UNIVERSIDADE FEDERAL FLUMINENSE

## MARCELO ANDRADE RODRIGUES D'ALMEIDA

# Matching Goals and Rewards for Reinforcement Learning Traffic Signal Control

NITERÓI 2021

### UNIVERSIDADE FEDERAL FLUMINENSE

### MARCELO ANDRADE RODRIGUES D'ALMEIDA

## Matching Goals and Rewards for Reinforcement Learning Traffic Signal Control

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Orientadora: Aline Marins Paes Carvalho

> Coorientador: Daniel Mossé

> > NITERÓI 2021

#### Ficha catalográfica automática - SDC/BEE Gerada com informações fornecidas pelo autor

D148m d'Almeida, Marcelo Andrade Rodrigues Matching goals and rewards for reinforcement learning traffic signal control / Marcelo Andrade Rodrigues d'Almeida ; Aline Marins Paes Carvalho, orientadora ; Daniel Mossé, coorientador. Niterói, 2021. 101 f. Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2021. D0I: http://dx.doi.org/10.22409/PGC.2021.m.06070084748 1. Aprendizado de maquina. 2. Inteligencia artificial. 3. Produção intelectual. I. Carvalho, Aline Marins Paes, orientadora. II. Mossé, Daniel, coorientador. III. Universidade Federal Fluminense. Instituto de Computação. IV. Título. CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

### MARCELO ANDRADE RODRIGUES D'ALMEIDA

Matching Goals and Rewards for Reinforcement Learning Traffic Signal Control

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação

Aprovada em Setembro de 2021.

BANCA EXAMINADORA

Profa. ALINE MARINS PAES CARVALHO, D.Sc. -

Orientadora, UFF

Prof. DANIEL MOSSÉ, Ph.D. - Coorientador, University of Pittsburgh

Gabriel Ramos

Prof. GABRIEL DE OLIVEIRA RAMOS, D.Sc. - Unisinos

de Oli-

Prof. DANIEL CARDOSO MORAES DE OLIVEIRA, D.Sc. - UFF

> Niterói 2021

# Agradecimentos

Agradeço aos meus orientadores, Aline e Daniel, por toda a paciência e contribuição dedicada ao trabalho. Agradeço também ao time de desenvolvimento do simulador de tráfego Sumo, o qual foi muito prestativo em diversas ocasiões, tirando dúvidas e consertando bugs.

# Resumo

Nos últimos anos, diversos estudos propuseram diferentes métodos e formulações de Aprendizado por Reforço (*Reinforcement Learning*, RL) para o problema de Otimização de Sinais de Trânsito (Traffic Signal Control, TSC), apresentando resultados promissores e soluções de semáforos flexíveis. Esses estudos geralmente decidem otimizar o tempo de viagem como objetivo. No entanto, o tempo de viagem tem algumas deficiências cruciais: não é facilmente decomposto em recompensas; impede a análise em um momento qualquer da simulação, exceto no final; produz resultados irreais para situações de Deadlock (interbloqueio, blocagem, impasse) e Starvation (inanição). Neste trabalho, usamos o framework Frap, estado-da-arte para RL TSC, para testar nossas propostas de objetivo baseado na perda de tempo, a saber, perda de tempo percentual por motorista e perda de tempo acumulada por motorista, para resolver as deficiências do tempo de viagem como objetivo. Mostramos também que melhorar a perda de tempo implica em melhorar o tempo de viagem e que existe uma relação direta entre o *objetivo* e a *recompensa* dos agentes. Os resultados experimentais apontam que a proposta de formulação de RL baseada em perda de tempo melhora a economia de tempo em 6% quando comparada a outras formulações do estado da arte comumente adotadas. Além disso, a maioria das soluções propostas (incluindo Frap) usa métodos livres de modelo em vez de métodos baseados em modelo (planejamento), com base em sua mecânica mais simples e resultados eficazes. No entanto, RL baseada em modelo pode melhorar a economia de tempo e a convergência do aprendizagdo, através da continuação das interações com os estados encontrados (isto é, percorrendo caminhos futuros), reunindo mais amostras de aprendizado e experiências diversas. Avaliamos os benefícios do uso de planejamento para o controle de sinais de trânsito. Nos experimentos realizados, o uso de planejamento produziu uma convergência consideravelmente mais rápida (até 34%), porém mais volátil após este ponto.

**Palavras-chave**: Otimização de Sinal de Trânsito, Sistemas de Transporte Inteligentes, Aprendizado por Reforço, Planejamento.

## Abstract

In recent years, several studies proposed different Reinforcement Learning (RL) methods and formulations for the Traffic Signal Control (TSC) problem, presenting promising results and flexible traffic light solutions. These studies generally decide for optimizing travel time as the objective. However, travel time has some crucial shortcomings: it is not easily decomposable into rewards; it hinders analysis at any simulation time but the very end; it produces unrealistic results for deadlock and starvation situations. In this work, we use the state-of-the-art RL TSC framework Frap to test our propositions of objectives based on time loss, namely percentage time loss per driver and cumulative time loss per driver, to address travel time shortcomings. We also show that improving time loss implies improving travel time and that there is a direct relationship between the time loss *objective* and the agents' *reward*. Our experimental results point out that our time loss-based RL formulation improves the time savings by 6% when compared to other commonly-adopted state-of-the-art formulations. Additionally, most of the proposed solutions (including Frap) use model-free instead of model-based methods, based on their simpler mechanics and effective results. However, model-based RL can improve time savings and learning convergence by further interacting with states (*i.e.*, walking future paths), gathering more learning samples and diverse experiences. We assess the benefits of using Planning (*i.e.*, model-based RL) for Traffic Signal Control. In our tests, the use of Planning produced a considerable (up to 34%) faster convergence at the beginning, but more volatile learning thereafter.

**Keywords**: Traffic Signal Control, Intelligent Transportation Systems, Reinforcement Learning, Planning.

# List of Figures

2.1	Reinforcement Learning process	11
2.2	Reinforcement Learning regular action-selection and learning sample gen- eration	16
2.3	RL Planning (decision-time planning) regular action-selection and learning sample generation	17
2.4	RL Planning (background planning) regular action-selection and learning sample generation	18
2.5	PlanningOnly's planning process	19
2.6	Overall Frap's learning process. 1. For training, Frap runs three simula- tions in parallel. 2. The tuples $s_t, a_t, s_{t+1}, r_{t+1}$ are stored in the memory at every instant t. 3. At the end of the training round, the stored learning samples are used to update the neural network that represents the policy. 4. The testing rounds generate the objective data	20
4.1	Synthetic scenarios' movement and phase definition	29
5.1	The scenarios	37
5.2	Regular policy (no WTR) phases and demand distribution	39
5.3	WTR policy phases and demand distribution	40
5.4	Single-Intersection (8 phases) learning process	49
5.5	Single-Intersection (14 phases) learning process	50
5.6	Multi-Intersection (8 phases) learning process	51
5.7	Multi-Intersection (14 phases) learning process	52
5.8	Planning RLF1 (sample of 1000) results	65
5.9	Planning RLF5 (sample of 1000) results	66

5.10	Planning RLF6 (sample of 1000) results	67
5.11	Planning RLF1 (sample of 10000) results	68
5.12	Planning RLF5 (sample of 10000) results	69
5.13	Planning RLF6 (sample of 10000) results	70

# List of Tables

5.1	Waiting Time Restriction experiment (WTR on and off)	41
5.2	Phase set definition (conflicting-movements phases highlighted)	43
5.3	Phase set configuration results	44
5.4	RL formulations tested	44
5.5	Time loss-based objectives results for the single-intersection scenario	45
5.6	Time loss-based objectives results for the multi-intersection scenario	46
5.7	RL formulation convergence and stability results for the single-intersection scenario	47
5.8	RL formulation convergence and stability results for the multi-intersection scenario	48
5.9	Other objectives' results for the single-intersection scenario $\ldots \ldots \ldots$	53
5.10	Other objectives' results for the multi-intersection scenario $\ldots \ldots \ldots$	53
5.11	Pending vehicles impact on metrics for the single-intersection scenario	54
5.12	Pending vehicles impact on metrics for the multi-intersection scenario $\ . \ .$	54
5.13	Time loss-based objectives results for 'Planning Only'	55
5.14	Different RL formulation results for 'Planning Only'	56
5.15	Types of planning and action sampling tested	58
5.16	Types of planning and action sampling policy results	59
5.17	Sample size based on action sampling size and number of iterations $\ldots$ .	59
5.18	Sample size and policy update time	60
5.19	Action sampling size, number of iterations, and sample size results	62
5.20	RL formulations tested	62

5.21	RL formulation results	64
A.1	Frap's default hyperparameters	81
A.2	Planning exclusive parameters	81
B.1	Time loss-based objectives results (single-intersection scenario - best round)	83
B.2	Time loss-based objectives results (multi-intersection scenario - bbest round)	83
B.3	Other objectives' results for the single-intersection scenario (best round) $\ .$	84
B.4	Other objectives' results for the multi-intersection scenario (best round)	84
B.5	Pending vehicles impact on metrics for the single-intersection scenario (best round)	85
B.6	Pending vehicles impact on metrics for the multi-intersection scenario (best round)	85

# List of Abbreviations and Acronyms

DRL	:	Deep Reinforcement Learning;
GCNN	:	Graph Convolutional Neural Networks;
LSTM	:	Long Short-Term Memory;
RNN	:	Recurrent Neural Networks;
RL	:	Reinforcement Learning;
TSC	:	Traffic Signal Control;
TTI	:	Texas A&M Transportation Institute;

# Contents

1	Introduction					
<b>2</b>	Bac	Background				
2.1 Traditional Traffic Signal Control						
	2.2	Machine Learning	9			
		2.2.1 Reinforcement Learning	10			
		2.2.1.1 Components	11			
		2.2.2 Deep Reinforcement Learning	15			
		2.2.3 Planning	16			
		2.2.3.1 "Search-only" Planning	17			
	2.3	Frap	17			
3	Rel	elated Work 2				
	3.1	Objective	22			
	3.2	RL Formulation	23			
4	Pro	blem Definition	<b>24</b>			
	4.1	Objective	24			
		4.1.1 Time Loss and Travel Time Relationship	27			
	4.2	Movements and Phases	28			
	4.3	Problem Modeling	29			
		4.3.1 State	29			
		4.3.2 Action	31			

		4.3.3	Reward	31
		4.3.4	Policy	32
			4.3.4.1 Excessive Waiting Time	33
			4.3.4.2 Starvation	33
		4.3.5	Environment	34
			4.3.5.1 Deadlock	34
<b>5</b>	$\mathbf{Exp}$	oerime	nts	36
	5.1	Scenar	rios	36
	5.2	Baseli	nes	36
	5.3	Non-P	Planning Experimental Results	38
		5.3.1	Waiting Time Restriction	38
		5.3.2	Phase Set Configuration	41
		5.3.3	RL Formulation	42
			5.3.3.1 Average Travel Time Objective Comparison	46
			5.3.3.2 Impact of Pending Vehicles on Metrics	50
	5.4	Planning Experimental Results		
		5.4.1	Planning Only	55
		5.4.2	Types of Planning and Action Sampling Policy	57
		5.4.3	Action Sampling Size, Number of Iterations, and Sample Size $\ . \ . \ .$	59
		5.4.4	RL Formulation and Planning	61
6	Con	nclusio	n	71
	6.1	Limita	ations	72
	6.2	Future	e Work	73
Re	efere	nces		<b>74</b>

Appendix A – List of Parameters and Hyperparameters					
A.1	Environment	80			
A.2	Agent	80			

### Appendix B – RL Formulation Results: Rolling Average Window's Best Round 82

## Chapter 1

# Introduction

The Traffic Congestion Problem Overcrowding and congestion on urban transportation systems are ongoing concerns for governments worldwide. Such concern requires comprehensive studies to assess the causes and possible solutions. One of the studies, the INRIX 2019 Global Traffic Scorecard [41], evaluated more than 970 cities globally, covering 29 countries from all continents. The study consolidates how many extra hours the drivers spend per year during peak hours compared to free-flow traffic (*i.e.*, vehicles traveling without delays). The INRIX's Impact Rank measures the impact of congestion by normalizing the time lost by the city's population. The top five include Bogota, Colombia, which reached 191 hours per driver a year, followed by Rio de Janeiro, Brazil (190), Mexico City, Mexico (158), Istanbul, Turkey (153), and São Paulo, Brazil (152). If considered the time loss alone, the top five includes Rome, Italy (166), Paris, France (165), and Belo Horizonte, Brazil (160), as third to fifth positions.

The study also analyzes the financial costs of congestion in the U.S., UK, and Germany both per driver and nationally for non-freight travels. In the United States (286 cities analyzed), the average driver lost 99 hours, costing them \$1,377 each and \$88 billion nationally. For the European countries, 115 hours average for the UK (101 cities), costing respectively \$1,162 and \$8.9 billion, and 46 hours average for Germany (74 cities), costing \$412 and \$3.1 billion, respectively.

Taking a more in-depth look at the United States situation, the Texas A&M Transportation Institute (TTI)'s 2019 Urban Mobility Report<sup>1</sup> [46] presented how traffic congestion has been increasing over the years for 494 U.S. urban areas. The yearly delay had increased by 14% and congestion cost by 19% nationally in five years (from 2012 to

<sup>&</sup>lt;sup>1</sup>Developed in partnership between TTI and INRIX.

2017) and has experienced a steady increase since  $1982^2$ . The total cost for 2017, including freight travels, was estimated at 179 billion dollars nationally. The usual (passenger travel) drivers spent an average of 54 hours and \$1,080 yearly in congestion<sup>3</sup>. Note that these expenses are usually measured by the median hourly wage [41, 46] and the extra fuel consumption [46] cost. The costs are far greater if reflected indirect effects on personal and business perspectives (*e.g.*, cost of opportunity, supply chain efficiency, quality of life, air and noise pollution, and others).

Another critical finding by TTI is that congestion has grown in areas of all sizes and that the growth trend itself has also increased. From 2000 to 2010, congestion increased 25% in small areas (less than 500,000 citizens), 13% in medium areas (500,000 to 1 million citizens), 10% in large areas (1 million to 3 million citizens), and 18% in very large areas (more than 3 million citizens). From 2010 to 2017, congestion in small areas increased by 23% (33% if linearly adjusted to the same ten years period), in medium areas by 22% (32% adjusted), large areas by 26% (37% adjusted), and very large areas by 26% (37% adjusted) [46].

Furthermore, congestion levels are often associated with economic growth [41, 46]. Except for the 2008/2009 U.S. economic recession, in which the traffic has improved slightly, the congestion has grown in every other period [46]. Projecting for 2025 (under the assumption of no recession in the period), the 2019 TTI study indicates national congestion cost reaching \$237 billion (a 32% increase), the cost for the driver up to \$1280 (a 19% increase), and the time loss to 62 hours (a 15% increase) [46]<sup>4</sup>. They also warn that traffic congestion can contribute to the potential reduction of further economic growth [46].

**Causes and Solutions** The general cause for congestion growth results from the increasing disparity between travel demand and transportation capacity supply. The addition of residents and jobs and the increase in the number or length of trips for the existing population are factors of demand increases [46]. Regarding transportation increase in supply, each city has a unique set of traits such as topography, demographic density, and zoning policies (to name a few) that pose significant challenges to adequate network-wide infrastructure to ever-increasing traffic demand. Additionally, the historical

 $<sup>^{2}2017</sup>$  is the last year presented in the study, 1982 is the first.

<sup>&</sup>lt;sup>3</sup>Besides the year and number of cities differences, both studies also have different methodologies. Additionally, INRIX refined its methodology over the years, making it non-comparable with the INRIX Global Traffic Scorecard's past editions.

<sup>&</sup>lt;sup>4</sup>Dollars as valued in 2017.

context does affect how the authorities manage a city's mobility. For example, INRIX mentions that geography, age, and density can be the reasons why Boston, Chicago, and Philadelphia are the three most congested cities in the U.S. (149, 145, and 142 in time loss, respectively). Other good examples are the European cities, which grew before vehicles' widespread adoption, making the cities have denser centers, narrower roads, and more complex road structures [41]. Other cities may have worse city organization and lack of public transportation modes as historical problems.

To alleviate the congestion symptoms, cities such as London, Stockholm, and Singapore, for example, opted for charging congestion fees for vehicles to use certain parts of the urban network. New York City plans to implement the fee in 2021, and Chicago and Los Angeles already study the measure [11]. Other cities like São Paulo, Beijing, Bogota, Delhi, Mexico City, among others, adopt a license plate-based driving restriction, where certain vehicles (*e.g.*, selected by the first or last numbers in the license plate) cannot drive in some sections of the city on pre-specified weekdays [12]. Although possibly effective, these extreme solutions raise social concerns once those expenses do not affect the population equally: the richer can just pay the fees or even buy another vehicle [12]. Other solutions to fight the problem involve investments in infrastructure and public policies to encourage people to use public transportation and alternatives like walking and cycling. While better infrastructure and environmental-friendly solutions should naturally be the long-term goal for cities, adopting them requires considerable time, money, and effort to implement them correctly and educate the citizens.

Another popular idea, which is the focus of this dissertation, is to optimize the traffic light system. When it comes to controlling vehicle flow, traffic signal control (TSC) plays a fundamental role in the urban network organization. Arguably, such an optimized system would reduce the traffic bottleneck in some regions while looking almost transparent to the citizens.

The Traffic Signal Control (TSC) Problem The TSC problem consists of deciding the amount of time and the lanes of traffic the signal should prioritize. The traditional *static* formulation defines those variables in advance according to the historical demand distribution [28, 30, 43]. However, the demand can vary significantly depending on some unexpected conditions. Crashes, bad weather, special events, and other irregular congestion causes can add up to 70% in total travel time (*i.e.*, approximately 41% of the trip as time loss<sup>5</sup>) to trips [46]. Traditional adaptive approaches [33, 50] (*e.g.*, actuated signals)

 $<sup>^5\</sup>mathrm{As}$  a reference, 100% more travel time imply 50% of trip time loss.

usually stick to the original traffic time plans and only make small adjustments in traffic light duration. On the other hand, Reinforcement Learning (RL) [48] methods, which we study and adopt in this dissertation, bring more flexibility by learning the policy to control the traffic, aiming to adjust the traffic lights to the *current* traffic condition.

From the existing RL methods, Frap [68] and its derived methods [15, 52, 64] propose a relatively simple Deep Reinforcement Learning (DRL) architecture that achieves good results in large-scale real-world scenarios and can be combined with different state and reward formulations. Other studies focus on investigating explicit coordination, such as incorporating temporal and spatial influences of neighboring intersections into the agent decision [31, 51, 53] or joint action decision [1, 20, 61]; exploring policy-based and actorcritic methods [7, 17]; different neural network architectures, such as Recurrent Neural Networks (RNN) [51], Long short-term memory (LSTM) [1], and Graph Convolutional Neural Networks (GCNN) [38, 51]; the methods' generability, focusing on the transference of learning [64, 66]; and the presence of disruption events and pedestrian crossing [7].

These solutions, however, generally suggest the optimization of *travel time* as the objective [15, 31, 38, 51, 52, 53, 64, 68]. The main problem with travel time is that it is not easily decomposable into rewards, which is what effectively guides the RL agent to optimize it. Thus, the methods decouple reward formulations from the objective. In practice, this limits the travel time to serve as a comparison metric, not a directly optimizable goal. Also, travel time is subject to deadlock and starvation issues since, in the simulation, it requires vehicles to finish their trips, and it may need to be paired up with a throughput co-objective to be accurately assessed.

Moreover, according to a 2020 survey [54], published RL methods for the TSC problem explore model-free methods [2, 3, 5, 7, 8, 9, 10, 13, 14, 15, 16, 17, 19, 21, 25, 34, 37, 38, 39, 40, 45, 49, 52, 53, 55, 62, 63, 64, 69, 68] much more extensively than modelbased ones [26, 27, 29, 44, 47, 56, 57, 58]. The model-free solutions are usually easier to implement, as they do not require learning a model of the environment. However, the model-based approach can reduce the time to train the agents and possibly the overall time loss levels.

**Research Questions** This dissertation aims to evaluate whether a direct match of goals and rewards leads to better time savings and whether RL Planning would benefit traffic signal control. By adequately modeling state and reward, we want to assess how the direct association with the objective can improve convergence and time savings results.

For the Planning, we want to assess to what degree future experience can help the agent in the learning process.

**Contributions** In this dissertation, we enhance the usual RL formulation to TSC to contemplate a tied integration between the objective, the rewards, and the states. We propose a **time loss-based objective**, which is related to the travel time but without inheriting its problems. Furthermore, we propose two time loss-based state/reward configurations that leverage the RL formulation relationship with the proposed objective (something infeasible to the travel time objective). The time-loss RL formulations are evaluated through the traffic simulator Sumo [32] in single and multi-intersection scenarios and compared with static and RL baselines.

Additionally, we assess a model-based variant (Frap + Planning). Our insight is to use the simulator's state-saving mechanism to extrapolate future paths instead of learning the environment mechanics. This way, we can take advantage of planning but avoid the complexity of model-based approaches.

Our specific contributions in this dissertation are as follows<sup>6,7</sup>.

- We propose a time loss objective compelled by the advantages over the travel time as an assessment metric and the correlations between them.
- We propose and evaluate a time loss and a time loss pressure variant for the RL state/reward formulation. We show that the use of time loss-based formulations results in an up to 55% reduction in total time lost by vehicles.
- We present and evaluate a Frap model-based variant. By using planning, we achieved a considerable (up to 34%) faster convergence at the beginning, but results were more volatile.

**Outline** The rest of this dissertation is organized as follows. Chapter 2 covers the relevant background information to our analysis of the problem. We describe some methods used in traditional traffic control (both adaptive and static) and the learning aspect, presenting the theory behind Reinforcement Learning (RL) and Deep Reinforcement Learning (DRL), explaining their components applied to the TSC domain. Finally, we introduce Planning in RL and how it can benefit the TSC problem decision process.

<sup>&</sup>lt;sup>6</sup>This dissertation also produced an accepted paper named "Designing Reinforcement Learning Agents for Traffic Signal Control with the Right Goals: a Time-Loss based Approach" at ITSC 2021

<sup>&</sup>lt;sup>7</sup>The code is available at https://github.com/MeLLL-UFF/urban-semaphore-optimization

Chapter 3 presents an overview of RL formulations, methods, experiment settings, and objectives choices from the literature. We highlight the Frap method and its variations due to its proposition of being a simple but effective method, a good foundation for further experiments and improvements. We also present the directions other works have taken to tackle the TSC problem.

In Chapter 4, we detail the problem definition, describing the time loss objective, discussion around the movements and phases, the RL formulations of the Frap correlated methods, and the time loss-based proposition. We also discuss some policy and environment-related problems.

Chapter 5 describes the experiments and discusses the results. We also describe methods, scenarios, and configurations. The focus is to initially present exploratory tests. Later we test the RL formulations and the Frap with Planning. Chapter 6 concludes the dissertation, presenting the limitations, future work, and final remarks.

## Chapter 2

## Background

The most traditional approach for TSC is to use static methods, which define *fixed sig*naling plans according to the pre-anticipated traffic demand. These plans consist of describing fixed time intervals where each traffic signal remains green, yellow, or red. The plan's revisions occur periodically after a substantial change in the city dynamics. These characteristics mean that static methods are not usually flexible to changes in the traffic network (*i.e.*, the modeled city roads). Alternatively, though still based on the same fixed plan principle, the traditional adaptive methods (*e.g.*, actuated signals) bring some flexibility by allowing the traffic light to sense the traffic conditions and slightly adjust the plan to better accommodate fluctuations in the traffic.

A more elaborate approach, the *learning methods* propose discovering the city's dynamics and understanding which behavior is more suitable for each circumstance. They are also adaptive methods, but they do not rely on the descriptions of traffic plans. Instead, they learn the adaptation itself by finding rules of engagement (*i.e.*, policies) that better fit the traffic light.

In this chapter, we introduce the fundamental concepts addressed by this dissertation. Section 2.1 reviews the traditional traffic control (non-learning) methods. Section 2.2 addresses the learning perspective, which usually leverages techniques from Reinforcement Learning (RL) [48].

### 2.1 Traditional Traffic Signal Control

A traffic light is composed of selected phases and their organization. A phase is a combination of lights (*i.e.*, signalization that instructs the drivers how to proceed) for the intersection's existing paths.

The most recent survey [54] about applying RL for the TSC problem mentions some relevant traditional traffic light systems. They are also called *traditional* because they are not based on any AI technique. The survey categorizes them as either operating statically or adaptively. The survey covers those categories in greater detail. In this section, we present a quick summary.

Static methods define a fixed phase sequence, a fixed cycle length (total time for the phase sequence), and a fixed phase split (the division of time among the phases). When composed with other signals, a new characteristic is added: the offsets, which are the differences between the start of cycles of adjacent signals. For example, the Webster method determines a formula to compute the cycle length and phase split for a single intersection. For offset signal coordination, other methods must be used with Webster. The GreenWave method focuses on the offset of a single direction at a time, while MaxBand seeks to handle offsets in both directions. Both methods require the same cycle length definition for all intersections.

Adaptive methods rely on sensors to recognize some traffic characteristics and activate pre-specified rules that adapt the standard traffic light configuration. SCATS and Max-pressure are examples of adaptive solutions. SCATS decides which traffic signal configuration to use from a selection of pre-defined static signal plans. Alternatively, Max-pressure aims at minimizing the pressure of an intersection, defined by the difference between the number of vehicles in the entering and exiting lanes.

Considering the division between adaptive vs. non-adaptive traffic light methods, Webster [28], GreenWave [43], and MaxBand [30] are examples of non-adaptive (static) solutions. SCATS [33] and Max-pressure [50] are examples of adaptive systems.

The traffic simulator Sumo [32] also provides a static traffic light generation<sup>1</sup> and two types of built-in adaptive methods<sup>2</sup>, one based on time-gaps and the other on time-loss. As the names suggest, the former resorts to extending the current phase when the time gap between successive vehicles is below a threshold while the latter when there are any vehicles with time loss greater than a threshold. The time gap refers to the distance (in time) between two vehicles and the time loss the wasted time driving below the street's maximum speed.

<sup>&</sup>lt;sup>1</sup>https://sumo.dlr.de/docs/Simulation/Traffic\_Lights.html#automatically\_generated\_ tls-programs

<sup>&</sup>lt;sup>2</sup>https://sumo.dlr.de/docs/Simulation/Traffic\_Lights.html#actuated\_traffic\_lights

The challenge for the standard adaptive solutions (and even more for the static) is to define some configuration, rule, or policy that characterizes the complex dynamics found in the city traffic. Instead of listing every possibility and defining the best curse of action, learning methods discover how to discern them.

### 2.2 Machine Learning

General computer programming consists of developing step-by-step solutions in the form of algorithms to solve a problem. From the computer point-of-view, the user is providing the input and the algorithm. That is, all the machine work is described as the simple execution of the given instructions. However, when addressing complex problems, specifying a set of instructions for an exact or optimal solution might not be conceivable or viable.

Machine Learning [35, 4] was developed to help with this matter. Machine Learning involves discovering how to (approximately) solve a problem without having the instructions for it. As a way to make a correspondence with general programming, Machine Learning is often explained in a ludic manner: instead of having the algorithm responsible for producing the desired result, the user provides the input and output, and the computer becomes responsible for producing the algorithm that correlates one to another. In practice, to fulfill this depiction, a set of tools, strategies, and algorithms is required so that the computer can fill the gaps in the search for patterns that constitute a solution to the problem.

In the TSC problem, writing a program that produces valid configurations for signal timings is a manageable task. The nontrivial part is when the problem is approached by an optimization nature: automatically adapting the control to different traffic conditions without extensively listing all possible scenarios. In this case, as in many other Machine Learning applications, the objective is to achieve results presumably unattainable by preprogrammed systems.

Machine Learning is divided into three broad fields: Supervised Learning; Unsupervised Learning; and Reinforcement Learning. Each one of them introduces a different paradigm.

In **Supervised Learning**, the computer is provided with curated examples of inputs and output pairs (*i.e.*, the features and a label, respectively). The objective is to learn a generalization of the inputs-output relation and correctly predict the label for unseen data. In **Unsupervised Learning**, there are no labels available. The strategy consists of approaches to find patterns in the data. Finally, **Reinforcement Learning** consists of trial-and-error learning, where the agent continuously interacts with an environment in a sequential decision setting. A reward signal guides the learning by providing feedback about the agent's decision-making.

As TSC is essentially a sequential decision problem, the RL approach is a more suitable candidate to tackle the problem. Next, we show the main concepts related to RL.

#### 2.2.1 Reinforcement Learning

Reinforcement Learning (RL) [48] consists of discovering how to map situations to actions in a sequential decision-making environment towards an objective. Basically, an agent perceives an observation o (with a partial or full information from the state s) of the environment and chooses an action a. This action may modify the environment, which also changes according to its own mechanics. As a consequence, the agent receives feedback for its action, in the form of a numerical reward signal r, so it can understand how good it was. Finally, the agent perceives a new state, repeating the above process until the end of an episode, or indefinitely. Several episodes are necessary so the agent can shape its understanding of how the action options relate to each observation in terms of expected reward collection.

The agent aims to maximize the cumulative reward it receives over time by learning a policy  $\pi$  that better fits the actions for the encountered states. The agent must explore the consequences of its actions to increase its knowledge of how to achieve better rewards, and, at the same time, it must also exploit the actions whose rewards are already known. In other words, if it explores too much, it misses the opportunity to exploit and maximize the return; if it explores too little, it does not unlock better reward possibilities. This exploration-exploitation trade-off is challenging to balance, as the sequence of actions influences the future states and, consequently, future rewards. When working with RL, one assumes that the *state*, *action*  $\rightarrow$  *state'* transitions are not available to the agent and must be inferred. Figure 2.1 illustrates the whole process.

In Section 2.2.1.1, the RL components are explained in greater detail. Some notions are provided of how one can model a RL solution according to the TSC problem's characteristics and objectives.



Figure 2.1: Reinforcement Learning process

#### 2.2.1.1 Components

**Objective** In general, the objective is entirely decomposed into the agent's feedback, and maximizing the reward (see Reward and Return, for more details) becomes identical to accomplish the objective. In other cases, as in Multi-Agent Reinforcement Learning problems, the objective might be easily described and measured but not sufficiently characterized by the agent's rewards. The TSC problem is one of the latter cases. The overall objective is beyond any individual agent's scope and is measured based on the entire environment.

A possible objective choice for the TSC problem could be to reduce the vehicles' waiting time in the entire traffic network. Note that the waiting time is a property of all the vehicles, even those that may not be currently in any agents' state, and also that optimizing all the agent's return does not necessarily mean to maximize the objective. For example, in a city grid, operational efficiency in a region (*e.g.*, at the borders or at the center of the city) increases the traffic load at the subsequent intersections, possibly reducing the reward the neighboring would receive otherwise.

**Agent and Environment** The agent interacts with the environment over a sequence of discrete time-steps t, where at each step, the agent perceives a state  $s_t$ , performs an action  $a_t$ , perceives (in the next step t + 1) the resulting state  $s_{t+1}$  and receives the respective reward  $r_{t+1}$ .

For the TSC problem, the environment is the traffic dynamics (network + demand), and the agents are the intersections' signal control modules. The network constitutes the road network itself: the configuration and disposition of streets and the intersection conversion possibilities. The demand comprises the distribution of vehicles traversing the traffic network over time. Each traffic signal perceives the traffic configuration in its surroundings (*i.e.*, through sensors and cameras), issues a command to switch the lights, senses the environment again, and computes the reward based on the sensing and the command choice. Please note that even with the traffic signal system doing all the work, there is a conceptual separation between agent and environment. The agent is limited to the control module's boundaries, and all the sensors and reward computations take care of everything else.

An RL agent can rely on model-based or model-free approaches. In model-based, the agent learns (or is provided with) a model that reproduces the environment's behavior, while in a model-free approach, the agent interacts only with the environment. The main distinction between the two is that model-free revolves around explicitly trial-and-error training, while model-based (if the model is accurate enough) can allow the agent to plan its actions (therefore called Planning). See more on Planning in Section 2.2.3.

**State** The set of states S comprises scenarios that make the agent differentiate one condition from another when deciding which action to make. It may include any feature that is useful for the decision-making process. For example, the number of vehicles on each road before an intersection can partially or entirely constitute the state definition.

Action The set of actions A is the set of possible actions that the agent executes to solve a given task. A simple action example is to define whether the agent will extend the phase time, rather than keeping the already defined cycle and phase sequence.

**Reward and Return** The reward  $R_t \in \mathbb{R}$  is how the environment evaluates the choice made by the agent at step t - 1. The agent's goal is to maximize the expected return  $G_t$ , that is, the sum of discounted reward it receives over time. Equation 2.1 displays it, where T is the final step or  $\infty$  and  $\gamma$  is the discount rate, with  $0 \leq \gamma \leq 1$ , provided that  $\gamma$  is not 1 at the same time T is  $\infty$ .

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1}$$
(2.1)

The discount rate  $\gamma$  determines how distant rewards are valued in comparison to imminent ones. If  $\gamma = 0$ , the agent is concerned about only maximizing immediate rewards. As  $\gamma$  approaches 1, future rewards are similarly valued to closer rewards.

For example, penalizing an agent, with the objective to reduce waiting time, based on the number of vehicle stops might make the vehicles stop fewer times but wait longer instead.

The reward should guide the agent about what it wants to achieve, not how it should achieve it. For example, penalizing an agent with the objective of reducing waiting time based on the number of vehicle stops might make the vehicles stop fewer times but wait longer instead. A reward formulation should consider alternatives that ensure the maximization of the objective. For example, a reward consisting of the number of different vehicles moving across the intersection can (possibly) reduce the overall waiting time. A more evident option (*i.e.*, more clearly related to the objective) is to measure the waiting time directly.

**Policy and Value Function** The policy  $\pi$  represents the actual decision-making, which matches each state with a choice of an action. As the agent learns, it changes the policy it is employing, resulting in a potentially different policy in each time step  $\pi_t$ , where  $\pi_t(a|s)$  is the probability of choosing the action a given the state s. A deterministic policy is defined as a function (*i.e.*, the probability for one of the actions is one and for the others are zero). In contrast, a stochastic policy is a family of conditional probability distributions.

In the TSC problem, the environment is usually partially observable given that neighbors' state and actions are unknown to the agent. A stochastic policy can be an option to be learned in these situations. However, as a predictable behavior in this domain is typically desired, the policy is usually deterministic. As an example, a resulting policy may discern that when one orientation has, say, 10% more vehicles than the other one, it may switch the signal to favor the orientation with more demand (or maintain it, in case it is already set to favor that orientation).

While in policy-based methods the policy is addressed directly, in value-based methods, the agent updates the estimate of a value function, which is used for deriving the policy  $\pi_t$ . The state-value function  $v_{\pi}(s)$  for policy  $\pi$  (Equation 2.2) denotes the expected return by starting at state s and following  $\pi$  thereafter. Similarly, the action-value function  $q_{\pi}(s, a)$  for policy  $\pi$  (Equation 2.3) also denotes the expected return, this time by starting at s, taking the action a, and then following the policy  $\pi$ .

$$v_{\pi}(s) = \mathbb{E}_{\pi}[G_t \mid S_t = s] \tag{2.2}$$

$$q_{\pi}(s,a) = \mathbb{E}_{\pi}[G_t \mid S_t = s, A_t = a]$$

$$(2.3)$$

By manipulating the state-value function into a recursive equation dependent on its successor state, the Bellman equation for  $v_{\pi}$  (Equation 2.4) arises. An analogous equation can be derived from the action-value function q (Equation 2.5). The Bellman equation represents an essential theoretical base for deriving value function estimations, and consequently, for policy generation.

$$v_{\pi}(s) = \sum_{a} \pi(a|s) \sum_{s',r} p(s',r|s,a) \left[ r + \gamma v_{\pi}(s') \right]$$
(2.4)

$$q_{\pi}(s,a) = \sum_{s',r} p(s',r|s,a) \left[ r + \gamma \sum_{a'} \pi(a' \mid s') q_{\pi}(s',a') \right]$$
(2.5)

A policy is deemed optimal  $(\pi_*)$  when its expected return is greater or equal for all states  $v_{\pi}(s)$  when compared to all other policies  $\pi$ . The respective state-value function and action-value function for the optimal policy are presented in Equations 2.6 and 2.7.

$$v_*(s) = \max_{\pi} v_{\pi}(s)$$
 (2.6)

$$q_*(s,a) = \max_{\pi} q_{\pi}(s,a)$$
(2.7)

Lastly, the respective Bellman optimality equations for  $v_*$  and  $q_*$  are displayed in

Equations 2.8 and 2.9.

$$v_*(s) = \max_{a} \sum_{s',r} p(s',r \mid s,a) \Big[ r + \gamma v_*(s') \Big]$$
(2.8)

$$q_*(s) = \sum_{s',r} p(s',r \mid s,a) \left[ r + \gamma \max_{a'} q_*(s',a') \right]$$
(2.9)

#### 2.2.2 Deep Reinforcement Learning

A class of Reinforcement Learning algorithms, known as tabular methods, use tables to track policy value functions. This approach has significant limitations regarding the representation of policies for large state-spaces and the ability to acknowledge similar states and update their policy at once. For anything but the most straightforward problems, function approximation methods are often used.

Neural Networks [24] are a popular approach to solve the tabular shortcomings. They represent a class of nonlinear function approximators that resort to the weight adjustments in a network of interconnected units, where the input layer consists of the environment's state and the output layer the actions. The typical strategy is to use Neural Networks with several intermediary (hidden) layers, as in this way, a Neural Network is proved to approximate any function. Nowadays, a Neural Network with several layers belongs to the class of Deep Learning [23] methods.

In value-based methods, the Deep Learning part can be used to approximate actionvalue functions  $\hat{q}(s, a, w) \approx q_*(s, a)$ , where  $w \in \mathbb{R}^d$  is the weight vector. In contrast, policy-based methods approximate the policy  $\pi(a|s, \theta)$  directly instead, where  $\theta \in \mathbb{R}^{d'}$ is policy's parameter vector. Finally, actor-critic methods combine both approaches by considering a policy-based method as the actor (the policy learner) and a value-based method as the critic (the policy evaluator) [6].

The unification of Deep Learning with Reinforcement Learning results in the class of Deep Reinforcement Learning (DRL) methods [36]. Due to the large state space and the great similarity between states, DRL methods are generally suggested for the TSC problem.

#### 2.2.3 Planning

In the RL context, Planning (or model-based RL) refers to using a model for searching for future paths (*i.e.*, simulate the future) and use it as contribution to the decisionmaking process (known as *state-space planning*<sup>3</sup>). This contribution can exist in two forms: decision-time planning or background planning. Unlike regular model-free RL, where the policy chooses which action to take (Figure 2.2), in decision-time planning, the agent simulates future experience with the intention to determine the next (nonsimulated) action (see Figure 2.3). In background planning, on the other hand, the simulated experience is used as an additional source (*i.e.*, other than the regular policy action-selection) of learning samples (*i.e.*, experiences to update the policy), as illustrated in Figure 2.4. Note that both planning styles can be combined by using the simulated experience for both direct action picking and policy update.



Figure 2.2: Reinforcement Learning regular action-selection and learning sample generation

To simulate future experience, a model of the environment is generally learned, where the model computes the dynamics to transform the state-action pair into the resulting state and reward. In the TSC problem, when using a traffic simulator with the possibility of (simulation) state saving, the planning interaction can be performed directly by simu-

<sup>&</sup>lt;sup>3</sup>Not to be confused with Planning in the broader sense of Artificial Intelligence, called by RL literature as *plan-space planning* [48].



Figure 2.3: RL Planning (decision-time planning) regular action-selection and learning sample generation

lation. In this case, the simulator acts as a perfect model, not requiring an actual model to be learned. Given the model, the agent builds a tree of future interactions through a distribution or search algorithm, also called, in this context, a planning algorithm.

#### 2.2.3.1 "Search-only" Planning

For the Planning experiments, we also use a PlanningOnly method to verify how the agent performs without any learning component. Since there is no learning, the planning algorithm (or search algorithm) is the only component the agent uses to pick actions. This way, the PlanningOnly method relies on an end-to-end decision-time planning. As illustred in Figure 2.5, the agent takes n steps ahead to check which path yields the most reward. The action derives from the best path by only taking the first step in the actual simulation.

#### 2.3 Frap

Given Frap is the state-of-the-art TSC solution [15] and its code's availability, we choose Frap as our RL base method, and all RL formulations proposed in this dissertation are tested based on it. Frap's primary focus is to reduce the state space, consequently im-



Figure 2.4: RL Planning (background planning) regular action-selection and learning sample generation

proving the learning process.

Figure 2.6 describes the learning process employed by Frap. Each round consists of three parallel simulations serving as the training round and one as the test round. From



Figure 2.5: PlanningOnly's planning process

the training rounds, all agent-environment interactions are stored as learning samples (– $s_t, a_t, s_{t+1}, r_{t+1}$  – tuples) for the policy update. The test rounds, in its turn, generate data for the evaluation of the objective. The Frap's DRL architecture is represented by the policy block, where, internally, a deep neural network receives the states as input and outputs the suggested action.



400 rounds

Figure 2.6: Overall Frap's learning process. 1. For training, Frap runs three simulations in parallel. 2. The tuples  $s_t, a_t, s_{t+1}, r_{t+1}$  are stored in the memory at every instant t. 3. At the end of the training round, the stored learning samples are used to update the neural network that represents the policy. 4. The testing rounds generate the objective data.
# Chapter 3

# **Related Work**

Previous RL solutions for TSC focused on different aspects of the problem. Some investigate policy-based and actor-critic methods [7, 17] while others focused on the different neural network architectures that can be used, including Recurrent Neural Networks (RNN) [51], Long Short-Term Memory (LSTM) [1], and Graph Convolutional Neural Networks (GCNN) [51, 38]. Other works focus on methods' generability (transfer learning and meta-RL) [64, 66]. Some other approaches investigate explicit coordination. For example, CoLight [53] investigates temporal and spatial influences through graph attentional networks to model neighboring information into the agent's state. The attention concept is also incorporated in the reward mechanism by Liu [31]. Wang *et al.* [51] treats coordination with an adjacency graph. Other works engage the coordination aspect from joint action decision [1, 20, 61]. Another important aspect studied is to expand the TSC problem to a broader formulation (*i.e.*, making it more true to life) by testing environments with disruption events (*e.g.*, accidents, constructions, emergencies, break-downs, debris, and weather conditions) and pedestrian crossing [7]. All of those works, however, consider objective and RL formulations only as a secondary matter.

A recent survey [54] shows that RL TSC solutions significantly vary in the objective, RL formulation, method, and experiment setting, making it difficult to compare them when no standard scenario and configuration is established. To understand different formulations, Egea [18] assesses reward functions, but only for a single-intersection case.

Therefore, there is a gap in the literature for studies investigating the objective and RL state and reward formulations and their impacts on the learning process. In this dissertation, we focus our analysis on assessing the use of the proposed time loss-based objective and RL formulations, using the Frap framework [68] as the RL method. Frap reports a relatively simple Deep Reinforcement Learning (DRL) architecture that achieves state-of-the-art results in large-scale real-world scenarios and can be combined with different state and reward formulations. Its highlight is the leveraging of invariance for symmetrical flow cases, such as inversion and rotation, based on the principle of phase competition in TSC (*i.e.*, giving priority to movements with higher demand).

Additionally, most works choose model-free approaches, given that model-based approaches, also called Planning since the model is used to plan, require a model of the environment. However, by interacting with 'future' states, the agent may gather more diverse experience. We assess how experiences resulting from planning can be used to reduce time loss and improve learning convergence.

In this dissertation, we propose new objective and RL state/reward formulations, and thus in the following sections (Section 3.1 and 3.2), we focus on briefly reviewing these aspects.

# 3.1 Objective

From the objectives raised by the survey [54], average travel time (O1, avg\_travel\_time) [7, 8, 15, 20, 21, 26, 27, 29, 31, 45, 49, 51, 52, 53, 55, 56, 57, 62, 64, 68, 69], is one of the most common goals presented in TSC. The metric consists of the average of all vehicles' time to complete their trips. Other objectives include the absolute number of vehicle stops (O2, #stops) [8, 26, 44], the number of vehicles stopped at traffic lights (O3, queue\_length) [16, 17, 21, 25, 34, 37, 42, 55, 67], and the number of vehicles to complete their trips in a given period (O4, throughput) [10, 13, 15, 21, 27, 39, 40, 42, 44, 45, 63].

From these, O2, #stops, is more related to the drivers' comfort, rather than time efficiency; for example, the vehicles might stop on a few occasions but wait long periods each time. O3, queue\_length, is associated with the magnitude of traffic jams. Its downside is that it does not account for vehicles that are not stopped but are instead traveling at very low speeds. O4, throughput, may represent time efficiency, but as the vehicle density increases, their speed naturally decreases while throughput is still high; compare, for example, higher speeds and lower densities vs. lower speeds and higher density (we should avoid the congested flow as much as possible). Lastly, O1,  $avg\_travel\_time$  seems to represent the time efficiency proposition well and avoid the possible pitfalls from the other approaches. Nevertheless, in Chapter 4 we discuss some practical problems of using average travel time as the objective and propose a different objective, based on time loss.

Other objectives used by the literature but not discussed by the survey include de-

lay [2, 3, 5, 8, 10, 13, 19, 20, 21, 37, 39, 55, 67], speed [10, 17, 27, 34, 42], waiting time [9, 21, 25, 27, 34, 38, 40, 42, 44, 45, 58, 67], pollution [7, 21, 42], and fuel consumption [7]. Although pollution and fuel consumption tend to reduce in a better time-optimized system, the result depends directly on vehicles' driving behaviors, such as maintaining constant velocities. Delay and speed are more closely related to our proposition. Specifically, our time loss proposal is delay-based and it is measured with regard to relative speed. The waiting time, similarly to queue length, only accounts for stopped periods.

From all objectives above, the less is better for the #stops, queue\_length, avg\_travel\_time, delay, waiting time, pollution, and fuel consumption, while the more is better for throughput and speed. The presented time loss objectives fit the first case.

## 3.2 RL Formulation

In RL, the *state* consists of the information the agent uses to distinguish different situations in the decision-making process. The *reward* is the feedback it receives about how good or bad were those decisions. Both state and reward need adequate representations so the agent can learn how to maximize cumulative reward. In this dissertation, we explore matching the RL formulation with the objective so that a better association between them leads to better reaching the objective.

We selected a few state and reward proposals from the literature that work on discrete information and do not rely on a combination of linear weights while also presenting good experimental results. They are proposed in Frap [68] ( $\langle$ Number of Vehicles, Vehicle Been Stopped $\rangle$  or  $\langle$ Number of Vehicles, Average Queue Length $\rangle$ ), PressLight [52]  $\langle$ PressLight's Pressure, PressLight's Pressure $\rangle$ , and MPLight [15]  $\langle$ MPLight's Pressure, MPLight's Pressure $\rangle$ . All the formulations rely on the movement modeling for the state (more on Section 4.2) and over the intersection for the reward.

The distinction between theirs' state and reward and ours is that their formulations do not relate directly to the goals, while ours do. Their objective uses average travel time as metric and density-related measurements (*e.g.*, number of vehicles, queue length, relative road density, and pressure) to state and reward. Ours use time loss information for both objective, state, and reward. We detail all the RL formulations in Chapter 4 and compare them with our time loss formulations ( $\langle \text{Time Loss}, \text{Time Loss} \rangle$  and  $\langle \text{Time Loss} \rangle$ Pressure, Time Loss Pressure $\rangle$ ) in Chapter 5.

# Chapter 4

# **Problem Definition**

This dissertation investigates how to address the traffic signal control (TSC) problem using Reinforcement Learning (RL) [48]. We adopt terminology from the traffic simulator Sumo [32], Frap [68], and RL literature.

This chapter aims to detail the TSC problem and model it according to the RL concepts. The problem objective is described in Section 4.1. Section 4.2 defines the movements of vehicles within an intersection as well as the traffic signal phases (green, red, yellow traffic light configurations) that allow specific vehicle movements to flow. RL components modeling the TSC problem (*i.e.*, state, action, reward, policy, and environment formulations) are presented in Section 4.3.

# 4.1 Objective

Based on the works described in Chapter 3, a common choice in the TSC problem is to use *average travel time* as the objective [15, 31, 51, 52, 53, 64, 68]. The metric consists of averaging the time each vehicle took to complete its own trip. Such a metric carries the notion of a utility function, representing what people likely value most: arriving faster at their destinations.

However, the average travel time metric has three major shortcomings when using it to train the RL agents. First, it is not accurate throughout the simulation runs because travel times are only determined as the vehicles complete their trips; however, in RL, the reward must credit each actions' outcome at every time. Therefore, travel time seems adequate to the objective but may not be accurate in deriving a reward mechanism. In general, this metric is not appropriate in analyzing intermediate simulation results. Second, and more importantly, this metric is not reliable across simulations, given that it is susceptible to starvation and deadlocks (vehicles "stuck in traffic" forever have a theoretic infinite travel time). In practice, simulators do not account for these vehicles' actual contribution, as it would have to wait indefinitely for them to finish. Hence, starvation and deadlocks actually contribute to decrease (instead of increasing) travel time, producing counter-intuitive and unrealistic results. Sections 4.3.4.2 and 4.3.5.1 further discuss starvation and deadlock in the TSC simulation, respectively.

Lastly, simply averaging travel time may fail to correctly assess unrealistic traffic shaping behavior that we called 'hold vehicles before depart'. This behavior represents the cases where methods artificially delay the vehicles' departure in favor of having fewer vehicles simultaneously in the simulation, effectively serving fewer vehicles at the end than expected but receiving better values of travel time. To alleviate this issue, MPLight, for example, includes throughput as a co-objective, such that the number of vehicles served can also be assessed [15].

Due to the issues aforementioned, we propose to use *time loss per driver* metrics. Time loss captures how much time (e.g., in seconds) vehicles lose in their trips, taking into account portions of time lost by not moving at the *free-flow speed* (*i.e.*, the maximum speed). Unlike travel time, time loss can be safely evaluated at any time, making it useful for simulation analysis and removing the reward formulation limitation. The per-driver element exists to make the metric quickly appraisable.

The time loss of each vehicle is measured at each instant and consists of the complementary of the relative speed. Equation 4.1 describes the instantaneous time loss for a given vehicle *i* at time *t*, where  $v_{i_t}$  is its current speed and  $v\_max_{i_t}$  is the allowed maximal speed. This definition is consistent with the one provided by the traffic simulator Sumo [32].

**Definition 4.1.1 (Instant Time Loss per Vehicle)** Instant Time Loss per Vehicle is the amount of time is lost in a given instant by a vehicle (smaller is better).

$$time\_loss_{vehicle_{i_{\star}}} = 1 - v_{i_t}/v\_max_{i_t}$$

$$(4.1)$$

The time loss for the traffic network is the sum of all vehicles' time losses in the simulation (including pending vehicles<sup>1</sup>) at a given time t. It is presented in Equation 4.2, where  $n_t$  is the number of vehicles at time t:

<sup>&</sup>lt;sup>1</sup>The pending vehicles are the ones that were programmed to enter the simulation but could not be inserted due to lack of physical space at the edges of the network.

**Definition 4.1.2 (Network Instant Time Loss)** Network Instant Time Loss is the time lost in a given instant by all vehicles in the simulation (smaller is better).

$$time\_loss_{network_t} = \sum_{i=1}^{n_t} time\_loss_{vehicle_{i_t}}$$
(4.2)

We formulate two different versions for the time loss per driver objective, namely percentage time loss per driver and cumulative time loss per driver. The first takes into account the instantaneous time loss per driver (including drivers from pending vehicles), as shown in Equation 4.3, where  $t_{total}$  is the total simulation period (*i.e.*,  $t_{finish} - t_{start}$ ) and  $n_t$  is the number of vehicles at time t. For  $t_{total} > 0$ :

**Definition 4.1.3 (Percentage Time Loss per Driver)** Percentage time loss per driver consists of the network time loss divided by all vehicles in the simulation at the respective instant, including the contribution of pending vehicles, averaged over the simulation period (smaller is better).

$$percentage\_time\_loss\_per\_driver = \frac{\sum_{t=t_{start}}^{t_{finish}} \frac{time\_loss_{network_t}}{n_t}}{t_{total}}$$
(4.3)

The second version considers the simulation's total time loss divided by all the drivers. Equation 4.4 illustrates it, where the time t ranges from  $t_{start}$  to  $t_{finish}$ , and  $n_{total}$  is the total number of all vehicles in the simulation (*i.e.*, both pending and departed vehicles). For  $n_{total} > 0$ :

**Definition 4.1.4 (Cumulative Time Loss per Driver)** Cumulative time loss per driver consists of all time loss accumulated in the simulation, divided by all vehicles, including the contribution of pending vehicles (smaller is better).

$$cumulative\_time\_loss\_per\_driver = \frac{\sum_{t=t_{start}}^{t_{finish}} time\_loss_{network_t}}{n_{total}}$$
(4.4)

The main difference between percentage time loss per driver and cumulative time loss per driver is that the former averages the percentages of the network time loss for the entire simulation period, thus outputting a mean percent value, while the latter computes the usual time loss per driver average, outputting the time loss per driver as the number of seconds spent. Note that both use time loss by time intervals rather than vehicle trips, avoiding the first two shortcomings present in the travel time metric. Any intermediate simulation results can safely be obtained by adjusting the t\_start and t\_finish values. Also, by accounting for the time loss of pending vehicles, both methods avoid the 'hold vehicles before depart' issue entirely, solving the third travel time shortcoming. At the same time, *cumulative time loss per driver* also obviates the coupling with the throughput co-objective; once that at the end, all vehicles were accounted at their respective times.

### 4.1.1 Time Loss and Travel Time Relationship

Note that time loss and travel time are different but also related to each other. The optimal travel time occurs when the vehicle is driving at the free-flow speed for the entire trip. The difference in the actual travel time characterizes the time loss, also defined by traffic engineering literature as travel-time delay [22] as illustrated in Equation 4.5. This relationship implies that optimizing the time loss also optimizes travel time.

$$time\_loss = travel\_time - travel\_time_{optimal}$$

$$(4.5)$$

Similarly, the  $time\_loss\%$  is the percentage of time lost in the travel (Equation 4.6).

$$time\_loss\% = 1 - \frac{travel\_time_{optimal}}{travel time}$$
(4.6)

Another common measure, the travel time index [46] describes the percentage of extra time drivers took to complete their trips. The relationship between  $travel\_time\_index$  and  $time\_loss\%$  is expressed in Equation 4.7.

$$travel\_time\_index = \frac{travel\_time}{travel\_time_{optimal}} = \frac{1}{1 - time\_loss\%}$$
(4.7)

For example, if *time\_loss*% is 0.5, then the *travel\_time\_index* is 2, meaning that half of the trip was time lost and took twice the time to make the trip. Similarly, the *time\_loss*% can be expressed in terms of *travel\_time\_index*, as in Equation 4.8.

$$time\_loss\% = 1 - \frac{1}{travel\_time\_index}$$
(4.8)

While the  $time\_loss$  is analogous to the cumulative time loss per driver (Equation 4.4) for the average driver, the  $time\_loss\%$  is analogous to the percentage time loss per driver (Equation 4.3) when defined for the entire demand and to the instantaneous time loss (Equation 4.1) when defined for a single driver. When put together, the  $time\_loss$  and

 $time\_loss\%$  values can derive the average travel time in seconds (as in Equation 4.9) and the average optimal travel time (by manipulating Equation 4.5 and using the average travel time).

$$time \ loss = travel \ time \times time \ loss\%$$
(4.9)

## 4.2 Movements and Phases

Movements are the directions of flow within an intersection, characterized by the traffic lanes turns; for example, in the 4-road intersection in Figure 4.1a, each road (0-3) has three movements: left-turn (L), straight (S), or right-turn (R). The *phases* are the movements that the traffic signal allows to proceed; for example, in the upper-left phase from Figure 4.1b, movements 0L and 0S are given green signals while all other movement changes to red.

Figures 4.1b and 4.1c show two *phase sets*: a typical phase set (as formulated in Frap [68], PressLight [52], and MPLight [15]) with 8 phases and an expanded phase set (Figure 4.1c) with 14 phases, that adds four non-conflicting-movements phases (*i.e.*, vehicles have segregated paths) and two conflicting-movements phases (*i.e.*, vehicles also coordinate themselves by following right-of-way rules). Also, the examples follow a right on red traffic rule (*i.e.*, vehicles can turn right even if the signal is red); hence we do not include the right-turn movements in the phase set.

The RL agent's responsibility is to change phases, given a set of phase options (*i.e.*, the phase set). Regarding those options, there is a trade-off: the agent should be allowed all possible phase options to guarantee freedom of action, but, at the same time, including many options may increase the state-action search-space unnecessarily. If the agent is given a too simplistic phase set, it may perform poorly, as it lacks the ability to choose how to direct traffic. A compromise can make the agent pick the more suitable phase for each case without consuming unnecessary time and effort during learning, consequently reducing the pressure on computer resources and the model's capability. For this reason, the Frap selection for the phase set allows agents to consider specific combinations of pairs of movements that do not conflict with each other (*i.e.*, their paths do not intercept). As a general rule, it seems a good compromise between freedom of action and simplicity. With that in mind, we experiment with several phase set configurations. See more in the experiments chapter (Chapter 5).



Figure 4.1: Synthetic scenarios' movement and phase definition

# 4.3 Problem Modeling

While we briefly covered TSC in Section 2.2.1.1, this section describes the problem modeling from the perspective of RL, and in particular of selected Frap correlated methods. That is, we need to describe the RL agent's state, reward, action, policy, and environment.

This dissertation also proposes a time loss formulation and a pressure variant to better relate to the time loss objective proposition (Section 4.1). The formulation details for the state and the rewards are described in Sections 4.3.1 and 4.3.3, respectively. Details about the action formulation are covered in Section 4.3.2, about policy in Section 4.3.4, and the environment in Section 4.3.5.

## 4.3.1 State

There are several approaches for defining states in RL for TSC. Frap state comprises (a) the current phase and (b) the number of vehicles in the entering lanes of each traffic movement (*i.e.*, in an array of movements); see Equation 4.10, where  $n(m_{enls})$  is the number of vehicles in the movement's entering lanes  $m_{enls}$ .

$$number\_of\_vehicles = \sum_{i}^{n(m_{enls})} 1$$
(4.10)

PressLight and MPLight, instead of vehicle count and phase, use phase and *pressure*, where pressure (of an intersection) is based on the difference of measurement between the movement's entering lanes  $m_{enls}$  and exiting lanes  $m_{exls}$ . For PressLight, pressure  $p_{PL\_state}$  is the difference between vehicles' density of  $m_{enls}$  and  $m_{exls}$  (Equation 4.11), where  $x(ls)/x_{max}(ls)$  is lanes ls's relative occupancy, x(ls) is the number of vehicles currently in lanes ls, and  $x_{max}(ls)$  is the maximum capacity of lanes ls. For MPLight, pressure  $p_{MP\_state}$  is the difference between the number of vehicles (Equation 4.12) in the movement's entering lanes  $m_{enls}$  and exiting lanes  $m_{exls}$ .

$$p_{PL\_state}(m_{enls}, m_{exls}) = \frac{x(m_{enls})}{x_{max}(m_{enls})} - \frac{x(m_{exls})}{x_{max}(m_{exls})}$$
(4.11)

$$p_{MP \ state}(m_{enls}, m_{exls}) = x(m_{enls}) - x(m_{exls})$$

$$(4.12)$$

In our approach, as mentioned before, we focus on time loss. We consider two versions of state, namely (1) time loss  $(tl_{state})$  and phase, and (2) time loss pressure  $(p_{tl_state})$  and phase. For (1), we use vehicles in the movement's entering lanes; see Equation 4.13, where n(ls) is the number of vehicles in the lanes ls, and  $time_loss_{vehicle_{i_t}}$  is from the Equation 4.1. For (2), the pressure variant, it is the difference between time loss in the movement's entering and exiting lanes; see Equation 4.14.

**Definition 4.3.1 (Time Loss** - **state)** For the state formulation, time loss is the total time lost in the entering lanes for each movement.

$$tl_{state}(m_{enls}) = \sum_{i}^{n(m_{enls})} time\_loss_{vehicle_{i_t}}$$
(4.13)

**Definition 4.3.2 (Time Loss Pressure** - **state)** For the state formulation, time loss pressure is the difference of time lost between the entering and exiting lanes for each movement.

$$p_{tl\_state}(m_{enls}, m_{exls}) = tl_{state}(m_{enls}) - tl_{state}(m_{exls})$$

$$(4.14)$$

## 4.3.2 Action

Similar to Frap [68], the agent decides which phases to activate (*e.g.*, switch from current phase to phase  $0S_2S$ ). After that, the controller grants a fixed amount of time (*i.e.*, 10 seconds) for that phase, including green-yellow-red transition. While the phase set pre-determines which actions are possible, the agent manages the sequence of phases and the resulting time distribution among them.

### 4.3.3 Reward

As presented in Section 2.2.1, the reward is the feedback the agent receives to figure out the consequences of its actions. With that in mind, the reward should be modeled in conjunction with the agent's decision process (*i.e.*, state and action) following the objective proposition.

The agent reward in our "time loss formulation" is similar to our state formulation: simply the negative value of total time loss of all the entering lanes, as illustrated in Equation 4.15 (the more time is lost, the bigger is the penalty). The pressure variant considers the difference between time loss in the entering and exiting lanes (Equation 4.16).

**Definition 4.3.3 (Time Loss – reward)** For the reward formulation, time loss is the total time lost in the intersection's entering lanes (the bigger the cumulative reward is the better).

$$tl_{rwd}(enls) = -\sum_{enl}^{enls} \sum_{i}^{n(enl)} time\_loss_{vehicle_{i_t}}$$
(4.15)

**Definition 4.3.4 (Time Loss Pressure** – **reward)** For the reward formulation, time loss pressure is the difference of time lost between the intersection's entering and exiting lanes (the bigger the cumulative reward is the better).

$$p_{tl \ rwd}(enls, exls) = -\left(tl_{rwd}(enls) - tl_{rwd}(exls)\right) \tag{4.16}$$

Frap defines reward as the average queue length where  $queue\_length(l)$  is the number of stopped vehicles in the lane l (Equation 4.17). In addition, Frap's GitHub code<sup>2</sup> adopts "vehicles been stopped" instead, which consists of the number of vehicles that stopped at least once (Equation 4.18); we are investigating both versions. The difference between

<sup>&</sup>lt;sup>2</sup>https://github.com/gjzheng93/frap-pub

them is that when vehicles start moving, they leave the queue, but the "been stopped" property only resets once the vehicles cross the intersection. Like a cumulative queue length record, the vehicles been stopped consist of the count of vehicles that stopped before the intersection at least once, considering the intersection's detection range.

$$average\_queue\_length(enls) = -\left(\frac{\sum_{enl \in enls} queue\_length(enl)}{\#enls}\right)$$
(4.17)

$$vehicles\_been\_stopped(enls) = -\sum_{enl}^{enls} \sum_{i}^{n(enl)} been\_stopped(i)$$
(4.18)

Similar to the pressure for movements in the state definition, PressLight and MPLight use the pressure of intersection as reward. For PressLight it is the absolute sum of movement pressures (Equation 4.19). For MPLight it is the difference between the entering and exiting lanes' queue length (Equation 4.20). Recall that  $p_{PL_state}$  is the pressure defined in Equation 4.11 and queue\_length is the number of stopped vehicles in the referred lane enl or exl.

$$p_{PL\_rwd}(enls, exls) = -\left|\sum_{(m_{enls}, m_{exls})}^{movements} p_{PL\_state}(m_{enls}, m_{exls})\right|$$
(4.19)

$$p_{MP\_rwd}(enls, exls) = -\left(\sum_{enl}^{enls} q\_length(enl) - \sum_{exl}^{exls} q\_length(exl)\right)$$
(4.20)

These formulations are assessed experimentally in Chapter 5 to understand how they perform in preliminary experiments and then select which one is further experimented on remaining tests.

### 4.3.4 Policy

Recall from Section 2.2.1.1 that the policy is a function that maps the action to be executed given a state. For non-adaptive methods, the policy is independent of context, performing actions regardless of the state. For adaptive methods, the policy is explicitly configured using rules (*i.e.*, whenever the time loss for a movement exceeds a threshold, change its light to green). In RL, instead of defining rules beforehand, an adaptive policy

is learned by iterative configuration, based on experience and experimentation (*i.e.*, learn which thresholds are essential to change to which behavior).

When evaluating the methods through their time loss reduction (or by any other metric, in fact), it is the policy that is being evaluated objectively. However, the drivers' perception of the phase-switching behavior is as important as the time loss result itself but much more subjective. With that concern, we discuss two policy-related problems, excessive waiting time and starvation (Sections 4.3.4.1 and 4.3.4.2). Despite that they are related to each other, they illustrate different policy behavior analysis.

#### 4.3.4.1 Excessive Waiting Time

Besides yielding satisfying time loss results, the policy has to be functional and take into account the users. A policy can achieve significant overall time loss reductions but constantly leave portions of traffic waiting unreasonably long. Other policies might achieve the same time loss levels or even a little higher, and at the same time, avoid making the driver angry or exhausted (*i.e.*, high time loss for a specific set of vehicles). Two possible strategies can be used to shape a policy: (1) tuning the reward function towards some behavior and (2) enforcing an environment restriction. To obtain a reward-based shaping, one could add some extra penalty from an impatience-based metric (*e.g.*, gradually subtract extra reward when vehicles strike waiting time milestones). As an environment's restriction, the environment could register vehicle waiting times and override the action before it violates the condition, like an external safety measure.

This dissertation chooses to employ the latter strategy. Transferring control to the environment avoids adding more complexity to the learning agent, including correctly mixing weights for different rewards. The implemented waiting time restriction tracks each lane's first vehicles' waiting time and examines how much time is available before the vehicles violate the constraint. It overrides the chosen action with the one with the most accumulated waiting time to fix the situation.

#### 4.3.4.2 Starvation

Starvation in a traffic signal happens when at least one of the movements does not receive enough attention from the agent (*i.e.*, the phases that control the movement are never or rarely activated). Policies that have not (yet) learned the environment dynamics (especially true in the early stages of the learning process) can produce starvation, which can happen independently of achieving high or low time loss reduction levels. This behavior can be generally mitigated by adequate state, action, and reward formulation and corrected by the learning process.

Although not initially driven by it, the waiting time restriction (presented in Section 4.3.4.1) also prevents starvation from happening by limiting any driver from waiting for more than the intended threshold. It would prevent starvation even on the agents whose learned policies would be susceptible to it. However, as the restriction is included from the beginning of the learning process, it may actually guide the learning process by restraining exploration into unacceptable policies.

### 4.3.5 Environment

As it is not advisable to directly train a trial-and-error agent with a real-world traffic light system, a traffic simulator is required to test methods for the TSC problem. Sumo is a very mature open-source traffic simulator and set of tools. It supports traffic network and demand modeling, baseline static methods design, and simulation environment. TraCi, from the Sumo suite, is used to interface with the simulation to collect all the state, reward, and report data. It also allows changes in the traffic signals dynamically and programmatically and, to some extent, supports extending some of the simulation functionality.

The simulation can be configured in many ways to reproduce the traffic dynamics, such as the consequence of collisions, the vehicles' possible routes, the traffic demand levels, and the overtaking of vehicles that are blocking the way. These traffic dynamics may lead to some real-world phenomena, for instance, local and generalized traffic deadlocks. Deadlocks can happen naturally or be caused by the agent's decisions. Following we describe the causes and possible solutions for deadlocks.

#### 4.3.5.1 Deadlock

In the simulation (and also in the real world), deadlocks in a single intersection can happen after a vehicle collision or when a vehicle advances and positions itself in the intersection causing a disturbance and blocking vehicles from completing their crossing. If the vehicles do not resolve the deadlocks within a reasonable time, they can be incredibly harmful because they waste much time prolonging a behavior entirely outside of the signal control. Consequently, the agent severely penalizes all possible actions (since they have little to no effect), hindering the learning process. Notice that it is not necessary to form a hard deadlock (*i.e.*, no physical space to maneuver with ease) to be harmful. Soft deadlocks (*e.g.*, one of the vehicles only having opposite moving traffic in its way) can cause many problems even they end up resolving themselves eventually. Sumo has an option to let vehicles resolve deadlocks themselves after some time. For this type of deadlock, a solution is to reposition the intersection waiting stops closer to the entering lane. This way, it reduces the vehicles' possibility of positioning themselves where they may get stuck or get other vehicles stuck.

Deadlocks can also happen in a multiple-intersection setting when vehicles block each other across a circular pattern in traffic jams along several streets. Inadequate traffic signal control can easily jeopardize traffic network function and lead to smaller demands to over-saturate the streets, causing traffic jams and possibly deadlocks, depending on the vehicles' flow patterns. In that case, bad policies (if in the early stages of the learning process) might be corrected during learning.

# Chapter 5

# Experiments

This chapter describes the executed experiments and details their configurations. Section 5.1 covers the scenarios. Section 5.2 describes the methods used. Section 5.3 details the regular (*i.e.*, non-planning) experiments, while Section 5.4 details the planning experiments. For the complete list of parameters and hyperparameters, check Appendix A.

## 5.1 Scenarios

The scenarios consist of the street network and its configured demand. In this dissertation, we consider two synthetic scenarios: a single 4-way intersection (Figure 5.1a); and a 4-intersection 4-way network (Figure 5.1b). Both scenarios simulate one hour of traffic, and all their edges are 300 meters long. We use a binomial distribution vehicle generation with a probability of 0.3, that is, 0.3 vehicles/s average (adjusted empirically for light to medium traffic), per edge, which are split with a uniform distribution of 10% turning left, 60% going straight, and 30% turning right (the same distribution used by Chen [15]). The total volume of vehicles for the scenarios is 4284 and 8631, respectively.

## 5.2 Baselines

To assess the Learning and Planning results, we use the static and unregulated configurations as baselines. Additionally, we use the Frap's state and reward formulation as the RL baseline for the Learning experiments and the best non-planning result as the RL baseline for Planning.

In Sumo, intersections can be configured to be *unregulated*. In this mode, the vehicles



Figure 5.1: The scenarios

completely ignore any traffic from other movements, disabling collisions, and deceleration at the intersection. For our purposes, it symbolizes a utopic traffic light control, serving as a hard lower-bound baseline for any other method. Although it is utopic, it still is a more reasonable lower boundary than the free-flow demand level since the vehicles' speed is compatible with the current traffic load. For the upper-bound baseline, we use Sumo's auto-generated static signaling.

A planning component is also analyzed. In this case, it refers to planning in the RL context, where the agent looks ahead of time, tests with actions, and foresight its value (as described in Section 2.2.3). For experimenting with planning, instead of constructing a model to simulate traffic's behavior (*i.e.*, predict next traffic states), parallel simulations are used to test future possibilities.

**Static** For the upper bound, we use Sumo's auto-generated right on red static configuration. While the RL agents choose which of the phase options to activate, the static method is preconfigured by default. The generated static configuration uses four phases, alternating most of its time between the conflicting-movements phases  $0S_2S_0L_2L$  and  $1S_3S_1L_3L$ , using the other two  $0L_2L$  and  $1L_3L$  only to clear out any vehicles in the minor lanes between transitions (*i.e.*, the left-turn lanes 0L, 2L, 1L, and 3L are given exclusivity for a short period).

**Unregulated** For the lower bound, we use Sumo's traffic to be unregulated.

**FRAP** We choose Frap as our RL base method for testing all RL formulations proposed in this dissertation (recall Section 2.3). We run a total of 400 learning episodes per RL experiment with Frap's default hyperparameters<sup>1</sup> (check Appendix A) for all scenarios.

**PlanningOnly** As covered in Section 2.2.3.1, PlanningOnly has no learning component and relies exclusively on decision-time planning.

**FRAP** + **Planning** The idea about including Planning to Frap is to increase the learning sample diversity and improve the learning speed and results. With planning embedded, Frap can either generate experience from unfulfilled prospective interactions, decide its action based on future experience, or both. The Frap+Planning follows the same base learning process from Frap (Figure 2.6), with the additional particularities discussed in Section 2.2.3 (illustrated by Figures 2.2, 2.3, and 2.4).

# 5.3 Non-Planning Experimental Results

In this section, we experiment with the waiting time restriction (Section 5.3.1), phase set configurations (Section 5.3.2), and RL formulations (Section 5.3.3). For the Planning-related experiments' result, check Section 5.4.

### 5.3.1 Waiting Time Restriction

In this experiment, we want to assess the vehicles' waiting time fairness. Since the agent can choose any next phase that seems suitable, some vehicles can wait way longer since the agent doesn't follow a predefined phase sequence, as the static configuration does; at the same time, it is looking to the intersection's overall time savings instead of each time loss individually.

As mentioned in Section 4.3.4.1, besides optimizing the time loss objective, the policy can also be shaped to favor a more user-friendly behavior. The waiting time restriction is an environment constraint designed to prevent vehicles from waiting a long time at the traffic signal. We use Frap's Vehicle been stopped - RLF1 - formulation ( $\langle$ Number of Vehicles, Vehicle Been Stopped $\rangle$ ; Table 5.4) in a single intersection (8 phases) for this experiment.

<sup>&</sup>lt;sup>1</sup>Note that the hyperparameters in the GitHub repository may or may not be equal to the ones used in the paper.



Figure 5.2: Regular policy (no WTR) phases and demand distribution



Figure 5.3: WTR policy phases and demand distribution

Configuration	Cumulative Time Loss per Driver	Percentage Time Loss per Driver
WTR off	44.03	0.50
WTR on $(120s)$	49.22	0.53

Table 5.1: Waiting Time Restriction experiment (WTR on and off)

As observed in Table 5.1, the restriction negatively affects the time loss result, leading to a difference of approximately 4s more time loss per driver than without the restriction. As discussed above, this is expected since the agent must regularly comply with the restriction and prioritize fewer vehicles with a significantly longer waiting time. However, Figures 5.2 and 5.3 show the difference from both policies' behavior. The policy generated without the restriction generates considerably larger waiting periods than the variant with 120 seconds restriction (*e.g.*, in the movement 2L, between seconds 540 and 940, vehicles wait up to 400s). Note that some phases on WTR may be greater than 120s since WTR considers the waiting time from the first vehicle when it stops at the signal. As the waiting time restriction significantly improves the policy fairness, it stays enabled for the remaining of the experiments.

### 5.3.2 Phase Set Configuration

The selection of the 'phase set' changes the possibility of options when the agent switches the traffic lights. More options mean the agent may take a longer time to test it all and understand the outcomes. Fewer options may seem more manageable, but the agent can be taken out the opportunity of choosing better actions for specific situations. Another complication is that the phase selection is not an unconnected decision. Transitions resulted from different phase sequences produce different outcomes, depending on the context. A particular sequence of phase switching, for example, can increase the number of accidents (collisions).

For that reason, instead of picking, for instance, the maximal set option, it is important to verify how the phases' options impact the learning speed and the result. In this experiment, we compare several phase set configurations. We include: the default 8 phases set configuration as the baseline — **PSC1**, a 12 non-conflicting phases set — **PSC2**, a simplified 4 phases set — **PSC3**; alternatives containing the combination of multiple phases, including the 14 phases set — **PSC4** — discussed in Section 4.2 (the 12-phases set plus the  $0S_2S_0L_2L$  and  $1S_3S_1L_3L$  combined phases) and other

similar-intended options (10 [8+2] — **PSC5**, 6 [4+2] — **PSC6**, and 2 [0+2] phases — **PSC7**); a further expanded 16-phases set — **PSC8**, which adds the opposing combined phases  $0S_2S_1L_3L$ ,  $1S_3S_0L_2L$  to the 14-phases set; a 10-phases set consisted of the 14-phases set subtracted of phases that are subset of any other phase in the original set — **PSC9**; and, finally, the phase set used by the static (baseline) method — **PSC10**, which includes the  $0S_2S_0L_2L$  and  $1S_3S_1L_3L$  combined phases. Table 5.2 put together all phases these definitions include. Table 5.3 shows the result for the experiment, where the regular 8 phases set (PSC1) is the baseline. For this experiment we also use Frap's Vehicle been stopped - RLF1 - formulation ((Number of Vehicles, Vehicle Been Stopped)) in a single intersection (8 phases); see formulation details in Table 5.4.

From the phase set configuration results, PSC8 presents the worst results. In that case, the RL agent either could not learn to isolate prejudicial phases and phase transitions (*e.g.*, 0S\_2S\_1L\_3L and 1S\_3S\_1L\_3L may cause more collision than other phases selections), or there were more phases that the agent could handle. On the other hand, the PSC3 presents bad results because it is too simplistic compared with the default 8 phases set (PSC1). All the configurations that include the combined phases 0S\_2S\_0L\_2L and 1S\_3S\_1L\_3L (except for PSC8, already explained above) present better results than those that do not. Even PSC7 that relies on those two phases alone presented better results than the default PSC1. Even though PSC7 has fewer phases than PSC3, it achieves much better results because it has fewer (but more effective) phases in its phase set. These results indicate that to include 0S\_2S\_0L\_2L and 1S\_3S\_1L\_3L and rely on right-of-way traffic rules does pay off.

#### 5.3.3 RL Formulation

In RL, the state and reward formulations (presented in Sections 2.2.1.1 and 4.3) heavily influence the agent's decision-making during the learning process. The experiments we conducted use the Frap agent to examine how different state and reward formulations affect the different metrics. In total, we test six  $\langle$ state; reward $\rangle$  formulations, where the first two come directly from Frap, the middle two come from subsequent proposals (PressLight [52] and MPLight [15]), and the last two are our time-loss-based proposals; the RL state includes the current phase, in addition to the value shown as follows (see also Table 5.4):

• (**RLF1**) (Number of Vehicles, Vehicle Been Stopped);

Label	Configuration	Phase Set
PSC1	Default 8 phases set (8 phases)	0L_0S, 0L_2L, 0S_2S, 1L_1S, 1L_3L, 1S_3S, 2L_2S, 3L_3S
PSC2	12 phases set (12 phases)	0L_0S, 0L_2L, 0S_2S, 1L_1S, 1L_3L, 1S_3S, 2L_2S, 3L_3S, 0L_3S, 0S_1L, 1S_2L, 2S_3L
PSC3	4 phases set (4 phases)	$0L_2L, 0S_2S, 1L_3L, 1S_3S$
PSC4	12 phases set + 2 combined phases (14 phases)	$\begin{array}{c} 0L\_0S, 0L\_2L, 0S\_2S, 1L\_1S, \\ 1L\_3L, 1S\_3S, 2L\_2S, 3L\_3S, \\ 0L\_3S, 0S\_1L, 1S\_2L, 2S\_3L, \\ + \ 0S \ \ 2S \ \ 0L \ \ 2L, + \ 1S \ \ 3S \ \ 1L \ \ 3L \end{array}$
PSC5	8 phases set + 2 combined phases (10 phases)	$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$
PSC6	4 phases set + 2 combined phases (6 phases)	$0L_2L, 0S_2S, 1L_3L, 1S_3S, \\ + 0S_2S_0L_2L, + 1S_3S_1L_3L$
PSC7	Combined phases only (2 phases)	$+ 0S\_2S\_0L\_2L, + 1S\_3S\_1L\_3L$
PSC8	12 phases set + 4 combined phases (16 phases)	$ \begin{array}{c} 0L\_0S, 0L\_2L, 0S\_2S, 1L\_1S, \\ 1L\_3L, 1S\_3S, 2L\_2S, 3L\_3S, \\ 0L\_3S, 0S\_1L, 1S\_2L, 2S\_3L, \\ + 0S\_2S\_0L\_2L, + 1S\_3S\_1L\_3L, \\ + 0S\_2S\_1L\_3L, + 1S\_3S\_0L\_2L \end{array} $
PSC9	12 phases set + 2 combined phases - 4 "redundant" phases (10 phases)	$0L_0S, 1L_1S, 2L_2S, 3L_3S, \\0L_3S, 0S_1L, 1S_2L, 2S_3L, \\+ 0S_2S_0L_2L, + 1S_3S_1L_3L$
PSC10	Static method phase set (4 phases)	$\begin{array}{cccccccccccccccccccccccccccccccccccc$

Table 5.2: Phase set definition (conflicting-movements phases highlighted)

- (RLF2) (Number of Vehicles, Average Queue Length);
- (**RLF3**) (PressLight's Pressure, PressLight's Pressure);
- (RLF4) (MPLight's Pressure, MPLight's Pressure);
- (RLF5) (Time Loss, Time Loss); and
- (RLF6) (Time Loss Pressure, Time Loss Pressure)

As discussed in Section 4.3, the pressure formulations (*i.e.*, RLF3, RLF4, and RLF6

Phase Set Configuration	Cumulative Time Loss per Driver	Variation	Percentage Time Loss per Driver	Variation
PSC1	44.93	-	0.51	-
PSC2	42.05	6.41%	0.49	3.92%
PSC3	57.90	-28.87%	0.56	-9.80%
PSC4	22.07	50.88%	0.33	35.29%
PSC5	23.46	47.79%	0.35	31.37%
PSC6	22.88	49.08%	0.34	33.33%
PSC7	22.10	50.81%	0.33	35.29%
PSC8	100.83	-124.42%	0.64	-25.49%
PSC9	21.60	51.93%	0.33	35.29%
PSC10	21.25	52.70%	0.33	35.29%

Table 5.3: Phase set configuration results

Table 5.4: RL formulations tested

<b>RL Formulation</b>	State		Reward
BLF1		number of vehicles	number of vehicles
		(per movement)	have been stopped
BLE5		number of vehicles	everage queue longth
		(per movement)	average queue length
BLF3		pressure [presslight]	pressure [presslight]
1(11)	curront phase	(per movement)	pressure [pressingine]
BLF4		pressure [mplight]	pressure [mplight]
		(per movement)	pressure [inplight]
BLF5		time loss	time loss
10110		(per movement)	
BLF6		pressure [time loss]	pressure [time loss]
10110		(per movement)	

recur to information from both entering and exiting lanes. One important consideration is that the exiting lanes naturally do not provide as much information in the singleintersection as they do in the multi-intersection case since, in the former, the vehicles just flow out of the city grid instead of sometimes waiting at the next intersection. Thus, the pressure formulations are better assessed in the Multi-Intersection scenario.

The experiments were executed in 8 and 14 phases set for each scenario. Table 5.5 and Figures 5.4 and 5.5 show the results for the Single-Intersection scenario and Table 5.6 and Figures 5.6 and 5.7 show the results for the Multi-Intersection scenario. We picked the best 10-round average results. For the best round in the specified window, check the

#### Appendix B.

Model	Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement						
Static	40.03	-	0.48	-						
Unregulated	4.23	-	0.09	-						
	8 phases									
RLF1	44.93		0.51							
RLF2	62.99	\	0.58	\						
RLF3	49.37	\	0.53							
RLF4	72.20		0.60							
RLF5	38.59	3.60%	0.47	2.28%						
RLF6	38.72	3.28%	0.47	2.07%						
		14 phases								
RLF1	22.07	44.87%	0.33	30.10%						
RLF2	115.56	\	0.64							
RLF3	22.78	43.11%	0.34	29.01%						
RLF4	28.74	28.21%	0.39	18.54%						
RLF5	21.37	46.61%	0.33	31.70%						
RLF6	21.41	46.52%	0.33	31.48%						

Table 5.5: Time loss-based objectives results for the single-intersection scenario

In both scenarios, the Time Loss (**RLF5**) and Time Loss Pressure (**RLF6**) formulations we propose performed better than the others in cumulative time loss per driver and percentage time loss per driver. Specifically, in the single-intersection case, our RL formulations produced up to 47% and 31% improvement over the static baseline (3% improvement over Frap's third place RLF1), for the respective objectives, while in the multi-intersection case, those numbers go up to 55% and 31% improvement (6% improvement over Frap's third place RLF1). Besides delivering better results, the Time Loss and Time Loss Pressure formulations also achieve faster convergence (17%-34% –26-81 rounds– from Time Loss Pressure and 4%-5% –6-12 rounds– from Time Loss over Frap's third place RLF1) and enhanced stability (1.66-7.08 std. dev. after convergence from Time Loss versus Frap's third place RLF1 13.12-10.29) in all the experiments, as seen in Tables 5.7 and 5.8. Figures 5.4, 5.5, 5.6, and 5.7 show the results for all experiments.

Another important result is the difference in the time loss metrics between 8 and 14 phases. The addition of phases to the default 8-phases configuration (recall Figure 4.1c) improves up to 45% for the cumulative time loss per driver and up to 30% for the per-

Model	Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement						
Static	105.77	-	0.61	-						
Unregulated	7.22	-	0.10	-						
	8 phases									
RLF1	96.19	9.06%	0.59	3.53%						
RLF2	145.34		0.67							
RLF3	128.57	\	0.65	\						
RLF4	128.69	\	0.65	\						
RLF5	83.87	20.70%	0.56	8.86%						
RLF6	104.85	0.87%	0.61	0.42%						
		14 phases								
RLF1	50.42	52.33%	0.43	29.15%						
RLF2	132.11	\	0.63	\						
RLF3	60.12	43.17%	0.47	22.31%						
RLF4	99.50	5.93%	0.56	7.91%						
RLF5	47.28	55.30%	0.42	31.65%						
RLF6	48.87	53.79%	0.42	30.36%						

Table 5.6: Time loss-based objectives results for the multi-intersection scenario

centage time loss per driver, when considered the results for each group. The reason is that more movements are serviced at the same time when using the *conflicting-movements* phases.

RLF2, which uses Average Queue Length, achieved worse results than the static baseline in all experiments. An explanation is that similarly to the queue length objective (O3, Section 3.1), average queue length fails to capture the time loss as vehicles traveling at very low speeds are not enqueued, differently from Vehicles Been Stopped (**RLF1**), which only considers the queue cleared after the vehicles crossed the intersection. This modification results in better traffic control (up to 62% improvement in the multi-intersection case) when comparing both. Similarly, the MPLight Pressure (**RLF4**), which is also based on the queue length, presented worse results, although it still achieved a modest improvement in the 14-phase experiments.

#### 5.3.3.1 Average Travel Time Objective Comparison

To understand how the RL formulations perform considering travel time, we collected results for the already-discussed (in Section 4.1) Average Travel Time and Throughput objectives (both used by Chen [15]) alongside the Average Time Loss objective (the time

Table 5.7:	$\operatorname{RL}$	formulation	convergence	and	stability	$\operatorname{results}$	for	${\rm the}$	${\it single-intersection}$
scenario									

Model	Cumu Time per I	ılative - Loss Driver	Conve Ro	ergence ound	Std. af Conve	Dev. ter ergence
	early	stable	early	stable	early	stable
	I	8	phases	1	1	1
RLF1	-	44.93	-	123	-	1.71
RLF2	-	62.99	-	315	-	39.00
RLF3	49.37	50.32	153	310	21.37	3.80
RLF4	-	72.20	-	226	-	21.94
RLF5	-	38.59	-	120	-	1.29
RLF6	-	38.72	-	113	-	1.50
		14	phases			
RLF1	-	22.07	-	153	-	13.12
RLF2	130.11	115.56	201	254	41.17	41.92
RLF3	-	22.78	-	160	-	2.03
RLF4	-	28.74	-	196	-	37.93
RLF5	-	21.37	-	147	-	2.04
RLF6	-	21.41	-	127	-	1.66
	Percentage           Iodel         Time Loss					
Model	Perce Time per I	entage Loss Driver	Conve Ro	ergence ound	Std. af Conve	Dev. ter ergence
Model	Perce Time per I early	entage Loss Driver stable	Conve Ro early	ergence ound stable	Std. af Conve early	Dev. ter ergence stable
Model	Perce Time per I early	entage 2 Loss Driver stable 8	Conve Ro early phases	ergence ound stable	Std. af Conve early	Dev. ter ergence stable
Model RLF1	Perce Time per I early	entage Loss Driver stable 8 0.51	Conve Ro early phases -	ergence bund stable 126	Std. af Conve early -	Dev. iter ergence stable
Model RLF1 RLF2	Perce Time per I early -	entage • Loss Driver stable 8 0.51 0.58	Conve Ro early phases - -	ergence ound stable 126 332	Std. af Conve early - -	Dev. ter srgence stable 0.01 0.06
Model RLF1 RLF2 RLF3	Perce Time per I early - - 0.53	entage 2 Loss Driver stable 8 0.51 0.58 0.53	Converse Ro early phases - - 157	stable 126 332 308	Std. af Conve early - - - 0.05	Dev. iter ergence stable 0.01 0.06 0.02
Model RLF1 RLF2 RLF3 RLF4	Perce Time per I early - 0.53 -	entage 2 Loss Driver stable 8 0.51 0.58 0.53 0.60	Converse Ro early phases - 157 -	ergence bund stable 126 332 308 223	Std. af Conve early - - 0.05 -	Dev. ter ergence stable 0.01 0.06 0.02 0.04
Model RLF1 RLF2 RLF3 RLF4 RLF5	Perce Time per I early - 0.53 - -	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47	Converse Ro early phases - - 157 - - - - -	ergence bund stable 126 332 308 223 120	Std. af Conve early - - - 0.05 - - -	Dev. iter stable 0.01 0.06 0.02 0.04 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF5	Perce Time per I early - 0.53 - - -	entage 2 Loss Driver 8 stable 0.51 0.58 0.53 0.60 0.47 0.47	Converse Ro early phases - 157 - - - - - - -	ergence         ound         stable         126         332         308         223         120         125	Std. af Conve early - - 0.05 - - - - -	Dev. ter stable 0.01 0.06 0.02 0.04 0.01 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6	Perce Time per I early - 0.53 - - - -	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47 0.47 14	Converse Ro early phases - - 157 - - - - - - - - - - - - - - - - - - -	ergence ound stable 126 332 308 223 120 125	Std. af Conve early - - - 0.05 - - - - -	Dev. iter ergence stable 0.01 0.06 0.02 0.04 0.01 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1	Perce Time per I early - 0.53 - - - 0.33	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47 0.47 14 0.33	Converse Ro early phases - - 157 - - - - - - - - - - - - - - - - - - -	ergence bund stable 126 332 308 223 120 125 313	Std. af Conve early - - 0.05 - - - 0.05	Dev. ter srgence stable 0.01 0.06 0.02 0.04 0.01 0.01 0.01 0.02
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF1 RLF2	Perce Time per I early - 0.53 - - 0.33 0.69	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47 0.47 14 0.33 0.64	Conve Ro early phases - 157 - - - - - - - - - - - - 225	ergence ound stable 126 332 308 223 120 125 313 360	Std. af Conve early - - - 0.05 - - - - - 0.06 0.06	Dev. ter stable 0.01 0.06 0.02 0.04 0.01 0.01 0.01 0.02 0.02 0.08
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF2 RLF2 RLF3	Perce Time per I early - 0.53 - - 0.53 - - 0.33 0.69 -	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47 0.47 14 0.33 0.64 0.34	Converse Ro early phases - 157 - - - - - - - - - - - - - - - - - - -	ergence         stable         126         332         308         223         120         125         313         360         160	Std. af Conve early - 0.05 - - - 0.05 - - 0.06 0.06 0.06 -	Dev. ter stable 0.01 0.06 0.02 0.04 0.01 0.01 0.01 0.02 0.08 0.02
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF2 RLF3 RLF3 RLF4	Perce Time per I early - 0.53 - - 0.53 - - 0.53 - - - - - - - - - - - - - - - - - - -	entage Loss Driver stable 8 0.51 0.58 0.53 0.60 0.47 0.47 14 0.33 0.64 0.34 0.39	Converse Ro early phases - - 157 - - - - - - - - - - 225 - - - - - - - -	ergence ound stable 126 332 308 223 120 125 313 360 160 196	Std. af Conve early - - - 0.05 - - - - 0.06 - - - - - - - - - - - - - - - - - - -	Dev. ter stable 0.01 0.06 0.02 0.04 0.01 0.01 0.01 0.02 0.02 0.08 0.02 0.02 0.10
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF2 RLF3 RLF4 RLF3 RLF4 RLF4	Perce Time per I early - 0.53 - - 0.53 - - 0.53 - - - - - - - - - - - - - - - - - - -	entage Loss Driver stable 8 0.51 0.53 0.60 0.47 0.47 14 0.33 0.64 0.34 0.39 0.33	Converse Ro early phases - - 157 - - - - - - - - - - - - - - - - - - -	ergence         stable         126         332         308         223         120         125         313         360         160         196         147	Std. af Conve early - - 0.05 - - - 0.06 0.06 0.06 - - - -	Dev. ter stable 0.01 0.06 0.02 0.04 0.01 0.01 0.01 0.02 0.08 0.02 0.08 0.02 0.10 0.02

loss equivalent of the average travel time metric).

As we can see in Tables 5.9 and 5.10, the results follow the same trend presented by the time loss-based objective results (Tables 5.5 and 5.6), with Time Loss **(RLF5)** and Time

Model	Cumu Time per I	Cumulative Time Loss per DriverConvergence RoundStd. I afte Convergence		Convergence Round		Dev. ær rgence
	early	stable	early	stable	early	stable
		8	phases			1
RLF1	-	96.19	-	220	-	2.47
RLF2	322.78	145.34	83	320	100.83	42.65
RLF3	-	128.57	-	116	-	16.18
RLF4	-	128.69	-	160	-	17.28
RLF5	-	83.87	-	122	-	8.22
RLF6	-	104.85	-	127	-	4.32
		$1_{4}$	4 phases	5		
RLF1	70.77	50.42	153	242	23.59	10.29
RLF2	132.11	269.22	133	250	59.82	28.13
RLF3	64.33	60.12	190	248	38.02	16.41
RLF4	99.5	199.43	293	360	59.26	37.61
RLF5	74.19	47.28	182	230	17.14	8.78
RLF6	-	48.87	-	161	-	7.08
		Percentage Time Loss				
Model	Perce Time per I	entage 2 Loss Driver	Conve Ro	ergence und	Std. aft Conve	Dev. Jer rgence
Model	Perce Time per I early	entage Loss Driver stable	Conve Ro early	ergence und stable	Std. aft Conver early	Dev. er rgence stable
Model	Perce Time per I early	entage e Loss Driver stable 8	Conve Ro early phases	ergence und stable	Std. aft Conver early	Dev. er rgence stable
Model	Perce Time per I early	entage 2 Loss Driver stable 8 0.59	Conve Ro early phases	ergence und stable 220	Std. aft Conver early -	Dev. cer rgence stable
Model RLF1 RLF2	Perce Time per I early -	entage e Loss Driver stable 0.59 0.67	Conve Ro early phases - -	ergence und stable 220 382	Std. aft Conve early - -	Dev. ser rgence stable 0.01 0.03
Model RLF1 RLF2 RLF3	Perce Time per I early - - -	entage e Loss Driver stable 8 0.59 0.67 0.65	Conversion Ro early phases - - - -	ergence und stable 220 382 116	Std. aft Conver early - - -	Dev. er rgence stable 0.01 0.03 0.02
Model RLF1 RLF2 RLF3 RLF4	Perce Time per I early - - - -	entage e Loss Driver stable 8 0.59 0.67 0.65 0.65	Conve Ro early phases - - - - -	ergence           und           stable           220           382           116           160	Std. aft Conve early - - - -	Dev. ser rgence stable 0.01 0.03 0.02 0.02
Model RLF1 RLF2 RLF3 RLF4 <b>RLF5</b>	Perce Time per I early - - - - - - - -	entage 2 Loss Driver 8 stable 0.59 0.67 0.65 0.65 0.56	Converse Ro early phases - - - - - - - - -	ergence und stable 220 382 116 160 <b>122</b>	Std. aft Conver early - - - - - - -	Dev. der rgence stable 0.01 0.03 0.02 0.02 0.02 0.01
Model RLF1 RLF2 RLF3 RLF4 <b>RLF5</b> RLF6	Perce Time per I early - - - - - - - - - -	entage 2 Loss Driver 8 stable 0.59 0.67 0.65 0.65 0.56 0.61	Conve Ro early phases - - - - - - - - - - -	220           382           116           160           122           127	Std. aft Conve early - - - - - - - - - - -	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.02 0.01 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6	Perce Time per I early - - - - - - - - -	entage 2 Loss Driver 8 stable 0.59 0.65 0.65 0.65 0.65 0.61 14	Conve Ro early phases - - - - - 4 phases	ergence and stable 220 382 116 160 <b>122</b> 127	Std. aft Conver early - - - - - - - - - -	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.02 0.01 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1	Perce Time per I early - - - - - - - - - - - - - - - - - - -	entage 2 Loss Driver 8 stable 0.59 0.65 0.65 0.65 0.65 0.61 14 0.45	Conve Ro phases - - - - - 4 phases 170	ergence         und         stable         220         382         116         160         122         127         5         290	Std. aft Conve early - - - - - - - - - - 0.05	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.02 0.01 0.01 0.01 0.03
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF1 RLF1	Perce Time per I early - - - - - 0.49 0.63	entage Loss Driver stable 8 0.59 0.65 0.65 0.65 0.65 0.61 14 0.45 0.76	Conve Ro early phases - - - - 4 phases 170 133	ergence and stable 220 382 116 160 122 127 5 290 250	Std. aft Conver early - - - - - 0.05 0.05	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.01 0.01 0.03 0.01
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF2 RLF2 RLF3	Perce Time per I early - - - - - - - - - - 0.49 0.63 0.49	entage 2 Loss Driver 8 stable 0.59 0.67 0.65 0.65 0.65 0.61 14 0.45 0.76 0.47	Conve Ro early phases - - - - 4 phases 170 133 190	ergence         und         stable         220         382         116         160         122         127         5         290         250         248	Std. aft Conve early - - - - - - 0.05 0.05 0.06	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.01 0.01 0.03 0.01 0.03 0.01 0.04
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF1 RLF2 RLF3 RLF3 RLF4	Perce Time per I early - - - - - 0.49 0.63 0.49 0.56	entage Loss Driver stable 8 0.59 0.65 0.65 0.65 0.65 0.61 14 0.45 0.76 0.47 0.69	Conve Ro early phases - - - - 4 phases 170 133 190 293	ergence and stable 220 382 116 160 122 127 5 290 250 248 360	Std. aft Conver early - - - - - - 0.05 0.05 0.05 0.06 0.07	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.01 0.01 0.03 0.01 0.03 0.01 0.04 0.04
Model RLF1 RLF2 RLF3 RLF4 RLF5 RLF6 RLF1 RLF2 RLF3 RLF4 RLF3 RLF4 RLF5	Perce Time per I early - - - - - - - 0.49 0.63 0.49 0.56 0.45	entage 2 Loss Driver stable 8 0.59 0.67 0.65 0.65 0.65 0.61 14 0.45 0.76 0.47 0.69 0.42	Conve Ro early phases - - - - 4 phases 170 133 190 293 162	ergence         und         stable         220         382         116         160         122         127         5         290         250         248         360         230	Std. aft Conve: early - - - - - - 0.05 0.05 0.05 0.06 0.07 0.06	Dev. ser rgence stable 0.01 0.03 0.02 0.02 0.02 0.01 0.01 0.03 0.01 0.04 0.04 0.04 0.03

Table 5.8: RL formulation convergence and stability results for the multi-intersection scenario

Loss Pressure (**RLF6**) as the best results. The conclusion is that the RL formulations presented consistent results between the different objectives addressed. Also, as discussed in Section 4.1.1, optimizing the time loss implies in optimizing the travel time since the former is naturally included in the latter.



Figure 5.4: Single-Intersection (8 phases) learning process

Also, in Section 4.1 we discussed the throughput role as co-objective for assessing the average travel time and that some methods might produce a traffic shaping behavior to get lower travel times. In the multi-intersection scenario, for example, RLF4 (14 phases) has, at the same time, lower average travel time and lower throughput than Static, for example. Our time loss-based proposition considers the total of vehicles (departed + pending) instead, which is consistent for all methods, not requiring the throughput as co-objective (as discussed in Section 4.1).



Figure 5.5: Single-Intersection (14 phases) learning process

#### 5.3.3.2 Impact of Pending Vehicles on Metrics

Similar to the average travel time, the average time loss also does not account for pending vehicles. When considering pending vehicles, each trip begins when the vehicle is supposed to enter the simulation grid, increasing its travel time and time loss in the cases where the vehicle delays its actual entry. Tables 5.11 and 5.12 present the impact of pending vehicles in both metrics.

Intuitively, the impact of pending vehicles on the metrics is more significant as the traffic becomes heavier, which in turn, it is more likely to occur on denser and more complex scenarios. However, even in the simple scenarios presented, this effect can be



Figure 5.6: Multi-Intersection (8 phases) learning process

seen. For example, in the multi-intersection scenario, the static controller yields on average 8.06 seconds per trip of time loss (approximately 8% of the total time loss) from pending vehicles, where the best result reduces this "extra" time loss to 0.02 seconds (about 0.02% of the total) per trip. That is, not to account for the time loss of pending vehicles can mask the actual improvement obtained by better methods, changing from 49% (without pending vehicles) to 53% improvement in the average time loss objective results. Our proposed time loss-based objectives account for pending vehicles and, therefore, they do not suffer from this discrepancy.

In the average time loss metric, similar to the average travel time, the vehicles need



Figure 5.7: Multi-Intersection (14 phases) learning process

to finish their trips so the objective can account for their time losses. Consequently, at intermediate steps, the average time loss only approximates the cumulative time loss per driver objective. Eventually, when all vehicles finish their trips, both objective results match.

# 5.4 Planning Experimental Results

In this section, we experiment with Planning (for non-planning experiments, check Section 5.3). Section 5.4.1 discusses what planning alone looks like. Section 5.4.2 evaluates

Model	Average Travel Time	Improv.	Throughput	Improv.	Average Time Loss	Improv.
Static	83.96	-	4181	-	40.45	-
Unregulated	47.80	-	4218	-	4.26	-
			8 phases			
RLF1	88.52		4170		45.01	
RLF2	103.47		4144	\	59.99	
RLF3	92.88		4165		49.37	
RLF4	108.95		4119	\	65.53	
RLF5	82.35	1.91%	4182	0.02%	38.84	3.99%
RLF6	82.47	1.77%	4180		38.96	3.70%
			14 phases	·	·	
RLF1	65.83	21.59%	4204	0.54%	22.31	44.85%
RLF2	113.95		3902		70.65	
RLF3	66.55	20.73%	4204	0.55%	23.03	43.06%
RLF4	72.55	13.59%	4201	0.48%	29.03	28.23%
RLF5	65.14	22.42%	4205	0.56%	21.61	46.57%
RLF6	65.16	22.39%	4205	0.57%	21.64	46.51%

Table 5.9: Other objectives' results for the single-intersection scenario

Table 5.10: Other objectives' results for the multi-intersection scenario

Model	Average Travel Time	Improv.	Throughput	Improv.	Average Time Loss	Improv.			
Static	161.01	-	8128	-	96.25	-			
Unregulated	72.41	-	8466	-	7.28	-			
8 phases									
RLF1	159.82	0.74%	8198	0.86%	94.91	1.40%			
RLF2	190.29		7857		125.76				
RLF3	192.66		8023	\	127.99				
RLF4	190.10		7988		125.44				
RLF5	148.17	7.97%	8221	1.15%	83.24	13.52%			
RLF6	167.01	\	8147	0.23%	102.21				
		-	14 phases						
RLF1	115.64	28.18%	8339	2.59%	50.59	47.44%			
RLF2	172.92		7923		108.32				
RLF3	124.95	22.40%	8301	2.13%	59.94	37.73%			
RLF4	149.92	6.89%	8016	\	85.20	11.48%			
RLF5	112.56	30.09%	8351	2.74%	47.51	50.64%			
RLF6	114.18	29.08%	8346	2.68%	49.13	48.96%			

Model	Average Travel Time	+ pending vehicles	Average Time Loss	+ pending vehicles	Cumulative Time Loss per Driver
Static	83.96	+ 0.01	40.45	+ 0.01	40.03
Unregulated	47.80	+ 0.01	4.26	+ 0.01	4.23
		8 ph	ases		
RLF1	88.52	+ 0.01	45.01	+ 0.01	43.93
RLF2	103.47	+ 2.61	59.99	+ 2.61	62.99
RLF3	92.88	+ 0.01	49.37	+ 0.01	49.30
RLF4	108.95	+ 6.37	65.53	+ 6.37	72.20
RLF5	82.35	+ 0.01	38.84	+ 0.01	38.59
RLF6	82.47	+ 0.01	38.96	+ 0.01	38.72
		14 ph	ases		
RLF1	65.83	+ 0.01	22.31	+ 0.01	22.07
RLF2	113.95	+ 33.28	70.65	+ 33.28	115.56
RLF3	66.55	+ 0.01	23.03	+ 0.01	22.78
RLF4	72.55	+ 0.05	29.03	+ 0.05	28.74
RLF5	65.14	+ 0.01	21.61	+ 0.01	21.37
RLF6	65.16	+ 0.01	21.64	+ 0.01	21.41

Table 5.11: Pending vehicles impact on metrics for the single-intersection scenario

Table 5.12: Pending vehicles impact on metrics for the multi-intersection scenario

Model	Average Travel Time	+ pending vehicles	Average Time Loss	$+ { m pending} { m vehicles}$	Cumulative Time Loss per Driver		
Static	161.01	+ 8.06	96.25	+ 8.06	105.77		
Unregulated	72.41	+ 0.01	7.28	+ 0.01	7.22		
8 phases							
RLF1	159.82	+ 0.32	94.91	+ 0.32	96.19		
RLF2	190.29	+ 9.89	125.76	+ 9.89	145.34		
RLF3	192.66	+ 4.31	127.99	+ 4.31	135.08		
RLF4	190.10	+ 2.49	125.44	+ 2.49	133.58		
RLF5	148.17	+ 0.12	83.24	+ 0.12	83.87		
RLF6	167.01	+ 0.70	102.21	+ 0.70	104.85		
14 phases							
RLF1	115.64	+ 0.04	50.59	+ 0.04	50.42		
RLF2	172.92	+ 16.93	108.32	+ 16.93	132.11		
RLF3	124.95	+ 0.06	59.94	+ 0.06	60.12		
RLF4	149.92	+ 8.26	85.20	+ 8.26	99.50		
RLF5	112.56	+ 0.03	47.51	+ 0.03	47.28		
RLF6	114.18	+ 0.02	49.13	+ 0.02	48.87		

different types of Planning and some policies for action-selection. Section 5.4.3 analyzes how the learning sample generation (*i.e.*, interactions with the environment) deals with the learning sample size configuration (*i.e.*, the portion of the interactions are really used during learning). Finally, Section 5.4.4 reviews the RL formulation tests with the Frap+Planning case.

## 5.4.1 Planning Only

In this experiment, we test the 2 phases set and 14 phases set configurations (PSC7 and PSC4, respectively, from Section 5.3.2) to evaluate the planning component alone. As explained in Section 5.2, the PlanningOnly agent takes the next action based solely on the total reward from future paths, that is, without any learning component (essentially a limited-depth breadth-first search). While the extensive search for the best path is clearly an exhaustive and impractical task, it does bring a sense of how learning methods work with and without planning. We test the 2-phases set with 3, 6, 9, and 12 iterations ahead (8, 64, 512, 4096 paths, respectively) and the 14-phases set with 3 iterations (2744 paths), where each iteration corresponds to 10 seconds. We use Frap's Vehicle been stopped - RLF1 - formulation ( $\langle$ Number of Vehicles, Vehicle Been Stopped $\rangle$ ) in a single intersection for this experiment.

Model		Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement		
Static		40.03	-	0.48	-		
Unregulated		4.23	-	0.09	-		
2 phases							
RLF1	3 iterations	46.49	\	0.50	\		
	6 iterations	40.49	\	0.47	2.08%		
	9 iterations	41.67		0.48	\		
	12 iterations	47.16	\	0.51	\		
14 phases							
RLF1	3 iterations	65.19	\	0.59	\		

Table 5.13: Time loss-based objectives results for 'Planning Only'

As we can see in Table 5.13, the simple planning itself did not achieve great results for the tested future extent, staying even behind the static configuration (between 40 and 46 for cumulative time loss per driver and 0.47 and 0.50 for percentage time loss per driver). This result denotes that planning alone may need to look way further to achieve comparable results and that RL solutions are implicitly doing much more than anticipating 12 iterations (120s).

The result of 9 iterations performing slightly worse than 6 and 12 worse than 3 iterations could be due to moments where more than one path yields the best reward, and the wrong choice can lead to worse later paths. We checked the time loss formulation for the same configurations, and the same behavior happened (34.23 and 33.73 for 9 and 6 iterations, respectively). Furthermore, the 14 phases 3 iterations result also points to the same conclusion, since all phases from the 2 phases set are also present in the 14 phases set, but the latter was significantly worse than the former for the same number of iterations along with having more action options to take a sub-optimal path.

Model		Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement	
Static		40.03	-	0.48	-	
Unregulated		4.23	-	0.09	-	
2 phases						
RLF1	3 iterations	46.49	\	0.50	\	
	6 iterations	40.49	\	0.47	2.08%	
RLF2	3 iterations	221.26	\	0.79	\	
	6 iterations	195.33	\	0.77	\	
RLF3	3 iterations	37.60	6.07%	0.46	4.17%	
	6 iterations	35.08	12.37%	0.44	8.33%	
RLF4	3 iterations	221.26	\	0.79	\	
	6 iterations	195.33		0.77	\	
RLF5	3 iterations	34.72	13.27%	0.44	8.33%	
	6 iterations	33.73	15.74%	0.43	10.42%	
RLF6	3 iterations	37.72	5.77%	0.46	4.17%	
	6 iterations	34.37	14.14%	0.44	8.33%	

Table 5.14: Different RL formulation results for 'Planning Only'

We also test the others RL formulations - RLF2, RLF3, RLF4, RLF5, and RLF6 - (recall Table 5.4) to verify how they differ in a scenario without any learning component (see Table 5.14). Compared to the baseline, the other formulations had a similar outcome, that is, equally distant from the non-planning results. Looking only for the RL formulations, RLF3 and our time loss formulations, RLF5, and RLF6, achieved better results when looking at three and six actions in the future (between 34 and 38 for cumulative time loss per driver, and 0.43 and 0.46 for percentage time loss per driver). Although the planning alone did not yield good results, at least up to the tested horizon of 120 seconds,
these results indicate that our time loss formulations are more suitable for representing the objective since the rewards alone led to better paths. As a further matter, the same reason why RLF2 and RLF4 did not work well in the RL formulation experiment (Section 5.3.3), that is, the notion of queue length does not adequately represent the time loss objective, also applies here.

#### 5.4.2 Types of Planning and Action Sampling Policy

To test Frap+Planning, we have to introduce some parameters of Planning and verify which configuration works best for the TSC problem. In this section, we evaluate the different types of planning and the action sampling policies.

For the different types of planning, recall from Section 2.2.3 we talked that Planning can be used either as background planning, decision-time planning, or both at the same time. That is, the agent can use future-interactions to either or both generate more learning samples and pick final actions. The action sampling policy, in turn, is necessary because we might want to test subsets of the actions instead of testing them all to avoid a combinatorial explosion from searching the future-state-space tree. For this parameter, we selected different configurations: best action (according to the agent's current policy), random action, and exploration-exploitation (the same trade-off found in Learning). Note that some of the configurations are not compatible, for example, we do not want to have 'random action' combined with the decision-time planning since this would defeat the reward exploitation. The combinations of planning and action sampling policy tested are the following: (background planning, random action), (background planning, best action), (decision-time planning, best action), (background and decisiontime planning, best action,  $\langle background planning, exploration-exploitation \rangle$ ,  $\langle decision$ time planning, exploration-exploitation, and (background and decision-time planning, exploration-exploitation, summarized in Table 5.15.

Concerning other parameters, for this experiment, we use the single intersection scenario with the Frap's Vehicle been stopped (RLF1) RL formulation, an action sampling size of 2 (*i.e.*, two sampled actions per iteration), and two iterations ahead (more about the latter two in Section 5.4.3). We chose these parameters because we wanted a simple planning baseline for further experimentation. Table 5.16 summarizes the results, where *early* and *stable* refers to different moments of convergence, being early either if the agent diverges afterwards or suddenly converges later at another result. In this experiment, the non-planning result is the baseline.

Configuration	Planning Type	Action Sampling Policy	
BP-RA	background planning	random action	
BP-BA	background planning	best action	
DTP-BA	decision-time planning	best action	
	background and	host action	
D&D11-DA	decision-time planning	Dest action	
BP-EE	background planning	exploration-exploitation	
DTP-EE	decision-time planning	exploration-exploitation	
B&DTP FF	background and	ovploration ovploitation	
	decision-time planning	exploration-exploitation	

Table 5.15: Types of planning and action sampling tested

Since the best RL formulations presented a very stable result for the non-Planning experiments, planning must at least improve time savings or convergence to make adding it advantageous. From the results, BP-BA, DTP-BA, BP-EE, DTP-EE achieved comparable results when considering only the time loss; however, only DTP-BA and DTP-EE also achieved comparable convergence (although slightly worse than no planning). All four of them also achieved good stability, with highlights for BP-BA and BP-EE being very close to no variation. However, combining background planning and decision-time planning did not work well, being significantly worse than their results when applied separately. We should note that we are using the same future experience for both strategies, while a model would make it easy to test this strategy with different action selections.

Concerning the action sampling policy, BA and EE results did not differ so much from each other, suggesting that the 'best actions' at the start of learning may appear as good as mainly taking random actions. Later in the learning process, when the actions are better known, EE favors mostly 'best actions', being naturally closer to BA.

Lastly, BP-RA had an interesting result. Although BP-RA has the worst time loss result in this experiment, its early convergence reached 58% of the 'No Planning' stable convergence. If by adjusting the Planning parameters, the learning becomes more stable, it can be a good configuration candidate for incorporating Planning to RL TSC. We are going to explore more of BP-RA in the following experiments.

Model	Cumulative Time Loss per Driver		Convergence Round		Std. Dev. after Convergence	
	early	early stable		stable	early	stable
No Planning	-	22.07	-	153	-	13.12
BP-RA	38.61	68.07	88	150	39.16	35.68
BP-BA	-	22.41	-	227	-	2.15
DTP-BA	-	22.20	-	158	-	21.78
B&DTP-BA	44.91	53.48	228	290	115.13	96.36
BP-EE	-	21.84	-	235	-	2.32
DTP-EE	-	22.22	-	168	-	10.10
B&DTP-EE	-	62.55	-	220	-	96.18

Table 5.16: Types of planning and action sampling policy results

# 5.4.3 Action Sampling Size, Number of Iterations, and Sample Size

When using Planning, we want to control how far we look in the future (search tree depth) and how many actions we consider each time (search tree breadth). For this, we adjust, respectively, the number of the iterations and the action sampling size (we already discussed how to choose the future actions in Section 5.4.2). Furthermore, the different settings of those two parameters, given the use of background planning, also impact how many learning samples are generated in the learning phase (see Table 5.17). Given this, the sample size parameter (*i.e.*, how much experience policy update uses) may need to change to keep up with a larger sample generation. Specifically, in this Section, we want to analyze how these parameters work together and their impacts on learning speed (wall-clock time and learning convergence).

Action Sampling Size	Number of Iterations	Learning Samples Generated by Round
No P	lanning	1080
2	2	7560 (7x)
5	2	33480 (31x)
2	5	68040 (63x)
3	2	14040 (13x)
2	3	16200 (15x)
14	1	16200 (15x)
1	14	16200 (15x)

Table 5.17: Sample size based on action sampling size and number of iterations

We want to measure how long it would take to update the policy when generating more samples, regardless of the extra simulation time, since a model can be used instead. For that concern, we measured how long it takes for sample size values of 1,000 (the default parameter in non-planning experiments), 10,000, 20,000, 30,000, 50,000, 100,000, and 1,000,000. We present relative numbers because actual time may vary depending on hardware and software aspects. See Table 5.18 for the update time results, where 1x is approximately four minutes, obtained empirically.

Sample Size	Policy Update Wall-clock Time Per-round
1000	1x
10000	1.94x
20000	2.27x
30000	4.80x
50000	8.17x
100000	17.50x
1000000	*(several hours)

Table 5.18: Sample size and policy update time

As sample size increases, time per round becomes unacceptable, in such a way that 100,000 takes approximately an hour to update the policy. Given that, we canceled the planned experiments "2; 5; 100,000" and "5; 2; 100,000" (action sampling size; number of iterations; sample size) and limited our tests to sample sizes at a maximum of 20,000. That way, we also could test more action sampling and iteration options.

Next, we want to assess how far we can go with the parameters and check if the sample size should follow the amount of experience generated. For this, we picked configurations that generate much more learning samples (*i.e.*, 7x, 13x, 15x, 31x, and 63x) and tested with different values of sample size. For the experiments between 10x and 20x of sample generation, we test samples sizes of 10,000 and 20,000 to verify if the amount of learning sample used does any difference. For the configurations which generate more than 20x learning samples, we tested sample sizes of 1,000 and 10,000 since they take longer in the simulation.

The configurations are:

- (1)  $\langle 2 \text{ sampled actions, } 2 \text{ iterations, sample size of } 1,000 \rangle$ ,
- (2)  $\langle 2 \text{ sampled actions, } 2 \text{ iterations, sample size of } 10,000 \rangle$ ,

- (3)  $\langle 5 \text{ sampled actions, } 2 \text{ iterations, sample size of } 1,000 \rangle$ ,
- (4)  $\langle 5 \text{ sampled actions, } 2 \text{ iterations, sample size of } 10,000 \rangle$ ,
- (5)  $\langle 2 \text{ sampled actions, 5 iterations, sample size of } 1,000 \rangle$ ,
- (6)  $\langle 2 \text{ sampled actions, 5 iterations, sample size of } 10,000 \rangle$ ,
- (7)  $\langle 3 \text{ sampled actions, } 2 \text{ iterations, sample size of } 10,000 \rangle$ ,
- (8)  $\langle 3 \text{ sampled actions, } 2 \text{ iterations, sample size of } 20,000 \rangle$ ,
- (9)  $\langle 2 \text{ sampled actions}, 3 \text{ iterations}, \text{ sample size of } 10,000 \rangle$ ,
- (10)  $\langle 2 \text{ sampled actions}, 3 \text{ iterations}, \text{ sample size of } 20,000 \rangle$ ,
- (11)  $\langle 14 \text{ sampled actions, 1 iteration, sample size of } 10,000 \rangle$ ,
- (12)  $\langle 14 \text{ sampled actions, 1 iteration, sample size of } 20,000 \rangle$ ,
- (13)  $\langle 1 \text{ sampled action, } 14 \text{ iterations, sample size of } 10,000 \rangle$ , and
- (14)  $\langle 1 \text{ sampled action, } 14 \text{ iterations, sample size of } 20,000 \rangle$ .

The results are presented in Table 5.19, where "No Planning" is the baseline, *early* and *stable* refers to different moments of convergence, *actn.* is the action sampling size, *iter.* is the number of iterations, and *samp.* is the sample size parameter.

From the results, (1), (3), (9), and (11) diverged, with (3) heavily diverging, (2) and (7) achieved comparable time loss results along with a slightly quicker convergence, (4) and (5) stabilized the converge too late, although (5) achieved closer time loss with a very early convergence. Finally, (6) and particularly (13) did not converge enough. While (1) and (3) dramatically improved their result when raising the sample size from 1,000 to 10,000, the same did not happen with (5). Based on this outcome, we could not find any correlation between different sample sizes or between different numbers of actions and iterations. Therefore we consider this as a significant setback for Frap+Planning.

#### 5.4.4 RL Formulation and Planning

Here we test how Planning impact RL formulation results and their convergence time when compared to non-Planning and if the RL formulations are influenced equally. For this we test the RL formulations which presented the best results from Section 5.3.3. They are (see

Model			Cumulative Time Loss per Driver		Convergence Round		Std. Dev. after Convergence	
planning	actn; iter.; samp.	id	early	stable	early	stable	early	stable
No Planning		(0)	-	22.07	-	153	-	13.12
	2; 2; 1000	(1)	38.61	68.07	88	150	39.16	35.68
	2; 2; 10000	(2)	26.52	21.42	105	127	16.23	7.69
	5; 2; 1000	(3)	52.58	156.62	35	205	81.32	73.38
	5; 2; 10000	(4)	36.17	27.28	75	240	60.98	83.47
	2; 5; 1000	(5)	24.63	21.85	50	240	22.03	25.99
	2; 5; 10000	(6)	36.02	46.06	65	356	32.65	20.46
BDBV	3; 2; 10000	(7)	-	23.33	-	119	-	21.72
DI -IIA	3; 2; 20000	(8)	23.04	23.12	103	355	48.99	6.81
	2; 3; 10000	(9)	26.93	44.90	97	283	50.07	18.64
	2; 3; 20000	(10)	42.56	22.22	56	370	86.27	1.73
	14; 1; 10000	(11)	32.87	51.13	82	120	29.49	28.52
	14; 1; 20000	(12)	28.06	22.72	66	357	38.48	13.63
	1; 14; 10000	(13)	64.91	60.14	135	200	32.95	35.10
	1; 14; 20000	(14)	71.45	23.22	89	330	32.80	2.20

Table 5.19: Action sampling size, number of iterations, and sample size results

Table 5.20): (**RLF1**) (Number of Vehicles, Vehicle Been Stopped); (**RLF5**) (Time Loss, Time Loss); and (**RLF6**) (Time Loss Pressure, Time Loss Pressure). We focus on the 14 phases set configuration in the single-intersection for this test and, since the results from Section 5.4.3 were somewhat inconclusive, we stick to the original 2 actions and 2 iterations and test two options of sample (1,000 and 10,000). Since the planning experiments seem more volatile in their learning, we ran 3 executions from each formulation and sample size, averaging the results at each group. The results are presented in Table 5.21. For the full convergence curves, see Figures 5.8, 5.9, 5.10, 5.11, 5.12, and 5.13, where *no planning* refers to the RLF6 (no planning) configuration.

Table 5.20: RL formulations tested

<b>RL</b> Formulation	State		Reward
	DI E1		number of vehicles
		(per movement)	have been stopped
DIF	current phase	time loss	time loss
IVEL D		(per movement)	
<b>BIE</b> 6		pressure [time loss]	prossure [time loss]
IULF 0		(per movement)	pressure [time loss]

The Planning helps the agent in the major convergence part (*i.e.*, the beginning of the

learning), achieving acceptable results with far fewer rounds (53-92 rounds, or 42%-72%, earlier in some cases). Still, right after it, the planning seems to obstruct the learning, making it, when not diverge, slowly converge to comparable results. Considering only the average results, RLF6 (sample of 1,000) achieved the best planning result. However, it still did not manage to stay ahead of its non-planning counterpart or other non-planning experiments. Similarly to non-planning experiments, the time loss formulations are more stable after convergence.

The results suggest that the generated extra experience becomes irrelevant, even detrimental to learning, right after converging faster, meaning that either a better planning algorithm should be used or that experiences should be filtered or worked on before serving as learning samples. One possible solution would be to filter repetitive interactions out to balance the importance the agent gives when sampling experience.

Model		Cumulative Time Loss per Driver		Convergence Round		Std. Dev. after Convergence		
planning	samp.	rl form. (try)	early	stable	early	stable	early	stable
	1000	RLF1	-	22.07	-	153	-	13.12
No Planning	1000	RLF5	-	21.37	-	147	-	2.04
	1000	RLF6	-	21.41	-	127	-	1.66
	1000	RLF1 (1st)	38.61	68.07	88	150	39.16	35.68
BP-RA	1000	RLF1 (2nd)	23.64	30.26	100	240	25.21	27.24
	1000	RLF1 (3rd)	40.32	22.57	97	300	31.95	7.97
	1000	RLF1 (avg)	34.19	40.30	95	230	32.11	23.63
	1000	RLF5 $(1st)$	27.05	32.61	82	240	13.90	11.75
BP-RA	1000	RLF5 $(2nd)$	23.22	24.20	94	160	9.78	7.18
	1000	RLF5 (3rd)	31.94	27.49	78	200	22.27	5.04
	1000	RLF5 (avg)	27.40	28.10	85	200	15.32	7.99
	1000	RLF6 (1st)	32.21	35.02	120	154	10.52	9.95
	1000	RLF6 (2nd)	-	21.15	-	71	-	2.81
DF-NA	1000	RLF6 (3rd)	-	21.25	-	82	-	4.66
	1000	RLF6 (avg)	24.87	25.81	91	102	6.00	5.81
	10000	RLF1 $(1st)$	26.52	21.42	105	127	16.23	7.69
	10000	RLF1 (2nd)	27.07	60.15	121	352	30.11	27.83
DF-NA	10000	RLF1 (3rd)	-	24.11	-	158	-	21.53
	10000	RLF1 (avg)	25.90	35.23	128	212	22.62	19.02
	10000	RLF5 $(1st)$	31.15	27.43	78	285	21.65	3.56
	10000	RLF5 $(2nd)$	-	22.27	-	96	-	5.97
DF-NA	10000	RLF5 $(3rd)$	28.93	20.91	89	248	20.65	5.73
	10000	RLF5 $(avg)$	27.45	23.70	88	210	16.10	5.09
	10000	RLF6 (1st)	36.61	21.74	74	158	24.48	35.31
	10000	RLF6 (2nd)	27.31	23.39	61	174	27.81	10.42
Dr-nA	10000	RLF6 (3rd)	-	24.52	-	123	-	4.25
	10000	RLF6 (avg)	29.48	23.22	86	152	18.85	16.66

Table 5.21: RL formulation results



Figure 5.8: Planning RLF1 (sample of 1000) results



Figure 5.9: Planning RLF5 (sample of 1000) results



Figure 5.10: Planning RLF6 (sample of 1000) results



Figure 5.11: Planning RLF1 (sample of 10000) results



Figure 5.12: Planning RLF5 (sample of 10000) results



Figure 5.13: Planning RLF6 (sample of 10000) results

## Chapter 6

### Conclusion

Most previous Reinforcement Learning (RL) works solve the Traffic Signal Control problem using travel time as the objective. Our insight is to identify shortcomings of such approaches, namely the difficulty of deriving a reward mechanism and the limitations on assessment of cases. Then, we proposed a time-loss-based objective, presented in two forms: *percentage* time loss per driver and *cumulative* time loss per driver. We show the relationship between them and the travel time objective and that optimizing the time loss also optimizes the travel time. We also conducted experiments with a time loss and a time loss pressure, which are state/reward formulations that better relate to the proposed objective.

Our formulations perform better than the baselines we established, achieving up to 55% of improvement over static for cumulative time loss per driver and 31% of improvement for percentage time loss per driver. Our formulations also achieve up to 34% - 81 rounds – faster convergence and enhanced stability than the state-of-art RL formulation. These results point out that our time loss RL formulations are better correlated with the time loss objective.

We also experimented with different phase set configurations, showing that using conflicting phases and relying on right-of-way traffic rules to coordinate vehicles reduces the time loss up to 53% in the single intersection scenario. Additionally, concerning user-friendliness traffic light policies, we implement a waiting time restriction in all the experiments. We present the differences between the constrained and unconstrained policy and that the restriction avoids vehicles excessive waiting time.

Furthermore, while most RL TSC proposals tackle model-free RL, we wanted to assess how a model-based (also called Planning) approach would benefit the agent since Planning allows for a more diverse experience through interactions with the future. We conducted experiments using simulation instead of using a model because we wanted to avoid modeling approximation errors.

In our Planning experiments, the results suggest that the experience of the future interactions helps the agent converge considerably faster at first but becomes irrelevant, even detrimental, in the middle of the learning process. We also could not find any correlation between the number of learning samples generated by background planning and used by the policy update.

#### 6.1 Limitations

We followed the idea of using simulation to assess how Planning can benefit the learning process so that the results would not be limited by any model particular design. Although using a simulation for Planning removes concerns about learning a model and its error, it adds a time constraint since it typically takes more time to simulate than to use a model. This way, the extra time may limit the number of interactions the agent can make, also limiting the number of experiments. Also, we store experience indistinctly; the agent may start visiting irrelevant future states, consequently reducing the overall quality of the learning samples.

For the experiments, we used the default hyperparameters available in Frap's [68] GitHub repository. On one hand, we saved some time and avoided tampering with the results presented by the RL formulations, on the other, we do not know if the parameters would also be ideal for the Planning experiments. Since there are already so many parameters in the regular Frap, and the RL experiments generally take much time, we stuck with the default parameters, only tweaking the sample size and the max memory length (updating it to remain at 10x the sample size, as the default).

There are many ways to build state and reward formulations for RL TSC. We chose some works that presented interesting results while conducting other exploratory tests to understand the agent's behavior. Therefore, there is a vast amount of RL formulations to be considered and tested when proposing new solutions.

#### 6.2 Future Work

The most evident extension is to investigate other formulations of state and reward. We would also like to remove the 10s time slots for action and make it more continuous, incorporating only a notion of minimum and maximum time.

Another extension of this work can investigate superior planning algorithms, such as Monte-Carlo Tree Search, and how to value and manage the experience to provide more relevant learning samples to policy update.

For both Planning and non-Planning experiments, further testing is needed to assess more complex scenarios. In future work, we will consider more demand variations (*e.g.*, dynamic distribution of departing vehicles) and different network structures (*e.g.*, realworld scenarios provided by open-source initiatives such as Sumo [32], Flow [59, 60], or CityFlow [65]).

Additionally, we would like to examine how to accommodate complementary objectives at once, such as time savings + safety + comfort, for example, and how to model RL states and rewards for the joint objective context. Also, add more information such as weather conditions and security data to help the agent's decision-making process and how to collect and incorporate such diverse data into the simulation itself.

## References

- ABDOOS, M.; BAZZAN, A. L. Hierarchical traffic signal optimization using reinforcement learning and traffic prediction with long-short term memory. *Expert Systems* with Applications 171 (2021), 114580.
- [2] ABDOOS, M.; MOZAYANI, N.; BAZZAN, A. L. Traffic light control in non-stationary environments based on multi agent q-learning. In 2011 14th International IEEE conference on intelligent transportation systems (ITSC) (2011), IEEE, pp. 1580– 1585.
- [3] ABDOOS, M.; MOZAYANI, N.; BAZZAN, A. L. Hierarchical control of traffic signals using q-learning with tile coding. *Applied intelligence* 40, 2 (2014), 201–213.
- [4] ALPAYDIN, E. Introduction to machine learning. MIT press, 2020.
- [5] AREL, I.; LIU, C.; URBANIK, T.; KOHLS, A. G. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Sys*tems 4, 2 (2010), 128–135.
- [6] ARULKUMARAN, K.; DEISENROTH, M. P.; BRUNDAGE, M.; BHARATH, A. A. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine 34*, 6 (2017), 26–38.
- [7] ASLANI, M.; MESGARI, M. S.; WIERING, M. Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events. *Transportation Research Part C: Emerging Technologies 85* (2017), 732–752.
- [8] ASLANI, M.; SEIPEL, S.; MESGARI, M. S.; WIERING, M. Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran. Advanced Engineering Informatics 38 (2018), 639–655.
- [9] BAKKER, B.; WHITESON, S.; KESTER, L.; GROEN, F. C. Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information* Systems. Springer, 2010, pp. 475–510.
- [10] BALAJI, P.; GERMAN, X.; SRINIVASAN, D. Urban traffic signal control using reinforcement learning agents. *IET Intelligent Transport Systems* 4, 3 (2010), 177– 188.
- [11] BELLON, T. As global traffic jams mount, cities try new ways to ease congestion: study. https://www.reuters.com/article/us-autos-congestion/as-global-trafficjams-mount-cities-try-new-ways-to-ease-congestion-study-idUSKBN20W18E, 2020. Reuters. Accessed in 2021-03-15.

- [12] BLACKMAN, A. License plate-based driving restrictions programs: Where do they make sense? https://blogs.iadb.org/ciudades-sostenibles/en/license-plate-baseddriving-restrictions-programs-where-do-they-make-sense/, 2020. IADB. Accessed in 2021-03-15.
- [13] BRYS, T.; PHAM, T. T.; TAYLOR, M. E. Distributed learning and multi-objectivity in traffic light control. *Connection Science* 26, 1 (2014), 65–83.
- [14] CASAS, N. Deep deterministic policy gradient for urban traffic light control. arXiv preprint arXiv:1703.09035 (2017).
- [15] CHEN, C.; WEI, H.; XU, N.; ZHENG, G.; YANG, M.; XIONG, Y.; XU, K.; LI, Z. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 3414–3421.
- [16] CHIN, Y. K.; LEE, L. K.; BOLONG, N.; YANG, S. S.; TEO, K. T. K. Exploring q-learning optimization in traffic signal timing plan management. In 2011 third international conference on computational intelligence, communication systems and networks (2011), IEEE, pp. 269–274.
- [17] CHU, T.; WANG, J.; CODECA, L.; LI, Z. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems 21*, 3 (2019), 1086–1095.
- [18] EGEA, A. C.; HOWELL, S.; KNUTINS, M.; CONNAUGHTON, C. Assessment of reward functions for reinforcement learning traffic signal control under real-world limitations. In 2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC) (2020), IEEE, pp. 965–972.
- [19] EL-TANTAWY, S.; ABDULHAI, B. An agent-based learning towards decentralized and coordinated traffic signal control. In 13th International IEEE Conference on Intelligent Transportation Systems (2010), IEEE, pp. 665–670.
- [20] EL-TANTAWY, S.; ABDULHAI, B. Multi-agent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc). In 2012 15th International IEEE Conference on Intelligent Transportation Systems (2012), IEEE, pp. 319–326.
- [21] EL-TANTAWY, S.; ABDULHAI, B.; ABDELGAWAD, H. Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions* on Intelligent Transportation Systems 14, 3 (2013), 1140–1150.
- [22] GARBER, N. J.; HOEL, L. A. Traffic and highway engineering. Cengage Learning, 2014.
- [23] GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A.; BENGIO, Y. Deep learning, vol. 1. MIT press Cambridge, 2016.
- [24] HAYKIN, S. Neural networks: a comprehensive foundation. Prentice-Hall, Inc., 2007.

- [25] IŠA, J.; KOOIJ, J.; KOPPEJAN, R.; KUIJER, L. Reinforcement learning of traffic light controllers adapting to accidents. *Design and Organisation of Autonomous* Systems (2006), 1–14.
- [26] KHAMIS, M. A.; GOMAA, W. Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In 2012 11th international conference on machine learning and applications (2012), vol. 1, IEEE, pp. 586–591.
- [27] KHAMIS, M. A.; GOMAA, W.; EL-SHISHINY, H. Multi-objective traffic light control system based on bayesian probability interpretation. In 2012 15th International IEEE conference on intelligent transportation systems (2012), IEEE, pp. 995–1000.
- [28] KOONCE, P.; RODEGERDTS, L. Traffic signal timing manual. Tech. rep., United States. Federal Highway Administration, 2008.
- [29] KUYER, L.; WHITESON, S.; BAKKER, B.; VLASSIS, N. Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases* (2008), Springer, pp. 656–671.
- [30] LITTLE, J. D.; KELSON, M. D.; GARTNER, N. H. Maxband: a versatile program for setting signals on arteries and triangular networks. Tech. rep., MIT, 1981.
- [31] LIU, J.; ZHANG, H.; FU, Z.; WANG, Y. Learning scalable multi-agent coordination by spatial differentiation for traffic signal control. *Engineering Applications of Artificial Intelligence 100* (2021), 104165.
- [32] LOPEZ, P. A.; BEHRISCH, M.; BIEKER-WALZ, L.; ERDMANN, J.; FLÖTTERÖD, Y.-P.; HILBRICH, R.; LÜCKEN, L.; RUMMEL, J.; WAGNER, P.; WIESSNER, E. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference* on *Intelligent Transportation Systems* (2018), IEEE.
- [33] LOWRIE, P. Scats-a traffic responsive method of controlling urban traffic. roads and traffic authority, sydney. *New South Wales, Australia* (1990).
- [34] MANNION, P.; DUGGAN, J.; HOWLEY, E. An experimental review of reinforcement learning algorithms for adaptive traffic signal control. *Autonomic road transport* support systems (2016), 47–66.
- [35] MITCHELL, T. Machine Learning. New York: McGraw-Hill, Inc, 1997.
- [36] MNIH, V.; KAVUKCUOGLU, K.; SILVER, D.; RUSU, A. A.; VENESS, J.; BELLE-MARE, M. G.; GRAVES, A.; RIEDMILLER, M.; FIDJELAND, A. K.; OSTROVSKI, G., ET AL. Human-level control through deep reinforcement learning. *nature* 518, 7540 (2015), 529–533.
- [37] MOUSAVI, S. S.; SCHUKAT, M.; HOWLEY, E. Traffic light control using deep policygradient and value-function-based reinforcement learning. *IET Intelligent Transport* Systems 11, 7 (2017), 417–423.

- [38] NISHI, T.; OTAKI, K.; HAYAKAWA, K.; YOSHIMURA, T. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In 2018 21st International Conference on Intelligent Transportation Systems (ITSC) (2018), IEEE, pp. 877–883.
- [39] PHAM, T. T.; BRYS, T.; TAYLOR, M. E.; BRYS, T.; DRUGAN, M. M.; BOSMAN, P.; COCK, M.-D.; LAZAR, C.; DEMARCHI, L.; STEENHOFF, D., ET AL. Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents* workshop (at AAMAS-13) (2013), vol. 10, IEEE, pp. 1196–1201.
- [40] PRASHANTH, L.; BHATNAGAR, S. Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC) (2011), IEEE, pp. 1640– 1645.
- [41] REED, T. Global traffic scorecard. Tech. rep., INRIX, 2019.
- [42] RIZZO, S. G.; VANTINI, G.; CHAWLA, S. Time critic policy gradient methods for traffic signal control in complex and congested scenarios. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (2019), pp. 1654–1664.
- [43] ROESS, R. P.; PRASSAS, E. S.; MCSHANE, W. R. Traffic engineering. Pearson/Prentice Hall, 2004.
- [44] SALKHAM, A. A.; CAHILL, V. Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In 13th International IEEE Conference on Intelligent Transportation Systems (2010), IEEE, pp. 531–538.
- [45] SALKHAM, A. A.; CUNNINGHAM, R.; GARG, A.; CAHILL, V. A collaborative reinforcement learning approach to urban traffic control optimization. In 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology (2008), vol. 2, IEEE, pp. 560–566.
- [46] SCHRANK, D.; EISELE, B.; LOMAX, T. 2019 urban mobility report. Tech. rep., Texas A&M Transportation Institute, 2019.
- [47] STEINGROVER, M.; SCHOUTEN, R.; PEELEN, S.; NIJHUIS, E.; BAKKER, B., ET AL. Reinforcement learning of traffic light controllers adapting to traffic congestion. In *BNAIC* (2005), pp. 216–223.
- [48] SUTTON, R. S.; BARTO, A. G. Reinforcement learning: An introduction. MIT press, 2018.
- [49] VAN DER POL, E.; OLIEHOEK, F. A. Coordinated deep reinforcement learners for traffic light control. Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016) (2016).
- [50] VARAIYA, P. The max-pressure controller for arbitrary networks of signalized intersections. In Advances in Dynamic Network Modeling in Complex Transportation Systems. Springer, 2013, pp. 27–66.

- [51] WANG, Y.; XU, T.; NIU, X.; TAN, C.; CHEN, E.; XIONG, H. Stmarl: A spatiotemporal multi-agent reinforcement learning approach for cooperative traffic light control. *IEEE Transactions on Mobile Computing* (2020).
- [52] WEI, H.; CHEN, C.; ZHENG, G.; WU, K.; GAYAH, V.; XU, K.; LI, Z. 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 (2019), pp. 1290–1298.
- [53] WEI, H.; XU, N.; ZHANG, H.; ZHENG, G.; ZANG, X.; CHEN, C.; ZHANG, W.; ZHU, Y.; XU, K.; LI, Z. Colight: Learning network-level cooperation for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019), pp. 1913–1922.
- [54] WEI, H.; ZHENG, G.; GAYAH, V.; LI, Z. A survey on traffic signal control methods. arXiv preprint arXiv:1904.08117 (2019).
- [55] WEI, H.; ZHENG, G.; YAO, H.; LI, Z. Intellight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018), pp. 2496–2505.
- [56] WIERING, M.; VEENEN, J. V.; VREEKEN, J.; KOOPMAN, A. Intelligent traffic light control, 2004.
- [57] WIERING, M.; VREEKEN, J.; VAN VEENEN, J.; KOOPMAN, A. Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium*, 2004 (2004), IEEE, pp. 453–458.
- [58] WIERING, M. A. Multi-agent reinforcement learning for traffic light control. In Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000) (2000), pp. 1151–1158.
- [59] WU, C.; KREIDIEH, A.; PARVATE, K.; VINITSKY, E.; BAYEN, A. M. Flow: Architecture and benchmarking for reinforcement learning in traffic control. arXiv preprint arXiv:1710.05465 10 (2017).
- [60] WU, C.; PARVATE, K.; KHETERPAL, N.; DICKSTEIN, L.; MEHTA, A.; VINITSKY, E.; BAYEN, A. M. Framework for control and deep reinforcement learning in traffic. In 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC) (2017), IEEE, pp. 1–8.
- [61] XIE, D.; WANG, Z.; CHEN, C.; DONG, D. Iedqn: Information exchange dqn with a centralized coordinator for traffic signal control. In 2020 International Joint Conference on Neural Networks (IJCNN) (2020), IEEE, pp. 1–8.
- [62] XIONG, Y.; ZHENG, G.; XU, K.; LI, Z. Learning traffic signal control from demonstrations. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019), pp. 2289–2292.
- [63] XU, L.-H.; XIA, X.-H.; LUO, Q. The study of reinforcement learning for traffic self-adaptive control under multiagent markov game environment. *Mathematical Problems in Engineering 2013* (2013).

- [64] ZANG, X.; YAO, H.; ZHENG, G.; XU, N.; XU, K.; LI, Z. Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence* (2020), vol. 34, pp. 1153–1160.
- [65] ZHANG, H.; FENG, S.; LIU, C.; DING, Y.; ZHU, Y.; ZHOU, Z.; ZHANG, W.; YU, Y.; JIN, H.; LI, Z. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The World Wide Web Conference* (2019), pp. 3620–3624.
- [66] ZHANG, H.; LIU, C.; ZHANG, W.; ZHENG, G.; YU, Y. Generalight: Improving environment generalization of traffic signal control via meta reinforcement learning. In Proceedings of the 29th ACM International Conference on Information & Knowledge Management (2020), pp. 1783–1792.
- [67] ZHANG, Z.; YANG, J.; ZHA, H. Integrating independent and centralized multiagent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651* (2019).
- [68] ZHENG, G.; XIONG, Y.; ZANG, X.; FENG, J.; WEI, H.; ZHANG, H.; LI, Y.; XU, K.; LI, Z. Learning phase competition for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management (2019), pp. 1963–1972.
- [69] ZHENG, G.; ZANG, X.; XU, N.; WEI, H.; YU, Z.; GAYAH, V.; XU, K.; LI, Z. Diagnosing reinforcement learning for traffic signal control. arXiv preprint arXiv:1905.04716 (2019).

# APPENDIX A – List of Parameters and Hyperparameters

In this appendix, we list the parameters and hyperparameters used by the environment (Section A.1) and the Frap and Frap+Planning agents (Section A.2).

#### A.1 Environment

- The yellow time is set to three seconds, matching the same conditions encountered on the baseline.
- No teleportation. Vehicles in Sumo teleport to the next lane after some time idling, overriding the signal control. Teleportation is completely undesired since traffic behavior shapes the reward. Setting time-to-teleport to -1 (minus one) deactivates it;
- Collision stop time. Vehicles stop for a total of 10 (ten) seconds when they collide;
- Collision detection boundary. Setting collision.minigap-factor to 0 (zero) makes collisions physical only (overlapping of vehicles);
- Collision consequence. Setting to warn the teleportations due to collision are also disabled;
- Collision check junctions. Setting to true to check for collisions in the intersection (always deactivated on unregulated junctions);
- Ignore junction blocker. It sets the time to wait for junctions blocker overtaking. As already mentioned, custom deadlock resolution is applied.

#### A.2 Agent

Parameter	Default Value
LEARNING_RATE	0.001
LR_DECAY	0.98
MIN_LR	0.001
SAMPLE_SIZE	1000
BATCH_SIZE	20
EPOCHS	100
UPDATE_Q_BAR_FREQ	5
UPDATE_Q_BAR_EVERY_C_ROUND	False
GAMMA	0.8
MAX_MEMORY_LEN	10000
PATIENCE	10
D_DENSE	20
EPSILON	0.8
EPSILON_DECAY	0.95
MIN_EPSILON	0.2
LOSS_FUNCTION	mean_squared_error
NORMAL_FACTOR	20
EARLY_STOP_LOSS	val_loss
DROPOUT_RATE	0
MERGE	multiply

Table A.1: Frap's default hyperparameters

Table A.2: Planning exclusive parameters

Parameter	Default Value
TIEBREAK_POLICY	random
PLANNING_ITERATIONS	2
ACTION_SAMPLING_SIZE	2
ACTION_SAMPLING_POLICY	random
BACKGROUND_PLANNING	True
DECISION_TIME_PLANNING	True

# APPENDIX B – RL Formulation Results: Rolling Average Window's Best Round

In this appendix, we report the best round results because, at the end of the day, only one model is chosen to be implemented. Tables B.1 and B.2 present the results for the time loss-based objectives. The major difference between these results and the ones presented in Chapter 5 is that for less stable RL formulation's learning processes, such as Average Queue Length (**RLF2**) and Pressure MPLight (**RLF4**), all reported values are closer to the best method result, although the difference in improvement is still noticeable.

Considering the best round, the Time Loss (**RLF5**) and Time Loss Pressure (**RLF6**) formulations we propose improved by, in the single-intersection case, up to 49.58% (6.35% improvement over the average results) and 34.16% (9.31% improvement) over the static baseline, while in the multi-intersection case, those numbers go up to 58.80% (6.32%) and 34.66% (11.27%) improvement, for cumulative time loss per driver and percentage time loss per driver, respectively.

We also present the results for average travel time, throughput, and average time loss objectives (Tables B.3 and B.4) as well as the impact of pending vehicles on the metrics (Tables B.5 and B.6). The same analysis from Section 5.3.3.1 also applies for the best round results.

Model	Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement		
Static	40.03	-	0.48	-		
Unregulated	4.23	-	0.09	-		
8 phases						
(RLF1)	43.93		0.50			
(RLF2)	53.59		0.55			
(RLF3)	46.16	\	0.51			
(RLF4)	58.62		0.57			
$(\mathrm{RLF5})$	36.93	7.75%	0.46	4.39%		
$(\mathrm{RLF6})$	37.67	5.91%	0.46	3.37%		
		14 phases				
(RLF1)	21.06	47.39%	0.33	32.03%		
(RLF2)	61.21		0.55			
(RLF3)	20.90	47.80%	0.32	32.59%		
(RLF4)	25.84	35.47%	0.37	22.81%		
(RLF5)	20.28	49.34%	0.32	34.01%		
(RLF6)	20.19	49.58%	0.32	34.16%		

Table B.1: Time loss-based objectives results (single-intersection scenario - best round)

Table B.2: Time loss-based objectives results (multi-intersection scenario - bbest round)

Model	Cumulative Time Loss per Driver	Improvement	Percentage Time Loss per Driver	Improvement			
Static	105.77	-	0.61	-			
Unregulated	7.22	-	0.10	-			
	8 phases						
(RLF1)	92.73	12.33%	0.58	4.82%			
(RLF2)	118.96		0.63				
(RLF3)	122.13		0.64				
(RLF4)	124.39	\	0.64				
(RLF5)	81.63	22.83%	0.55	9.72%			
(RLF6)	101.16	4.36%	0.60	1.61%			
		14 phases					
(RLF1)	47.98	54.64%	0.42	31.00%			
(RLF2)	71.57	32.34%	0.51	16.95%			
(RLF3)	55.19	47.82%	0.46	25.22%			
(RLF4)	80.26	24.12%	0.52	15.05%			
(RLF5)	43.58	58.80%	0.40	34.66%			
(RLF6)	46.45	56.08%	0.41	32.12%			

Model	Average Travel Time	Improv.	Throughput	Improv.	Average Time Loss	Improv.
Static	83.96	-	4181	-	40.45	-
Unregulated	47.80	-	4218	-	4.26	-
			8 phases			
(RLF1)	87.59		4167		44.08	
(RLF2)	97.00		4160		53.49	
(RLF3)	89.61	\	4162	\	46.11	
(RLF4)	102.04	\	4156	\	58.56	
(RLF5)	80.64	3.95%	4185	0.10%	37.12	8.23%
(RLF6)	81.33	3.13%	4179		37.81	6.52%
			14 phases			
(RLF1)	64.80	22.82%	4204	0.55%	21.28	47.39%
(RLF2)	95.13	\	4084	\	51.73	
(RLF3)	64.66	22.98%	4205	0.57%	21.14	47.75%
(RLF4)	69.68	17.01%	4208	0.65%	26.15	35.35%
(RLF5)	64.00	23.77%	4205	0.57%	20.48	49.38%
(RLF6)	63.89	23.90%	4206	0.60%	20.37	49.64%

Table B.3: Other objectives' results for the single-intersection scenario (best round)

Table B.4: Other objectives' results for the multi-intersection scenario (best round)

Model	Average Travel Time	Improv.	Throughput	Improv.	Average Time Loss	Improv.			
Static	161.01	-	8128	-	96.25	-			
Unregulated	72.41	-	8466	-	7.28	-			
8 phases									
(RLF1)	157.17	2.38%	8227	1.22%	92.22	4.19%			
(RLF2)	174.44		8018		109.82				
(RLF3)	183.16	\	8077	\	118.42	\			
(RLF4)	184.08		8050	\	119.38				
(RLF5)	146.18	9.21%	8234	1.30%	81.23	15.60%			
(RLF6)	164.56	\	8156	0.34%	99.72				
14 phases									
(RLF1)	113.28	29.65%	8351	2.74%	48.21	49.91%			
(RLF2)	134.81	16.27%	8192	0.79%	69.89	27.39%			
(RLF3)	120.58	25.11%	8331	2.50%	55.55	42.28%			
(RLF4)	138.35	14.07%	8168	0.49%	73.48	23.66%			
(RLF5)	109.01	32.30%	8369	2.97%	43.96	54.33%			
(RLF6)	111.78	30.57%	8351	2.74%	46.72	51.46%			

Model	Average Travel Time	+ pending vehicles	Average Time Loss	+ pending vehicles	Cumulative Time Loss per Driver				
Static	83.96	+ 0.01	40.45	+ 0.01	40.03				
Unregulated	47.80	+ 0.01	4.26	+ 0.01	4.23				
8 phases									
(RLF1)	87.59	+ 0.01	44.08	+ 0.01	43.93				
(RLF2)	97.00	+ 0.01	53.49	+ 0.01	53.59				
(RLF3)	89.61	+ 0.01	46.11	+ 0.01	46.16				
(RLF4)	102.04	+ 0.32	58.56	+ 0.32	58.62				
(RLF5)	80.64	+ 0.01	37.12	+ 0.01	36.93				
(RLF6)	81.33	+ 0.01	37.81	+ 0.01	37.67				
14 phases									
(RLF1)	64.80	+ 0.01	21.28	+ 0.01	21.06				
(RLF2)	95.13	+ 6.53	51.73	+ 6.53	61.22				
(RLF3)	64.66	+ 0.01	21.14	+ 0.01	20.90				
(RLF4)	69.68	+ 0.01	26.15	+ 0.01	25.84				
(RLF5)	64.00	+ 0.01	20.48	+ 0.01	20.28				
(RLF6)	63.89	+ 0.01	20.37	+ 0.01	20.19				

Table B.5: Pending vehicles impact on metrics for the single-intersection scenario (best round)

Table B.6	: Pending	vehicles	impact	on	metrics	for	the	${\it multi-intersection}$	$\operatorname{scenario}$	(best
round)										

Model	Average Travel Time	$+ {\ { m pending} \over { m vehicles}}$	$\begin{array}{c c} \mathbf{Average} \\ \mathbf{Time} \\ \mathbf{Loss} \end{array} + \mathbf{pending} \\ \mathbf{vehicles} \end{array}$		Cumulative Time Loss per Driver				
Static	161.01	+ 8.06	96.25	+ 8.06	105.77				
Unregulated	72.41	+ 0.01	7.28	+ 0.01	7.22				
8 phases									
(RLF1)	157.17	+ 0.05	92.22	+ 0.05	92.73				
(RLF2)	174.44	+ 3.62	109.82	+ 3.62	118.96				
(RLF3)	183.16	+ 1.68	118.42	+ 1.68	122.13				
(RLF4)	184.08	+ 0.71	119.38	+ 0.71	124.39				
(RLF5)	146.18	+ 0.07	81.23	+ 0.07	81.63				
(RLF6)	164.56	+ 0.53	99.72	+ 0.53	101.16				
14 phases									
(RLF1)	113.28	+ 0.01	48.21	+ 0.01	47.98				
(RLF2)	134.81	+ 0.68	69.89	+ 0.68	71.57				
(RLF3)	120.58	+ 0.04	55.55	+ 0.04	55.19				
(RLF4)	138.35	+ 4.45	73.48	+ 4.45	80.26				
(RLF5)	109.01	+ 0.01	43.96	+ 0.01	43.58				
(RLF6)	111.78	+ 0.01	46.72	+ 0.01	46.45				