z}_{\mathcal{M}(i,j)} + \frac{\rho}{2} \Big(||\boldsymbol{x}_i^{k+1}||_2^2 - 2(\boldsymbol{x}_i^{k+1})_j \boldsymbol{z}_{\mathcal{M}(i,j)} + z_{\mathcal{M}(i,j)}^2 \Big)
\end{aligned} \tag{4.65}$$

In the final equation, we have transformed the summation procedure from encompassing all agents and their decision vectors to focusing on the summation of all components of the global shared variable, considering only the agents that contribute to the determination of each component. Additionally, it's evident that the inner summation is confined to the local neighbors of intersection i, as other intersections do not share the same vector component and consequently don't influence the same elements of $\boldsymbol{z}$. As a result, mirroring the continuous case, we can effectively distribute the update locally among the agents. In this decentralized scheme, each agent i solves the local optimization problem independently, leveraging communication from its neighbors solely for the component $\boldsymbol{z}_g$:

$$\boldsymbol{z}_g^{k+1} = \underset{\boldsymbol{z}_g\in[0,1]}{\arg\min} \sum_{\mathcal{M}(i,j)=g} -(\boldsymbol{\lambda}_i^k)_j \cdot \boldsymbol{z}_{\mathcal{M}(i,j)} + \frac{\rho}{2} \Big(||\boldsymbol{x}_i^{k+1}||_2^2 - 2(\boldsymbol{x}_i^{k+1})_j \boldsymbol{z}_{\mathcal{M}(i,j)} + \boldsymbol{z}_{\mathcal{M}(i,j)}^2 \Big) \tag{4.66}$$

Finally, while the use of binary global variables might seem more intuitive for our goal of optimizing binary values, it comes with the requirement to solve an optimization problem (4.66) in each iteration for updating $\boldsymbol{z}$. To enhance computational efficiency, an alternative approach is to transition to the continuous case, where the update for $\boldsymbol{z}$ simplifies to a straightforward calculation (4.64) without the need for optimization. However, in Chapter 5, our main focus will be on utilizing the binary variable variant to remain consistent with the original formalism. Nevertheless, we wanted to emphasize the potential for enhanced speed through the continuous

Algorithm 2 Consensus ADMM Algorithm

1: **for** $k = 0, 1, \ldots,$ W **do**

2: **for** all intersections $i \in \mathcal{G}$ **do**

$$\boldsymbol{x}_i^{k+1} \leftarrow \arg\min_{\boldsymbol{x}_i} \left(f_i(\boldsymbol{x}_i) + \left[\boldsymbol{\lambda}_i^k\right]^T \boldsymbol{x}_i + \frac{\rho}{2} ||\boldsymbol{x}_i - \tilde{\boldsymbol{z}}_i^k||_2^2 \right) \tag{4.67}$$

3: **end for**

4: **for** all intersections $i \in \mathcal{G}$ **do**

$$\boldsymbol{z}_g^{k+1} = \arg\min_{\boldsymbol{z}_g \in [0,1]} \sum_{\mathcal{M}(i,j)=g} -(\boldsymbol{\lambda}_i^k)_j \cdot \boldsymbol{z}_{\mathcal{M}(i,j)} + \frac{\rho}{2} \left(||\boldsymbol{x}_i^{k+1}||_2^2 - 2(\boldsymbol{x}_i^{k+1})_j \boldsymbol{z}_{\mathcal{M}(i,j)} + \boldsymbol{z}_{\mathcal{M}(i,j)}^2 \right) \tag{4.68}$$

5: **end for**

6: **for** all intersections $i \in \mathcal{G}$ **do**

$$\boldsymbol{\lambda}_i^{k+1} = \boldsymbol{\lambda}_i^k + \rho(\boldsymbol{x}_i^{k+1} - \tilde{\boldsymbol{z}}_i^{k+1}) \tag{4.69}$$

7: **end for**

8: **end for**

modification, though we won't explore this further in subsequent chapters.

An important consideration in ADMM is determining the termination criterion. A common approach is to monitor the reduction in the primal and dual residuals and assess whether they fall below a predefined threshold. However, due to the discrete nature of our objective function, this method may not be as effective. Therefore, our strategy involves specifying a maximum iteration limit denoted as $W \in \mathbb{Z}$ for the algorithm. This approach proves to be effective in our problem context since, depending on the penalty function, the algorithm can converge to the global minimum within a limited number of iterations. The contrast in complexity between the two penalty functions will become evident in the results presented in Chapter 5. The entire algorithm can now be succinctly summarized as illustrated in Algorithm 2.

4.2.2 Greedy Consensus

In this section, our aim was to devise an alternative algorithm that offers a simpler approach than ADMM and requires fewer optimization steps, while still addressing the same consensus problem. Although this method lacks formal theoretical guarantees, we constructed it based on an intuitive rationale. The summation of pressures per intersection, as defined in (2.6), can serve as an indicator of an intersection's traffic status. A low sum might not definitively indicate fewer cars, as there could be a comparable number on both upstream and downstream lanes. Conversely, a high sum signifies a considerable number of cars on upstream lanes, waiting for a green light. Consequently, intersections with high sums should prioritize granting green lights to lanes in need. Nonetheless, this approach may result in some intersections being unable to attain their local optima due to the need to adapt their decisions. The core concept here is to prioritize intersections with high local objective values, indicating higher pressure, and to deprioritize intersections with lower objective values.

Algorithm 3 Synchronous Greedy Binary Consensus, Code for intersection i

1: **Initialize**: $\text{DET}_i = \text{False}, \forall i \in \mathcal{G}$

2: **while** $\text{DET}_i = \text{False}$ **do**

3: Compute and broadcast P_{i,ϕ_k}, $\forall \phi_k \in \Phi_i$ to neighbors
4: Optimize $\hat{P}_i(t, \boldsymbol{X}_i)$
5: Broadcast $\hat{P}_i(t, \boldsymbol{X}_i)$ and $\boldsymbol{X}_i$ to neighbors $\tilde{\mathcal{N}}$

6: **if** $\boldsymbol{X}_i = \boldsymbol{X}_j$, $\forall j \in \mathcal{N}_i$ **then**
7: Fix $\boldsymbol{X}_i$ and set $\text{DET}_i = \text{True}$
8: Broadcast $\text{DET}_j = \text{True}$ to neighbor $j \in \mathcal{N}_i$
9: Terminate
10: **end if**

11: **if** $\hat{P}_i(t, \boldsymbol{X}_i)) < \hat{P}_j(t, \boldsymbol{X}_j), \forall j \in \mathcal{N}_i$ **then**
12: Majority vote to determine ϕ_k
13: Fix ϕ_k and set $\text{DET}_i = \text{True}$
14: Broadcast DET_i and ϕ_k to neighbors
15: Terminate
16: **end if**

17: **if** Receive $\text{DET}_i = \text{True}$ from any neighbour **then**
18: Fix ϕ_k and set $\text{DET}_i = \text{True}$
19: Broadcast ϕ_k and $\text{DET}_i = \text{True}$ to neighbors
20: Terminate
21: **end if**

22: **end while**

For this purpose, we introduce the following framework. Each intersection is assigned a unique index, randomly allocated, to serve as a tie-breaker. Additionally, each intersection i has a flag denoted as DET_i, which indicates whether the intersection has completed its operation. This flag is initially set to False for natural reasons. Similar to the ADMM approach, we use $\boldsymbol{x}_i$ to represent the local decision of intersection i. Moreover, we use $\boldsymbol{X}_i = [\boldsymbol{x}_i, \boldsymbol{x}_i'], ; i' \in \tilde{\mathcal{N}}$, to refer to the tensor that includes all variables intersection i is aware of. This encompasses its own phases as well as the phases of all neighboring intersections.

The algorithm commences with the computation of the standard pressure definition (2.6) for all feasible phases. Subsequently, this information is broadcasted to the neighboring intersections, which is necessary for them to evaluate their local objectives as defined in (4.44). In lines 4-5, each intersection optimizes its local objective function and transmits its corresponding optimal decision along with the objective value. It's important to note that this optimal decision encompasses the decisions of its neighbors as well. Afterward, the algorithm checks whether its own decision aligns with those made by its neighbors. Consensus is achieved if the following two conditions are satisfied:

- $(\phi_k^{(i)})_i = (\phi_k^{(j)})_i, \forall j \in \mathcal{N}_i$

- $(\phi_k^{(i)})_j = (\phi_k^{(j)})_j, \forall j \in \mathcal{N}_i$

where we denote $(\phi_k^{(i)})_j$ as the decision made by intersection i concerning phase ϕ_k from intersection j. If consensus is reached, we utilize all these decisions, encompassing all $(\phi_k^{(i)})_j$ decisions for $j \in \mathcal{N}_i$, and refrain from altering them further until the algorithm's conclusion. Furthermore, we set our own flag to True and communicate that we have terminated. Additionally, this termination could extend to neighbors as well, as our consensus implies alignment between their decisions and ours.

However, if consensus is not achieved, we proceed to identify the intersection with the lowest objective value $\hat{P}_i(t, \boldsymbol{X}_i)$ among neighbors in line 11. This decision adheres to the aforementioned principle of prioritizing intersections with higher objective values and, consequently, greater pressures. Subsequently, the intersection with the lowest objective value, referred to as the "weakest" intersection, is compelled to yield and embrace the decisions of its neighbors. This decision will be established through a majority vote solely among the neighbors. Following this determination, the weakest intersection will terminate its operations.

In the event that we receive information from a neighbor indicating that it has achieved consensus among its variables, we respond by terminating our own operations. This occurs when we receive the termination signal in line 17. As part of this process, we retain the decision we formulated in line 5 and conclude our actions. This decision to terminate aligns with the concept that we can uphold consensus with that specific neighbor only if we refrain from making any further changes to our decision.

Although the algorithm lacks formal theoretical optimality guarantees, it demonstrates rapid convergence. Specifically, it converges within $\mathcal{O}(|\mathcal{G}|)$ iterations, with at least one intersection terminating in each iteration. In practice, multiple intersections terminate simultaneously in each round, further accelerating the algorithm's convergence speed.

Chapter 5

Performance Analysis

5.1 Simulation Environment

Several widely-used simulation environments are available for testing and evaluating our methods. Two notable options are SUMO [32] and Vissim [29], which are extensively employed by major companies like Bosch [11] and DHL [9] for optimizing routes and reducing emissions. However, for our purposes, we opted for a lightweight simulator called CityFlow [45]. CityFlow is an open-source traffic simulator known for its speed and efficiency. It is reported to outperform SUMO in various scenarios and can exhibit a speedup of up to 25 times in large-scale networks [7]. Notably, CityFlow offers a Python interface that simplifies both controlling and retrieving information from the network during simulations. The simulator supports flexible road network definitions and allows for customization of traffic flows, which aligns well with our needs.

In our thesis, we examine three distinct traffic grid networks. To begin, we explore a network comprising 4×3 intersections, each spaced at intervals of 300 m. Subsequently, we construct a second network characterized by shorter block lengths of only 150 m. This design aims to highlight the necessity for collaboration within a more compact network, where intersection decisions can exert a more pronounced influence on neighboring intersections. Throughout our discussion, we will refer to the former as the "coarse" network and the latter as the "fine" network. These networks are illustrated in Figure 5.1 and Figure 5.2.

We use a synthetically generated traffic flow with a peak inflow rate of approximately $0.72 \frac{\text{vehicles}}{\text{sec}}$ per lane. It's worth highlighting that we maintain the same flow rate for both networks to effectively emphasize the influence of network configuration disparities. The introduction of car inflow occurs in short intervals spanning a few seconds, and this continuous inflow remains active for a duration of 1000 sec. Subsequent to this period, we halt the ingress of new vehicles and analyze how efficiently the algorithm redirects the remaining vehicles out of the network over an additional 600 sec.

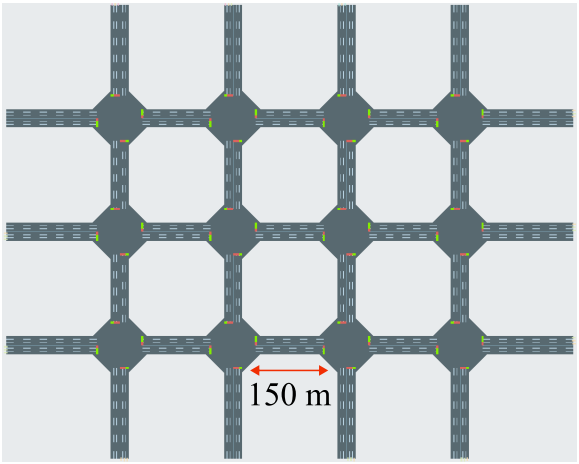


Fig. 5.1: The fine traffic network with a shorter block length 150 meters.

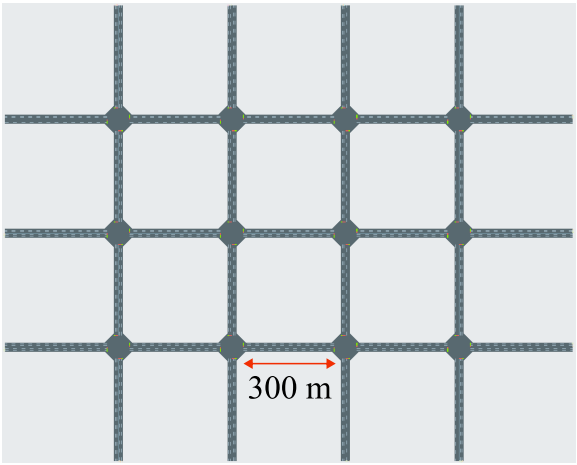


Fig. 5.2: The coarse traffic network with a larger block length of 300 meters.

To highlight the inherent advantages of a distributed formulation over a centralized one, we establish a network comprising 10×10 intersections, where the inter-intersection distances remain at 150 m. This network, referred to as the "large network," serves to emphasize the substantial escalation in optimization variables and constraints that the centralized formulation must contend with. However, in the context of the distributed formulation, the addition of more intersections has no discernible impact on the computational complexity.
For this particular network, we have formulated a new traffic flow scenario. This decision arises from the fact that in larger networks, maintaining the same inflow rate would result in severe congestion at the intersections located at the periphery, while the inner intersections would experience minimal traffic. The objective here is to gradually saturate the network to achieve a relatively uniform distribution of vehicles across the entire network. To achieve this, we have reduced the peak inflow rate to approximately 0.2 cars/sec per lane.

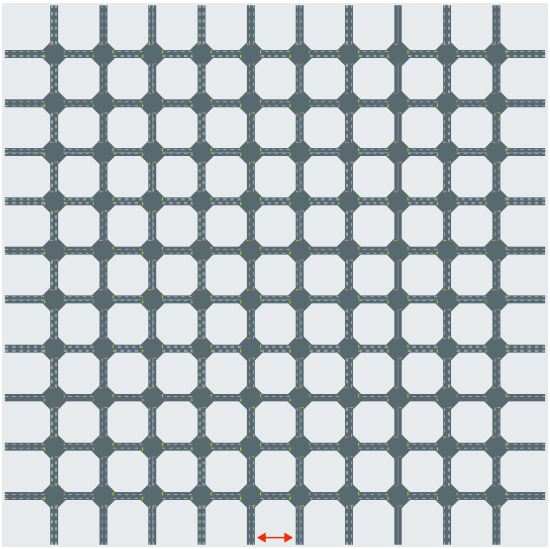


Fig. 5.3: A depiction of the large traffic network featuring a block length of 150 meters and an arrangement of intersections in a 10×10 grid layout.

5.2 Evaluation Metrics

We have now presented various algorithms for traffic flow control and will proceed to introduce the metrics that will serve as the basis for our comparison. We will record several metrics that often have close relationships with each other that are summarized in Table 5.1.

CityFlow offers the option to record travel times through their API. It's worth noting that the implementation of entry lanes ($l \in \mathcal{L}_{entry}$) includes a waiting buffer, which holds cars scheduled to enter the network but are temporarily held due to congestion. When requesting travel times from CityFlow, these buffered cars are also considered, potentially leading to longer recorded travel times. While this doesn't pose a problem, we aimed to provide a more specific comparison by focusing solely on the travel time of cars that have physically entered the network. This will be referred to as the travel time without the waiting buffer. The discrepancy between these two metrics can be attributed to the number of cars that have entered the network, which we'll denote as the throughput. Although throughput typically measures the number of cars that have traversed the network and exited, it strongly correlates with the number of cars that have entered the network, given the limited available space within the network.

Another essential consideration in simulations is the speed of the cars. Higher speeds generally translate to fewer stops for vehicles. This outcome could arise from better traffic management overall or the establishment of green waves that facilitate quicker passage throughout the network.

Two other significant metrics include the cars per lane, which measures the extent to which the network is utilized, and the average waiting time per car. Although these metrics are correlated, variations can emerge between them. High cars-per-lane values suggest the network is operating near its capacity. However, elevated average waiting times per car might arise from specific lanes with lighter traffic being overlooked. In such cases, the cars-per-lane average might remain stable, but individual waiting times could increase.

Finally, we record the frequency of traffic light changes within a specified timeframe. It's im-

Metric	Unit
Travel time (with a waiting buffer)	[sec]
Travel time (without a waiting buffer)	[sec]
Throughput	[cars/1600 sec]
Car speed	[m/sec]
Cars per lane	[cars/lane]
Waiting time per car	[sec]
Phase change probability	-

Tab. 5.1: All recorded metrics and units for comparison.

portant to note that all algorithms adhere to the rule that phase changes can only occur every Δ seconds. Calculating the average frequency provides insights into the likelihood of a phase change during each new traffic light update.

5.3 Simulation Results

Prior to delving into the comparison of the algorithms using the presented metrics, we will provide commentary on the queue model presented in equations (3.5) to (3.7).

5.3.1 Validation of queue model

In this section, we undertake an assessment of the accuracy of our previously introduced queue model. The validation of the model is carried out through the subsequent analysis. At each simulation time step, we compare the predicted number of cars with the actual simulated count for each lane. In instances where we have multiple future predictions, we compute the average across all these predictions for each lane. All this data is logged for every simulation time step. The validation results showcase both the standard deviation of the mean and the average of the mean for the recorded data.

Coarse Network

Commencing our analysis with the coarse network, Figure 5.4 presents a comparison between employing more frequent prediction steps versus utilizing only 1 step within the same period. It is essential to highlight that the frequency of traffic light changes remains constant in both scenarios, occurring every $\Delta = 20\,\text{sec}$. This signifies that when we establish the phase at time step $t = \tau$, it remains fixed until $t = \tau + \Delta$. Consequently, at the conclusion of Δ seconds, both methods predict an equivalent number of cars. Nevertheless, opting for smaller step sizes yields a marginal advantage. The mean error, representing the discrepancy between the predicted and simulated car counts, is smaller when multiple steps are incorporated within a single update. This advantage holds relevance for the centralized controller, as it capitalizes on future predictions to anticipate traffic with heightened accuracy within a single update period. This refined foresight can then be harnessed to formulate a more optimal solution.
In Figure 5.4, we observe the utilization of two distinct parameter settings. As depicted in the plot, the simulation incorporating multiple predictions within the same period exhibits a smaller mean error, a result in line with our earlier explanation. The saturation flow rate C and exogenous inflow d remain consistent for both cases: $\Delta \cdot C = 6$ cars and $\Delta \cdot d = 5$ cars over the

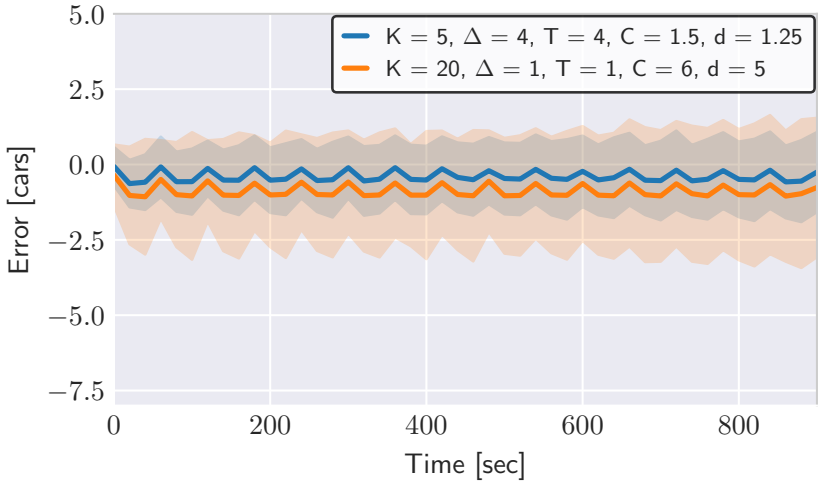


Fig. 5.4: The mean and standard deviation of the prediction error of simulations with two different Horizon lengths simulated in the coarse network. More prediction steps give a lower mean error.

period of $\Delta \cdot S$ seconds. For the purpose of this analysis, we did not consider extending the prediction horizon T beyond Δ, as our simulations only encompass the subsequent Δ before updating the traffic light. However, augmenting the prediction horizon could prove advantageous for optimizing average travel times, rather than the model itself. Regrettably, the computation time experiences a significant surge with the inclusion of more prediction steps. Hence, we made the decision to limit our predictions to just one step into the future.

While the mean error remains relatively modest, it can be further reduced by choosing lower values for the capacity and exogenous flow parameters. This trend can be attributed to the inherent simplicity of the model, which assumes conditions such as no traffic congestion and cars being able to leave a lane immediately upon arrival. These assumptions, while aiding in computational efficiency, do not capture the complexities of real-world traffic dynamics.

Indeed, opting for higher capacity and exogenous flow values can result in larger errors, especially when real traffic conditions do not align with the model's assumptions. Another contributing factor is the unequal inflow distribution among entry lanes. Despite the relatively balanced inflow, the single parameter used for the entire network may not fully capture these nuanced details. This discrepancy becomes evident when comparing simulations that consider predictions from all lanes to those that omit entry lanes. The comparison clearly shows that a substantial portion of the error originates from these entry lanes.

Reducing the exogenous flow might lead to neglecting the lanes that experience higher inflow. Interestingly, after simulation, it might become evident that maintaining the higher flow is more beneficial for the overall traffic dynamics. This underscores the complex interplay between different factors and the need for careful parameter selection.

Enlarging the prediction horizon leads to a significant increase in computational time. This is due to the need to add numerous constraints and variables to model relations such as $\min c, q_{l,m}$ and $\min \left\{c_{l,m}(t), q_{l,m}(t), \min_{p \in Out_m} \left\{q_{m,p}^{max} - q_{m,p}(t)\right\}\right\}$. To mitigate this computational complexity, an approach is to approximate these expressions. If we assume a minimal amount of cars present on all lanes, then we can write

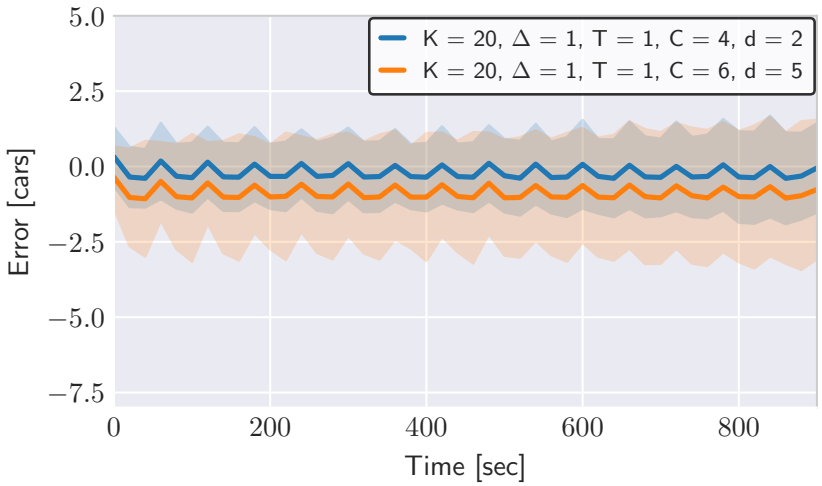


Fig. 5.5: The mean and standard deviation of the prediction error of simulations with two different capacity and exogenous flows simulated in the coarse network. A lower flow increases the accuracy of the model overall.

$$\text{Inflow} = \min\{C, q_{l,m}(t)\} \approx c \tag{5.1}$$

$$\text{Outflow} = \min\left\{C, q_{l,m}(t), \min_{p \in Out_m} \left\{q_{m,p}^{max} - q_{m,p}(t)\right\}\right\} \approx c \tag{5.2}$$

as previously introduced in equation (3.15). This approximation assumes that in each time step, the same amount of cars flows into and out of each lane, considering the respective traffic phase that is active. In the simulation, it can be observed that this approximation doesn't result in significant deviations from the original model.

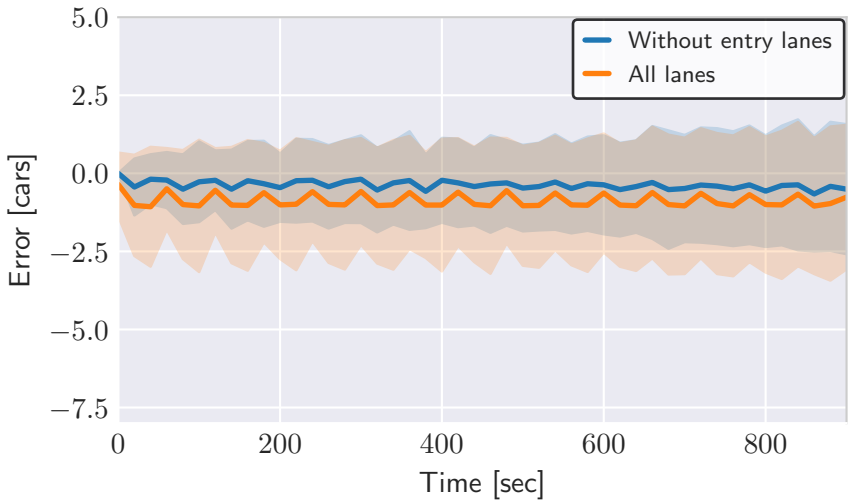


Fig. 5.6: The average and standard deviation of prediction errors from simulations, comparing scenarios with and without the inclusion of entry lanes in the coarse network. Notably, the error decreases when entry lanes are omitted.

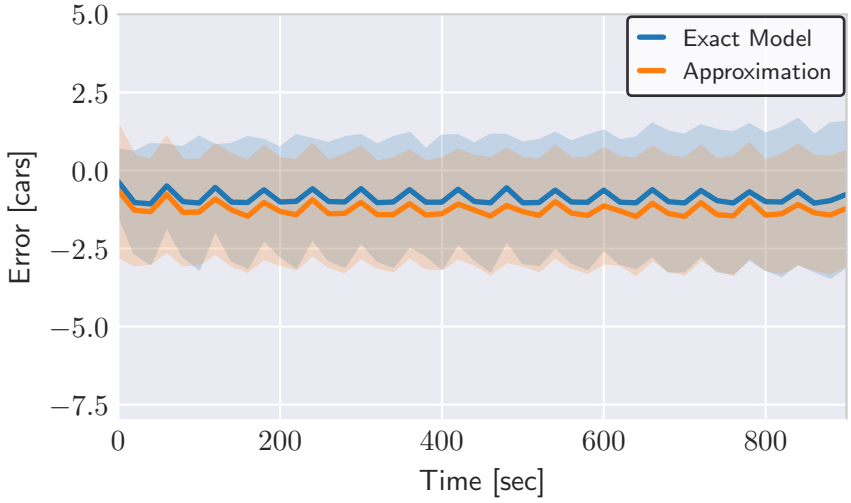


Fig. 5.7: The mean and standard deviation of prediction errors in simulations in the coarse network, both when utilizing the queue model approximation and when using the original model. Notably, the approximation yields slightly inferior results compared to the original model.

Fine Network

The fine network exhibits similar behavior to the coarse one. The primary distinction from the coarse model lies in our selection of capacity and exogenous flow. Interestingly, despite increasing both parameters to 10 cars per Δ seconds leading to a worse mean error, the travel time per car has reduced. This can be attributed to the fact that the lanes are shorter in the fine network. As cars arrive at a lane, they tend to leave that lane more rapidly compared to the larger network. The figures depicting the comparisons for the fine network can be found in Appendix D, offering a comprehensive view of the analysis conducted for this network configuration.

5.3.2 Performance Comparison

In the upcoming chapter, we will proceed to compare all the mentioned algorithms using various metrics. Our initial focus will be on the fine network this time.

Parameter Selection

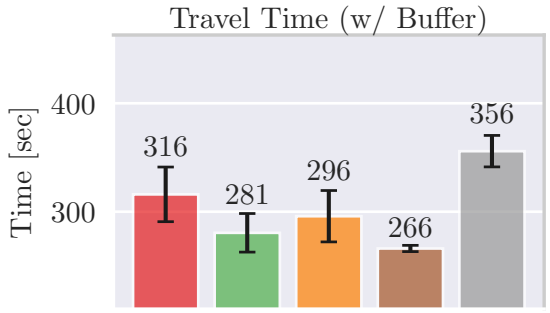
Throughout all simulations, we adopted an update interval of $\Delta = 20\,\text{sec}$, as it represents a balanced compromise between the controller's autonomy and the necessity for a minimum green time to ensure a certain flow of vehicles through intersections. The specific parameter choices used in the simulations are provided in Appendix E.

Fine Network

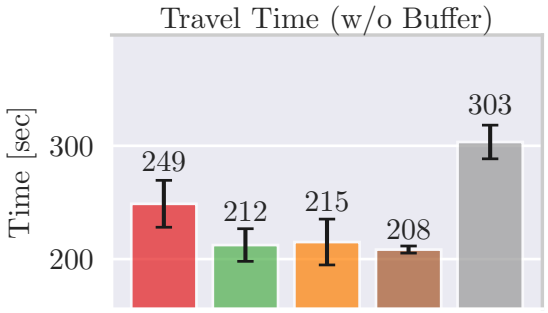
We start by addressing arguably the most crucial metric - travel time. An insightful overview of their performance in terms of travel time is provided by Figure 5.8. Among these algorithms, the greedy approach exhibits the most significant divergence, amounting to a substantial difference of 81 seconds between the travel time measured with and without a waiting buffer. This discrepancy suggests that the greedy algorithm tends to prioritize specific entry lanes, resulting in a higher accumulation of cars in the buffer of the neglected entry lanes. Interestingly, this strategic decision can foster an increase in the service rate of some entry lanes, resulting in the greedy algorithm nearly closing the gap with ADMM (continuous penalty). This trend is highlighted in the plot, where Greedy achieves a similar level of performance in terms of throughput as ADMM (continuous penalty) over the entire simulation period.

The choice of the binary penalty with ADMM resulted in a decrease in performance, leading to fewer cars being able to enter the network. An explanation for this difference in performance can be attributed to the nature of the penalties employed. The binary penalty penalizes intersections when a lane becomes congested and exceeds its capacity. On the other hand, the continuous penalty encourages neighboring intersections to create a more continuous flow of green lights. Consequently, if a lane is granted the right of way, the downstream lane is also more likely to receive a green light. This approach benefits vehicles by allowing them to maintain their speed and avoid unnecessary stops at subsequent red lights. As a result, this not only reduces travel times but also increases overall speed and flow within the network, as demonstrated in Figure 5.9.

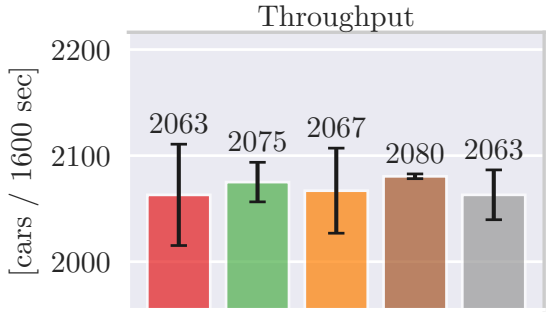
As expected, the centralized solution outperformed the other algorithms not only in terms of travel time but also in the number of vehicles that successfully entered the network. It's important to emphasize that the centralized algorithm exhibited the smallest variance among all the other algorithms. This advantage arises from the algorithm's access to comprehensive information and its ability to integrate a prediction horizon, effectively considering future events. In contrast, the Maximum Pressure algorithm performed less favorably. This can be attributed to each intersection independently optimizing its decisions without communicating with neighboring intersections. Notably, when considering the travel time without the buffer, ADMM was able



(a) Measured travel time including the waiting buffer



(b) Measured travel time without including the waiting buffer



(c) The count of cars that have passed through the network during the simulation time, a metric closely correlated with the throughput

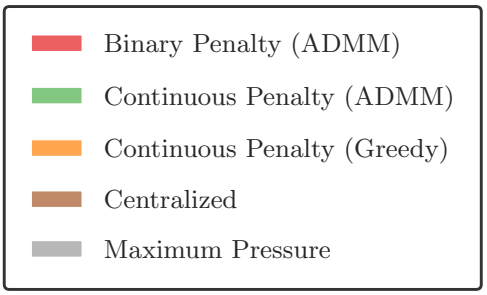
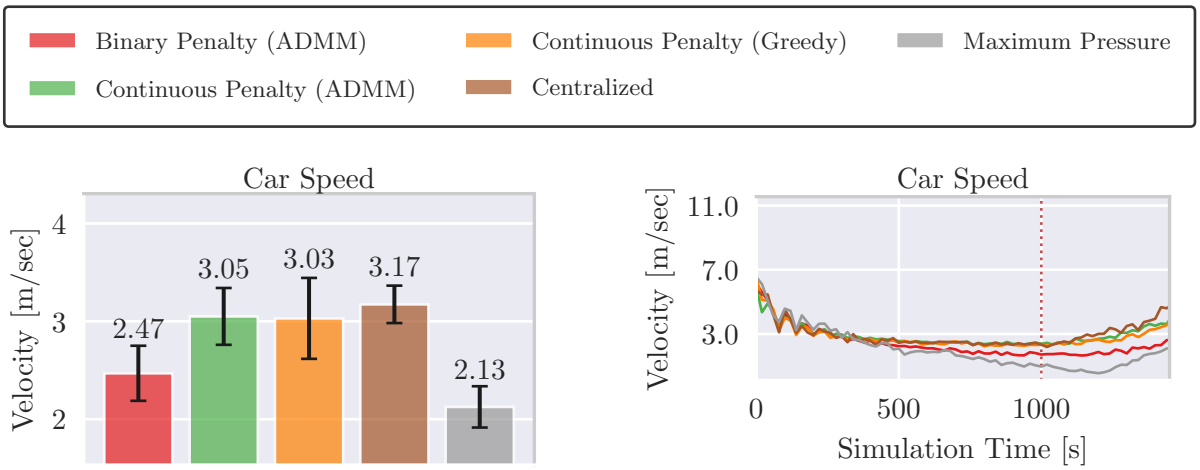


Fig. 5.8: Comparison of travel time and throughput of all algorithms in the fine network

to achieve a significant improvement over MP, with an advantage of up to 30 %.

A similar trend emerges when analyzing the average car speed. At the beginning of the simulation with an empty network, the speed is high. As more cars enter the network, the speed naturally decreases across all algorithms. However, after the last car enters the network at around 1000 sec, the average speed starts to rise again. This phenomenon highlights the varying abilities of different algorithms to efficiently divert cars out of the network. Notably, the centralized algorithm demonstrates the most effective performance in this aspect, while the maximum pressure algorithm faces considerable challenges, achieving results over 40% lower than those of ADMM. The reduced speed under the maximum pressure algorithm isn't solely due to a slower pace in redirecting the remaining cars out of the network; it's also a consequence of longer queues that accumulate in comparison to the centralized algorithm. This observation is visually apparent in Figure 5.10, which clearly illustrates that lanes remain occupied for longer durations under the maximum pressure algorithm compared to other algorithms.

It's also important to note that during the first 500 sec, the performance of all algorithms is roughly similar. Thus, when the network has a relatively low number of cars, the maximum pressure algorithm performs relatively well compared to the other algorithms. The issue arises when the network becomes more congested, as the assumption underlying the maximum pressure algorithm - that there are infinitely long queues - becomes increasingly violated as lane occupancy increases. In such settings, cars have to wait up to 40% longer than when using ADMM.



(a) Mean car speed averaged across the entire simulation duration

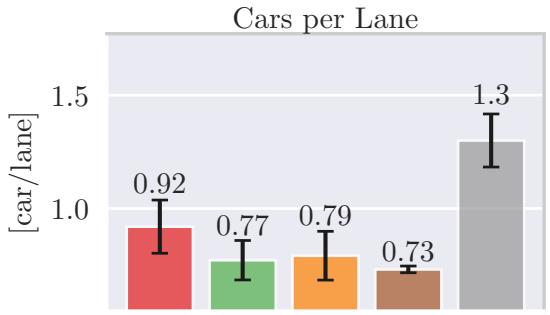
(b) Mean car speed measured over the whole simulation time

Fig. 5.9: Comparison of car speeds for different algorithms in the fine network. The red dotted line indicates the timestamp at which the last car enters the network.

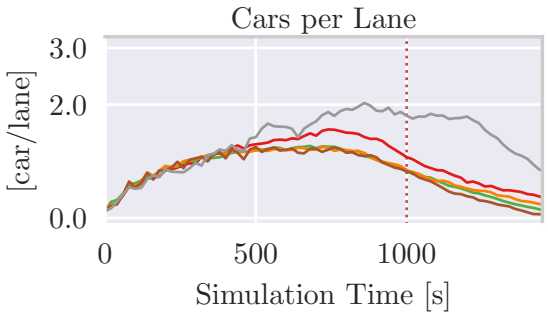
Even though ADMM with the binary penalty attempts to mitigate this problem, it consistently performs worse than ADMM with the continuous penalty. This observation is noteworthy because ADMM, while using the continuous penalty, struggles to reach the true optimal solution within 30 ADMM iterations, whereas using the binary penalty allows it to match the true optimal solution at every time step. This distinction is clearly evident in Figure 5.11. Therefore, the difference in performance primarily arises from the penalty function used, rather than the algorithm itself. Figure 5.11 further illustrates why ADMM slightly outperforms the Greedy algorithm.

While ADMM consistently reaches around 80% to 90% of the true optimal solution, the greedy algorithm only attains around 50% to 60% of the optimal solution. It's important to note that the ADMM algorithm is formulated as a minimization problem, whereas Greedy is a maximization problem. However, pushing the solution to higher levels only leads to diminishing returns, yielding minimal improvements for any metric. This justifies the use of the greedy algorithm. Another significant advantage of the greedy algorithm is its computational speed compared to ADMM. The greedy algorithm is guaranteed to converge in at most $\mathcal{O}(n)$ iterations, while Maximum Pressure converges even faster, as seen in Figure 5.12.

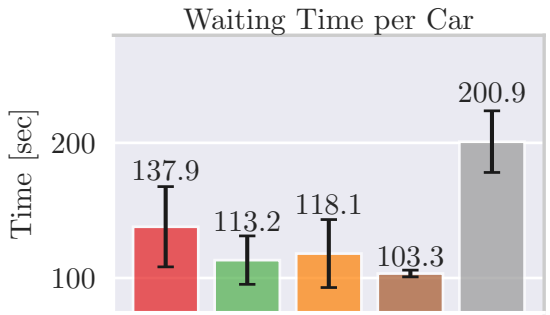
The speed of the centralized algorithm is a notable observation. It appears to be even faster than ADMM, which holds true in this specific scenario. However, it's important to maintain perspective. Our analysis is based on a network with only one prediction horizon and a limited size of 12 intersections. This relatively small scale likely contributes to the apparent speed advantage of the centralized algorithm. While a computation time of 12.8 sec might seem lengthy, especially considering the 20-second decision interval, the true strength of the centralized algorithm lies in its ability to distribute the problem across different intersections. The extended computation time observed in our simulation primarily results from our sequential implementation on a single machine. In a real-world deployment, the computational load can be efficiently distributed across various intersections, leveraging parallel processing capabilities. This parallelization significantly mitigates computation time, with the primary bottleneck shifting to the speed of communication between neighboring intersections.



(a) The average number of cars per lane averaged over the whole simulation time and all lanes



(b) The average number of cars per lane during the whole simulation time



(c) The average waiting time per car, averaged over the entire simulation time

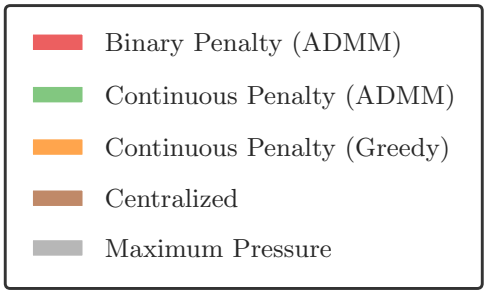
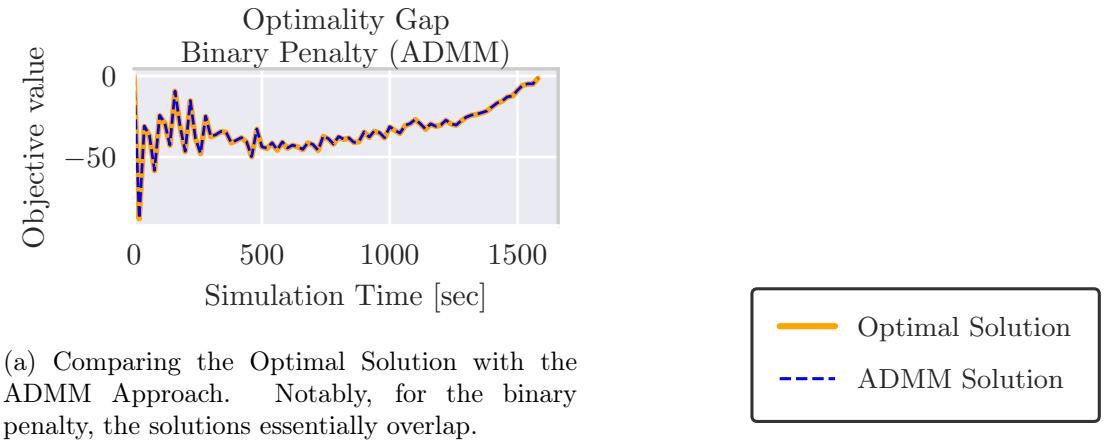


Fig. 5.10: Comparison of congestion metrics and waiting time metrics within the fine network. The red dotted line indicates the timestamp at which the last car enters the network.

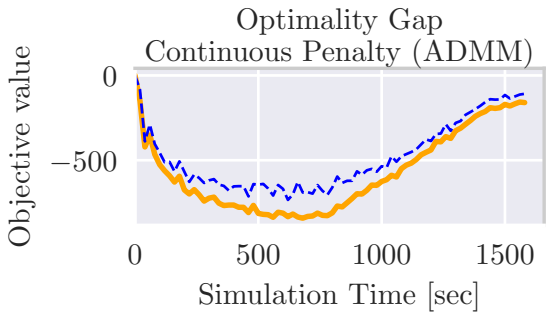
Furthermore, the computation time of both the centralized formulation and ADMM with the continuous penalty seems to be influenced by the traffic situation. During high traffic periods, the computation time increases, while it decreases during periods of lower traffic. This behavior likely arises from the increased complexity introduced by the continuous penalty, as evidenced in Figure 5.11. Conversely, the binary penalty formulation, while not achieving the same solution quality, managed to halve the computation time for this scenario, reducing it to 12.8 seconds per time step compared to the queue penalty.

The final metric to consider is the probability of a traffic light change at each update. A value close to 1.0 indicates a high likelihood of a traffic light change in the subsequent update.

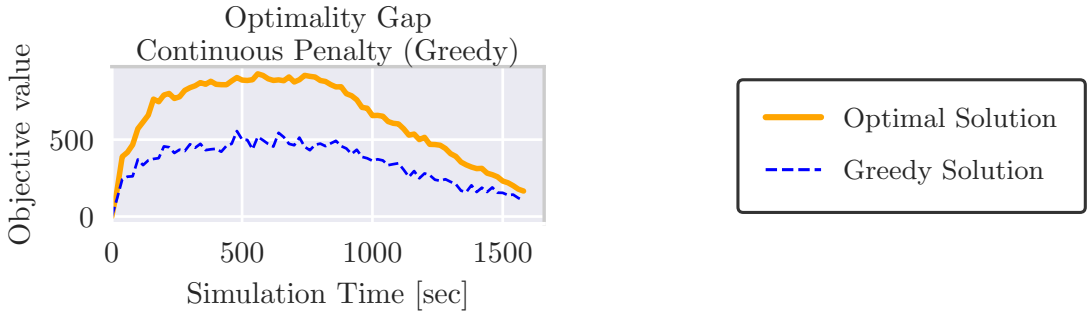
It's worth noting that adjusting the frequency of traffic light changes has a significant impact on performance in this traffic network. For instance, changing the traffic light too infrequently, as observed in the case of the Maximum Pressure algorithm, leads to poorer performance. Conversely, changing it too frequently, as seen in ADMM with the binary penalty, does not necessarily yield optimal results compared to the other three algorithms. This comparison provides more of an intriguing insight rather than a definitive analysis.



(a) Comparing the Optimal Solution with the ADMM Approach. Notably, for the binary penalty, the solutions essentially overlap.

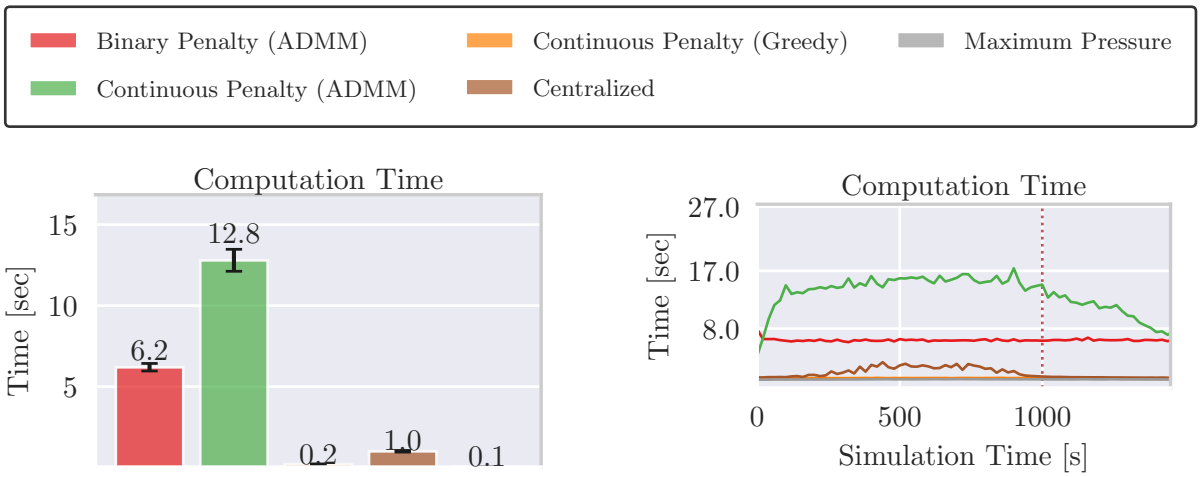


(b) Comparing the Optimal Solution with the ADMM Approach. Importantly, for the continuous penalty, the ADMM only achieves solutions that are close to optimal.



(c) Comparing the Optimal Solution with the Greedy Approach. Worth noting that the Greedy method yields a solution that is inferior to the ADMM approach.

Fig. 5.11: Comparison of different optimality gaps in the fine network. ADMM is formulated as a minimization problem, whereas Greedy is a maximization problem



(a) The average computation time for one simulation time step

(b) The average computation time during the whole simulation.

Fig. 5.12: Comparison of computation times for one simulation time step in the fine network

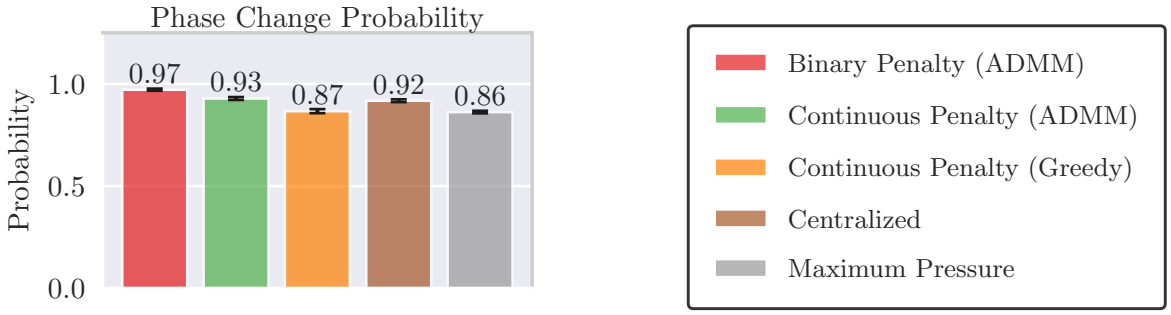


Fig. 5.13: Comparative analysis of the algorithmic phase transition probabilities in the fine network

Coarse Network

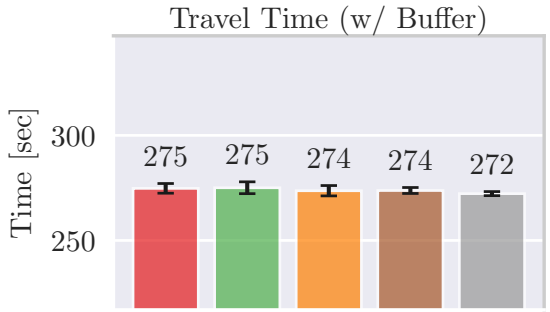
As we did for the fine network, let's begin by examining the travel time for the small network. Notably, the distinctions between different algorithms become less pronounced, and it becomes increasingly challenging to outperform the maximum pressure algorithm. This phenomenon can be attributed to the longer road lanes present in the small network. In the fine network, the closer proximity of intersections allowed for a more significant impact of their decisions on their neighbors, as cars could swiftly traverse between intersections. However, with increased distances between intersections in the small network, this interdependency diminishes. Intersections are better off making independent decisions, as the opinions of other intersections could potentially lead them away from optimal solutions. This highlights the advantage of allowing intersections to prioritize their own decisions rather than being overly influenced by neighbors.

The longer road length in the small network brings another benefit compared to the smaller network. By employing the exact same flow rates as used in the small network, we can make a direct comparison between the two. It can be observed that the average travel time with a buffer decreases for all algorithms in the large network, except for the centralized formulation. However, the travel time with the waiting buffer increases. This can be attributed to fewer cars waiting in the buffer and more cars being able to physically enter the network. Unfortunately, this led to a slightly higher travel time inside the network. This trend indicates that the speed at which cars were able to enter the network, as depicted in Figure 5.15, can achieve higher values. The entry lanes were one of the critical points in the fine network. The same pattern is noticeable in Figure 5.16, where the overall number of cars per lane has decreased.

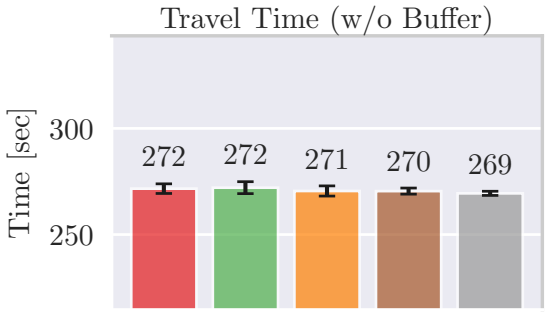
During the simulation, an interesting observation emerged: lower penalty weights for the algorithms tended to result in better performance than higher weights. As discussed earlier, this suggests that the maximum pressure algorithm performed well, and it was advantageous to adjust the parameters of the other algorithms to align them more closely with the principles of the maximum pressure formulation. Consequently, the problem becomes less influenced by neighbors' decisions, and even the greedy algorithm managed to achieve an average optimality gap of around 10 %. This is roughly the same level of performance that ADMM achieved, while still maintaining a lower computation time, as demonstrated in Figure 5.18.

This plot further highlights the dependency of the computation time for both the centralized and ADMM with the queue penalty on the traffic situation. This characteristic is not favorable in high-density cities where traffic conditions can vary significantly.

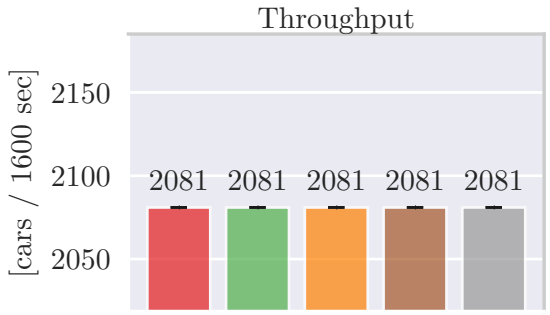
Once more, we'll delve into the probability of traffic light switches. Figure 5.19 unveils a similar pattern among the algorithms. Curiously, it appears that the frequency of phase changes doesn't significantly affect the overall performance.



(a) Measured travel time including the waiting buffer



(b) Measured travel time without including the waiting buffer



(c) The count of cars that have passed through the network during the simulation time, a metric closely correlated with the throughput

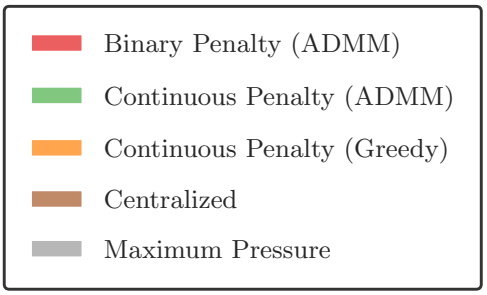
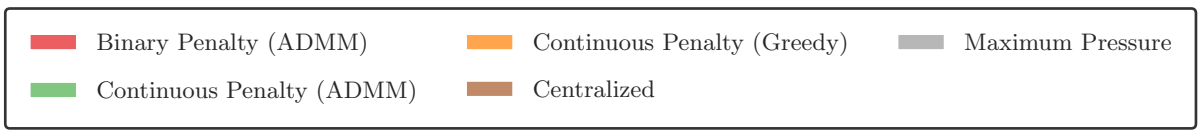
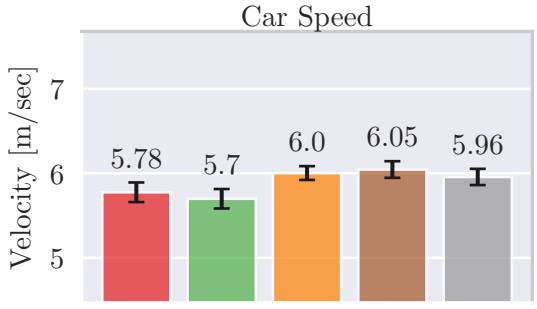
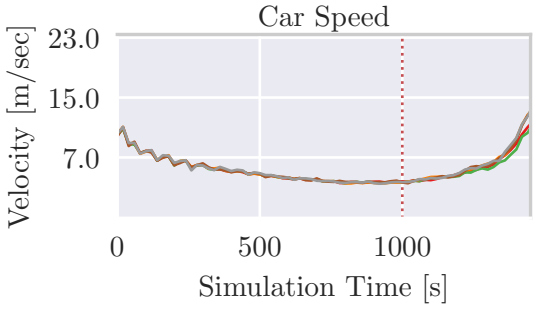


Fig. 5.14: Comparison of travel time and throughput of all algorithms in the coarse network



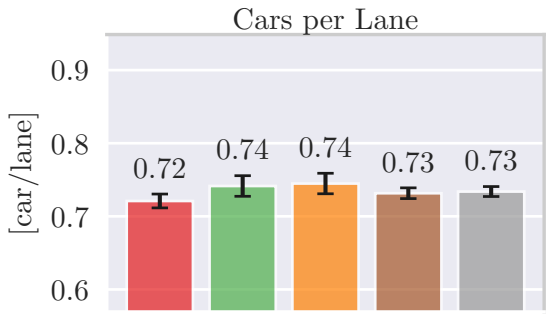


(a) Mean car speed averaged across the entire simulation duration

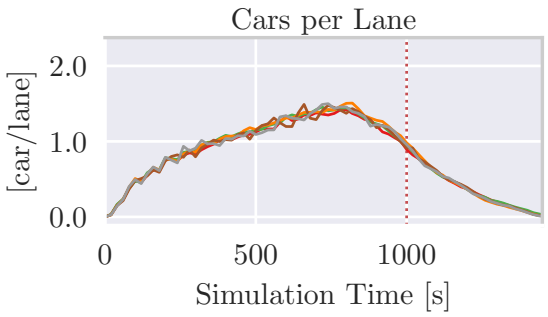


(b) Mean car speed measured over the whole simulation time

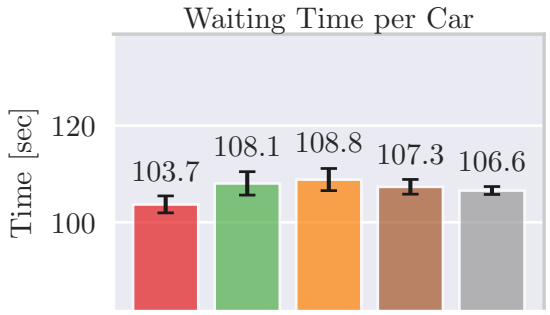
Fig. 5.15: Comparison of car speeds for different algorithms in the coarse network. The red dotted line indicates the timestamp at which the last car enters the network

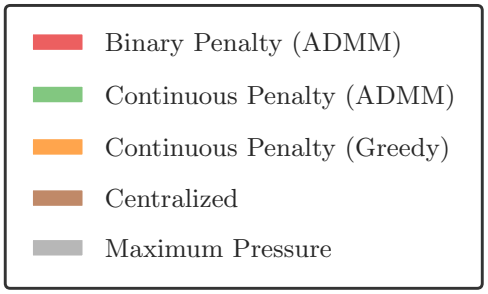


(a) The average number of cars per lane averaged over the whole simulation time all lanes



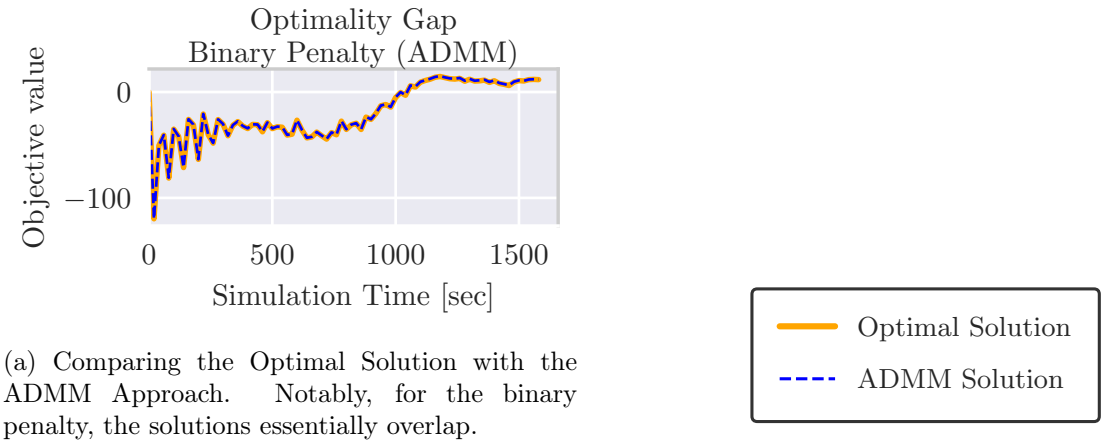
(b) The average number of cars per lane during the whole simulation time



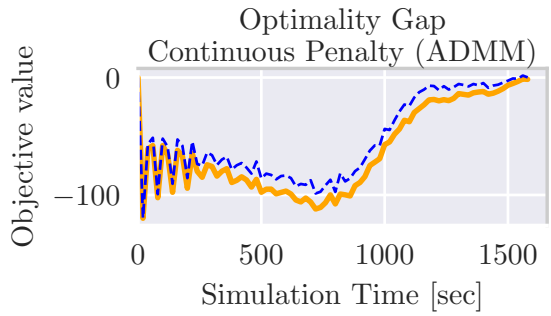


(c) The average waiting time per car averaged over the entire simulation time.

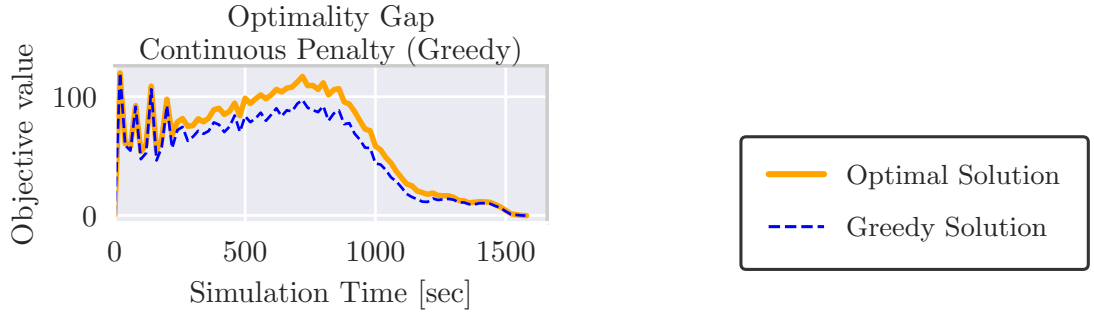
Fig. 5.16: Comparison of congestion metrics and waiting time metrics within the coarse network



(a) Comparing the Optimal Solution with the ADMM Approach. Notably, for the binary penalty, the solutions essentially overlap.

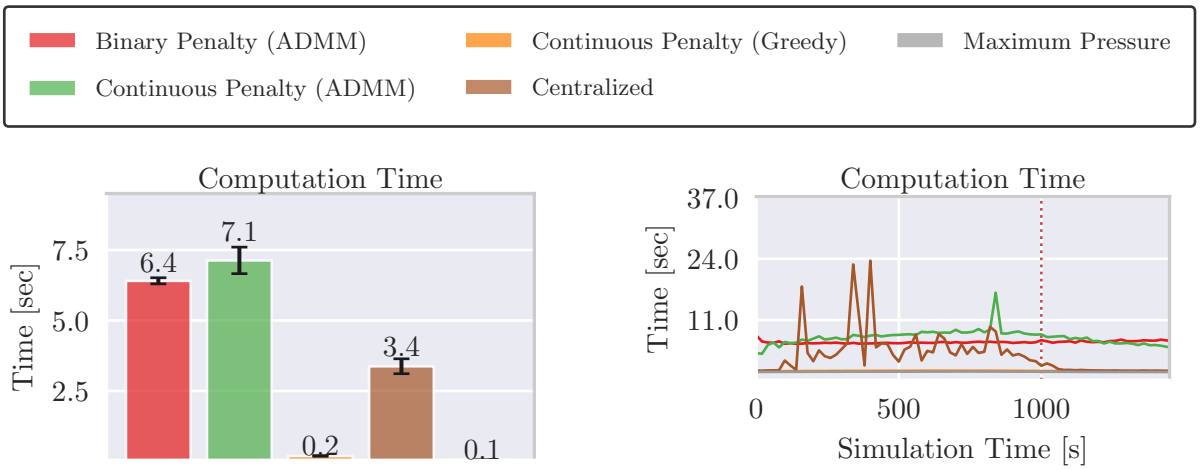


(b) Comparing the Optimal Solution with the ADMM Approach. Importantly, for the continuous penalty in the coarse network, ADMM achieves solutions too that are close to optimal.



(c) Comparing the Optimal Solution with the Greedy Approach. Worth noting that the Greedy method for the coarse network yields a solution that is comparable to the ADMM approach.

Fig. 5.17: Comparison of different optimality gaps in the coarse network. ADMM is formulated as a minimization problem, whereas Greedy is a maximization problem



(a) The average computation time for one simulation time step

(b) The average computation time during the whole simulation

Fig. 5.18: Comparison of computation times for one simulation time step in the coarse network.

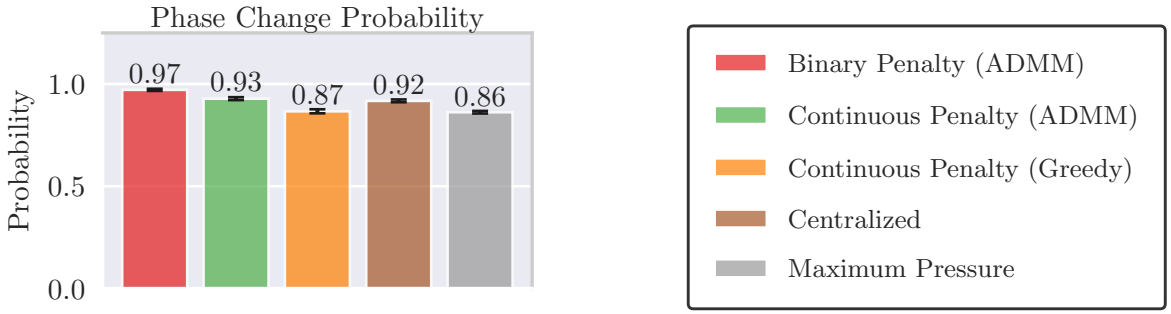


Fig. 5.19: Comparative analysis of the algorithmic phase transition probabilities in the coarse network

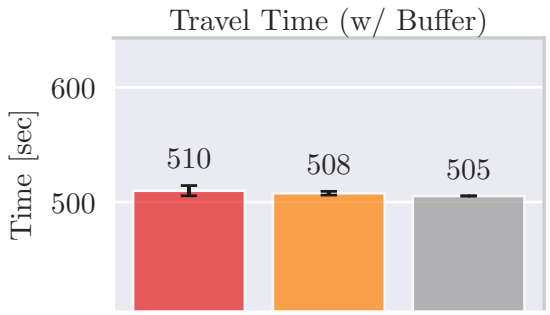
Large Network

We now present a concise overview of the outcomes on the large network. As emphasized earlier, the centralized approach is not scalable and would take too long to solve even with a prediction horizon of $T = 1$. Therefore, we exclude it from this analysis and concentrate solely on the distributed methods. Furthermore, our previous analysis highlighted that, for the fine network, the continuous penalty outperformed the binary one. Thus, for the subsequent analysis, we solely consider the continuous penalty formulation.

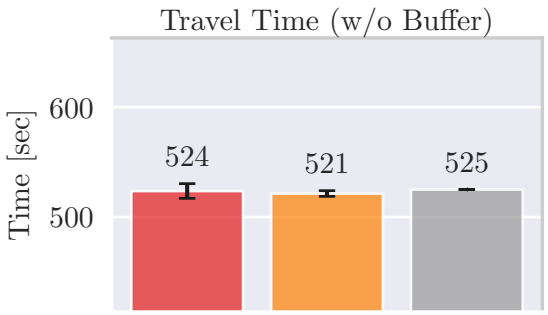
The outcomes for the large network, as depicted in Figure 5.20, differ from what was observed in the fine network. This divergence can be attributed to the distinct traffic flow introduced into the network. Despite reducing the peak flow rate, a uniform distribution of cars across the network remains challenging due to its size. Consequently, the current traffic flow predominantly concentrates on intersections located at the edges of the network. As a result, the algorithm's challenge lies in prioritizing inflowing lanes to expedite the diversion of traffic toward the network's center. Achieving this could entail assigning different penalty weights to intersections at the edge versus those in the center. However, in the present simulation, all intersections are treated equally.

As evident in Figure 5.20, both the Greedy algorithm and ADMM effectively enable more cars to enter the network while maintaining comparable travel times. Although the difference in throughput may appear relatively minor, achieving this in simulation is a non-trivial accomplishment.

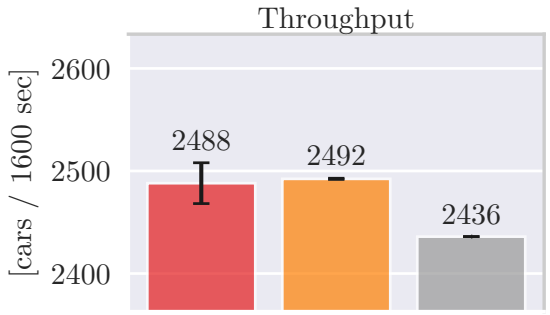
The remaining metrics exhibit marginal differences, with both ADMM and the Greedy algorithm slightly outperforming the Max Pressure algorithm. While these outcomes might deviate from the expectations derived from the earlier analysis, they highlight a slight weakness in our formulation. Specifically, the performance of our approach is heavily reliant on the chosen parameters, the specific network configuration, and the traffic inflow pattern. While our formulation presents an opportunity to capitalize on particular situations, the theory does not provide explicit guidance on parameter selection. Consequently, while there might exist parameter configurations that could yield superior results for this network, there are also parameter combinations that result in worse performance compared to the MP algorithm.



(a) Measured travel time including the waiting buffer.



(b) Measured travel time without including the waiting buffer.



(c) The count of cars that have passed through the network during the simulation time, a metric closely correlated with the throughput.

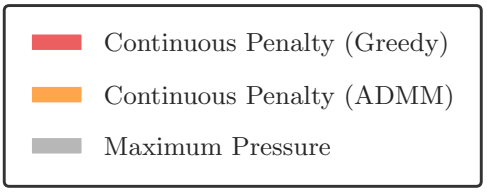


Fig. 5.20: Comparison of travel time and throughput of all algorithms in the large network.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

The escalating challenge of traffic congestion necessitates innovative solutions for regulating traffic flow and optimizing traffic light systems. In this thesis, we introduce a novel approach to traffic light control, aiming to reduce car travel times within the network while considering various associated metrics. Our focus centers on a newly developed distributed controller, designed to surmount limitations observed in the popular Max Pressure technique while retaining theoretical guarantees.

Initially, we adopt a centralized strategy akin to Model Predictive Control (MPC), leveraging a predictive traffic model to address the intricacies of traffic dynamics and signal control. Recognizing the scalability constraints of centralized systems for expansive networks, we swiftly transition to a distributed framework. Here, we integrate the principles of Lyapunov Drift-Plus-Penalty to establish mean rate stability, resulting in a bounded congestion level. This achievement stems from a carefully devised penalty function that not only stabilizes the network but also facilitates concurrent penalty minimization. By incorporating neighboring intersections' actions into this function, we effectively shift the problem from individual signal optimization to a consensus challenge among intersections, thereby fostering collaborative global optimization.

To attain consensus, we introduce the ADMM algorithm along with an innovative Greedy consensus algorithm, an approximation of the former. Our algorithms undergo rigorous testing across diverse network scenarios, spanning from compact configurations to extensive networks with up to 100 intersections. While the MP algorithm demonstrates its effectiveness in networks characterized by larger inter-intersection distances, our approach excels in densely populated networks where the impact of communication overhead becomes more pronounced. Remarkably, in such scenarios, our method demonstrates substantial performance enhancements - reducing travel times by up to 30% and increasing average vehicle speeds by up to 40% compared to the established MP algorithm.
The potency of our approach can be attributed to two primary factors. Firstly, by nurturing collaboration among intersections, we fundamentally enhance overall system efficiency. Secondly, we adeptly address the challenge of infinite queues, which contributes to our performance gains. Despite the supremacy of the ADMM algorithm over the greedy counterpart, the latter retains notable advantages. Notably, the computational swiftness of the greedy approach exceeds that of ADMM. Concurrently, it simplifies the parameter space requiring precise definition, since the greedy algorithm entails the configuration of just one parameter.

The proliferation of parameters poses a concern, as their variability can render simulations highly

sensitive. As such, our study not only makes strides in the realm of traffic signal control but also underscores the intricate interplay between advanced algorithmic techniques, computational efficiency, and the effective management of parameters.

Although simulations involving a large number of intersections showcase modest improvements compared to MP, they underscore two pivotal insights. Primarily, the significance of distribution in adapting algorithms to urban network scales becomes pronounced when contrasted with a centralized approach. Secondly, the algorithm's susceptibility to parameter adjustments emphasizes again the necessity for meticulous parameter calibration in forthcoming research pursuits.

6.2 Future Work

While our algorithm shows promise, there exists ample room for refinement and enhancement. Addressing the sensitivity issue is a prime concern, and one approach involves redefining various parameters. This can be achieved through the utilization of machine learning or reinforcement learning techniques. These approaches would facilitate the adjustment of parameters to harmonize effortlessly with both the complex road network and the dynamic traffic flow.

Exploring the integration of variable parameters presents an intriguing avenue for development. These adaptable parameters could dynamically adjust during simulations, thereby capturing real-time scenarios. As previously highlighted, introducing individualized weights for intersections emerges as an appealing proposition. This direction, while potentially involving a greater number of parameters, holds the promise of delivering solutions with superior quality.

Beyond these adjustments, it's imperative to assess the real-world implications of our methodology, given the heightened complexities compared to our simulations. Delving into the dynamics of actual traffic scenarios adds depth to our analysis. An intriguing facet to consider is the inclusion of individual road user behavior. Such behavior encompasses instances where commuters alter their routes due to traffic congestions in specific directions.

Moreover, it is worth delving into the exploration of incorporating pedestrians into the model. These pedestrian components could be portrayed as supplemental queues, each assigned extra weights to incentivize the algorithm to prioritize pedestrian signals. This strategic incorporation ensures that pedestrian waiting times are minimized, thereby contributing to a more holistic traffic management framework.

An alternative approach to tackling this issue involves drawing inspiration from the realm of game theory. While this avenue has been explored before [1, 14], it offers an intriguing opportunity to amalgamate our existing penalty-based approach with the rich foundation of game theory. By melding these two perspectives, we can potentially unlock novel insights and strategies for addressing the problem at hand. This synthesis of methodologies could pave the way for innovative solutions that capitalize on the strengths of both approaches and offer a more comprehensive framework for resolving the challenges we have encountered.

Bibliography

[1] Hossam M. Abdelghaffar, Hao Yang, and Hesham A. Rakha. "Isolated traffic signal control using a game theoretic framework". In: *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. 2016, pp. 1496–1501.

[2] K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos. "Store-and-forward based methods for the signal control problem in large-scale congested urban road networks". In: *Transportation Research Part C: Emerging Technologies* 17.2 (2009), pp. 163–174.

[3] Bram Bakker et al. "Traffic Light Control by Multiagent Reinforcement Learning Systems". In: vol. 281. Mar. 2010, pp. 475–510.

[4] Simanta Barman et al. "Towards Implementation of Max-Pressure Signal Timing on Minnesota Roads". In: 2022.

[5] Andrew G. Barto and Sridhar Mahadevan. "Recent Advances in Hierarchical Reinforcement Learning". In: *Discrete Event Dynamic Systems* 13.1 (Jan. 2003), pp. 41–77.

[6] Stephen Boyd et al. "Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers". In: *Foundations and Trends in Machine Learning* 3 (Jan. 2011), pp. 1–122.

[7] *CityFlow*. `https://cityflow-project.github.io`. [Accessed 17-08-2023].

[8] Carlos F. Daganzo. "The cell transmission model, part II: Network traffic". In: *Transportation Research Part B: Methodological* 29.2 (1995), pp. 79–93.

[9] DHL. `https://www.dhl.com/ch-en/home.html`. [Accessed 17-08-2023].

[10] Deepeka Garg, Maria Chli, and George Vogiatzis. "Deep Reinforcement Learning for Autonomous Traffic Light Control". In: *2018 3rd IEEE International Conference on Intelligent Transportation Engineering (ICITE)*. IEEE, Oct. 2018, pp. 214–218.

[11] Bosch Global. `https://www.bosch.ch`. [Accessed 17-08-2023].

[12] Roland Glowinski. "On Alternating Direction Methods of Multipliers: A Historical Perspective". In: vol. 34. June 2014, pp. 59–82.

[13] Jean Gregoire et al. *Capacity-aware back-pressure traffic signal control*. 2014.

[14] Shenxue Hao et al. "Distributed Cooperative Backpressure-Based Traffic Light Control Method". In: *Journal of Advanced Transportation* 2019 (Mar. 2019), pp. 1–14.

[15] J.J. Henry and J.L. Farges. "PRODYN". In: *Control, Computers, Communications in Transportation*. Ed. by J.-P. PERRIN. IFAC Symposia Series. Oxford: Pergamon, 1990, pp. 253–255.

[16] PB Hunt et al. *SCOOT-a traffic responsive method of coordinating signals*. Tech. rep. 1981.

[17] S.A.C.S. Jayasooriya and Y.M.M.S. Bandara. "Measuring the Economic costs of traffic congestion". In: *2017 Moratuwa Engineering Research Conference (MERCon)*. 2017, pp. 141–146.

[18] R.W. Jr, E. Curtis, and P. Olson. "The national traffic signal report card". In: 82 (June 2012), pp. 22–26.

[19] Tung Le et al. *Decentralized Signal Control for Urban Road Networks*. 2014.

[20] Xiaoyuan Liang et al. "A Deep Reinforcement Learning Network for Traffic Light Cycle Control". In: *IEEE Transactions on Vehicular Technology* 68.2 (Feb. 2019), pp. 1243–1253.

[21] Wei-Hua Lin and Chenghong Wang. "An enhanced 0-1 mixed-integer LP formulation for traffic signal control". In: *IEEE Transactions on Intelligent Transportation Systems* 5.4 (2004), pp. 238–245.

[22] Hong K. Lo, Elbert Chang, and Yiu Cho Chan. "Dynamic network traffic control". In: *Transportation Research Part A: Policy and Practice* 35.8 (2001), pp. 721–744.

[23] PR Lowrie. "Scats, sydney co-ordinated adaptive traffic system: A traffic responsive method of controlling urban traffic". In: (1990).

[24] Pedro Mercader, Wasim Uwayid, and Jack Haddad. "Max-pressure traffic controller based on travel times: An experimental analysis". In: *Transportation Research Part C: Emerging Technologies* 110 (2020), pp. 275–290.

[25] Ted Moskovitz et al. *ReLOAD: Reinforcement Learning with Optimistic Ascent-Descent for Last-Iterate Convergence in Constrained MDPs*. 2023.

[26] Michael Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Vol. 3. Jan. 2010.

[27] Tobias Pohlmann and Bernhard Friedrich. "Online control of signalized networks using the Cell Transmission Model". In: *13th International IEEE Conference on Intelligent Transportation Systems*. 2010, pp. 1–6.

[28] Mădălin-Dorin Pop. "Traffic Lights Management Using Optimization Tool". In: *Procedia - Social and Behavioral Sciences* 238 (2018), pp. 323–330.

[29] *PTV Vissim — myptv.com*. `https://www.myptv.com/de/mobilitaetssoftware/ptv-vissim`. [Accessed 17-08-2023].

[30] Syed Shah Sultan Mohiuddin Qadri, Mahmut Ali Gökçe, and Erdinç Öner. "State-of-art review of traffic signal control methods: challenges and opportunities". In: *European Transport Research Review* 12.1 (Sept. 2020), p. 55.

[31] Thomas Riedel and Monica Menendez. "7 - Switzerland". In: *Global Practices on Road Traffic Signal Control*. Ed. by Keshuang Tang et al. Elsevier, 2019, pp. 99–115.

[32] *SUMO Documentation — sumo.dlr.de*. `https://sumo.dlr.de/docs/index.html`. [Accessed 17-08-2023].

[33] Xiaotong Sun and Yafeng Yin. "A Simulation Study on Max Pressure Control of Signalized Intersections". In: *Transportation Research Record* 2672.18 (2018), pp. 117–127.

[34] Xiaotong Sun and Yafeng Yin. "A Simulation Study on Max Pressure Control of Signalized Intersections". In: *Transportation Research Record: Journal of the Transportation Research Board* 2672 (July 2018), p. 036119811878684.

[35] L. Tassiulas and A. Ephremides. "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks". In: *IEEE Transactions on Automatic Control* 37.12 (1992), pp. 1936–1948.

[36] *The Leader in Decision Intelligence Technology - Gurobi Optimization — gurobi.com*. `https://www.gurobi.com`. [Accessed 15-08-2023].

[37] Stelios Timotheou, Christos G. Panayiotou, and Marios M. Polycarpou. "Towards distributed online cooperative traffic signal control using the cell transmission model". In: *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 2013, pp. 1737–1742.

[38] Ishu Tomar, Indu Sreedevi, and Neeta Pandey. "State-of-Art Review of Traffic Light Synchronization for Intelligent Vehicles: Current Status, Challenges, and Emerging Trends". In: *Electronics* 11.3 (2022).

[39] Department for Transport. "National Road Traffic Projections 2022". In: (2022).

[40] Pravin Varaiya. "Max pressure control of a network of signalized intersections". In: *Transportation Research Part C: Emerging Technologies* 36 (2013), pp. 177–195.

[41] Hua Wei et al. "PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network". In: *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. New York, NY, USA: Association for Computing Machinery, 2019, pp. 1290–1298.

[42] Hua Wei et al. "Recent Advances in Reinforcement Learning for Traffic Signal Control: A Survey of Models and Evaluation". In: *SIGKDD Explor. Newsl.* 22.2 (Jan. 2021), pp. 12–18.

[43] Tichakorn Wongpiromsarn et al. "Distributed Traffic Signal Control for Maximum Network Throughput". In: *Conference Record - IEEE Conference on Intelligent Transportation Systems* (May 2012).

[44] Nan Xiao et al. "Pressure releasing policy in traffic signal control with finite queue capacities". In: *53rd IEEE Conference on Decision and Control*. 2014, pp. 6492–6497.

[45] Huichu Zhang et al. "CityFlow: A Multi-Agent Reinforcement Learning Environment for Large Scale City Traffic Scenario". In: *The World Wide Web Conference*. ACM, May 2019.

[46] Liang Zhang et al. "Expression might be enough: representing pressure and demand for reinforcement learning based traffic signal control". In: *Proceedings of the 39th International Conference on Machine Learning*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2021, pp. 26645–26654.

[47] Guanjie Zheng et al. *Diagnosing Reinforcement Learning for Traffic Signal Control*. 2019.

[48] Guanjie Zheng et al. "Learning Phase Competition for Traffic Signal Control". In: *CoRR* abs/1905.04722 (2019).

Appendix A

Proof of Theorem 4.1.1

We will first prove part (b). Using (4.6) and taking the expectation of (4.7) yields:

$$\mathbb{E}\{L(\mathbf{Q}(\tau+1))\} - \mathbb{E}\{L\mathbf{Q}(\tau)) \leq B - \epsilon \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\{|Q_{l,m}(t)|\} \tag{A.1}$$

Next, we take the sum over $\tau = \{0, 1, \ldots, t-1\}$ for some slot $t > 0$ and use the law of telescoping sums, which yields:

$$\mathbb{E}\{L(\mathbf{Q}(t))\} - \mathbb{E}\{L(\mathbf{Q}(0)) \leq Bt - \epsilon \sum_{\tau=0}^{t-1} \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\{|Q_{l,m}(t)|\} \tag{A.2}$$

as the remaining terms on the left-hand side cancel out. If we assume that $\epsilon > 0$, we can divide by $t \cdot \epsilon$, rearrange terms and use the fact that $\mathbb{E}\{L(\mathbf{Q}(0))\} \geq 0$ yields:

$$\frac{1}{t}\sum_{\tau=0}^{t-1} \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\{|Q_{l,m}(\tau)|\} \leq \frac{B}{\epsilon} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{\epsilon \cdot t} \tag{A.3}$$

This relation holds for all slots $t > 0$ and taking the limit as $t \to \infty$ proves part (b).

To prove part (a), we have from (A.2) that for all slots $t > 0$:

$$\mathbb{E}\{L(\mathbf{Q}(t))\} - \mathbb{E}\{L(\mathbf{Q}(0))\} \leq Bt \tag{A.4}$$

since $\mathbb{E}\{|Q_{l,m}(\tau)|\} \geq 0$. Using the definition of $L(\mathbf{Q}(t))$ yields:

$$\frac{1}{2}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} \mathbb{E}\{Q_{l,m}(t)^2\} \leq \mathbb{E}\{L(\mathbf{Q}(0))\} + Bt \tag{A.5}$$

Since the variance of $|Q_{l,m}(t)|$ cannot be negative, we have $\mathbb{E}\{Q_{l,m}(t)^2\} \geq \mathbb{E}\{|Q_{l,m}(t)|\}^2$. Therefore, for all $(l, m) \in \mathcal{L}$ and $t > 0$, we have:

$$\mathbb{E}\{Q_{l,m}(t)\} \leq \sqrt{2\mathbb{E}\{L(\mathbf{Q}(0))\} + 2Bt} \tag{A.6}$$

Dividing by t and taking a limit as $t \to \infty$ proves that:

$$\lim_{t\to\infty} \frac{\mathbb{E}\{|Q_{l,m}(t)|}{t} \leq \lim_{t\to\infty} \sqrt{\frac{2\mathbb{E}\{L(\mathbf{Q}(0))\}}{t^2} + \frac{2B}{t^2}} = 0 \tag{A.7}$$

Thus, all queues $Q_{l,m}(t)$ are mean rate stable, proving part (a).

Appendix B

Proof of Theorem 4.1.2

Fix any slot τ. Because (4.10) holds for this slot, we can take expectations of both sides and use the law of iterated expectations to yield:

$$\mathbb{E}\{L(\mathbf{Q}(\tau+1))\} - \mathbb{E}\{L(\mathbf{Q}(\tau))\} + V\mathbb{E}\{p(\tau)\} \leq B + Vp^* - \epsilon \sum_{\substack{l\in\mathcal{L} \\ m\in Out_l}} \mathbb{E}\{|q_{l,m}(\tau)|\} \tag{B.1}$$

Summing over $\tau \in \{0, 1, \ldots, t-1\}$ for some $t > 0$ and using the law of telescopic sums yields:

$$\mathbb{E}\{L(\mathbf{Q}(t))\} - \mathbb{E}\{L(\mathbf{Q}(0))\} + V\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\} \leq (B+Vp^*)t - \epsilon\sum_{\tau=0}^{t-1}\sum_{\substack{l\in\mathcal{L} \\ m\in Out_l}} \mathbb{E}\{|q_{l,m}(\tau)|\} \tag{B.2}$$

Rearranging terms and neglecting non-negative terms when appropriate, we can show that the above inequality directly implies the following two inequalities for all $t > 0$:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\} \leq p^* + \frac{B}{V} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{Vt} \tag{B.3}$$

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{\substack{l\in\mathcal{L} \\ m\in Out_l}} \mathbb{E}\{|q_{l,m}(\tau)|\} \leq \frac{B+V(p^*-p_{min})}{\epsilon} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))}{\epsilon t} \tag{B.4}$$

where we used that $\mathbb{E}\{L(\mathbf{Q}(t))\} \geq 0$ for any t and $V\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\} \geq Vtp_{min}$. Taking limits of the above as $t \to \infty$ proves (4.11) and (4.12).

Rearranging (B.2) also yields:

$$\mathbb{E}\{L(\mathbf{Q}(t))\} \leq \mathbb{E}\{L(\mathbf{Q}(0))\} + (B + V(p^* - p_{min}))t \tag{B.5}$$

from which mean rate stability follows by an argument similar to that given in the proof of Theorem 4.1.1).

Appendix C

Proof of Theorem 4.1.5

From Lemma 4.1.4, we have that for any controller it holds that:

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(t)|\mathbf{\Theta}(t)\} \geq B + V\mathbb{E}\{p^*(t)|\mathbf{\Theta}(t)\} + \sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} q_{l,m}(t)\mathbb{E}\{a^*_{l,m}(t) - b^*_{l,m}(t)|\mathbf{Q}(t)\} \quad \text{(C.1)}$$

where $a^*_{l,m}, b^*_{l,m}, p^*$ are the resulting arrival, service and penalty values under any ω-only decision $s^*(t) \in \mathcal{A}_{\omega(t)}$. Now fix $\delta > 0$, and consider the ω-only policy $\phi^*(t)$ that yields (4.35) and (4.20). Because this is an ω-only policy, and $\omega(t)$ is i.i.d. over slots, the resulting values of $a^*_{l,m}(t), b^*_{l,m}(t)$ are independent of the current queue backlogs $\mathbf{Q}(t)$, and we have from (4.35) and (4.20):

$$\mathbb{E}\{p^*(t)|\mathbf{Q}(t)\} \leq p_{max} \quad \text{(C.2)}$$

$$\mathbb{E}\{a^*_{l,m}(t) - b^*_{l,m}(t)|\mathbf{Q}\} = \mathbb{E}\{a^*_{l,m}(t) - b^*_{l,m}(t)\} \leq \delta \qquad \begin{matrix}\forall l \in \mathcal{L}\\ \forall m \in \mathcal{D}_l\end{matrix} \quad \text{(C.3)}$$

Plugging these into the right-hand-size of (C.1) and taking $t \to 0$ yields:

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(t)|\mathbf{Q}(t)\} \leq B + Vp_{max} \quad \text{(C.4)}$$

This is in the exact form for application of the Lyapunov Optimization Theorem (Theorem 4.1.2). Hence, all queues are mean rate stable. Further, from the above drift expression, we have for any $t > 0$ from (4.11):

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\} \leq p_{max} + \frac{B}{V} + \frac{\mathbb{E}\{L(\mathbf{Q}(0))\}}{Vt} \quad \text{(C.5)}$$

which proves part (a) by taking a lim sup as $t \to \infty$.

To prove part (c), we assume that Slater's assumption holds. Plugging the ω-only policy that yield (4.25) and (4.26) into the right-hand-side of the drift bound (C.1) yields:

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{p(t)|\mathbf{Q}(t)\} \leq B + V\Psi(\epsilon) - \epsilon\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}} Q_{l,m}(t) \quad \text{(C.6)}$$

Taking iterated expectations, summing the telescopic series, and rearranging terms as usual yields:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_l}}\mathbb{E}\{Q_{l,m}(\tau)\} \leq \frac{B+V(\Psi(\epsilon)-\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\}}{\epsilon}+\frac{\mathbb{E}\{L(\mathbf{Q}(0))}{\epsilon t} \tag{C.7}$$

However, because our algorithm satisfies all the desired constraints of the optimization problem (4.18) - (4.19), its limiting time average expectation for $p(t)$ cannot be better than:

$$\liminf_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{p(\tau)\} \geq p_{min}=0 \tag{C.8}$$

Taking the limit of (C.7) as $t\to\infty$ and using (C.8) yields:

$$\limsup_{t\to\inf}\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{\substack{l\in\mathcal{L}\\ m\in\mathcal{D}_L}}\mathbb{E}\{Q_{l,m}(\tau)\} \leq \frac{B+V\Psi(\epsilon)}{\epsilon} \tag{C.9}$$

Appendix D

Queue Model Validation (Fine Network)

For the sake of completeness, we provide the figures below to illustrate the validation of the queue model for the fine network.

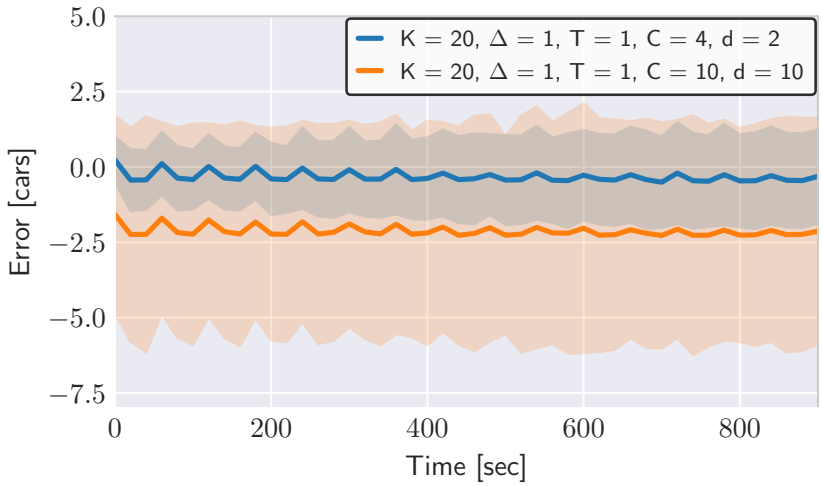


Fig. D.1: The mean and standard deviation of the prediction error of simulations with two different capacity and exogenous flows simulated in the fine network. A lower flow increases the accuracy of the model overall.

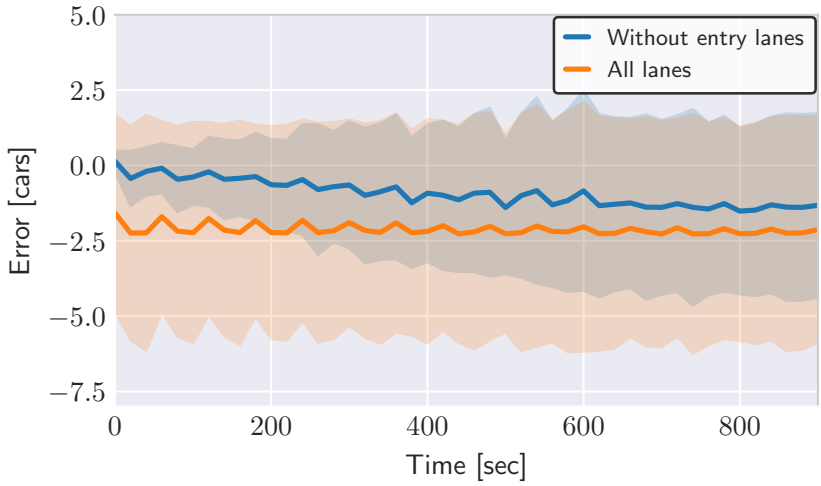


Fig. D.2: The mean and standard deviation of the prediction error of simulations with and without considering entry lanes in the fine network. The error decreases as we do not consider the entry lanes.

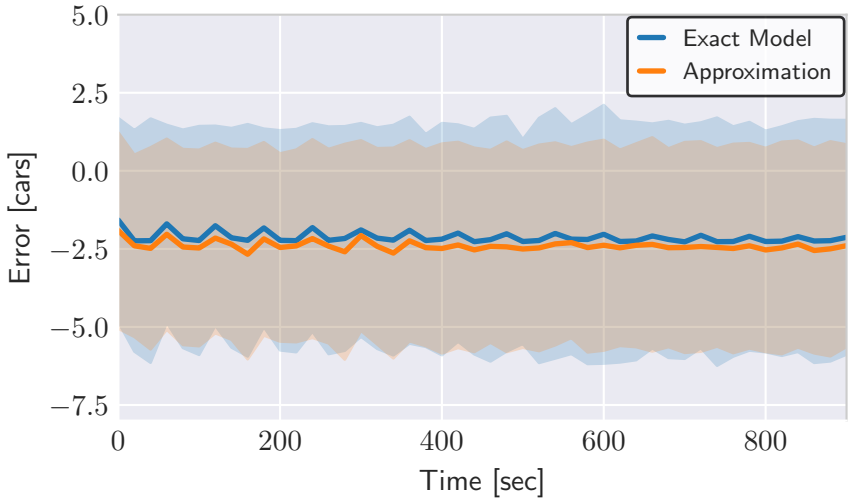


Fig. D.3: The mean and standard deviation of the prediction error of simulations with and without approximating the queue model in the fine network. The approximation is slightly worse than the original model.

Appendix E

Simulation Parameters

Centralized Controller	α	C	d	q_{max}
Fine Network	0.6	10	10	30
Coarse Network	0.3	6	5	40

Tab. E.1: The simulation parameters for the centralized controller.

Binary Penalty (ADMM)	V_1	V_2	V_3	L	ρ	q_{thresh}	W	C
Fine Network	2	1	0.5	7	0.5	25	30	15
Coarse Network	0	0	0.1	20	0.5	25	30	15

Tab. E.2: The simulation parameter for ADMM with the binary penalty.

Continuous Penalty (ADMM)	V_1	ρ	W
Fine Network	10	8	30
Coarse Network	0.5	2	30
Large Network	1	1	30

Tab. E.3: The simulation parameter for ADMM with the continuous penalty.

Continuous Penalty (Greedy)	V_1
Fine Network	10
Coarse Network	0.5
Large Network	0.5

Tab. E.4: The simulation parameter for the Greedy algorithm with the continuous penalty.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Declaration of originality

The signed declaration of originality is a component of every semester paper, Bachelor's thesis, Master's thesis and any other degree paper undertaken during the course of studies, including the respective electronic versions.

Lecturers may also require a declaration of originality for other written papers compiled for their courses.

I hereby confirm that I am the sole author of the written work here enclosed and that I have compiled it in my own words. Parts excepted are corrections of form and content by the supervisor.

Title of work (in block letters):

Authored by (in block letters):
For papers written by groups the names of all authors are required.

Name(s):	**First name(s):**

With my signature I confirm that

- I have committed none of the forms of plagiarism described in the 'Citation etiquette' information sheet.
- I have documented all methods, data and processes truthfully.
- I have not manipulated any data.
- I have mentioned all persons who were significant facilitators of the work.

I am aware that the work may be screened electronically for plagiarism.

Place, date	**Signature(s)**

For papers written by groups the names of all authors are required. Their signatures collectively guarantee the entire content of the written paper.