HEC MONTRÉAL
École affiliée à l'Université de Montréal

Three Essays on Inductive Graph Reinforcement Learning for Scalable Traffic Signal Control

par
François-Xavier Devailly

Thèse présentée en vue de l'obtention du grade de Ph. D. en administration
(spécialisation Ph.D. en Science des données)

Décembre 2022

HEC MONTRÉAL

École affiliée à l'Université de Montréal

Cette thèse intitulée :

Three Essays on Inductive Graph Reinforcement Learning for Scalable Traffic Signal Control

Présentée par :

François-Xavier Devailly

a été évaluée par un jury composé des personnes suivantes :

Michel Denault
HEC Montréal
Président-rapporteur

Denis Larocque
HEC Montréal
Directeur de recherche

Laurent Charlin
HEC Montréal
Codirecteur de recherche

Marilène Cherkesly
ESG UQAM
Membre du jury

Vikash V. Gayah
The Pennsylvania State University
Examinateur externe

Jean-François Cordeau
HEC Montréal
Représentant du directeur de HEC Montréal

Résumé

L'optimisation de l'utilisation des infrastructures routières existantes par le biais d'un contrôle adaptatif des feux de circulation (ATSC) basé sur l'état en temps réel du réseau routier a un rôle crucial à jouer dans la réduction de la congestion et de ses conséquences sociales et environnementales. L'écart de sophistication entre le contrôle adaptatif des feux de signalisation dans la littérature académique et le contrôle à temps fixe utilisé dans la pratique (la proportion d'intersections qui ne sont pas sous contrôle à temps fixe reste anecdotique) ne cesse de se creuser. L'ATSC nécessite actuellement à la fois 1) l'installation de capteurs coûteux sur les réseaux routiers (ce qui implique parfois des travaux de voirie) et 2) une expertise humaine (généralement des ingénieurs des transports) pour régler avec précision les systèmes correspondants dans le contexte dans lequel ils doivent être appliqués. Étant donné que la plupart des localités n'ont ni l'un ni l'autre, il semble évident que toute tentative de lutte contre la congestion à grande échelle devra se concentrer sur l'atténuation de ces obstacles. Si l'apprentissage automatique, et l'apprentissage par renforcement (RL) en particulier, offre un moyen séduisant d'automatiser le réglage par essais-erreurs dans l'environnement, il remplace ce coût par l'introduction de risques de performance et de sécurité durant l'entrainement, ce qui rend son acceptabilité sociale questionnable. De plus, le RL pour l'ATSC repose toujours sur des capteurs au niveau du réseau routier pour recueillir des données. Dans ce travail, nous tirons parti de la flexibilité des réseaux à convolution sur des graphes et du "data efficiency" de l'apprentissage par renforcement basé sur des modèles (ou "model-based RL") pour rendre l'ATSC aussi prêt à l'emploi que possible. Plus précisément, nous fournissons de nouvelles approches pour 1) réduire (et même contourner complètement) le besoin d'exploration de l'environnement, 2) fournir de la flexibilité en assurant la robustesse (généralisation) des politiques aux changements de l'environnement (par exemple, la structure du réseau routier, la distribution du trafic et les contraintes des contrôleurs), 3) permettre l'extensibilité totale à d'immenses réseaux routiers, et 4) offrir des moyens alternatifs de représenter le trafic en utilisant des sources de données plus granulaires au niveau du véhicule.

Mots-clés

Apprentissage par renforcement profond, Apprentissage par transfert, Contrôle adaptatif des feux de signalisations, Réseaux à convolutions sur des graphes, Transfert zero-coup, Apprentissage de fonction Q indépendant, Apprentissage par renforcement multi-agent, Modélisation conjointe des actions, Apprentissage par renforcement basé sur un modèle, Apprentissage par renforcement hors ligne.

Méthodes de recherche

Forage de données, Analyse multivariée

Abstract

Optimizing the use of existing road infrastructures via adaptive traffic signal control (ATSC) based on the real-time state of the road network has a crucial role to play in mitigating congestion and its social and environmental tolls. The gap of sophistication between ATSC in the academic literature and Fixed-Time control used in practice keeps widening. While ATSC does exist in the real world, the vast majority of intersections remains under nonadaptive control as ATSC currently requires both 1) the installation of costly sensors on road networks (which sometimes involves roadwork) and 2) human expertise (typically transportation engineers) to design and fine-tune the corresponding systems in the specific context in which they are to be applied. As most localities have none, it appears clear that any attempt to tackle congestion at scale will have to focus on alleviating these barriers. While machine learning, and reinforcement learning (RL) in particular, offers a seducing way to automate design and tuning by trial-and-error in the environment, it introduces performance and safety hazards during training which hinders its social acceptability. Furthermore, RL for ATSC still relies on road-network level sensors to gather data. In this thesis, we leverage the flexibility of graph convolutional networks and the data efficiency of model-based reinforcement learning to make ATSC as plug-and-play as possible. Specifically, we provide new approaches to 1) reduce (and even fully bypass) the need for exploration in the environment, 2) provide flexibility by ensuring robustness (generalization) of policies to changes in the environment (e.g. new road network structures, traffic distributions and constraints of controllers), 3) enable full scalability to immense road networks, and 4) offer alternative ways to represent traffic by making use of more granular sources of data at the vehicle level.

Keywords

Deep reinforcement learning, Transfer learning, Adaptive traffic signal control, Graph neural networks, Zero-Shot Transfer, Independent Q-Learning, Multi-Agent Reinforcement Learning, Joint Action Modeling, Model-Based Reinforcement Learning, Offline Reinforcement Learning,

Research methods

Multivariate analysis, Data Mining

Contents

Résumé iii

Abstract v

List of Tables xi

List of Figures xiii

List of acronyms xv

Acknowledgements xix

General Introduction 1

0.1 Key concepts 4

1 IG-RL: Inductive Graph Reinforcement Learning for Massive-Scale Traffic Signal Control 5

Abstract 6

1.1 Introduction 6

1.1.1 Contribution 6

1.2 Related Work 8

1.2.1 Decentralization 8

1.2.2 Graph-Based Representation 8

1.2.3 Parameter sharing 9

1.3 Background 9

1.3.1 Reinforcement Learning (RL) 9

1.3.2 Graph-Convolutional Networks 11

1.4 Inductive Graph Reinforcement Learning (IG-RL) 12

1.4.1 Decentralized ATSC Decision Processes . . . 13
1.4.2 Model . . . 14
1.5 Experiments . . . 18
1.5.1 General Setup . . . 19
1.5.2 IG-RL models . . . 22
1.5.3 Baselines . . . 23
1.5.4 Experiment 1: Evaluating Generalization using Synthetic Road Networks . . . 24
1.5.5 Experiment 2: Transfer and Scaling to Manhattan . . . 30
1.6 Complexity . . . 32
1.6.1 Building the computational graph . . . 32
1.6.2 Training & Evaluation . . . 33
1.6.3 Scaling & Transfer . . . 34
1.7 Algorithm (pseudocode) . . . 34
1.7.1 Interaction with SUMO . . . 34
1.8 Conclusion . . . 37
1.9 Acknowledgements . . . 37
References . . . 38

2 Model-Based Graph Reinforcement Learning for Inductive Traffic Signal Control 43

Abstract . . . 43
2.1 Introduction . . . 44
2.1.1 Contribution . . . 44
2.2 Related Work . . . 46
2.2.1 Scalability . . . 46
2.2.2 Coordination . . . 46
2.2.3 Data Efficiency & Model-Based RL . . . 47
2.3 Background . . . 47
2.3.1 Model-Based Reinforcement Learning . . . 47
2.3.2 Heterogeneous Graph Convolutional Networks (GCNs) . . . 48
2.3.3 ATSC Decision Process . . . 48
2.4 MuJAM . . . 50
2.4.1 Dynamics Model . . . 50

2.4.2 Coordinated priors for parsimonious planning . . . 51
2.4.3 Architecture . . . 52
2.4.4 Planning with a tree search . . . 54
2.4.5 Training . . . 56
2.4.6 Algorithms (pseudocodes) . . . 56
2.5 Experiments . . . 62
2.5.1 General Setup . . . 62
2.5.2 Baselines . . . 63
2.5.3 Experiment 1: Inductive Learning & Zero-shot transfer . . . 64
2.5.4 Experiment 2: Scaling to Manhattan . . . 68
2.6 Conclusion . . . 70
2.6.1 Future work . . . 70
References . . . 71

3 IORL : Inductive-Offline-Reinforcement-Learning for Traffic Signal Control Warmstarting 77
Abstract . . . 78
3.1 Introduction . . . 78
3.1.1 Contribution . . . 79
3.2 Background . . . 80
3.2.1 Graph Convolutional Networks . . . 80
3.2.2 Markov Decision Process . . . 81
3.2.3 Reinforcement Learning . . . 81
3.2.4 Model-Based Reinforcement Learning . . . 81
3.2.5 Offline Reinforcement Learning . . . 82
3.3 Related Work . . . 83
3.3.1 Multi-Agent RL-ATSC . . . 83
3.3.2 Parameter Sharing & Transfer Learning . . . 84
3.3.3 Model-Based . . . 84
3.3.4 Offline RL-ATSC . . . 85
3.4 IORL: Inductive-Offline-Reinforcement-Learning . . . 85
3.4.1 ATSC MDP . . . 85
3.4.2 Simulator and model-based planning . . . 86

3.4.3 Offline Learning Formulation . . . 89
3.4.4 Algorithms (pseudocodes) . . . 90
3.5 Experiments . . . 93
3.5.1 General Setup . . . 93
3.5.2 Experiment 1: Inductive Learning & Zero-Shot Transfer . . . 95
3.5.3 Experiment 2: Online fine-tuning from offline initialization . . . 97
3.5.4 Experiment 3: Ablation study . . . 98
3.5.5 Experiment 4: Real road networks . . . 101
3.6 Conclusion . . . 102
3.6.1 Future Work . . . 103
References . . . 104

General Conclusion **109**

Bibliography **111**

List of Tables

1.1 Initial state variables for every type of node. 18
1.2 Durations on synthetic road networks 33
1.3 Training (5 runs) 33
1.4 Evaluation (5 runs) 33

2.1 Node features 53
2.2 Edge features 53
2.3 Hyperparameters 63

3.1 Description of the datasets used for offline learning 98

List of Figures

1.1 Local computational graph. 16
1.2 Four randomly generated road networks. 20
1.3 Distributions of trips durations: **Default** Traffic Regime | Synthetic Road Networks. 25
1.4 Distributions of trips duration: **Heavy** Traffic Regime | Synthetic Road Networks. 25
1.5 Total Delay Evolution: **Default** Traffic Regime | Synthetic Road Networks 26
1.6 Total Delay Evolution: **Heavy** Traffic Regime | Synthetic Road Networks. 27
1.7 Differences of Paired Trips Durations (compared to G-IG-RL-V): **Default** Traffic Regime | Synthetic Road Networks 27
1.8 Differences of Paired Trips Durations (compared to G-IG-RL-V): **Heavy** Traffic Regime | Synthetic Road Networks 28
1.9 Manhattan road network. 30
1.10 Total Delay Evolution: **Light** Traffic Regime | Manhattan Island 32

2.1 Illustration of MuJAM's model-based computational graph 55
2.2 Distributions of trips delays 66
2.3 Evolution of average total delay during training 67
2.4 Distributions of delays' differences 68
2.5 Steps delays in Manhattan (compared to Fixed Time policy) 69

3.1 Online Vs. Offline learning paradigms 83
3.2 Evolution of generalization 88
3.3 Offline scaling 89
3.4 Distributions of total delay for the main baselines and proposed methods (OR and IORL) 95
3.5 Distributions of paired (per trip) differences in delays between any given method and IORL (30K). 96
3.6 Average total delay (cumulative delay for all trips and steps in a given episode) 98

3.7 Distributions of total delay for IORL using different offline datasets. T-tests are paired (per trip). . 99
3.8 Distributions of paired (per trip) differences in delays between any given method and IORL(30K). 100
3.9 Central Manhattan road network. 102
3.10 Manhattan - Cumulative difference in total delay (compared to the fixed-time policy) 103

List of acronyms

ATSC Adaptive traffic signal control

DEC-RL Decentralized reinforcement learning

DRL Deep reinforcement learning

FRQNT Fonds de Recherche du Quebec: Nature et Technologies

GCN Graph convolutional network

G-IGRL Generalist inductive graph reinforcement learning

HEC Hautes études commerciales

HGCN Heterogeneous graph convolutional network

IGRL Inductive graph reinforcement learning

IGRL-V Inducrive graph reinforcement learning - Vechicles

IGRL-L Inductive graph reinforcement learning - lanes

IORL Inductive offline reinforcement learning

IQL Independent Q-learning

MARL Multi-agent reinforcement learning

MBRL Model-based reinforcement learning

MCTS Monte carlo tree search

MDP Markov decision process

MFRL Model-free reinforcement learning

MLP Multilayer perceptron

MuJAM Joint action modeling with MuZero

MuJAM-C joint action modeling with MuZero - cyclic

MuJAM-A joint action modeling with MuZero - acyclic

MuIM independent action modeling with MuZero

MuJAM-NNL joint action modeling with MuZero - no noisy layers

MuJAM-NR joint action modeling with MuZero - no reanalyze

NSERC Natural Sciences and Engineering Research Council

OOD Out of distribution

ORL Offline reinforcement learning

OW Offline warmstart

PhD Doctorat

PUCT Probabilistic upper confidence tree

RL Reinforcement learning

S-IGRL Specialist inductive graph reinforcement learning

SUMO Simulation of urban mobility

TD Temporal difference

TS Tree search

TSC Traffic signal control(lers)

Acknowledgements

First and foremost, I would like to thank my advisors, Denis Larocque and Laurent Charlin, for their unwavering support and kindness, as well as their time. I could hardly have hoped for a better complementarity of expertise and experience for the supervision of this research, or for better paragons from which to draw inspiration on scientific, pedagogical, and human dimensions.

I would like to thank the reviewers of the *Inductive Graph Reinforcement Learning for Massive-Scale Traffic Signal Control* article (corresponding to the 1st chapter of this thesis) for their interesting and constructive comments that lead to improved versions of this work and were also taken into account in the 2 subsequent articles which build upon the first.

I would also like to thank Maxime Gasse for helpful discussions of deep learning on graphs which helped formulate GCN architectures included in this work,

I also acknowledge that this work was partially funded by the Natural Sciences and Engineering Research Council (NSERC), Fonds de Recherche du Quebec: Nature et Technologies (FRQNT), Samsung, and Fondation HEC Montreal.

General Introduction

The social, economical, and environmental tolls of congestion are well documented (Schrank et al., 2012; Barth and Boriboonsomsin, 2008; Schrank et al., 2015, 2019) [1]. The cost of hardware (sensors) as well as the time and expertise required to manually design and tune efficient adaptive-heuristic-based policies remain significant. Furthermore, as budgets of localities are limited and countries with the highest public spending per capita also tend to have smaller populations (OECD, 2021), tackling congestion globally will require cost-efficiency. Machine learning offers an appealing self-optimizing alternative to move beyond such approaches (Koseki et al., 2022). Recent developments and successes of deep reinforcement learning (deep RL or DRL), in particular, enable learning optimal policies from high dimensional observations of trajectories, gathered via interaction with the environment (i.e. actual control of TSCs) in a trial-and-error spirit. However, one of the curses of RL is precisely this continuous need for interaction (i.e. experimentation and data collection) with the environment. Combined with notorious data inefficiency, this broadly curbs real-world applicability and social acceptability of RL as 1) data collection can be expensive and time-consuming, and 2) random exploration from untrained policies might cause both safety hazards and catastrophic performance. For ATSC in particular, the joint action space corresponding to a given road network grows exponentially with the number of intersections, and while decentralization in the form of multi-agent RL can make it slightly more scalable, without parameter sharing, computational resources and time required to train MARL approaches grow with every additional signalized intersection as it requires the training of additional agents. A key obstacle to data efficiency with current RL-ATSC approaches is the fact that the specialization of any agent on the particular environment experienced by its jurisdiction during training hinders generalization to changes in the environment and transferability to new environments. Therefore, applying RL on any new road network or after any modification to a road network already under RL control typically requires training from scratch.

[1] According to TomTom Traffic Index, for instance, rush hour congestion in London is associated to 1) massive delays (42.5 minutes for a 10km drive with 43% of this duration being due to congestion), 2) increased CO2 emissions (282kg of additional CO2 released yearly by a petrol vehicle for a one-way commute of 10km), and 3) increased fuel price (25% of the cost, 0.5 USD, being due to congestion)

Throughout this thesis, we aim to equip RL-ATSC with the following capabilities to specifically address the data inefficiency of RL-ATSC: 1) Inductive learning [2], 2) Coherent exploration [3], 3) Sample efficiency [4], 4) Offline learning [5].

The first chapter introduces IGRL[6], an architecture that exploits the inductive properties of GCNs to allow the learning of policies that generalize to new road network architectures and traffic distributions with no additional training. In terms of applicability, this means that after the model is trained *once*, on a set of example scenarios, it can be applied on new settings (i.e. new intersection(s), road network(s), and traffic distributions) immediately, without the need for data collection, exploratory behavior, or training on these new settings. IGRL also addresses the exploration side of data inefficiency by leveraging noisy parameters to promote coherence in exploratory behavior. This improved exploration lowers, for the original training, the required amounts of hazardous interactions with the environment and data to be collected.

The second chapter introduces another model, MuJAM[7], which builds upon the advantages of IGRL and introduces planning based on the use of a simulator (trained automatically using available data). This approach, beyond improving performance, allows better sample efficiency (i.e. reaching a higher level of performance for a given amount of collected data), and a new form of generalization to controller constraints (e.g. between cyclic or acyclic constraints). Concretely, instead of having to adapt, train, evaluate and deploy a new version of the model for the new set of constraints a particular intersection or road network might require for safety and efficiency concerns, this last part enables training a unique policy which can be used *as is* with different constraints.

Finally, the third paper introduces a method, IORL[8], which completely bypasses interaction with the environment. This method thus allows the learning of inductive policies without ever introducing the threat of catastrophic performance or safety hazards. In other words, it only requires collecting observations under the current (historical) policy (adaptive or not) to train simulators and improved policies. In practice, this approach, which can still be directly applied to new settings after learning from a set of example scenarios, does not need to disrupt these example scenarios in any way.

[2]Learning generalizable patterns which can transfer efficiently at application time to new instances (i.e. variations) of the underlying task(s).

[3]Short-term correlation in exploratory behaviour (successive transitions in the environment) which can lead to the collection of more informative samples (i.e. sequences of transitions in the environment).

[4]Refers to the relative ability (measured as reachable performance wrt. a given task) to exploit a given amount of signal.

[5]Learning using strictly historical data (i.e. a dataset of transitions), without ever interacting with the real environment during training.

[6]Published in IEEE Transactions on Intelligent Transportation Systems (Volume: 23, Issue: 7, July 2022)

[7]Submitted (August 2022)

[8]Submitted (March 2023)

For all approaches, the ability to decouple training scenarios and application scenarios greatly limit computational requirements. The set of parameters is small and ubiquitous (applied everywhere), and it can be trained on sets of small road networks using only consumer-grade processing units. For application (i.e. at inference time) that is much less demanding, a single consumer-grade GPU can be used to jointly control thousands of traffic signal controllers. All approaches can be used in a fully decentralized fashion if required (at the potential cost of coordination ability), enabling the use of local processing units to avoid the risk of a single point of failure (SPOF).

Most importantly, the combination of the automated training of a simulator and the ability to learn fully offline (without interaction) detaches training and experimentation from many real-world constraints. Training of an RL-ATSC policy for the control of any new real-world intersection would typically require 1) extensive discussions and agreements with a city to (even temporarily) obtain some level of control of the corresponding intersection, and 2) strictly real-time data collection corresponding to every exploratory action performed under training. This drastically limits experimentation ability and the exploitability of such data by third-party contributors. By comparison, with just a regular dataset of past observations, contributors could leverage IORL to train a simulator and a policy that can both not only be used on the observed scenarios but also directly transferred to new settings (thanks to its inductive capabilities).

In addition to the focus on data efficiency, the use of GCNs offers the possibility to exploit demand information at its finest level of granularity. All proposed methods represent each vehicle with its features and do not have to resort to arbitrary lane-level aggregations. This representation is shown to result in improved performance. All methods can still use lane-level information (e.g., if this is the only level at which information is captured), providing flexibility in this regard. Moreover, MuJAM and IORL rely on joint modeling of actions (joint simulation) and allow explicit coordination of agents at scale. To test our claims we introduce a new inductive evaluation setting for RL-ATSC which involves evaluation of the learned policies on a set of synthetic and large real road networks never seen in training, and we compare the performance of various instantiations of our methods and domain-specific baselines compatible with the inductive setting. On top of aggregated performance, we study how the delay is distributed among trips under different policies to ensure fairness (i.e. that an increase in performance does not translate into an unfair treatment of some trips).

0.1 Key concepts

The following section aims to define some key concepts used in this thesis.

- *Graph convolutional network (GCN)*: Flexible deep learning architecture (model) designed for graph-structured data that relies on local (neighbor-to-neighbor nodes) parameterized message-passing. These local aggregation functions (GCN layer) are trained end-to-end via gradient descent and applied in parallel for every node in the graph. The stacking of multiple GCN layers applied sequentially can enable the learning of aggregation functions representing an arbitrarily large neighborhood for all nodes.

- *Decentralized RL*: Decomposition (e.g. spatial decomposition) of a large action space into multiple local action spaces. In its most typical form, each action space is controlled by a different agent which observes only a (relevant) portion of the state space. The objective (e.g. reward function) can also be decomposed (individualized) per agent.

- *Embedding*: An embedding is a (typically) vectorized and lower dimensional space (or representation) translation of a large and/or complex-structure-space.

- *Latent space*: Space consisting of (random) variables that cannot be observed directly. Embedding space and latent space are typically used interchangeably in the context of deep learning.

- *Parameter sharing*: In deep learning computational graphs (the sequence of operations that defines the particular architecture of a neural network), a given parameter can often be used on different original (i.e. covariate) or intermediary (i.e. a given neuron) features (e.g. a convolutional filter used in different parts of a network/image/sentence).

- *Value-equivalent decision process (& MuZero)*: The key concept behind MuZero (latent-based planning used in chapters 2 & 3) is the creation of a general Markov decision process (MDP) model that mirrors the actual environment so that planning in the abstract MDP is tantamount to planning in the real-world setting. This correspondence is established by guaranteeing value equivalence, which means that the cumulative reward obtained by starting from the same actual state must be the same as the cumulative reward gained by starting from the equivalent abstract state.

Chapter 1

IG-RL: Inductive Graph Reinforcement Learning for Massive-Scale Traffic Signal Control

François-Xavier Devailly, Denis Larocque, Laurent Charlin

Abstract

Scaling adaptive traffic signal control involves dealing with combinatorial state and action spaces. Multi-agent reinforcement learning attempts to address this challenge by distributing control to specialized agents. However, specialization hinders generalization and transferability, and the computational graphs underlying neural-network architectures—dominating in the multi-agent setting—do not offer the flexibility to handle an arbitrary number of entities which changes both between road networks, and over time as vehicles traverse the network. We introduce Inductive Graph Reinforcement Learning (IG-RL) based on graph-convolutional networks which adapts to the structure of any road network, to learn detailed representations of traffic signal controllers and their surroundings. Our decentralized approach enables learning of a transferable-adaptive-traffic-signal-control policy. After being trained on an arbitrary set of road networks, our model can generalize to new road networks and traffic distributions, with no additional training and a constant number of parameters, enabling greater scalability compared to prior methods. Furthermore, our approach can exploit the granularity of available data by capturing the (dynamic) demand at both the lane level and the vehicle level. The proposed method is tested on both road networks and traffic settings never experienced during training. We compare IG-RL to multi-agent reinforcement learning and domain-specific baselines. In both synthetic road networks and in a larger experiment involving the control of the 3,971 traffic signals of Manhattan, we show that different

instantiations of IG-RL outperform baselines.

1.1 Introduction

The steady growth of the world's population, combined with a lack of space in urban areas, leads to intractable road congestion, the social and environmental costs of which are well documented (Barth and Boriboonsomsin, 2008). The adaptive control of traffic signal systems based on real-time traffic dynamics could play a key role in alleviating congestion. The framing of adaptive traffic signal control (ATSC) as a Markov decision process (MDP) and the use of reinforcement learning (RL) to solve it via experiencing the traffic system is a promising way to move beyond heuristic assumptions (Chu et al., 2019; Mannion et al., 2015; Abdulhai et al., 2003; Bingham, 2001).

Value-based methods like Q-Learning constitute a cornerstone of RL. In practice, however, using a single centralized learner (Genders and Razavi, 2016; Prashanth and Bhatnagar, 2010) in an ATSC setting with numerous traffic signal controllers (TSCs) involves a combinatorial action space (Bakker et al., 2010). Multi-agent reinforcement learning (MARL), where each agent controls a single traffic light is appealing, but the proliferation of parameters, nonstationarity, and a lack or transferability (§ 1.2.1) make training and scaling challenging (Chu et al., 2019; Mannion et al., 2016; Wang et al., 2019a; Kuyer et al., 2008; Houli et al., 2010; Mikami and Kakazu, 1994; Prabuchandran et al., 2014; Arel et al., 2010).

1.1.1 Contribution

We introduce inductive graph reinforcement learning (IG-RL), which combines the inductive capabilities of graph-convolutional networks (GCNs) (Hamilton et al., 2017) with a new decentralized RL (DEC-RL) framework. In our independent Q-Learning (IQL) formulation, multi-agent RL is replaced by a shared policy, learned and applied in a decentralized fashion.

We define the topology of the computational graph of the GCN based on the current dynamic state of the road network (e.g., including the position of moving entities such as cars).

The GCN learns and exploits representations of a neighborhood of arbitrary order for every entity of the road network, in the form of node embeddings. TSCs' node embeddings are used to evaluate action-values (i.e., Q-values). The entire model is differentiable and we train it using backpropagation of the temporal difference error (TD error), following a standard Deep Q-Learning setting (Szepesvári, 2010). By having the computational graph adapt to the road network's state, we are able to:

- Train a ubiquitous policy which can adapt to new road networks, including topologies and traffic never experienced during training.

- Exploit the vehicular data at its finest granularity by representing every vehicle as a node and its corresponding vectorized representation (i.e., embedding). Current neural-network architectures used in RL-ATSC do not enable dealing with a changing number of inputs and state dimensionality. For this reason, in a traffic scenario where various types of entities such as cars and pedestrians enter, move inside of, and leave the network, these methods do not enable full granularity exploitation. They typically resort, upstream of learning, to aggregations at the lane-level (e.g., the number of vehicles approaching the intersection) (Chu et al., 2019; Wei et al., 2019; Wang et al., 2019b) or queues lengths (Chu et al., 2019; Wei et al., 2019; Wang et al., 2019b). Alternatively, they fix the number of entities to be represented in detail. For instance, Chu et al. (2019) represents the cumulative delay of the first vehicle.

To test our claims, we define a new evaluation setting in which RL-ATSC methods are tested on road networks which are never experienced during training. In particular, we design two experiments involving non-stationary traffic distributions and different road networks.

The first experiment, on synthetic road networks, evaluates IG-RL's performance against both learned and domain-specific baselines. We compare a *specialist* instance of IG-RL trained on the road network used during evaluation to a *generalist* instance that is trained on a set of road networks not including the target road network. These instances are compared to assess generalizability, transferability, and the effect of specialization on performance. In addition, we also compare two GCN architectures which respectively capture traffic demand at the vehicle and lane levels, to measure the flexibility of IG-RL in different data conditions, and its ability to exploit granularity.

In the second experiment, an IG-RL instance is transferred, with no additional training (i.e., zero-shot transfer), to the road network of Manhattan and its 3,971 TSCs. This evaluation constitutes, by far, the largest RL-ATSC experiment to date. This final transfer from small synthetic road networks to a large real-world network aims to demonstrate the scalability of learning an agnostic policy.

1.2 Related Work

1.2.1 Decentralization

The main thread of research this contribution draws on is decentralized RL (DEC-RL). In RL-ATSC, decentralization and distribution of control via multi-agent RL (MARL) is a popular approach (Chu et al., 2019; Mannion et al., 2016; Wang et al., 2019a; Kuyer et al., 2008; Houli et al., 2010; Mikami and Kakazu, 1994; Prabuchandran et al., 2014; Arel et al., 2010). However, current MARL approaches used for ATSC suffer from the following caveats:

- From the perspective of a given agent (e.g., controller), the evolving behaviors of other agents (e.g., controllers) in the global environment cause nonstationarity and instability during training (Papoudakis et al., 2019; Omidshafiei et al., 2017). As a remedy, researchers include the recent policies of other agents as part of the state of every local agent which seems to help, but does not yet solve the problem (Chu et al., 2019).

- Without parameter sharing, though more scalable than previous approaches, computational resources and time required to train MARL approaches grow with every additional signalized intersection as it requires the training of additional agents.

- The specialization of every agent on the particular environment experienced by its jurisdiction during training hinders generalization and transferability to new environments. Therefore, applying MARL on any new road network or after any modification to a road network already under RL control requires training from scratch while interacting with road users and pedestrians to gather experience.

Scalability of MARL and RL-ATSC in general, has therefore typically been limited. In our work, instead of using MARL, we approach DEC-RL as a set of similar decision processes which can be controlled by a single agent and show results on a network of 3,971 lights, which is, to the best of our knowledge, the RL-ATSC setting involving the largest number of TSCs to date.

1.2.2 Graph-Based Representation

A second thread which we draw from is graph-neural networks, which have been shown to provide richer spatio-temporal representations of the road network and facilitate coordination (Wei et al., 2019; Wang et al.,

2019b). Instead of studying coordination however, we focus on leveraging the flexibility provided by graph-neural networks.

1.2.3 Parameter sharing

Concurrent research Wei et al. (2019) and Chen et al. (2020) leverage parameter sharing to enable further scalability. However, our contribution differs significantly from the proposed methodologies in the following ways:

- Our main contribution is an inductive method which generalizes and scales to massive networks at test time. Our parameter-sharing scheme enables zero-shot learning—adaptation without additional training—to different road networks. In particular, we demonstrate that our method, trained on small random networks, can control the lights on much larger networks (e.g., with over 3,900 lights) with completely different structures (e.g., traffic lights on the whole of Manhattan).
- Our method does not resort to (arbitrary) aggregations of local traffic statistics as state features for parameter sharing. The architecture we use is specifically built to enable full granularity exploitation of demand information sensed at the lane or vehicle levels.

1.3 Background

1.3.1 Reinforcement Learning (RL)

The optimization of a decision process under uncertainty can be framed as an MDP. An MDP is defined by a set of states S, a set of actions A that the agent can take, a transition function $\mathscr{T}$ defining the probabilities of transitioning to any state s' given a current state s and an action a taken from s, and a reward function R mapping a transition (i.e., s, a, and s') to a corresponding reward. Optimizing such a decision process means finding a policy π—a policy is a mapping from a state to the probability of taking an action—which maximizes the sum of the future rewards discounted by a temporal factor γ:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} \tag{1.1}$$

where r_t represents the reward at time step t. Reinforcement learning (RL) can be used to solve an MDP whose transition and value dynamics are unknown, by learning from experience gathered via interaction with the corresponding environment (Sutton and Barto, 2018). We introduce the key RL concepts that we will use

in this paper: 1) Q-Learning a standard algorithm for optimizing a policy in RL problems, 2) the concept of adding parameterized noise for exploration, and 3) decentralized RL. This introduction is self-contained but not exhaustive (an in-depth introduction to the field is available in Sutton and Barto (2018)).

Q-Learning

The state-action-value function (Q-function) under policy π is the expected value of the sum of the discounted future rewards, starting in state s, taking action a, and then following policy π:

$$Q^{\pi}(s,a) = E_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} r_{t+k+1} \mid s_t = s, a_t = a\right] \tag{1.2}$$

From the optimal Q-function $Q^{*} = \max_{\pi} Q^{\pi}$, we can derive the optimal policy $\pi^{*}(a|s) : a = argmax_{a'} Q^{*}(s,a')$. Q^{*} can be obtained by solving the Bellman equation (Bellman, 1957):

$$Q(s,a) = r(s,a) + \gamma \sum_{s' \varepsilon S} \mathscr{T}\left(s' \mid s,a\right) \max_{a' \varepsilon A} Q\left(s',a'\right) \tag{1.3}$$

where $r(s,a)$ is the immediate reward received after taking action a from state s. Q-Learning uses transitions sampled from the environment and dynamic programming to solve the Bellman equation with no initial knowledge of transition and reward functions.

There are standard approaches that enable Q-Learning to model environments with large discrete or continuous state spaces (as is the case in this work). One of them consists in fitting the Q-function using buffers of experienced transitions that are large enough to prevent excessive correlations between observed transitions. In continuous state spaces, tabular Q-Learning (shown above) is replaced by a parametric model Q_{θ} such as linear regression or a deep neural network (Szepesvári, 2010). The latter approach is known as Deep Q-Learning.

To improve stability a common approach is to keep a lagging version of the model with recent parameters θ^{-}, called *target model* $Q_{\theta^{-}}$, and use it to compute the temporal difference error (TD error): $r(s,a) + \gamma \max_{a'} Q_{\theta^{-}}(s',a') - Q_{\theta}(s,a)$. The TD error is backpropagated to update the parameters θ following the standard Q-Learning update rule:

$$\begin{aligned} Q_{\theta}(s,a) \leftarrow\ & Q_{\theta}(s,a) \\ & + \alpha \underbrace{\left[r(s,a) + \gamma \max_{a' \varepsilon A} Q_{\theta^{-}}\left(s',a'\right) - Q_{\theta}(s,a)\right]}_{\text{TD error}} \end{aligned} \tag{1.4}$$

where α is the learning rate.

Two standard extensions of Q-Learning led to promising results in our early RL-ATSC experimentation, and we use them in all our experiments:

- Double Q-Learning (Hasselt, 2010), which consists in reducing the overestimation of action values by decoupling action selection and action evaluation.

- Dueling Q-networks (Wang et al., 2015), which consists in enabling some parameters to focus on the relative advantage of actions by using additional parameters to evaluate the average of Q-values.

Noisy Networks for Exploration

During training of RL agents, interaction with the environment involves a fundamental dilemma between exploiting what has been learned so far by the agent and gathering additional information (Sutton and Barto, 2018, Chapter 1.1). Trading off *exploration-exploitation* is critical to learning good policies in a fast and robust fashion, and it remains an active research area. Exploration via noisy networks refers to the use of parameterized noise, in the form of independent Gaussian noise added to the parameters of a neural network, in order to favor consistent and state-dependent exploratory behavior (Fortunato et al., 2017a). In comparison to decaying ε-greedy exploration—which consists in decaying the probability of acting randomly during training—noisy-network exploration often leads to better performances, enables online learning since it does not involve limiting exploration as time passes, and does not require tweaking sensitive learning schedules (Fortunato et al., 2017a). This form of exploration resulted in considerable improvements during our experiments and we use it to train all models presented in this work.

Decentralized Reinforcement Learning

Decentralized reinforcement learning (DEC-RL) distributes the control based on the assumption that the global Q-function is decomposable (Busoniu et al., 2006). Independant Q-Learning (IQL) is a straightforward DEC-RL method in which the global MDP is decomposed into DEC-MDPs which are solved using independent Q-Learners.

1.3.2 Graph-Convolutional Networks

Enabling machine learning model to learn from complex relationships between objects in graphs is challenging. Graph-Convolutional Networks (GCNs) are a recent and fruitful attempt to leverage neural-network architectures and backpropagation to address the complexity of graph data. GCNs consist in stacking k (a hyperparameter to be defined) convolutional layers as a neighborhood-information aggregation framework (Kipf and Welling, 2016). At every layer of a GCN, every node aggregates *communications* sent to it by both its neigh-

bors and itself into an embedding. This form of embedding both exploits the graph structure and the features of the nodes belonging to its neighborhood (up to order k). Using different parameters for different types of relations (Schlichtkrull et al., 2018) yields the following message propagation equation, applicable to every node:

$$n_i^{(l)} = f\left(\sum_{e \varepsilon E, j \varepsilon N_t(i)} C_{i,j,e} \cdot (W_{l_e} \cdot n_j^{(l-1)})\right) \tag{1.5}$$

where n_i^l is the embedding of node i at layer l, f is a non-linear differentiable function, E is the set of relation types, $N_e(i)$ is the 1st-order neighborhood of node i in the graph of relation type e, $C_{i,j,e}$ is a relation-specific normalization constant, and W_{l_e} is the l^{th} layer's weight matrix for message propagation corresponding to a relation of type e. Embeddings can be used to perform a supervised or unsupervised learning task, and an error signal corresponding to the task can be backpropagated through the entire model to perform gradient based optimization of its parameters.

1.4 Inductive Graph Reinforcement Learning (IG-RL)

We now introduce IG-RL, a scalable DEC-RL method. IG-RL models objects (e.g., lanes, traffic signals, vehicles) as nodes in a graph. Edges of this (dynamic) graph represent the physical connections between objects (e.g., a vehicle node is connected to its current lane node).

A GCN models this graph. Initial node embeddings encode observable features of the objects (e.g., the vehicle speed). Then, each inferred TSC-node-embedding is used to predict the Q-values of that TSC.

The key for generalization is this graph representation. By modeling road networks at the object-level (vehicles, lanes, traffic lights) parameters are independent of any intersection or road-network structure. Therefore, the GCN only instantiates a small number of parameters which are jointly learned across an entire road network or even across multiple road networks. The same parameters can then generalize and transfer, without additional training (zero-shot), to completely different intersections and road networks (e.g., different structures) with different states (e.g., different traffic conditions). In turn, this generalization translates into immediate scaling to larger networks independently of the number of intersections.

We first discuss how to decompose the road network into small MDPs and then introduce our GCN model.

1.4.1 Decentralized ATSC Decision Processes

The state and action spaces of the global ATSC MDP, which consists in the management of all controllers in a given road network, can be large and intractable in practice. We therefore decompose it into smaller decentralized decision processes. In this context, the management of every intersection by a TSC becomes an MDP.

Each MDP uses continuous states (which represent the state of nearby roads), discrete actions (e.g., whether or not to change phase), and discrete time steps (the length of which determines when the next action can be taken).

State

The state of every decentralized decision process consists in real-time information coming from local sensors and is said to be partially observable because it does not include all information about the global MDP. This state encodes both 1) current connectivity in the network, and 2) demand. Connectivity of a given intersection is defined by the current phase of the corresponding TSC, while demand can be sensed either at the level of lanes, or at the level of vehicles (§ 1.4.2 and 1.4.2).

Action

At every intersection of the road network, the flow of traffic is managed by a logical program, composed of a given number of phases, depending on the number of roads, the number of lanes, and the connectivity between lanes. This program cycles through phases, influencing connectivity between lanes, in a constant predefined order that is often known to the road users and pedestrians. In our studies, the agent must respect the program and chooses, every step, whether to switch to the next phase or prolong the current phase. The action space at every intersection is therefore binary. Compared to methods which enable complete freedom of the controller, with the agent being able to choose among any phase every time it picks an action (Chu et al., 2019; Prashanth and Bhatnagar, 2010), this formulation of the ATSC problem is more constrained. The agent must learn transition dynamics ruled by a complex cycle influencing connectivity. It is however more adaptive than methods determining duration upstream of any given phase (Aslani et al., 2017). We leave the exploration of other policy constraints for future work, although the model we introduce in § 1.4.2 is general and is not tailored to the aforementioned constraints. In fact, we discuss modelling an environment where the next action is arbitrary in § 1.4.2.

Reward

The reward for a given agent is defined as the negative sum of local queues lengths q, which corresponds to the total number of vehicles stopped[1] on a lane leading to the corresponding intersection, at a maximum distance of 50 meters of the intersection. The reward at intersection j is

$$r_j = -\sum_{i \varepsilon L_j} q_i \tag{1.6}$$

where L_j is the set of lanes leading to intersection j. This measure can both be obtained via lane or vehicular sensors. We choose this measure instead of alternatives such as wave (Aslani et al., 2017) or average trip waiting time (Mannion et al., 2016) because it is correlated to the local transition mechanisms of each DEC-MDP, and it is dense since actions tend to impact this measure quickly. These last two points translate into eased attribution of reward signals (Chu et al., 2019).

Step

We discretize time into one-second time steps. In other words, actions are selected every second.

Episode

Depending on the experiment (§ 1.5), an episode either ends as soon as all trips are completed or after a fixed amount of time.

1.4.2 Model

The control of all traffic signals is a family of decision processes. Each decision process in this family involves a controller, lanes, connections between lanes, and vehicles (e.g., nearby ones). A single GCN is used to model these entities and their dynamic relationships. IG-RL proposes systematic sharing of the parameters of the GCN among all objects and relationships of the same nature.

In addition, the flexibility of GCNs enables representing an arbitrary number of entities that changes over time (e.g., vehicles moving in and out of the network) in a detailed way (i.e., as nodes in a graph).

Architecture

Fig. 1.1 illustrates the GCN for modelling a traffic signal control. It includes 4 types of nodes:

[1] A vehicle is considered stopped if its speed is inferior to 0.1km/h.

- *TSC node*, which represents the state of a controller.
- *Connection node*, which represents the state of an existing link between an entry lane and an exit lane. A link (i.e., connection node) exists between an entry lane A and an exit lane B if, under at least one phase of the TSC program, a vehicle on A is allowed to continue its travel on B.
- *Lane node*, which represents the state of a lane.
- *Vehicle node*, which represents the state of a vehicle.

The GCN uses the following edges and edge types:

- An edge between every node and itself (4 types of edges).
- Bidirectional edges between every *TSC node* and *Connection nodes* corresponding to connections the TSC can influence by changing its phase (2 types of edges).
- Bidirectional edges between every *Connection node* and *Lane nodes* corresponding to its entry lane (2 types of edges) and its exit lane (2 types of edges).
- Bidirectional edges between every *Lane node* and *Vehicle nodes* corresponding to vehicles located on the corresponding lane, given that the GCN exploits detailed vehicular data (2 types of links).

Every layer of the GCN uses one set of parameters per edge type to perform message-propagation (W in Eq. 1.5). Our early experimentation with IG-RL and GCNs suggested that using normalization constant (C in Eq. 1.5) was detrimental to performance. In contrast to Equation 1.5, we do not include a normalization constant during message-passing

$$n_i^{(l)} = f\left(\sum_{r \varepsilon R, j \varepsilon N_r(i)} W_{l_r} \cdot n_j^{(l-1)}\right). \tag{1.7}$$

The embedding n_j^0 of a node j is initialized using its features (§ 1.4.2). A fully connected layer, with a single set of parameters, maps the final node embedding of a TSC to the Q-values corresponding to its selectable actions.

Since preliminary experiments demonstrate that noisy exploration enables faster and more robust training, as well as better performance compared to ε-greedy exploration, parameterized Gaussian noise is added to the parameters of this final mapping. An initialization of 0.017 for the variance parameters performs well in multiple supervised (Fortunato et al., 2017a) and reinforcement learning tasks (Fortunato et al., 2017b), and so we also use that value. Fig. 1.1 is an illustration of the architecture of the full model for a simple road network with a single TSC and three vehicles.

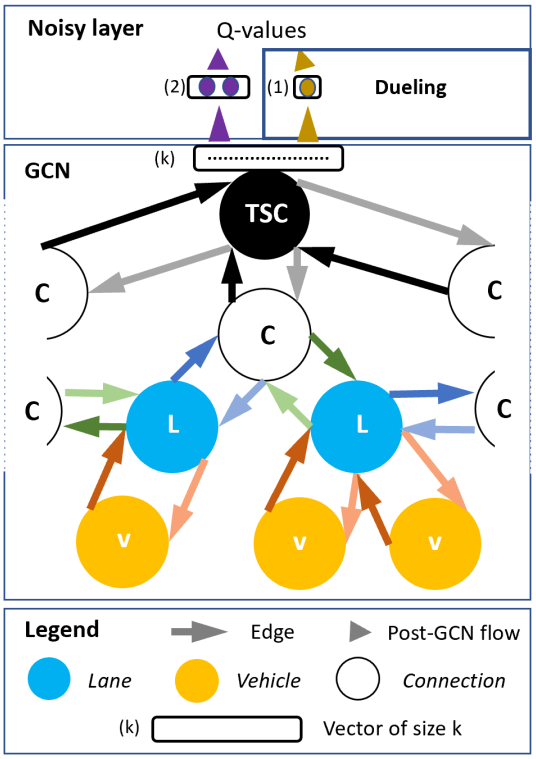

Figure 1.1: Local computational graph.

We illustrate the computational graph corresponding to one of the connections a TSC observes at its intersection. One vehicle is located on the connection's inbound lane while two are located on its outbound lane. In this particular example, both lanes are only involved in a single connection on the intersection and respectively start from and lead to other intersections. For every layer of the GCN, message-passing is performed along every edge following Equation 1.7. Edges of the same type share the same color and parameters. The final embedding of the TSC node is then passed through a fully connected layer to obtain its Q-values (one per possible action).

Action Space Flexibility

Given the computational flexibility of GCNs, the model can be easily adapted to different action spaces. For instance, enabling the choice of an arbitrary phase (instead of a program that cycles through phases) is achieved by representing phases as nodes of a new type, and linking these nodes to both their corresponding TSC node and the nodes representing connections they influence. This way, one Q-value can be computed per phase node. Preliminary experiments using this formulation seem promising.

Parameter Sharing

Using a graph representation at the object level (TSC, connection, lane, car) enables sharing parameters between objects of the same type, both inside each decision process (e.g., two lanes on the same intersection), and across all decision processes belonging to the same family (e.g., two lanes on unrelated intersections). This

parameter sharing is useful because:

- It uses a constant number of parameters independent of the number of underlying processes.
- During training, these shared parameters are updated using experience gathered from a variety of decision processes (a variety of intersection architectures belonging to a variety of road network architectures).

Generalization and Transfer via Inductive Learning

Consistent parameter sharing enables training a unique set of parameters using experience gathered from a set of decision processes involving diverse intersection topologies, road network topologies, and traffic distributions. Enforcing such diversity can translate into better generalization to unseen MDPs from the same family of MDPs (Sadeghi et al., 2018; Akkaya et al., 2019). In this work, the learning of general and transferable patterns is based on two key components:

- The proposed architecture, based on a GCN, which enables parameter sharing without sacrificing granularity.
- The sampling strategy over the family of MDPs, which in this context refers to the way decision processes are created. The sampling strategy should ensure that the training set is representative of the corresponding family of decision processes.

Marginalizing Nonstationarity

Nonstationarity, caused by the evolution of the behavior of neighbor agents during training, remains a critical challenge for DEC-RL (§ 1.2.1). We hypothesize that emphasizing diversity in the training set, as described in § 1.4.2, can limit nonstationarity and stabilize learning by marginalizing out the experience of a particular intersection with respect to its neighbor intersections.

Features

Our approach exploits data that can be accessed from sensors representing the various entities that make up the road network and traffic. At any time step in a given road network, every element defining the state is represented through a node in the GCN. The architecture of the GCN is built on the assumption that the current state of the controller (i.e., the current phase and its corresponding local connectivity) is known, and that traffic information coming either from lane sensors or vehicle sensors is also available. Other assumptions would

result in the choice of a different GCN architecture or in the use of different features. The features used for different types of nodes are summarized in Table 1.1. *Current speed* represents the current speed of a vehicle in km/h, *Position on lane* represents the relative location of a vehicle with respect to the lane it is on. It is defined as a proportion of the lane's length. *Length* is the length of a lane in meters, *Number of vehicles* is the total number of vehicles captured by the lane's sensor, *Average vehicles speed* is the average speed of vehicles captured by the lane's sensor, *Is open* represents whether a connection is opened under the current phase, *Has priority* represents whether, if an open connection between an entry and an exit lane has priority[2] or not. *Number of switches* to open is the number of switches the controller has to perform before the next opening of a given connection. *Next open has priority* defines whether the next opening of the connection will have priority or not. *Time since last switch* is the number of seconds since a traffic controller performed its last phase switch.

Table 1.1: Initial state variables for every type of node.

Type of node	**State features**
Vehicle	*Current* speed, *Position on lane*
Lane	*Length*, *Number of vehicles*, *Average vehicles speed*
Connection	*Is open*, *Has priority*, *Number of switches to open*, *Next opening has priority*
TSC	*Time since last switch*

1.5 Experiments

Using the SUMO traffic simulator (Lopez et al., 2018), we evaluate the performance of several IG-RL instantiations using small synthetic road networks and a large real road network, and compare it to the results of several baselines.

Our study demonstrates that:

- IG-RL outperforms all included baselines on the ATSC task in a traditional evaluation setting.
- Training IG-RL under several varying initial conditions (including different road networks) yields the best performing method and enables zero-shot transfer.

[2]Vehicles may use a connection (i.e., go from its entry lane to its exit lane) if no vehicle uses a higher prioritised connection

- Training IG-RL on small synthetic road networks with limited computational resources enables efficient zero-shot transfer to real road networks, and such a transfer translates into improved scalability.

The design of a synthetic training set (of random road networks) before transferring the learned policy to a more realistic setting (Manhattan) is inspired by other zero-shot transfer approaches in reinforcement learning (Oh et al., 2017; Higgins et al., 2017).

1.5.1 General Setup

Network Generation

The structure of a road network is picked, at random, from a set which consists in the cross product of the number of intersections (typically between 2 and 6), the structure of every intersection, the length of every edge (between 100 and 200 meters), the number of lanes per route (between 1 and 4 per edge), and general connectivity. A given random seed will lead to the generation of the same network, enabling exact comparison of performance between policies. Examples of randomly generated networks from our procedure are shown in Fig. 1.2.

TSC-Programs

TSC-programs, or cycles, are generated by SUMO and typically include several phases, corresponding to the legal states at a given intersection.[3] Programs are therefore not restricted to abstracted axis connectivity such as North-South or East-West. Our synthetic network generation procedure (§ 1.5.1) leads, on average, to the creation of TSCs whose programs are composed of 5.5 phases. We leave experimentation with other types of TSC-programs to future work.

Traffic Generation

A given number of trips are to be generated per second, on average, during an episode. This value is referred to as the traffic regime. Every generated trip is assigned a trajectory. A trajectory is defined by a starting lane, a sequence of intermediary lanes, and a final lane. Every valid trajectory includes at least 2 lanes. To simulate the urban setting, a trajectory can start and end anywhere in the network. Parameters defining non-uniform distributions used for assigning trajectories to trips are re-sampled every 2-minutes to ensure non-stationary distributions.

[3] "All red" phases are not included as part of SUMO's default TSC-Programs generation.

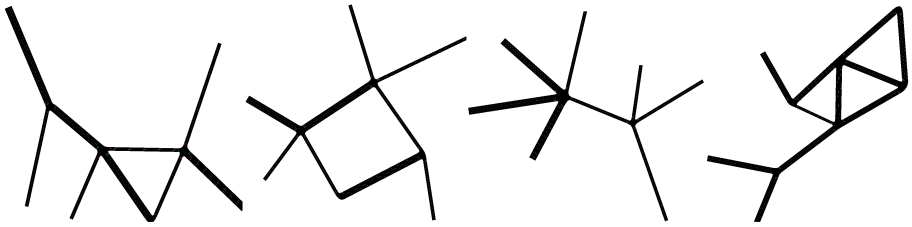

Figure 1.2: Four randomly generated road networks.

Thickness indicates the number of lanes per direction (between 1 and 2 per direction for a maximum of 4 lanes per edge

Training

Models are trained on a set of 30 simulations running in parallel. During training, exploratory behaviors tends to lead to catastrophic congestion from which recovery becomes unlikely. This is a common RL challenge which makes training from long episodes inefficient in practice. For this reason, as in Chu et al. (2019), every simulation runs 30,000 steps divided into independent short episodes of 500 steps (seconds) each. All simulations run on a single randomly generated road network. We also use this network for evaluation in our first experiment (§ 1.5.4). Further, each simulation is initialized using a different random seed to ensure a variety of observed transitions. Models are trained using the exact same environment (network and traffic). The parameters of target Q-networks are updated every 100 parameters updates of the main Q-networks (§ 1.3.1) The learning rate is 0.001 and the batch size is 16.[4]

Evaluation

All methods are evaluated using the same target networks and identical traffic. However, evaluation introduces traffic regimes (densities) never experienced during training, to evaluate how the methods generalize to new traffic distributions. We evaluate performance by measuring the evolution of the instantaneous delay:

$$\begin{aligned} s_v^* &= min(s_{v^*}, s_l), \\ d_t &= \sum_{v \varepsilon V} (s_v^* - s_v)/s_v^* \end{aligned} \tag{1.8}$$

where V is the set of all vehicles in the road network, s_{v^*} is the maximum speed of the vehicle, s_l is the maximum allowed speed on the lane the vehicle is on, s_v is the current vehicle speed, and d_t is the total instantaneous delay at time step t. Given that all models are evaluated on the same sets of networks and trips, we can pair the same trips together for hypothesis testing (using paired t-tests), and to study distributions of trips-durations differences.

[4] A batch is composed of networks. A network typically involves multiple intersections

Constraints

Phases involving yellow lights last exactly five seconds, as suggested by Aslani et al. (2017). For all other phases, in order to maximize adaptiveness, instead of using a fixed time of 5 seconds between every action as suggested by Chu et al. (2019), a TSC must spend at least 5 seconds in each phase before being allowed to switch. The parameters associated with these time constraints can be modified and their values are expected to influence achievable performance. We leave thorough experimentation with time constraints (e.g., minimum green time for pedestrians) to future work. For trainable methods, constraints are enforced by ignoring invalid actions selected by the agent. In other words, in a given time step, even though the agent can pick an action, its decision will have no influence on the MDP if its chosen action breaks a constraint.

Action-Correction

Ignoring an agent's invalid actions forces it to learn that the effect of the available actions on connectivity are dependent on other logical factors like the time since the last phase change and the nature of the current phase. To ease learning, and since it is preferable to focus on the actual influence an agent can have on its MDP, we replace, during experience replay, binary decisions which were chosen by the agent at every time step, by whether a light switch was performed. We also force the greedy policy, in the creation of the temporal difference target, to only consider actions which could be performed at the corresponding step. To adapt the proposed action-correction methodology to a policy which is able to choose an arbitrary phase, the selected action can simply be replaced by the action corresponding to the selection of the phase that is actually active at the next step.

Robustness

The seeds we use in the experiment influence the random generation of road networks, traffic distributions, initial weights of neural networks, and the Gaussian noise used for exploration during training. To ensure the robustness of our conclusions, every experiment, including both training and test, is repeated 5 times using different seeds. For instance, the training on 30 different simulations occurs 5 times for a total of 150 simulations. The same goes for evaluation. Reported test results are an aggregation over these 5 runs for both experiments.

1.5.2 IG-RL models

Architectures

Our complete GCN architecture, referred to as IG-RL-Vehicle (IG-RL-V), uses a 3-layer GCN and vehicle-level traffic information for both state[5] and reward (§ 1.4.1).

We also introduce a lighter GCN architecture, IG-RL-Lane (IG-RL-L) which uses a 2-layer GCN and lane-level traffic information for both state[6] and reward (§ 1.4.1). Even though IG-RL-L does not exploit demand at the vehicle-level, it has the added benefit of requiring constant computational time with respect to the traffic demand.

For both IG-RL-L and IG-RL-V, the number of layers is chosen to be the minimum which ensures that information representing demand can reach the nearest TSC(s) nodes. For both GCN architectures, we use node embeddings of size 32 as further increasing this hyperparameter did not result in improved performances.

General versus Specialized

The target network refers to the road network used to test the performance of all methods after training and must be distinguished from the training set of road networks since the same IG-RL instance can be applied, and therefore trained on, different road networks. Two training sets of road networks are built for IG-RL.

The *specialist set*, used to train Specialist-IG-RL (S-IG-RL), aims to evaluate intentional specialization on a given network. It consists in training the model exclusively on the target road network, as described in § 1.5.1.

The *generalist set*, used to train Generalist-IG-RL (G-IG-RL), aims to study IG-RL's transfer abilities. This set is composed of independent road networks and does not include the target network. In such a context, performance reflects generalizability to new road network and intersection architectures. In addition, as different road networks imply different traffic distributions even when the regime (§ 1.5.1 and § 1.5.4) is left unchanged, performance reflects generalizability to new traffic distributions for all studied regimes, even the *default* one which is used during training.

Our experiments with hybrid training sets involving both the target road network and random road networks did not lead to improved performance compared to the generalist approach and are omitted for brevity.

[5] Lane level traffic features are not included in IG-RL-V.
[6] Vehicle nodes are not included in IG-RL-L.

Evaluating Action-Correction

To evaluate the importance of action-correction (§ 1.5.1), we introduce a version of IG-RL-L for which no action-correction is performed during training. This ablation study is referred to as IG-RL-no-correction (IG-RL-n-c).

1.5.3 Baselines

We now describe the models used as baselines in both of our experiments.

Fixed-Time Baseline

This baseline follows the cycle of phases using predefined and constant durations generated by SUMO (Lopez et al., 2018) based on the architecture of each intersection.

Max-moving-car heuristic (Greedy)

This dynamic heuristic-based method aims at ensuring that as many vehicles as possible are moving on inbound lanes at any given time, in the spirit of the popular baseline MaxPressure (Varaiya, 2013) under a cyclic setting. At every intersection, the controller switches to the next phase if, on inbound lanes, the number of stopped vehicles is superior to the number of moving vehicles, and prolongs the current phase otherwise.

MARL-IQL

MARL-IQL is a learned baseline. In this model, every intersection is controlled by its corresponding agent (i.e., with its own unique parameters), which learns to solve its own local MDP. Every agent, parameterized by a deep Q-Learner network, takes the exact same variables that are available to IG-RL-L as inputs (Table 1.1 and § 1.5), with the exception of lanes' *lengths*, which is constant for each MDP. Every Q-Learner consists of a first hidden layer of 256 neurons, a second hidden layer of 128 neurons, and a last hidden layer of 64 neurons, which is the result of a grid-hyperparameter search. While this model would constitute a "vanilla" multi-agent implementation, we chose to add both Double Q-Learning and a Dueling architecture to every Q-Learner (§ 1.3.1) as done in Liang et al. (2019). In addition, the introduction of noisy parameters (§ 1.3.1) and action-correction (§ 1.5.1), were the elements which led the the biggest improvements during training. These improvements have the two following objectives:

- Create a strong baseline.

- Enable an ablation study. The only difference between MARL-IQL and S-IG-RL-L is that the former uses one neural network per intersection to learn the local Q-function and compute the local Q-values, while the latter uses the same GCN to do so at every intersection. Apart from this, all the information and methods they use for training are identical.

1.5.4 Experiment 1: Evaluating Generalization using Synthetic Road Networks

Description

In this experiment, we evaluate policies on a single road network. The same road network is used to train all methods except G-IG-RL. As MARL-IQL does not scale as efficiently as IG-RL methods, we picked a small road network.

Evaluation setting During evaluation, trips are generated during the first hour. An episode ends as soon as all trips are completed. Performance of all methods are evaluated using 30 different random seeds for traffic generation. Evaluation is run twice, with two distinct traffic regimes (denoted *default* and *heavy*), in order to evaluate the ability of models to generalize to traffic densities never experienced in training, as well as to study the performance in different density settings. The first evaluation traffic regime, which is the one used during training, introduces an expected one vehicle every 4 seconds in the network, while the second traffic regime introduces twice as many per second on average.

Overcoming overfitting While methods are evaluated during episodes of more than an hour, they are trained on shorter episodes of 500 seconds (§ 1.5.1). As more vehicles enter the network, congestion can therefore increase past levels experienced during training (which again implies new traffic distributions) even in the *default* traffic regime used during training. Since conditions differ between training and evaluation, the task is very challenging and the learning of generalizable patterns and behaviors (policies) is key to perform in both traffic regimes.

Results

IG-RL models We now refer to the group of methods using both IG-RL and action-correction as IG-RL models, and highlight them in all following figures (green font). Means and medians of trip durations, reported in Figs. 1.3 and 1.4, are lower when using IG-RL models, compared to other methods. Instantaneous total delays, reported for every simulation step in Figs. 1.5 and 1.6, show that congestion increase is slower, and

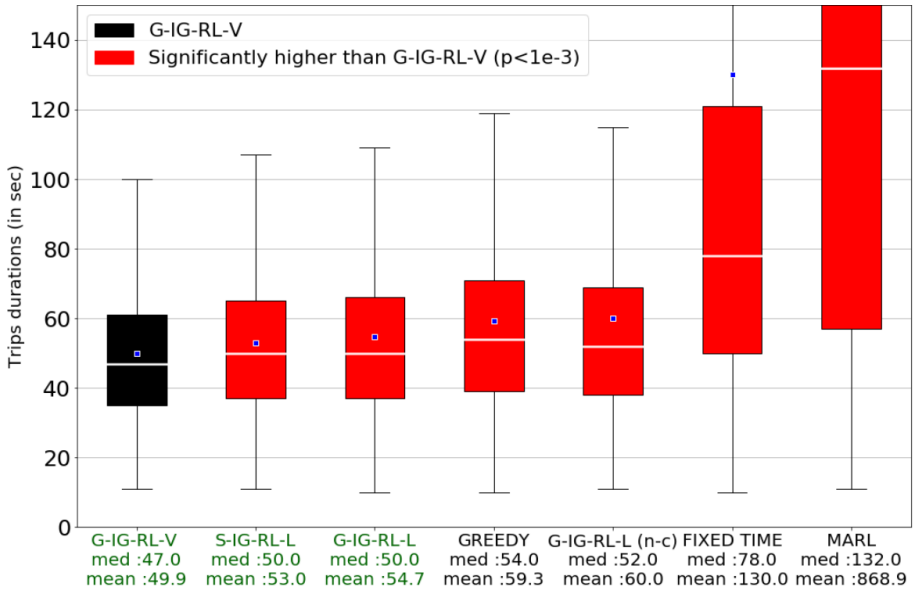

Figure 1.3: Distributions of trips durations: **Default** Traffic Regime | Synthetic Road Networks.

This figure presents the distribution of trips durations during test in the same traffic regime used for training. The vertical axis is cut at MARL's median trip duration for readability. Since every method is evaluated using the exact same trips, for every method, we perform t-tests on the paired sample of: 1) The durations of the trips when using the given method and 2) The durations of the trips when using G-IG-RL-V. The results of these paired t-tests suggest that G-IG-RL-V significantly outperforms every other method, while the boxplots, means, and medians of trips durations suggest that IG-RL models using action-correction outperform all other methods in general.

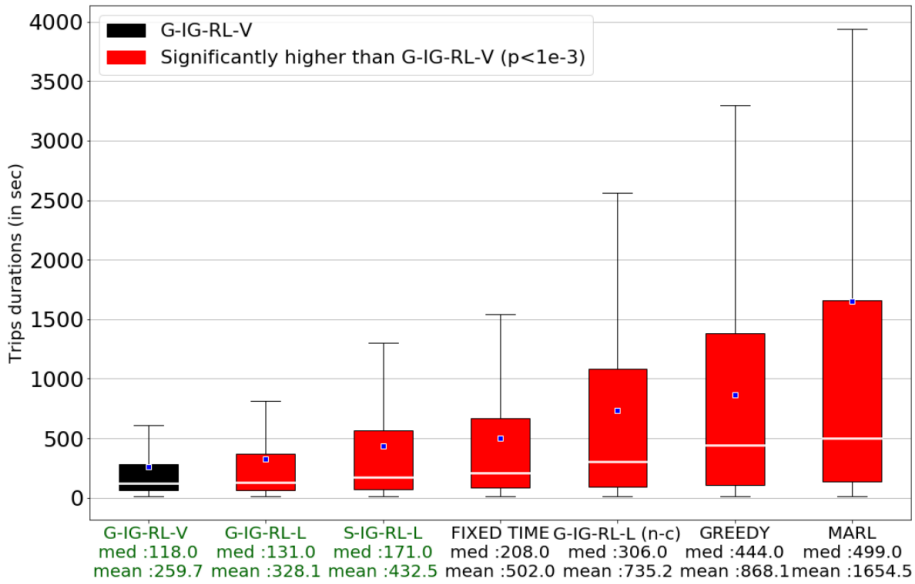

Figure 1.4: Distributions of trips duration: **Heavy** Traffic Regime | Synthetic Road Networks.

This figure presents the distribution of trips durations during test in the heavier traffic regime that was not experienced during training. Results are again in favor of G-IG-RL-V and IG-RL methods in general. Performance gains are larger in this setting compared to the *default* traffic regime. (Fig. 1.3

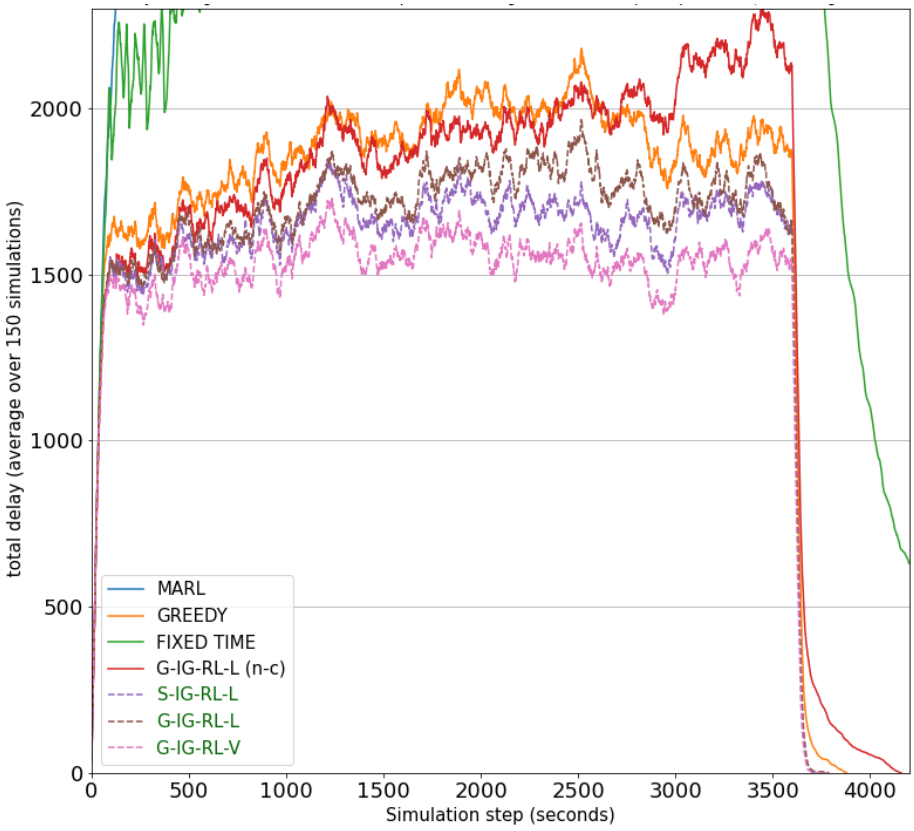

Figure 1.5: Total Delay Evolution: **Default** Traffic Regime | Synthetic Road Networks

For clarity, this figure focuses on competitive approaches with lower delays (which stabilize early on). The delays for Fixed-Time and MARL continue to increase and do not converge within an hour. A complete version of this figure is provided as supplementary material (§ **??**).

recovery[7] is faster using IG-RL models in both traffic regimes. In addition, in the *default* traffic regime, these methods reach the lowest steady-state in terms of congestion level.

G-IG-RL-V is the best performing method, with both the lowest trip-durations (Figs. 1.3 and 1.4), the slowest congestion increase and the fastest recovery (Figs. 1.5 and 1.6). In addition, distributions of paired-trips-durations differences, between every method and G-IG-RL-V, reported in Fig. 1.7 and 1.8, show that a large majority of trips are completed faster using G-IG-RL-V. In the *default* traffic regime, some trips are delayed by up to a thousand seconds when using MARL or Fixed-Time. In comparison, no trips are delayed by more than a hundred seconds under the G-IG-RL-V policy. In the *heavy* traffic regime, the conclusions are similar, with even higher differences compared to other methods. Fig. 1.7 and 1.8 also show that in addition to improving average performance, G-IG-RL-V tends to distribute delay in a more equitable way than any other method (numerous extreme delays are avoided and replaced by a few shorter delays).

[7]Recovery refers to the speed at which delay decreases after the simulation reached 1 hour and no additional trips are generated.

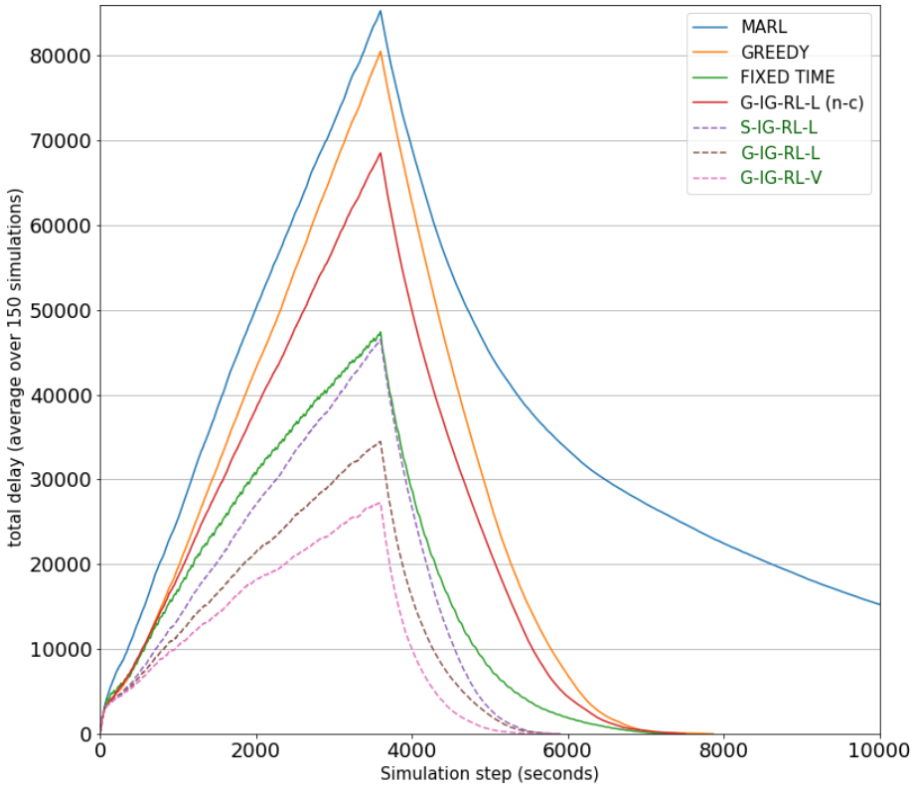

Figure 1.6: Total Delay Evolution: **Heavy** Traffic Regime | Synthetic Road Networks.

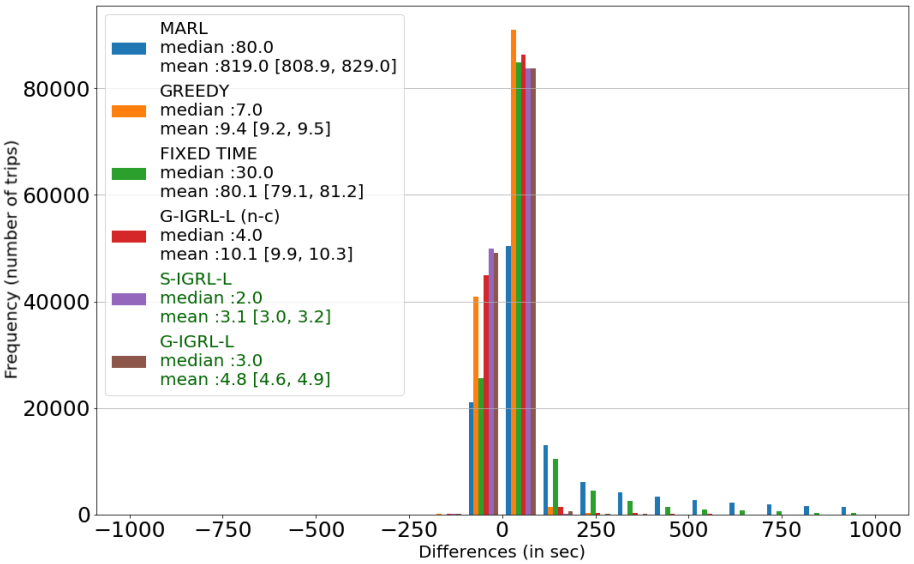

Figure 1.7: Differences of Paired Trips Durations (compared to G-IG-RL-V): **Default** Traffic Regime | Synthetic Road Networks

This figure presents the histogram of the differences of paired trips durations (§ 1.5.1) between G-IG-RL-V and each other method. Evaluation involved the same traffic regime as used in training. Positive values indicate trips which were completed faster using G-IG-RL-V than the compared method. Negative values indicate trips which were completed faster using the compared method than G-IG-RL-V. Medians, means, and confidence interval for means of the differences are also reported. All these results show that when using G-IG-RL-V, trips tend to be completed faster.

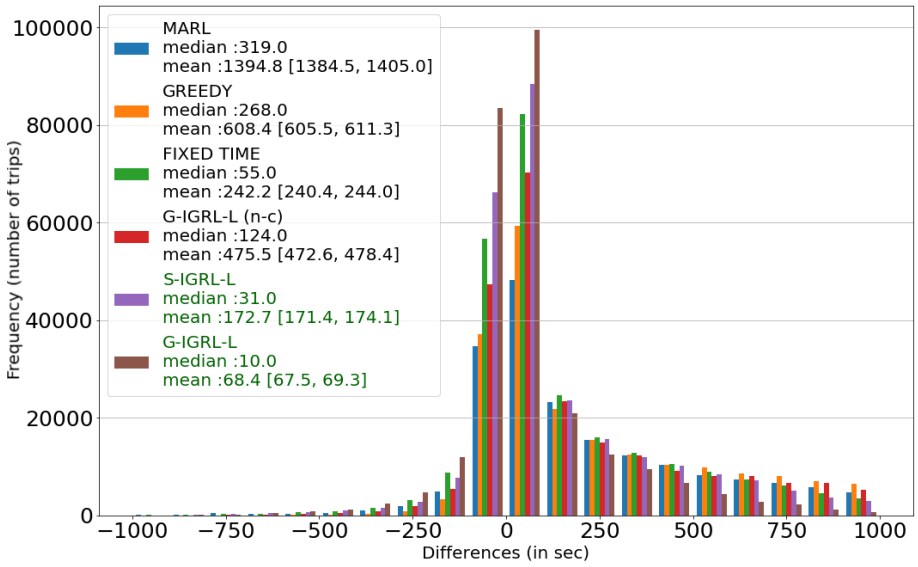

Figure 1.8: Differences of Paired Trips Durations (compared to G-IG-RL-V): **Heavy** Traffic Regime | Synthetic Road Networks

This figure reports the same metrics as Fig. 1.7, but the evaluation uses the heavier traffic regime not experienced during training. Results are again in favor of G-IG-RL-V, with larger differences against all compared methods than under the *default* traffic regime. In addition, extreme delays are avoided when using G-IG-RL-V.

G-IG-RL-L and S-IG-RL-L A key finding emerges from comparing G-IG-RL-L and S-IG-RL-L. Even though their performance is similar in the *default* regime, G-IG-RL-L outperforms S-IG-RL-L in the *heavy* regime. This shows that G-IG-RL enables the efficient learning of transferable patterns, and that learning from a variety of road networks is key to enable generalization to new traffic regimes. This result further emphasizes the limits of training any policy, whether it is a single policy in the context of centralized RL or multiple policies in the context of MARL, on a single road architecture. (Recall that allowing for modelling different sets of train and test networks is one of our key contributions.)

IG-RL-V and IG-RL-L G-IG-RL-V outperforming other methods in both traffic regimes highlights the flexibility of the GCN to exploit a finer level of granularity (i.e., data sensed at the vehicle level). However, G-IG-RL-L and S-IG-RL-L being the second bests (ex-aequo) performing methods in the *default* traffic regime, and G-IG-RL-L being the second best performing method when generalizing to a heavier traffic regime, we can conclude that IG-RL can also perform well with demand sensed at the lane level. As the comparison between S-IG-RL-V and S-IG-RL-L leads to the same conclusion as the one between G-IG-RL-V and G-IG-RL-L, we did not include S-IG-RL-V in this paper to improve readability.

Action-Correction The performance of G-IG-RL-L(n-c) shows that removing action-correction (§ 1.5.1) has an important negative impact on performance and generalization to the *heavy* traffic regime, with trips lasting longer, congestion increasing faster, and recovering slower than every other IG-RL-based method.

MARL-IQL MARL-IQL is the worst performing model (it is even outperformed by the fixed-time baseline). We hypothesize that two reasons explain its performance:

- Overfitting: As mentioned in § 1.5.4, the current experiment is very challenging with respect to overfitting in both traffic regimes. During training, we observed that MARL-IQL largely outperforms the Fixed-Time policy, and its performance is both: 1) on par with the performance of the Max-moving-car heuristic (Greedy) policy, and 2) much closer to the performances of other learned models as compared to the gap in performance at test time. This observation suggests that policies learned by MARL-IQL do not generalize well to new demand and traffic distributions.

- Nonstationarity: As described in § 1.2.1, nonstationarity is a key reason for instability in MARL. It is no surprise that naively applying deep neural networks to MARL-IQL can lead to subpar performance. Note that this is the case even though we use an improved version MARL-IQL (§ 1.5.3). In Chu et al. (2019), for instance, a MARL-IQL method performed significantly worse than all other formulations, including one based on vanilla linear regression.

Conclusions This experiment highlights that IG-RL's 1) parameter-sharing (§ 1.4.2), 2) training schemes (§ 1.4.2), and 3) vehicle-level-demand representation (§ 1.5.2) are key to its performance. The first is observed when comparing MARL-IQL to IG-RL models (parameter sharing distinguishes S-IG-RL-L from MARL-IQL as stated in § 1.5.3). The second is shown by the performance of the G-IG-RL models (especially in the *heavy* traffic regime). Further, training all parameters with a variety of experiences (from multiple intersections using S-IG-RL and even multiple road networks using G-IG-RL) and avoiding overparameterization helps IG-RL prevent overfitting (a key challenge in this experiment as stated in § 1.5.4). We hypothesize that this variety also marginalizes out nonstationarity (§ 1.4.2). Finally, comparison of G-IG-RL-V and G-IG-RL-L demonstrates the third highlight.

1.5.5 Experiment 2: Transfer and Scaling to Manhattan

Description

In this experiment the target network is Manhattan (Fig. 1.9), which involves 3,971 TSCs, some complex intersections being managed by several TSCs, and 55,641 lanes. This network provides a testbed for evaluating the generalization and scaling properties of G-IG-RL approaches.

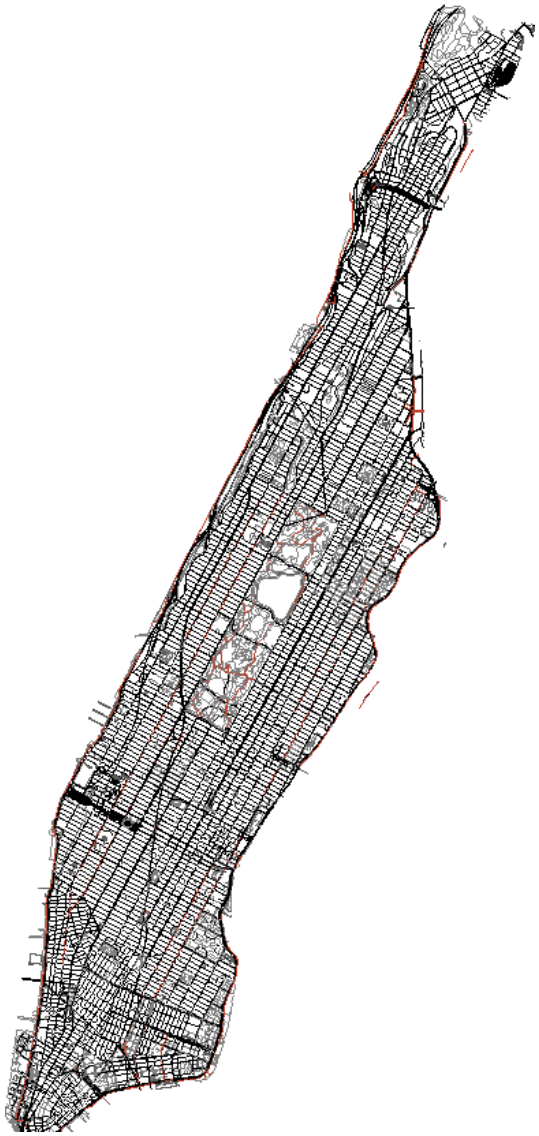

Figure 1.9: Manhattan road network.

Training MARL based methods on that many intersections would imply massive time (running the corresponding simulations) and memory (storing and training one model per intersection) resources. Training S-IG-RL would enable the use of a constant number of parameters, but gathering the amount of experience

required for RL on such a large road network would not be possible in a reasonable time with the simulator and computational resources that we use. Since it can leverage zero-shot transfer (i.e. no additional training) and keep the number of parameters constant, G-IG-RL, which is trained on a set of small networks as described earlier, is the only learned method usable in this large-scale context.

For a given road network, G-IG-RL-L requires fixed computational time, while both the computational time and memory requirements of G-IG-RL-V increase linearly with the number of vehicles in the network. Considering the size of the experiment, even though we expect G-IG-RL-V to perform better than G-IG-RL-L considering the results of the previous experiment, we only evaluate the latter.

Evaluating methods on this large network is more computationally demanding compared to Experiment 1. Every method is only evaluated on 5 instances of randomly generated traffic for a fixed duration of one hour and run using a single traffic regime which introduces a vehicle every second in the network. Compared with what G-IG-RL experienced during training, and what was evaluated in Experiment 1, adding a vehicle every second in a network as large as Manhattan leads to light traffic density which increases in certain areas as vehicles enter the network following asymmetric distributions based on a given random seed. Waiting for all trips to end in a large network would mean prolonging the duration of some simulations in an unreasonable way, considering that some trips might require a lot of time to be completed. For computational concerns, episodes end after an hour. We report aggregated results, but trips cannot be compared via pairing, as done in Experiment 1, without introducing bias.

Results

A first result is the bare fact that by using G-IG-RL-L, the 3,971 traffic signal controllers are operable, using a single GPU, and a model involving only 11,076 parameters. In addition, we can see that the total instantaneous delay, measured at every time step and reported in Fig. 1.10, shows that G-IG-RL outperforms baselines which can scale to Manhattan, in terms of the slowest congestion increase and the lowest congestion-level steady-state. Furthermore, it confirms the ability of IG-RL to generalize to different (here lighter) traffic regime than experienced in training. These results demonstrate that using synthetic randomly generated road networks for training is a viable and efficient way to learn policies that can transfer well to real road networks. This evaluation setting is similar in spirit to what is done in sim2real for robotics (Sadeghi et al., 2018; Akkaya et al., 2019).

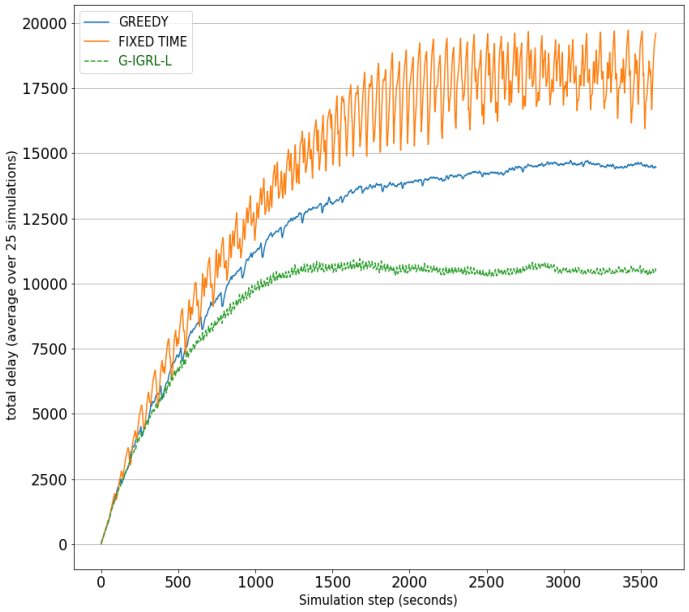

Figure 1.10: Total Delay Evolution: **Light** Traffic Regime | Manhattan Island

As all simulations are running on the same road network, and the fixed time controls of many intersections are synchronized, the local delays are highly correlated across intersections and across simulations under this method, hence the periodic pattern.

1.6 Complexity

In this section, we discuss time complexity of IG-RL with respect to :

1. Preprocessing states and building the computational graph required to perform convolutions.
2. Training & Evaluation under the settings described in the manuscript
3. Scaling & Transfer

1.6.1 Building the computational graph

The computational graph is the preprocessed/network representation of the state of the environment at a given timestep. It is on this graph that convolutions are performed to obtain Q-values. There are two 'parts' in the computational graph: The 1st part is the fixed part (the architecture of the road network which does not evolve). It includes TSC nodes, connection nodes, lane nodes, and the lane length feature for all lanes. This part is only built once per road network (no need to update it). The 2nd part is the dynamic part. It includes vehicle nodes, and most node features (see § 1.4.2. This is the part that is updated every timestep.

1.6.2 Training & Evaluation

To run the experiments presented in the manuscript, we used 2 GPUs (RTX 2080Ti, Memory : 11GB) for training and 1 for evaluation, and a single CPU (AMD Ryzen Threadripper 2950X) for both training and evaluation.

Synthetic road networks

As mentioned in 2.5.1 : Experiments - General Setup - Robustness, to ensure the robustness of our conclusions, every experiment, including both training and evaluation, is repeated 5 times using different seeds. We now provide average estimates (over these 5 instances) of the required time to complete training and evaluation steps for IG-RL (once).

Table 1.2: Durations on synthetic road networks

Architecture	IG-RL-L		IG-RL-V
Training	Specialist	Generalist	
mean (min)	69.54	66.97	79.6
std dev	6.07	0.48	1.09

Table 1.3: Training (5 runs)

Architecture	IG-RL-L		IG-RL-V
Training	Specialist	Generalist	
mean (min)	4.29	4.27	5.24
std dev	0.38	0.40	0.66

Table 1.4: Evaluation (5 runs)

IG-RL-L Vs. IG-RL-V

- IG-RL-L does not use vehicle nodes. For this reason, the graph will only be built once (all nodes are fixed) and only features will need to be updated between timesteps. Message-passing complexity is constant between timesteps (does not vary with demand) for this version, hence the lower training and evaluation durations.

- IG-RL-V uses vehicle nodes. For this, reason, the graph needs to be updated at every timestep. Features must also be updated. Message-passing complexity increases linearly with the number of vehicles (between timesteps) for this version, hence the longer training and evaluation durations.

Generalist Vs. Specialist The training duration of G-IG-RL(L and V) has relatively low variability because of the variety of road network architectures used to train it. In comparison, the single road network architecture used for S-IG-RL can have a significant impact on training duration.

Manhattan road network

To build the Manhattan road network and all included objects, we imported all road network data/details from OpenStreetMap on SUMO. We generate traffic/demand using the same method that is described in the 1st experiment (§ 1.5.1).

For the corresponding experiments, we completed all 5 evaluation runs simultaneously (in parallel). Completing the evaluation scenario described in Experiment 2 for G-IG-RL-L took ∼23 hours and 30 minutes.

1.6.3 Scaling & Transfer

Scaling (number of intersections)

Every intersection operates independently (in a decentralized fashion). Both the construction of the computational graph and the message-passing can be performed independently as well. Consequently:

- The total computational cost of both the construction of the computational graph and message passing scales linearly with the number of intersections (assuming intersections have architectures of comparable complexity).
- Control of all TSCs can be parallelized (performed independently).

Transfer complexity

Where other methods would require to re-train from scratch on any new intersection and/or road network, requiring many experiences and substantial training for any new setting, one of the main advantage of G-IG-RL is the ability to make training time independent of the applications of the model after it is trained.

1.7 Algorithm (pseudocode)

In this section, we provide a pseudo-code which describes how the gathering of experiences and the training of the model are orchestrated.

1.7.1 Interaction with SUMO

SUMO's Traffic Control Interface (TraCI) enables retrieving values of simulated objects and manipulating their behavior "on-line". Simulation processes run multiple instances of SUMO in parallel and are used to both 1)

gather experiences for training and 2) perform evaluation.

Interaction with SUMO for IGRL

At every timestep t of a given episode (on a given simulation process), data retrieved using TraCI constitutes the current state (s_t) of the environment/road network (and therefore the state of every included MDP). This data is preprocessed into a network G_{s_t} (nodes, node embeddings/features, and edges) on which we perform graph convolutions to obtain the Q-values for every TSC (§ 1.4). For every TSC, the action with the highest predicted Q-value is selected and fed back to SUMO to compute/obtain the next state. Exploration is enabled by the use of a noisy fully connected layer (§ 1.4).

Algorithme 1 : Training IG-RL (with double Q-Learning)

Input : M : Total number of steps per simulation process

Input : T : Episode duration

Input : J : Number of training iterations per training epoch

Input : N : Batch size

Initialize Q-Network (GCN + noisy fully connected layer) with random weights θ

Initialize target Q-Network with $\theta^{'} = \theta$

Initialize replay buffer R

do in parallel

- **Simulation processes (gathering experiences by interacting with SUMO) :**
  - m=0
  - **while** *m<M* **do**
    - Generate and start episode (road network and trips)
    - Build the root computational graph G_{root} corresponding to the empty road network (§ 1.6.1)
    - Observe initial state s_1
    - **for** *t=1,T* **do**
      - Update G_{root} to obtain the current computational graph G_{s_t} (§ 1.6.1)
      - Select action $a_t = argmax_a Q^{\theta}(G_{s_t}, a)$
      - Execute action a_t, observe reward r_t and new state s_{t+1}
      - m = m + 1
    - Send all transitions from the episode $(G_{s_t}, a_t, r_t, G_{s_{t+1}})$ to the training process
  - Terminate simulation process
- **Training process (training the Q-Network) :**
  - **while** *simulation processes are not all terminated* **do**
    - **for** *each simulation process* **do**
      - Verify if new transitions are available and store them in R
    - **for** *j=1,J* **do**
      - Sample a random minibatch of N transitions from R
      - Set double Q-Learning target, $y_i = r_i + \gamma Q^{\theta}(G_{s_{t+1}}, argmax_{a'} Q^{\theta^{'}}(G_{s_{t+1}}, a'))$
      - Update Q-Network (θ) by minimizing the loss : $L = \frac{1}{N}\sum(y_i - Q^{\theta}(G_{s_t}, a_t))$
    - Update target Q-Network with $\theta^{'} = \theta$
    - Send updated θ to simulation processes
  - Save Q-Network (θ)

1.8 Conclusion

We introduce IG-RL, a reinforcement-learning trained method that leverages the flexible computational graphs of GCNs and their inductive capabilities (Hamilton et al., 2017) to obtain a single set of parameters applicable to the control of a variety of road networks. Experiments show that training using small-synthetic networks is enough to learn generalizable patterns, which in turn enables zero-shot transfer to new road networks, as well as new traffic distributions. Experiments also show that IG-RL outperforms MARL-IQL, as well as both dynamic heuristics and fixed-time baselines. Further, generalizability over architectures, which emerges from a generalist IG-RL (G-IG-RL), helps improve performance and generalization to new regimes of traffic. In addition, we showed that the flexibility of GCNs enables a flexible representation of the road network. Demand and structure can be represented and exploited in various ways, at their finest level of granularity, no matter the evolution of the number of entities nor their respective locations in the road network.

Future work IG-RL opens a path for the following future works: 1) The flexibility of GCNs could be key to the optimization of multi-modal transportation. We represent vehicles as nodes in IG-RL-V and we could experiment with alternative types of nodes such as pedestrian nodes (for pedestrian-aware ATSC as proposed in (Liang et al., 2020)), cyclist nodes, and many more, to ensure that traffic signal control accounts for all road-users. 2) In this work, the reward function used for every MDP focuses on local queues (Equation 1.2). In such a setting, coordination between MDPs is limited, and a shallow GCN is able to perform well since local information is sufficient. Evaluating whether the use of additional coordination mechanisms can further improve global performance could be an interesting avenue for future work. One way of addressing coordination, in the context of more global reward functions, would be to perform message passing on longer network distances by adding more layers to the GCN, or by adding recurrent mechanisms to it as done in Zhang et al. (2018). 3) The present work assumes data availability but some events (e.g. sensor failures) could generate various types of missing data. Evaluating IG-RL's robustness under several missing data mechanisms may constitute the object of future studies.

1.9 Acknowledgements

We would like to thank the three anonymous reviewers for their interesting and constructive comments that lead to an improved version of this article. We would also like to thank Maxime Gasse for helpful discussions of deep learning on graphs which helped formulate GCN architectures included in this work.

References

Abdulhai, B., Pringle, R., and Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135.

Aslani, M., Mesgari, M. S., and Wiering, M. (2017). 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*, 85:732–752.

Bakker, B., Whiteson, S., Kester, L., and Groen, F. C. (2010). Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information Systems*, pages 475–510. Springer.

Barth, M. and Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Bingham, E. (2001). Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241.

Busoniu, L., De Schutter, B., and Babuska, R. (2006). Decentralized reinforcement learning control of a robotic manipulator. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. (2020). Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017a). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

Fortunato, M., Blundell, C., and Vinyals, O. (2017b). Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Genders, W. and Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Hasselt, H. V. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org.

Houli, D., Zhiheng, L., and Yi, Z. (2010). Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. *EURASIP journal on advances in signal processing*, 2010(1):724035.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kuyer, L., Whiteson, S., Bakker, B., and Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer.

Liang, X., Du, X., Wang, G., and Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology*, 68(2):1243–1253.

Liang, X., Guler, S. I., and Gayah, V. V. (2020). Traffic signal control optimization in a connected vehicle environment considering pedestrians. *Transportation Research Record*, 2674(10):499–511.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Mannion, P., Duggan, J., and Howley, E. (2015). Parallel reinforcement learning for traffic signal control. *Procedia Computer Science*, 52:956–961.

Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pages 47–66. Springer.

Mikami, S. and Kakazu, Y. (1994). Genetic reinforcement learning for cooperative traffic signal control. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 223–228. IEEE.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017). Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2661–2670. JMLR. org.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2681–2690. JMLR. org.

Papoudakis, G., Christianos, F., Rahman, A., and Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

Prabuchandran, K., AN, H. K., and Bhatnagar, S. (2014). Multi-agent reinforcement learning for traffic signal control. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2529–2534. IEEE.

Prashanth, L. and Bhatnagar, S. (2010). Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. (2018). Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4691–4699.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Varaiya, P. (2013). The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pages 27–66. Springer.

Wang, X., Ke, L., Qiao, Z., and Chai, X. (2019a). Large-scale traffic signal control using a novel multi-agent reinforcement learning. *arXiv preprint arXiv:1908.03761*.

Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., and Xiong, H. (2019b). STMARL: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. *arXiv preprint arXiv:1908.10577*.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. (2019). Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. (2018). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Chapter 2

Model-Based Graph Reinforcement Learning for Inductive Traffic Signal Control

François-Xavier Devailly, Denis Larocque, Laurent Charlin

Abstract

Most reinforcement learning methods for adaptive-traffic-signal-control require training from scratch to be applied on any new intersection or after any modification to the road network, traffic distribution, or behavioral constraints experienced during training. Considering 1) the massive amount of experience required to train such methods, and 2) that experience must be gathered by interacting in an exploratory fashion with real road-network-users, such a lack of transferability limits experimentation and applicability. Recent approaches enable learning policies that generalize for unseen road-network topologies and traffic distributions, partially tackling this challenge. However, the literature remains divided between the learning of cyclic (the evolution of connectivity at an intersection must respect a cycle) and acyclic (less constrained) policies, and these transferable methods 1) are only compatible with cyclic constraints and 2) do not enable coordination. We introduce a new model-based method, MuJAM, which, on top of enabling explicit coordination at scale for the first time, pushes generalization further by allowing a generalization to the controllers' constraints. In a zero-shot transfer setting involving both road networks and traffic settings never experienced during training, and in a larger transfer experiment involving the control of 3,971 traffic signal controllers in Manhattan, we show that MuJAM, using both cyclic and acyclic constraints, outperforms domain-specific baselines as well as another transferable approach.

2.1 Introduction

Adaptive-traffic-signal-control (ATSC) aims at minimizing traffic congestion which gives rise to a plethora of environmentally and socially harmful outcomes (Schrank et al., 2012; Barth and Boriboonsomsin, 2008; Schrank et al., 2015). Reinforcement learning (RL), via trial-and-error, of ATSC policies, is popular to move beyond heuristic-based approaches which rely on both manual design and domain knowledge (Hunt et al., 1981; Lowrie, 1990). Multi-Agent RL (MARL) in particular, promises scalability of RL approaches by dividing control among intersections, but has mainly been restricted to training *specialist agents* which can only be applied on the exact road-network-topology, traffic distribution, and under the same constraints as experienced in training (see § 2.2). This lack of transferability, combined with notorious data inefficiency of these methods (i.e. the need to gather a massive amount of experience in order to train RL policies) (Wei et al., 2019b) poses a challenge to real world applicability as the social acceptability of prolonged exploration via interaction with real road-network-users remains questionable. Furthermore, most MARL-ATSC approaches sacrifice coordination ability between agents (see § 2.2).

2.1.1 Contribution

Model-based RL (MBRL), which explicitly models the dynamics of the environment tend to both outperform model-free RL (MFRL) in complex planning tasks and enable better sample efficiency (Silver et al., 2017; Schrittwieser et al., 2020). We introduce joint action modeling with MuZero (Schrittwieser et al., 2020), MuJAM, a new family of MBRL approaches for RL-ATSC which models dynamics of the ATSC environment at the lane level using a latent space.

Inherited Capabilities

Most neural-network architectures do not enable dealing with changing number of inputs and state dimensionality. For this reason, in a traffic scenario where various types of entities such as cars enter, move inside of, and leave the network, these methods do not enable full granularity exploitation. Instead of using loop sensor information and resorting to manual and arbitrary aggregations, flexibility in the computational graph offered by graph convolutional networks (GCNs) enables the exploitation of available data at its finest level of granularity (e.g., at the vehicle level) (Devailly et al., 2021). These approaches leverage inductive capabilities of GCNs to enable transfer, with no additional training, to new intersections and traffic distributions. This in turn translates into massive scalability as demonstrated in related MFRL approaches (Devailly et al., 2021).

On top of these inherited abilities, MuJAM outperforms its MFRL peers and offers, the following advantages:

Acyclic-Inductive-Graph RL

RL-ATSC remains divided between the learning of cyclic and acyclic policies (Wei et al., 2019b). This comes at a cost for urban-mobility-planning as experimentation with different constraints systematically requires retraining using a different approach. Even transferable approaches are limited in this regard. Only when the policy consists in switching to the next phase or remaining in the current phase (using cyclic constraints) does the action space remain binary and identical across intersections. For this reason, methods which aim to generalize over road-network architectures and traffic distributions are limited to the use of cyclic constraints (Devailly et al., 2021). MuJAM enables the use of transferable approaches, which were limited to cyclic constraints until now (see § 2.2), with acyclic constraints. Policies under the latter, looser type of constraints, typically perform better.

Constraint-Agnostic ATSC

MuJAM pushes generalization ability further and a unique instantiation of this model can generalize over constraints and be used, for instance, with both cyclic or acyclic policies on any new intersection. This additional level of generalization and transferability is intended to ease urban-mobility-planning by limiting the costs of experimentation and application.

Coordination *at scale*

It enables explicit joint action modeling *at scale* for the first time (see § 2.2)

Data efficiency

A) Improved Exploration: We adapt recent MBRL approaches to enable more efficient exploration which translates into better data efficiency as we gather more relevant signal with fewer interactions with the environment. B) Sample Efficiency: One of the main motivations behind MBRL is the ability to facilitate learning and reach better sample efficiency by leveraging a model of the dynamics of the environment. MuJAM does enable greater sample efficiency compared to related approaches.

To test our claims, we introduce and evaluate various instantiations of MuJAM (and perform ablation studies) on zero-shot transfer settings involving road networks and non-stationary traffic distributions both

never experienced during training, and show that our various instantiations outperform trained-transferable and domain-specific baselines

2.2 Related Work

2.2.1 Scalability

Even though some works propose to control a few intersections using a single agent (Casas, 2017; Prashanth and Bhatnagar, 2011), the explosion of both state and action space dimensionalities with the number of intersections limits the scalability of such approaches. MARL-ATSC aims at making RL scalable by decentralizing control (El-Tantawy and Abdulhai, 2010; Abdoos et al., 2011; El-Tantawy et al., 2013; Khamis and Gomaa, 2012; Steingrover et al., 2005; Salkham et al., 2008; Van der Pol and Oliehoek, 2016; Arel et al., 2010; Balaji et al., 2010; Aslani et al., 2018; Pham et al., 2013; Kuyer et al., 2008; Zheng et al., 2019; Xiong et al., 2019; Nishi et al., 2018; Wei et al., 2019a; Xu et al., 2013; Khamis et al., 2012; Aslani et al., 2017; Chu et al., 2019; Iša et al., 2006; Wiering et al., 2004b; Wiering, 2000; Zhang et al., 2019; Mannion et al., 2016; Wiering et al., 2004a). However, an increase of parameters leads to an explosion of costs (computational power, memory, and data requirements) with the number of intersections, which hinders scalability. Parameter sharing in MARL methods (Devailly et al., 2021; Zang et al., 2020; Zheng et al., 2019; Wei et al., 2019a; Chen et al., 2020; Wang et al., 2019) has enabled far greater scalability (up to a thousand controllers). Such methods usually rely on GCNs. Transfer learning is a promising way to reach even further scalability while drastically diminishing the training requirements. To limit the quantity of experience required to training under a new setting, Zang et al. (2020) uses meta-learning, and Devailly et al. (2021) achieves zero-shot transfer from small synthetic networks to massive real world networks using GCNs' inductive capabilities.

2.2.2 Coordination

Most MARL approaches for ATSC methods offer either no means of coordination between intersections, or only the ability to communicate (El-Tantawy and Abdulhai, 2010; Nishi et al., 2018; Zhang et al., 2019; Wei et al., 2019a; Devailly et al., 2021; Arel et al., 2010). Joint action modeling is challenging because of the non-tractability of exploding action spaces. Some approaches do tackle joint action modeling on small road networks (El-Tantawy et al., 2013; Kuyer et al., 2008; Van der Pol and Oliehoek, 2016; Xu et al., 2013).

2.2.3 Data Efficiency & Model-Based RL

One of the main challenges to real-world applicability of RL-ATSC is the notorious data inneficiency of RL algorithms. This data inefficiency can be decomposed into 2 sub-challenges 1) Exploration: Guiding the collection of experiences to gather insightful observations/transitions in as few interactions with the environment as possible. 2) Sample Efficiency: Learning from the collected observations/transitions as efficiently as possible. Despite the fact that one of the main appeals of model-based RL (MBRL) is precisely to improve the latter, only a few works have explored such approaches for ATSC (Salkham and Cahill, 2010; Khamis et al., 2012; Wiering, 2000; Wiering et al., 2004a; Steingrover et al., 2005; Khamis and Gomaa, 2012; Wiering et al., 2004b; Kuyer et al., 2008), and this research avenue has not brought much attention since the early 2010s.

2.3 Background

2.3.1 Model-Based Reinforcement Learning

Sequential decision making can be framed as a Markov decision process (MDP). An MDP is defined by: S a set of states, A, a set of actions, $\mathscr{T}$, a transition function defining the dynamics of the system (the probabilities of transitioning from a state to another state given an action), and R, a reward function defining utility (i.e. performance w.r.t. the underlying task). Solving an MDP means finding a policy, π, which maximizes the expected accumulation of future value (discounted by a temporal factor γ): $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where r_t represents the reward experienced at time step t. RL is a popular framework to solve MDPs via trial-and-error. When the learning of value and transition functions are confounded, RL is said to be model-free. Alternatively, MBRL approaches explicitly leverage transition dynamics to enable planning (i.e. simulating trajectories to select promising ones up-front). In MBRL, models of the dynamics which are perfectly known in advance can be provided to the RL method. This, combined with tree search (TS)[1] planning, has enabled AlphaZero to reach superhuman performance in complex planning tasks such as Chess, Shogi, and Go (Silver et al., 2016, 2017). However, in most real-world environments, dynamics are complex and unknown and must be learned. Learning transition functions (i.e. $s_{t+1} = \mathscr{T}(s_t, a_t)$) in high dimensional state spaces is expensive and challenging as the performance of MBRL approaches significantly lagged that of MFRL approaches in corresponding tasks (Schrittwieser et al., 2020). Recent approaches aim to include implicit feature selection and dimensionality reduction by planning in a value-equivalent-latent-space, instead of planning in the original state space of

[1] Tree search is an heuristic planning method simulating many roll-outs by randomly sampling the search space to identify and analyze the most promising actions.

the MDP (Oh et al., 2017a; Schrittwieser et al., 2020). In other words, a dynamic model learns, given a starting state, 1) to map the original state space to a latent (vectorized) space and 2) to simulate trajectories in this latent space under the only constraint that the prediction of value-related quantities based on the latent space must match those observed in the real state-space. Such approaches have 2 main advantages: 1) The dynamics model can ignore all dynamics unrelated to the task at hand (dynamics which do not influence value-related metrics), 2) computations are projected into a latent vectorized space instead of predicting (usually reconstructing) high dimensional states such as images. With these advances, MuZero (Schrittwieser et al., 2020) matches the performance of MBRL algorithms in their favored domains (complex planning tasks) and outperforms MFRL algorithms in their favored domains (states involving high dimensional state spaces).

2.3.2 Heterogeneous Graph Convolutional Networks (GCNs)

Graph convolutional networks consist in stacking convolutional layers to recursively aggregate transformations of neighborhood-information in a graph (Kipf and Welling, 2016). As a message passing framework, GCNs enable the learning and exploitation of nonlinear patterns involving both structural and semantic (nodes and edges features) signal. Their extensions to heterogeneous graphs (with multiple types of nodes and edges), Heterogeneous GCNs or HGCNs (Schlichtkrull et al., 2018), work by applying the following message-passing equation on every node, at every layer:

$$n_i^{(l)} = f\left(\sum_{e \varepsilon E, j \varepsilon N_e(i)} W_{l_e} \cdot n_j^{(l-1)}\right) \tag{2.1}$$

where n_i^l is the embedding at layer l of node i, f is a non-linear differentiable function, $N_e(i)$ is the immediate neighborhood of node i in the graph of relation type e, E is the set of relation types, and W_{l_e} is the l^{th} layer's weight matrix for message propagation corresponding to a relation of type e. Parameters are typically learnt by backpropagation of an error obtained using nodes embeddings. For simplicity, we will refer to HGCNs as GCNs for the rest of the paper.

2.3.3 ATSC Decision Process

Decision Process

The decision process consists in the simultaneous and coordinated control of all TSCs in a given road network.

State

The state of a road network comprises 1) *connectivity*: the status of all existing connections between lanes at intersections, and the status of their respective controllers' constraints, and 2) *demand*: vehicles positions and speeds. State information is encoded using vector representations which qualify nodes and edges of the GCN. (see 2.4.3).

Action

The joint action at a given time step, is a combination of local actions, each consisting in the selection of a legal[2] phase among all legal phases. Once a phase is selected to be the next phase, one of two things happen: If the selected phase is the current one, then no switch is performed. On the other hand, if the selected phase is different from the current phase, the chosen phase becomes the target phase and the switch to this target phase begins. If an intermediary phase involving yellow lights is required (e.g. if some lights are green in the current phase and will be red in the destination phase), then this intermediary phase is enforced for a fixed duration before activating the destination phase.

Transition constraints

A TSC can only select a phase different from the current one after a minimum duration has passed. All intermediary phases involving yellow lights last for a fixed duration. If a TSC is under cyclic constraints, then it can only select the current phase or the next phase in the cycle. If a TSC is under acyclic constraints, then it can select any legal phase.

Reward

The reward is the negative sum of local queues lengths.[3]

Step

The duration of each step is one second.

Episode

Episodes can either end after all trips are completed, or after a given amount of time.

[2] A legal phase is a phase which can be selected by a TSC at a given time step according to this TSC's constraints

[3] A vehicle is counted in a queue if it is stopped at a maximum of 50 meters of the intersection

2.4 MuJAM

Our new family of approaches for RL-ATSC builds upon inductive-graph-reinforcement-learning (with GCNs) and models dynamics of the ATSC environment using a latent space to facilitate the combination of MBRL and GCNs. Modeling is decentralized at the level of lanes and performed jointly for all lanes in the road-network to enable full coordination (joint action modeling).

2.4.1 Dynamics Model

We distinguish two parts of environmental dynamics in the ATSC MDP.

1. *Connectivity dynamics* are simple and known in advance. If a given action is taken, connectivity (e.g. which connections between lanes at a given intersections will be opened) is influenced in a fully deterministic and predictable way.

2. *Traffic dynamics* (i.e. the way vehicles behave under a given state of traffic and connectivity) are complex and unknown.

Traffic dynamics have to be learned. MBRL with a learned dynamics model usually involves learning to predict the next state conditioned on the current state and a given action. A challenge is that in the current setting, states are complex graphs with varying number of nodes, connectivity, and features. All of these elements change from one time step to another (as vehicles move along the road network for instance). Learning to predict such a complex object in a learnable/differentiable way is non trivial. It becomes even more challenging if planning has to be performed in real-time which is required for real-world ATSC. For this reason, we use MuZero (Schrittwieser et al., 2020), which enables planning in a latent vectorized space by only using value-related scalars as targets to learn the model.

Connectivity dynamics, on the other hand, are directly provided to the model as features in the GCN and are manually updated during model-based-planning. Note that the model is informed of the type of constraints used (via features in the GCN).

Our dynamics model is learned at the lane level. For planning, we start by obtaining an initial state representation for all lanes (a vector obtained using a GCN which aggregates information about structure, connectivity, traffic, etc.). Then, to model a transition given an action, we 1) manually update features related to connectivity dynamics in the GCN, 2) use the GCN to update representations (vectors) for all lanes, 3) predict rewards r and long-term values estimates v (i.e. predicted future cumulative reward) for all lanes based on their respective

vectorized representations. The final value-related metrics (r,v) associated to a given transition for the entire road network is then simply obtained by summing all lane level estimates.

2.4.2 Coordinated priors for parsimonious planning

In a coordinated setting, the action space rapidly becomes intractable with the number of intersections. The total number of actions at a given time step is $\prod_{i=1}^{n} |A_i|$ where n is the number of intersections and $|A_i|$ is the size of the action space for intersection i according to its current state. Even for a small road network of 20 intersections, modeling all transitions for a single time step (i.e. performing an exhaustive search of the first layer of the TS) could require querying our GCN-based-dynamics-model more than a million times. Instead, we sample the action space parsimoniously. To do so, we adapt the TS sampling strategy proposed in Sampled MuZero (Hubert et al., 2021) to a coordinated setting. Namely, we train a prior multilayer perceptron (MLP) ϕ to predict, at the intersection level, the local action which will be part of the best coordinated set (according to the results of the TS).

Local priors

Logits are obtained using ϕ on phase nodes representations, and used to define multinomial distributions with parameters: $p_i = exp(l_i)/\sum_{j \varepsilon A} exp(l_j)$, where p_i is the probability of sampling action i, A is the set of actions (i.e. legal phases) for the TSC at the current state, and l_i is the logit corresponding to action i. Each of these distributions can be used to sample local actions (i.e. actions for each TSC),

Best coordinated set

A TS is used to identify the best coordinated set of actions among all candidate sets. To sample a candidate set, a local action is sampled for all intersections using the local multinomial distributions. The best candidate set (joint action a) is the one maximizing: $\sum_{i \varepsilon L} r_i + \gamma . v_i$, where L is the set of all Lanes, r_i is the predicted reward for the transition $\mathscr{T}(s,a) \rightarrow s'$, and v_i is the estimated (via TS) long term value for state s' (i.e. the state reached after taking action a).

Improving priors

To train ϕ, local actions are extracted from the best coordinated set and become training targets. As training progresses, local priors increasingly favor the sampling of local actions belonging to the best joint action set, improving TS estimates, which in turn improve priors in a positive feedback loop.

Predicting local actions belonging to the best coordinated sets based on neighborhing information (i.e. using representations obtained from the GCN) makes our model-based approach scalable as it can then be used with an arbitrary planning budget. In the extreme case of null budget (i.e. no TS), actions can still be selected by acting greedily with respect to these learned "coordinated priors". In other words, we can simply sample a candidate set using the trained local priors without running a TS.

2.4.3 Architecture

Road-network objects are modeled as nodes in the GCN. The 4 types of nodes are: vehicle, lane, connection[4] and phase.[5] The main difference between the GCN architectures of IG-RL (Devailly et al., 2021) and MuJAM is that the latter uses one node per phase whereas the former uses a single node to represent the controller. The types of edges[6] are:

- Edge linking the node of a vehicle to the node of a lane it is currently on.
- Edge linking the node of a lane to the node of another inbound lane
- Edge linking the node of a lane to the node of another outbound lane
- Edge linking the node of a connection to the node of its inbound lane
- Edge linking the node of a connection to the node of its outbound lane
- Edge linking the node of a connection node to the node of a phase

Graph Features

Node and edge features are summarized in Tables 2.1, 2.2. *Current speed* represents the normalized current speed of a vehicle (between 0 and 1: speed in km/h divided by the maximum allowed speed of 50 km/h), *Position on lane* represents the relative location of a vehicle on a given lane (between 0 and 1, expressed as a proportion of the lane). *Length* is the length of a lane in meters, *Is open* indicates if a given connection is currently opened (i.e. green), *Is yellow* indicates if a given connection is currently yellow. *Has priority* represents, when a connection is opened, if it has priority (i.e. if vehicles following a connection have priority over vehicles following alternative connections at an intersection). *Next switch open*[7] indicates if the connection

[4] A connection exists between two lanes if one can lead to the other given a state of connectivity.
[5] A phase defines a state (green/yellow/red) for all connections at a given intersection.
[6] Every type of edge uses its own set of parameters.
[7] *Next switch open, Number of switches to open & Next opening has priority are set to -1 when using acyclic constraints*

Table 2.1: Node features

Node type	Features
Vehicle (V)	*Current speed*, *Position on lane*
Lane (L)	*Length*
Connection* (C)	*Constraints type*, *Time since last switch*, *Is open*, *Is yellow*, *Has priority*, *Next switch open*, *Number of switches to open*, *Next opening has priority*
Phase (P)	*None*

Table 2.2: Edge features

Edge type	Features
V to L	*None*
L to L	*Is inbound* + all C features
L to C	*Is inbound* + all C features
C to P	*Opens connection*, *Is legal*

will be opened on the next phase of the cycle. *Number of switches to open*[7] indicates, in the case of cyclic constraints, in how many switches the connection will be opened (i.e. green). *Next opening has priority*[7] indicates if the next opening of the connection has priority. *Time since last switch* is the number of seconds since the last change in connectivity (switch) at the corresponding intersection. *Is inbound* represents whether an edge (in the GCN) follows the direction of traffic (i.e. propagates information in the direction of traffic) or follows the opposite direction of traffic (i.e. propagates information in the opposite direction of traffic). *Opens connection* indicates, for a connection-phase pair, if the connection is opened (i.e. green) under the corresponding phase. *Is legal* indicates if a given phase can legally be selected at the current time step.

Computational Graph & Propagation rules

Initial Representation: The *initial representation* for the road network aims at obtaining, for all lanes, a representation of its current surroundings (connectivity and traffic). To obtain this representation, messages are first propagated once along V-to-L edges to obtain a representation of demand on all lanes. Then, in order to contextualize this representation, messages are propagated K (a hyperparameter) times[8] along L-to-L edges.

Dynamics: Based on a representation s_t of the road network, and given a joint action a_t, to obtain the

[8]Every GCN layer uses its own set of parameters.

representation of the next time step (as part of a simulated trajectory), we first manually update all C features (on both nodes and edges) (see Tab. 2.1, 2.2). This informs the model of the evolution of connectivity dynamics (i.e. the immediate impact of a_t). Then, we propagate messages alongside L-to-L edges K' (a hyperparameter) times[8]. This last step aims at modeling traffic dynamics between lanes based on the previous representation s_t and updated connectivity features.

Value-related-metrics computation: For all lanes, reward (r) and value (v) estimates for a given time step are obtained by feeding lane's representation (i.e. L nodes embeddings) to 2 respective MLPs.

Prior computation: Given a representation of the road network (i.e. contextualized representations for all lanes), messages are propagated once along L-to-C edges, and then once along C-to-P edges. For all intersections, the obtained representations of legal phases nodes (i.e. nodes embeddings of phases which are legally selectable at the corresponding time step) are independently fed to a MLP (ϕ) and then normalized (per intersection) using a softmax to obtain probabilities (i.e. the predicted local action distribution, used as prior during planning).

Fig. 2.1 illustrates and describres a slice of the computational graph (i.e. how MuJAM models dynamics and predicts value-related quantities and coordinated priors).

2.4.4 Planning with a tree search

At any given time step, in order to select a joint action, TS can be performed according to a predefined search budget β (a hyperparameter). If $\beta = 0$, we sample an action per intersection using the local priors (distributions) obtained from the representation of the current state (i.e. original node). The combination of these local actions constitutes the joint action. Alternatively if $\beta > 0$, β trajectories will be simulated as part of the TS (trajectories must not exceed the maximum search depth δ).

Growing the tree

Inspired by the approach proposed in Hubert et al. (2021) we perform guided sampling of the action space. In our coordinated setting, we use local priors, trained on coordinated targets and combine locally sampled actions to form joint actions. As the number of possible actions 1) varies dramatically across time steps (because of constraints) and across road networks 2) explodes exponentially with the number of intersections, we propose the following progressive widening strategy for the TS: If $samples < ((actions/C1)^{C2}).(visits^{C3})$ (where $samples$ is the number of sampled actions at the current node, $actions$ is the number of selectable actions at the current node, $visits$ is the number of times the current node has been visited in the current TS, and $C1, C2, C3$

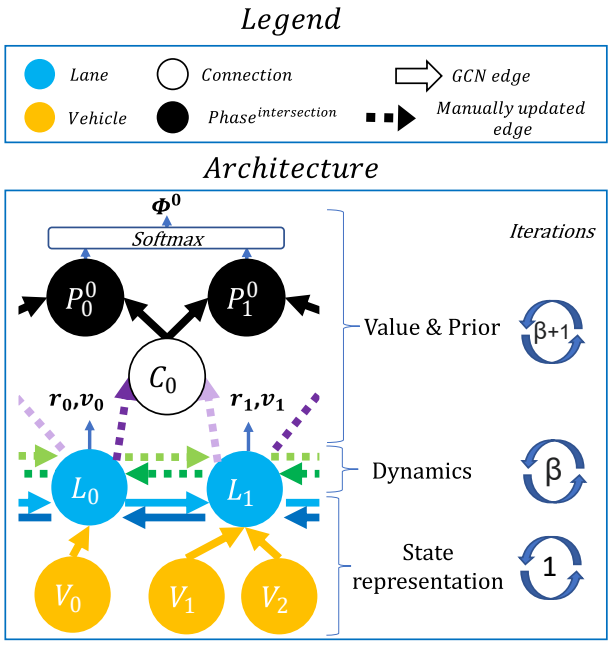

Figure 2.1: Illustration of MuJAM's model-based computational graph

For simplicity, we illustrate the computational graph under a cyclic policy. First, we obtain the current state representation of all lanes by aggregating vehicle level information and propagating representations along connected lanes. Then, we unroll our dynamics model β times by first manually updating connectivity-related features (i.e. *connectivity dynamics*) in the GCN, and then propagating updated representations (i.e. modeling *traffic dynamics*) along the updated edges. For both the initial representation and simulated steps: 1) value related metrics (r_i,v_i) are obtained using MLPs on lanes' representations, and these values are then summed to obtain network-level-estimates 2) local priors are obtained using the second part of the GCN which first computes connections representations from their respective inbound and outbound lanes' representations, and then computes phases representations from the representations of connections they control. For all intersections, final representations of all legally selectable phases (current phase and next phase in the particular cyclic setting) are then fed to a final MLP and all outputs from a given intersection are normalized using a softmax to get the final priors Φ (i.e. probabilities of selecting each phase).

are hyperparameters), then a new joint action is sampled using all local priors computed from the current node representation. This new progressive widening strategy enables controlling the rate at which alternative trajectories are evaluated at a given node based on the size of the action space *at that node*. For instance, this enables ensuring that the width of the search increases (i.e. more actions are sampled) with the action space's size, *ceteris paribus*.

Exploring the tree

During the simulation of a trajectory, at a given node, an action is selected (i.e. a child node is selected) using the following probabilistic upper confidence tree (PUCT) bound (Silver et al., 2016):
$argmax_a \left[Q(s,a) + c(s).\Phi(s,a)\frac{\sqrt{P_{visits}}}{1+Ch_{visits}} \right]$, with $c(s) = log((P_{visits} + C_{base} + 1)/C_{base}) + C_{init}$ where P_{visits} is the number of times the parent/current node was visited, Ch_{visits} is the number of times the child node (i.e.

corresponding action) was visited, Φ(s, a) is the joint prior (i.e. the probability of sampling the given joint action obtained by the product of local probabilities), and C_{base}, C_{init} are hyperparameters. Additionally, as done in Schrittwieser et al. (2020), we add Dirichlet noise to local priors of the source node to favor exploration at the root of the TS.

Backpropagating values

When simulating a new transition (i.e. exploring a new node in the TS), the estimated value of the trajectory is backpropagated along the TS. However, unlike in Monte Carlo TS (MCTS) where the estimated value of starting from a given node is based on the average of the estimated values of trajectories simulated from that node, in our experiments, we define it as the value of the best simulated trajectory, as it improved learning and convergence speeds significantly in our experiments.

2.4.5 Training

MuJAM is trained end-to-end by backpropagation of prediction errors' (on r,v, and Φ) using observed sequences of transitions and corresponding TS results.

Improving data efficiency

Targets for v and Φ are built using the result of TS when interacting with the environment. However, as proposed in Schrittwieser et al. (2020), we continuously reanalyze episodes in the replay buffer to refresh TS estimates using updated parameters. This is done to improve sample efficiency and training speed. Furthermore, to enforce exploration in the environment during training, instead of using independent noise at each time step as done in MuZero, we use noisy networks (Fortunato et al., 2017) for the prior. More specifically, the weights composing the layers of the MLP used for prior computation are defined as parameterized Gaussian distributions. These weights are periodically re-sampled from these distributions. This adaptation of MBRL enables coherent exploration as the behavior of consecutive time steps is nonlinearly correlated. In consistence with Devailly et al. (2021), coherent exploration improved training speed and asymptotic performance in our experiments.

2.4.6 Algorithms (pseudocodes)

This section provides pseudocodes used in MuJAM for both training and inference.

Initial Representation

This algorithm transforms the graph representation of the current state of a given road networks and returns an updated (contextualized) graph in which all lane nodes' (L) embeddings have been updated by aggregating surrounding state and structure information (i.e. surrounding lanes and vehicles) using the corresponding GCN layers. The embedding of a given lane node correspond to its latent state representation.

Algorithme 2 : Initial Representation

Input : K : Number of representation (contextualization) GCN layers

Input : G : Graph with nodes and edges corresponding to the current state of the road network(see § III-C)

Get initial lanes' representations:

Represent local traffic (i.e. vehicles on a given lane):

Update all vehicle nodes (V) initial representations with current timestep information (i.e. current position speed).

GCN-layer propagation(see Eq.1):

A) Propagate messages from V-to-L nodes using the 1st *representation* layer of the GCN.

B) Aggregate (sum) all messages received by a given lane node (L) to obtain it's updated (**isolated**) representation.

Contextualize representations using surrounding lanes' representations:

GCN-layer propagation(see Eq.1):

for $k \leftarrow 1$ *to* K **do**

A) Propagate messages from L-to-L nodes using subsequent *representation* (contextualization) layers of the GCN.

B) Aggregate (sum) all messages received by a given lane node (L) to obtain it's updated (**contextualized**) representation.

Return z (the updated graph with contextualized latent representations of lane-nodes)

Simulation (of a transition)

This algorithm takes 1) a contextualized graph representation of a state, in which lanes nodes' embeddings represent their respective current latent states, and 2) a joint set of actions (i.e. one action per intersection). It first updates all connectivity features of the graph based on the joint action, and then updates lane nodes' embeddings with the corresponding GCN layers.

Algorithme 3 : Simulation (of a transition)

Input : K' : Number of dynamics GCN layers

Input : z_n : Graph with contextualized-latent lanes' representations (see § III-C)

Input : A : Joint Action (i.e. a set of local actions for all intersections in the road network)

Simulate transition by updating lane-nodes' (L) representations:

- **Manually update connectivity features (see § III-A):**
  - According to A (i.e. potential changes in connectivity):
  - A) Update all connection nodes' (C) features B) Update all L-to-L and L-to-C edges' features
- **Simulate transition according to updated connectivity:**
  - **for** $k \leftarrow 1$ *to* K' **do**
    - A) Propagate messages from L-to-L nodes using *dynamics* layers of the GCN.
    - B) Aggregate (sum) all messages received by a given lane node (L) to obtain it's updated (**simulated**) representation.

Return $z_{n'}$ (the graph with updated latent lanes representations).

Inference$_{value}$(r,v)

This algorithm takes a contextualized graph representation of a state, in which lanes nodes' embeddings represent their respective current latent states, and predicts a reward and value per corresponding latent state (i.e. per lane).

Algorithme 4 : Inference$_{value}$(r,v)

Input : L : The set of lanes in the road network

Input : z : Graph with contextualized-latent lanes' representations (see § III-C)

Predict, per lane, transition-rewards (r) and states values (v):

- **for** $l \in L$ **do**
  - $r_l, v_l = w(l_{rep}), q(l_{rep})$

Return r, v

Inference$_{prior}(\Phi)$

This algorithm takes a contextualized graph representation of a state, in which lanes nodes' embeddings represent their respective current latent states. It first aggregates lanes representation to obtain representations of connection nodes (C) and phase nodes (P), and then outputs logits per selectable phase. All logits corresponding to the same intersection (i.e. selectable phases/actions) are then normalized using the softmax function and the corresponding outputs define, for the multinomial distribution, probabilities of selecting corresponding actions.

Algorithme 5 : Inference$_{prior}(\Phi)$

Input : z : Graph with nodes and edges with contextualized-latent lanes' representations(see § III-C)

Input : U : The set of all intersections in the road network

Input : L : The set of lanes in the road network

Input : P : The set of all locally selectable phases in the road network

Predict, per intersection, the local action a **corresponding to the best joint action** A**:**

A) Propagate messages from L-to-C nodes using *connection* layers of the GCN.

B) Aggregate (sum) all messages received by a given connection node (C) to obtain its representation.

Get phases (i.e. local actions) representations:

A) Propagate messages from C-to-P nodes using *phase|action* layers of the GCN.

B) Aggregate (sum) all messages received by a given phase node (P) to obtain its representation.

for $p\ \varepsilon P$ **do**

$logit_i = \phi(p_{rep})$

for $u\ \varepsilon U$ **do**

$p_i = exp(logit_i)/\sum_{j\varepsilon A_u} exp(logit_j)$*(see § III-B)*

$\Phi_u = Multinomial(p_i, \forall i \varepsilon A_u)$

Return Φ

Tree Seach

This algorithm defines how to first obtain, or update, for a given state s in the replay buffer, estimates of v, the maximum future value starting from s, and A^*, the target local actions (for all intersections) corresponding to the best joint (i.e. coordinated) action.

Algorithme 6 : Tree Search

Input : β : Number of transitions to be simulated in a given tree search (planning budget)

Input : $C1, C2, C3$: Progressive widening hyperparameters

Input : G_{s_t} : Graph with nodes and edges corresponding to the current state of the road network(see § III-C)

Gather initial representation z_{init} from G_{s_t} (see Algo.2)

$k = 0$

while $k<\beta$ **do**

  Start new rollout at the root node:

    $n = n_{init}$

    Initialize *Path* (empty list)

    Progressive widening (see§ III-B):

    while $samples_n \geq ((actions_n/C1)^{C2}).(visits_n^{C3})$ **do**

      Extend the rollout with an existing node:

      Select child node n' **(i.e. joint action A)**:

      $_{A \varepsilon A_n} \left[Q(z_n, A) + c(z_n).\phi(z_n, A) \frac{\sqrt{P_{visits_n}}}{1 + Ch_{visits_n}} \right]$

      Prepend n' to *Path*

      $n = n'$

    Sample a new node (transition):

    Sample A from Φ_n and create a new node n'

    Simulate transition using (z_n, A) to obtain $z_{n'}$ (see Algo.3)

    Infer $r_{n'}$ and $v_{n'}$ from $z_{n'}$ (see Algo.4)

    Infer Φ'_n from $z_{n'}$ (see Algo.5)

  Backpropagate value along the completed rollout:

  for *n in* Path **do**

    $value_n = max_{A_n} \sum_{l \varepsilon L} r_{A_n} + \gamma v_{A_n}$

  $k = k + 1$

$A^*_{init} = argmax_{A_{init}} \sum_{l \varepsilon L} r_{A_{init}} + \gamma v_{A_{init}}$

Return $z_{init}, value_{init}, A^*_{init}$

Training

This algorithm defines asynchronous processes used in training. Experience is gathered by interacting with the environment in an exploratory fashion. Target values and target actions (v and A^*) are obtained (and updated) using tree searches, and estimators are trained using gradient based optimization.

Algorithme 7 : Training

Input : T : Episode duration

Input : N : Batch size

Input : H : Training depth

Input : $Stop$: Stopping criterion (e.g. no improvement over a number of training or evaluation steps)

Initialize model (GCN & MLPs) with random weights θ

Initialize replay buffer D

do in parallel

- **Centralized process (enables asynchronous access to D from the other processes):**
- **Self-play processes (gathering experiences by interacting with the environment) :**
  - **while** *not Stop* **do**
    - Generate and start episode (road network and trips)
    - **for** t=0,T **do**
      - Build the graph corresponding to s_t, G_{s_t}
      - Get $z_{init}, value_t, A_t^*$ from running the tree search on G_{s_t} (see Algo.6)
      - Get Φ_{noisy} using parameter noise in $\phi(z_{init})$
      - Sample an exploratory joint action A_t from Φ_{noisy}
      - Execute A_t and observe (per-lane) rewards r_t and new state s_{t+1}
    - Add all transitions $(G_{s_t}, A_t, A_t^*, r_t, G_{s_{t+1}}, value_t)$ to D
- **Simulation processes (continuous update of training targets with tree searches):**
  - **while** *not Stop* **do**
    - Sample a random episode e from D
    - Sample a random step t from e
    - Update $value_t, A_t^*$ in D by re-running the Tree Search (see Algo.6) with fresh parameters
- **Training process (training the simulator and actor) :**
  - **while** *not Stop* **do**
    - Sample a random minibatch of N sequences of transitions of length H from D
    - Gather initial representations z_t from G_{s_t} (see Algo.2)
    - loss = 0
    - **for** $t \rightarrow t+H$ **do**
      - $r_t^{pred}, v_t^{pred} = Infer_{value}(z_t)$ (see Algo.4)
      - $\Phi_t = Infer_{prior}(z_t)$ (see Algo.5)
      - $z_{t+1} = Simulate(z_t, A_t)$ (see Algo.3)
      - $loss+ = MSE\_Loss(r_t^{pred}, r_t) + MSE\_Loss(v_t^{pred}, v_t) + Cross\_Entopy\_Loss(\Phi_t, A_t^*)$
    - Backpropagate the loss to compute gradients and perform gradient descent based optimization.

Save model (GCN & MLPs weights : θ)

2.5 Experiments

In our experiments, we use a zero-shot transfer setting proposed by Devailly et al. (2021) and inspired by other transfer settings in RL (Oh et al., 2017b) (Higgins et al., 2017).

We compare the performence of instantiations of MuJAM and several baselines on both small synthetic road networks and the road network of Manhattan. The following experiments demonstrate that:

- MuJAM enables both higher asymptotic performance and higher data efficiency
- MuJAM enables both the use of acyclic constraints (which typically enable better performance than cyclic constraints) and the learning of policies that are agnostic to these constraints (i.e. perform well in both settings).
- Joint action modeling contributes to the performance of MuJAM.
- MuJAM distributes delay between trips in a more equitable fashion.

2.5.1 General Setup

Network generation

Road networks are randomly generated (i.e. random connectivity and structure) using SUMO (Lopez et al., 2018). The generated networks typically include between 2 and 6 intersections. The length of edges is between 100 and 200 meters, and between 1 and 4 lanes compose a given road.

Traffic generation

Traffic generation (i.e. the generation of trips) follows asymetric trajectory distributions (probabilities for a trip to start and finish on any given lane in the road network) which are resampled every 2 minutes to ensure non-stationarity.

Evaluation

We use the same networks and trips to evaluate all methods. Performance is evaluated using the instantaneous delay, defined as: $d_t = \sum_{v \epsilon V}(s_v^* - s_v)/s_v^*$, where $s_v^* = min(s_{v^*}, s_l)$, V is the set, at time step t, of all vehicles in the road network, s_{v^*} is the maximum speed of a vehicle, s_l is the maximum speed a vehicle can legally reach considering the lane it currently is on, s_v is the vehicle speed at time step t.

Table 2.3: Hyperparameters

learning rate	1e-3	C_{init}	1.25
ω	1e4	K	3
batch size	16	K'	3
$C1$	5	γ	0.997
$C2$	0.5	δ	1
$C3$	0.5	β	50
C_{base}	19652	*optimizer*	Adam

Robustness

We run all experiments 5 times to ensure the robustness of our conclusions as random seeds influence network and traffic generation, initial parameters' values for learnable approaches, and exploratory noise during training.

2.5.2 Baselines

Fixed Time

The Fixed Time baseline follows SUMO default cyclic programs which is defined based on the structure of a given intersection

Max-moving-car heuristic (Greedy)

This dynamic baseline aims to enable the movement of a maximum number of vehicles at any given time in the spirit of the popular baseline MaxPressure (Varaiya, 2013) under a cyclic setting. To do so, it switches to a next phase as soon as there are more immobilized vehicles than moving vehicles in lanes which are inbound to the intersection. Otherwise, it prolongs the current phase.

IG-RL

This is the only learnt baseline for which zero-shot transfer to new road architectures and traffic distributions is achievable. It consists in a GCN which gathers demand at the vehicle level and is trained via RL using noisy parameters for exploration (Devailly et al., 2021).

MuJAM

For MuJAM, 50% of intersections used during training are under cyclic constraints and 50% are under acyclic constraints to enforce generalization to these constraints. For MuJAM-C and MuJAM-A, all intersections used

in both training and test are under the corresponding constraints (cyclic and acyclic, respectively). The suffix *NNL* (no noisy layers) indicates that exploration in the environment was performed as done in the original MuZero formulation (i.e. exploratory behavior is independent between 2 consecutive time steps), instead of using noisy layers (which ensure coherence in exploratory behavior). The suffix *NR* (no reanalyze) indicates that we use the default (more mainstream) instantiation of MuZero (Schrittwieser et al., 2020) which does not periodically reanalyzes transitions to refresh training targets with updated parameters. Finally, for MuIM (independent modeling), a TS is performed independently for each intersections instead of performing a joint TS for the entire road-network. (i.e. joint-action-modeling is removed).

Training and hyperparameters

Training episodes last 10 minutes (simulation time). During training, performance is continuously evaluated on a separate set of 10 road networks. If a method does not reach a higher average reward on this set for ω steps, early stopping is enforced (i.e. training ends). For IG-RL, hyperparameters are the same as those used in the original paper (Devailly et al., 2021). We now refer to all methods based on MuZero as *Mu methods*. For *Mu methods*, hyperparameters are listed in Tables 2.3. and are either chosen according to other works or based on computational constraints.

2.5.3 Experiment 1: Inductive Learning & Zero-shot transfer

Traffic is generated for the first 10 minutes (simulation time). On average, a vehicle is introduced every 4 seconds in a given road-network. Episodes are terminated as soon as all trips have been completed. As all methods are evaluated using the same set of trips, they are paired together to study differences in delays. We report paired t-tests and distributions of differences in delays, when compared to the best performing method (MuJAM with acyclic constraints), in Fig. 2.2 and Fig. 2.4, respectively.

Model-based inductive learning

Mu methods enable lower trips delays (lower means, medians and quartiles) compared to all baselines, as shown in Fig. 2.2. These methods also lead to higher asymptotic performance (i.e. lower average total delay after training is completed) (see Fig. 2.3). With the only notable exception of MuJAM-NR-C, which is discussed later in this subsection, *Mu methods* offer higher data efficiency as these methods start outperforming all baselines early in training. This demonstrates that recent advances and successes in model-based RL on games (e.g. Chess, Go, Atari games) can also be leveraged in the challenge of ATSC.

Constraint Agnosticism

First, we observe that methods under acyclic constraints outperform methods under cyclic constraints as 1) the distribution of trips delays (means, medians and quartiles) is the lowest for MuJAM with acyclic constraints (see Fig.2.2) and 2) both data efficiency and asymptotic performance are better for MuJAM-A than MuJAM-C (see Fig.2.3). Although this result is not surprising as the acyclic setting is less constraining than the cyclic setting, it it the first time that an acyclic approach is transferable (with zero-shot transfer) to new road-network and traffic distributions. Furthermore, the evaluation of our hybrid approach, MuJAM, under both cyclic and acyclic constraints yields similar asymptotic performance compared to corresponding specialists formulations (MuJAM-A and MuJAM-C). As show in Fig.2.3, it is slightly higher for acyclic constraints, and slightly lower for cyclic constraints. This demonstrates the ability of MuJAM to generalize to constraints and the viability of training a single agnostic method to tackle both a variety of road-network architectures, traffic distributions, and behavioral constraints simultaneously.

Coordination

The ablation of coordination ability (i.e. MuIM-C) leads to a lower asymptotic performance when compared to MuJAM-C (see Fig. 2.3). In fact this ablation makes MuIM-C one of the worst performing *Mu methods*. This demonstrates that coordination ability (jointly maximizing performance for the entire road-network) can improve performance compared to locally-greedy approaches (maximizing reward per intersection).

Data Efficiency

Removing coherence in exploratory behavior (i.e. MuJAM-NNL-C) leads to both a slower increase in performance during training and a lower asymptotic performance when compared to MuJAM-C (see Fig. 2.3). This ablation makes MuJAM-NNL-C the worst performing *Mu method*. This demonstrates that coherence in exploration is key to both data efficiency and performance in RL-ATSC, confirming what was reported in Devailly et al. (2021). Removing the ability to refresh training targets with updated parameters (i.e. MuJAM-NR-C) negatively impacts asymptotic performance (see Fig. 2.3), although this method still remains the second best performing *Mu method* under cyclic constraints. However, this ablation impairs data efficiency the most as performance even lags behind IG-RL during the first 10k training iterations.

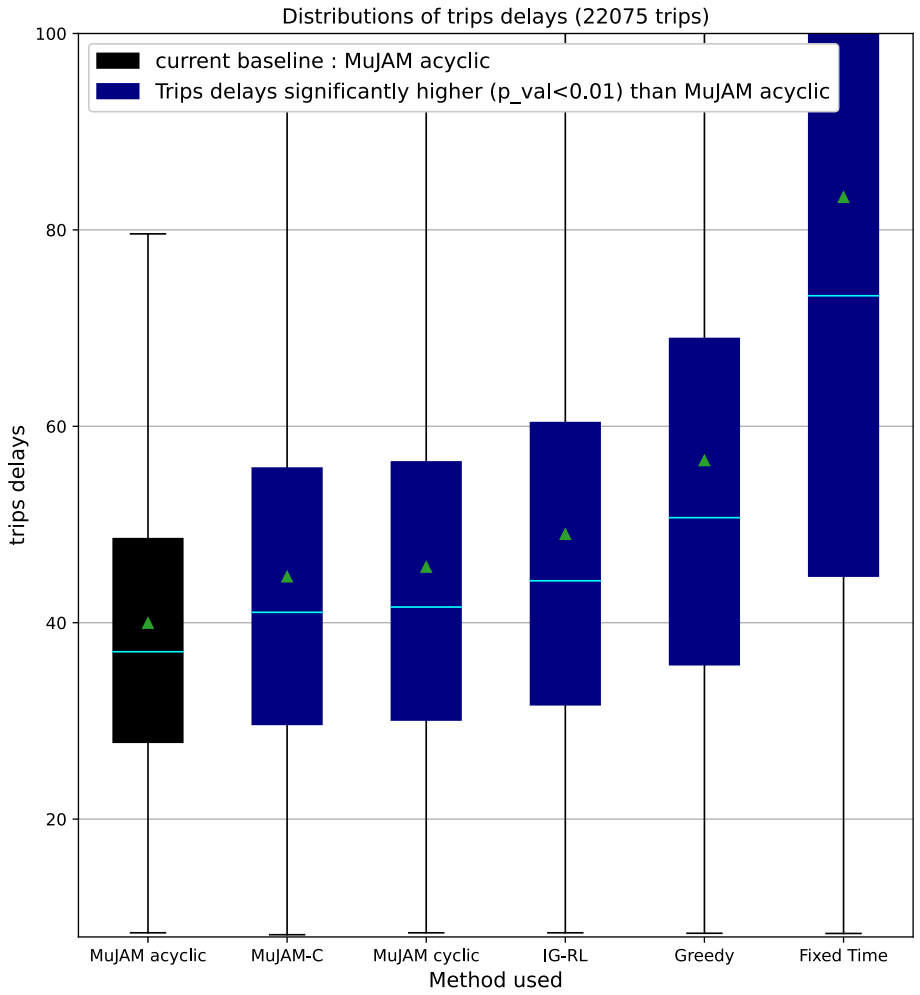

Figure 2.2: Distributions of trips delays

This figure displays the distributions of total delays experienced per trip.

Performance Vs. Fairness

ATSC distributes delay among trips in a road-network as connectivity at any intersection does not enable all vehicles to travel at full speed (i.e. some connections are always closed). An improvement in global performance should not be at the expense of equity in this distribution (i.e. it is socially unacceptable to disproportionately delay some trips for the sake of the average performance). As shown in Fig. 2.4, in addition to accelerating most of the trips (diminishing their delays), the best performing method (MuJAM with acyclic constraints) also happens to distribute delay in a more equitable fashion. Indeed, trips which are advantaged by this method can be drastically delayed by substituting this method with other methods, but trips which are disadvantaged by this method are only marginally delayed.

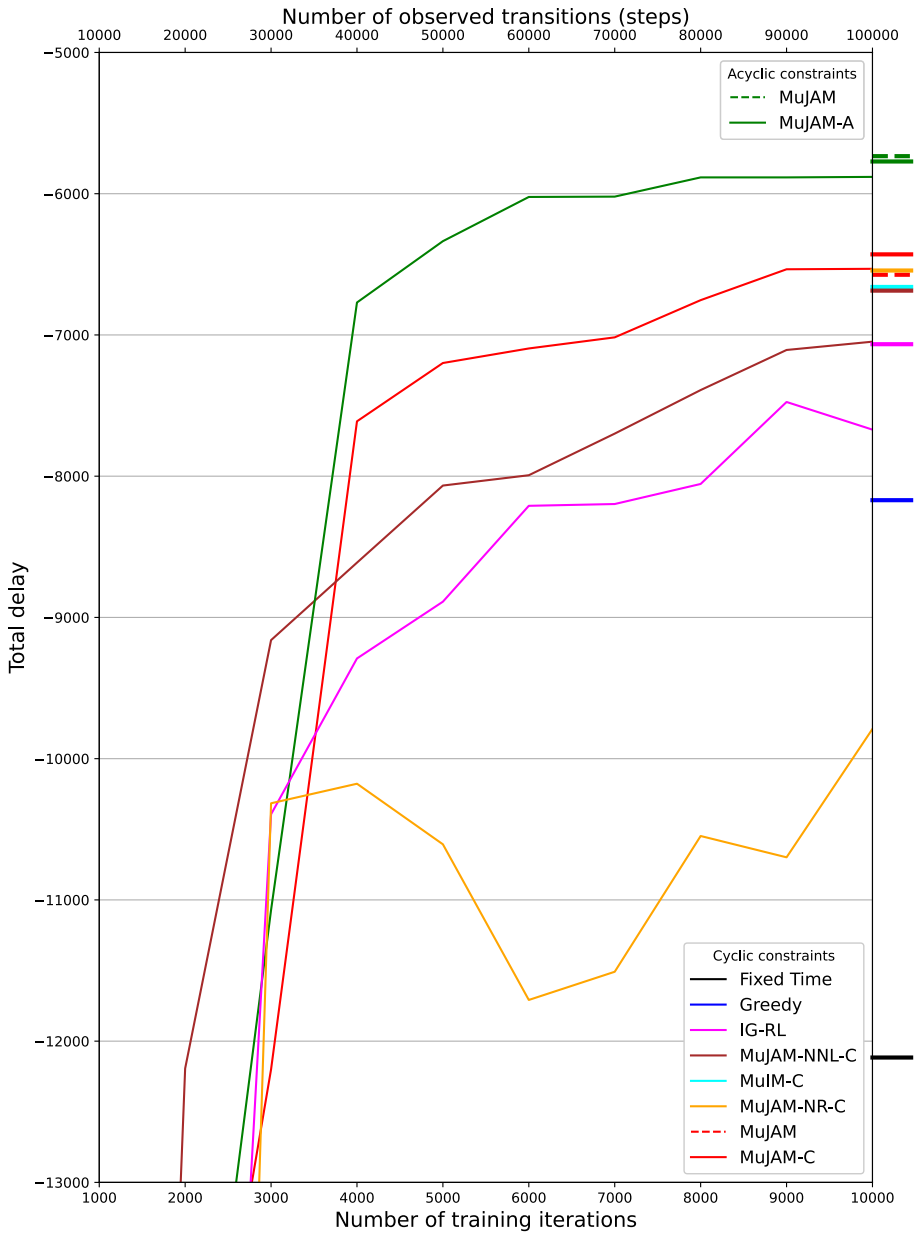

Figure 2.3: Evolution of average total delay during training

This figure displays the average of the total delay (summed over all trips and all time steps) in a given episode (i.e. for a given road-network). Asymptotic performance (after training is completed) is represented as a thick line at the right extremity of the figure. For some methods (for which we specifically want to compare data efficiency) the ratio of training steps to observed transitions (i.e. interactions with the environment) is fixed to 0.1 for the first 10k training steps. Every 1k training steps in this interval, average performance of is measured and reported.

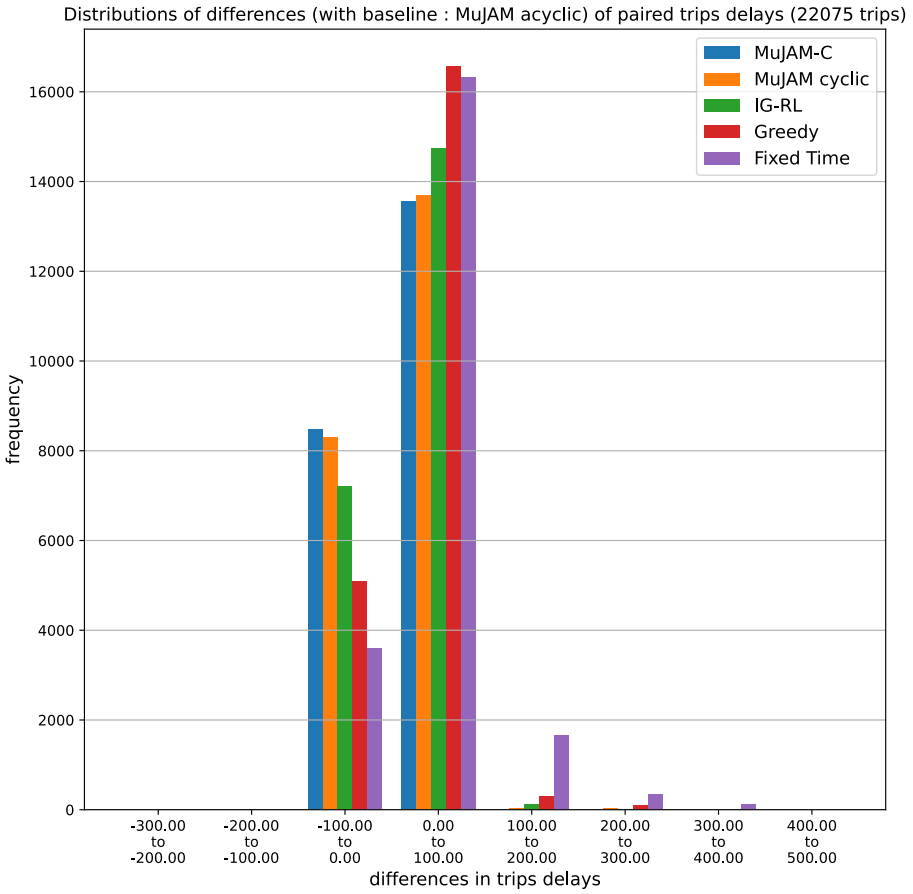

Figure 2.4: Distributions of delays' differences

This figure displays the distributions of the differences of trips delays when compared to MuJAM acyclic.

2.5.4 Experiment 2: Scaling to Manhattan

In this experiment, we push zero-shot transfer to the large scale real-world-network of Manhattan (3,971 TSCs and 55,641 lanes) using a heavy traffic [9] (tens of thousands of vehicles with asymmetric traffic distributions) to evaluate the extent of generalizability and scalability of approaches studied in 2.5.3. The combination of new network structure (including some intersections with complex patterns), new (heavier) traffic distributions, and scaling to a network much larger than anything experienced in training is expected to be a challenging task.

Zero-shot transfer is what enables running this large scale experiment with learned methods as training on such a large network (particularly if using a different set of parameter per intersection as typically done in MARL-ATSC) would involve prohibitive computational costs.

In this experiment, traffic is generated for 30 minutes (warm-start) using the Fixed Time policy, so that the

[9] Even though the road network is real and extracted from openstreetmaps.org, asymmetric traffic distributions used in evaluation are not based on observations of real-world distributions in Manhattan.

traffic is already dense[10] before evaluation starts for all approaches. At this point an episode lasts one hour (density continues to increase continuously during this time). As not all trips are completed at the end of an episode, we cannot pair them for comparison without introducing bias. We therefore report aggregated metrics.

As the cost of evaluation on Manhattan is still more expensive than for synthetic networks used in 2.5.3, we only evaluate methods on 6 instances of randomly generated traffic per run (for a total of 30 instances as we repeat all experiments 5 times as described in 2.5.1).

Results

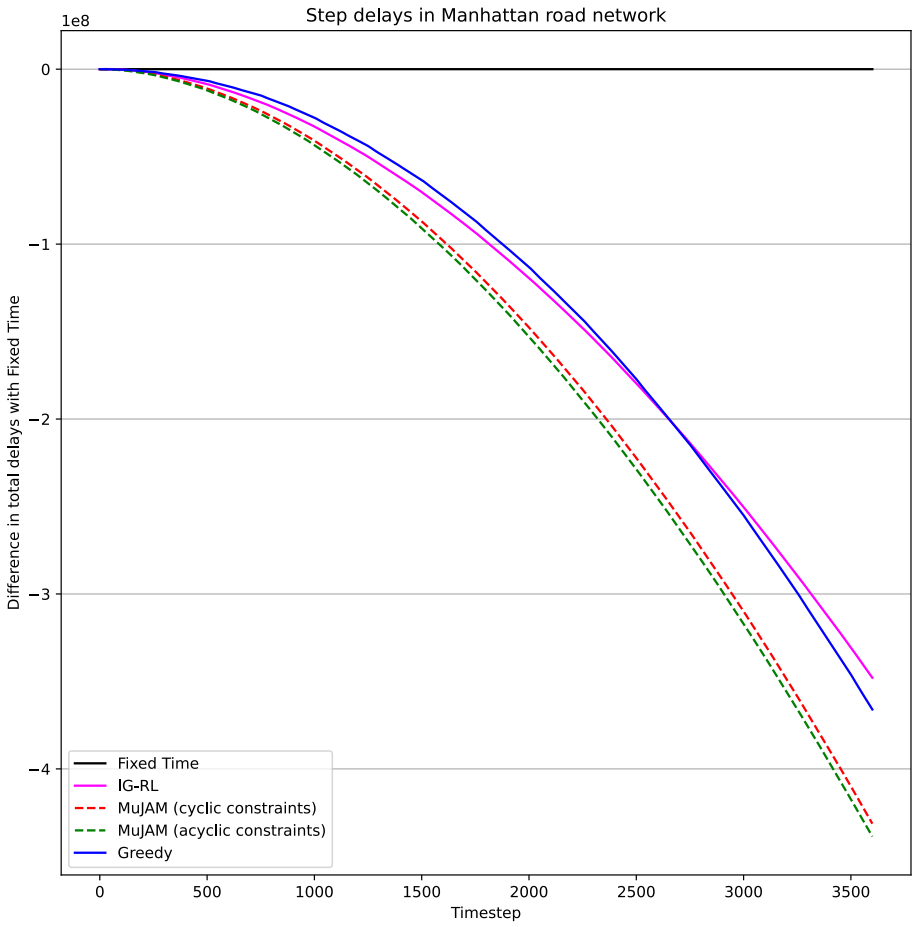

Figure 2.5: Steps delays in Manhattan (compared to Fixed Time policy)

This figure displays the cumulative difference (comparing a given method to the Fixed Time policy) in total delay experienced in the road network.

A first observation is that with MuJAM, we are able to use, for the first time, a coordinated RL-ATSC approach trained with explicit joint action modeling (not only the ability to communicate) on such a large

[10] At this point, 18,000 vehicles have been inserted on average.

setting. To enable this, we only use coordinated priors (see §.2.4.2) to select an action and ignore planning. In other words we use a null search budget for planning β=0. Under this condition, computational complexity is similar to that of IG-RL. Moreover, the same instance of MuJAM is evaluated on both cyclic and acyclic control.

Fig. 2.5 displays for MuJAM, IGRL & Greedy, the cumulative difference with Fixed Time (baseline) in total instantaneous delay per timestep for the entire duration of evaluation (one hour). All methods outperform Fixed Time. MuJAM, even under cyclic constraints, outperforms both IG-RL and the dynamic baseline. This demonstrates that in this challenging setting, this new model-based approach seems to improve generalizability and scalability. Finally, generalization to constraints is once more demonstrated as the same instantiation of MuJAM ends up outperforming all other approaches. We also note that after a certain time, IG-RL seems to underperform the dynamic baseline. We hypothethize that IG-RL suffers from a gradual shift in distribution. As density increases in the road network with time, observed distributions of traffic get further and further away from distributions used during training.

2.6 Conclusion

We introduce MuJAM, a method for zero-shot-transfer-ATSC based on MBRL and GCNs. On top of enabling representation of traffic at the finest level of granularity and transferability to new road-network-architectures and traffic distributions, MuJAM constitutes the first bridge between learnt-cyclic and learnt-acyclic methods, which not only enables training specialist methods on either types of constraints, but also yields hybrid methods which can generalize across these constraints. It is also the first approach to enable joint action modeling at scale for RL-ATSC. MuJAM introduces MBRL developments 1) learning complex graph dynamics models in a latent space 2) improving coherence in exploration by leveraging noisy layers in MuZero, and 3) enabling multi-agent joint action modeling at scale. For RL-ATSC, this work introduces a new level of generalization in line with recent interest/development in this aspect (Devailly et al., 2021), which constitutes a promising way to ease experimentation in urban-mobility-planning without having to train a new model for any tweak of a behavioral constraint, which we hope contributes to real-world applicability.

2.6.1 Future work

MuJAM opens a path for the following future works: 1) MuJAM offers explicit coordination. However, further investigating the advantages of said coordination and how behavior differs from independent modeling could

reveal interesting patterns for RL-ATSC coordination. 2) Offline RL, which consists in training RL approaches using historical data only (i.e. no interaction with the environment), seems appealing for RL-ATSC. As MuJAM only needs to learn local-transferable-traffic-dynamics, it could prove to be less dependent on the online control of TSCs, and easier to train using offline data. Transferable-Offline RL-ATSC would enable both safe and cost-efficient training on offline data, and zero-shot transferability across networks, traffic, and constraints.

References

Abdoos, M., Mozayani, N., and Bazzan, A. L. (2011). Traffic light control in non-stationary environments based on multi agent q-learning. In *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*, pages 1580–1585. IEEE.

Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135.

Aslani, M., Mesgari, M. S., and Wiering, M. (2017). 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*, 85:732–752.

Aslani, M., Seipel, S., Mesgari, M. S., and Wiering, M. (2018). Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran. *Advanced Engineering Informatics*, 38:639–655.

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

Barth, M. and Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171.

Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. (2020). Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Devailly, F.-X., Larocque, D., and Charlin, L. (2021). Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

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

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org.

Hubert, T., Schrittwieser, J., Antonoglou, I., Barekatain, M., Schmitt, S., and Silver, D. (2021). Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pages 4476–4486. PMLR.

Hunt, P., Robertson, D., Bretherton, R., and Winton, R. (1981). Scoot-a traffic responsive method of coordinating signals. Technical report.

Iša, J., Kooij, J., Koppejan, R., and Kuijer, L. (2006). Reinforcement learning of traffic light controllers adapting to accidents. *Design and Organisation of Autonomous Systems*, pages 1–14.

Khamis, M. A. and Gomaa, W. (2012). Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In *2012 11th international conference on machine learning and applications*, volume 1, pages 586–591. IEEE.

Khamis, M. A., Gomaa, W., and El-Shishiny, H. (2012). Multi-objective traffic light control system based on bayesian probability interpretation. In *2012 15th International IEEE conference on intelligent transportation systems*, pages 995–1000. IEEE.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kuyer, L., Whiteson, S., Bakker, B., and Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Lowrie, P. (1990). Scats-a traffic responsive method of controlling urban traffic. *Sales information brochure published by Roads & Traffic Authority, Sydney, Australia*.

Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pages 47–66. Springer.

Nishi, T., Otaki, K., Hayakawa, K., and Yoshimura, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International conference on intelligent transportation systems (ITSC)*, pages 877–883. IEEE.

Oh, J., Singh, S., and Lee, H. (2017a). Value prediction network. *Advances in neural information processing systems*, 30.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017b). Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2661–2670. JMLR. org.

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. (2013). Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, volume 10, pages 1196–1201. IEEE.

Prashanth, L. and Bhatnagar, S. (2011). Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1640–1645. IEEE.

Salkham, A. a. and Cahill, V. (2010). Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th international IEEE conference on intelligent transportation systems*, pages 531–538. IEEE.

Salkham, A. a., Cunningham, R., Garg, A., and Cahill, V. (2008). A collaborative reinforcement learning approach to urban traffic control optimization. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 560–566. IEEE.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Schrank, D., Eisele, B., Lomax, T., and Bak, J. (2015). 2015 urban mobility scorecard.

Schrank, D., Lomax, T., and Eisele, B. (2012). 2012 urban mobility report. *Texas Transportation Institute,[ONLINE]. Available: http://mobility. tamu. edu/ums/report*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B., et al. (2005). Reinforcement learning of traffic light controllers adapting to traffic congestion. In *BNAIC*, pages 216–223.

Van der Pol, E. and Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*.

Varaiya, P. (2013). The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pages 27–66. Springer.

Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., and Xiong, H. (2019). STMARL: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. *arXiv preprint arXiv:1908.10577*.

Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. (2019a). Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2019b). A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*.

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158.

Wiering, M., Veenen, J. v., Vreeken, J., and Koopman, A. (2004a). Intelligent traffic light control.

Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004b). Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 453–458. IEEE.

Xiong, Y., Zheng, G., Xu, K., and Li, Z. (2019). Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2289–2292.

Xu, L.-H., Xia, X.-H., and Luo, Q. (2013). The study of reinforcement learning for traffic self-adaptive control under multiagent markov game environment. *Mathematical Problems in Engineering*, 2013.

Zang, X., Yao, H., Zheng, G., Xu, N., Xu, K., and Li, Z. (2020). Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1153–1160.

Zhang, Z., Yang, J., and Zha, H. (2019). Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651*.

Zheng, G., Xiong, Y., Zang, X., Feng, J., Wei, H., Zhang, H., Li, Y., Xu, K., and Li, Z. (2019). Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972.

Chapter 3

IORL : Inductive-Offline-Reinforcement-Learning for Traffic Signal Control Warmstarting

François-Xavier Devailly, Denis Larocque, Laurent Charlin

Abstract

Reinforcement learning is notoriously data inefficient, and usually requires direct interaction with the environment in a trial-and-error fashion. This constitutes a key challenge for the social acceptability of many of its applications. Traffic signal control is no exception as the typical interaction of untrained reinforcement learning policies with real intersections may cause catastrophic congestion and safety hazards. Recent works rely on the inductive capabilities of graph convolutional networks to transfer learnings to new intersections and traffic distributions. This alleviates the need to retrain a model for a new context (e.g. new road networks, traffic distributions, constraints). In practice, such networks can be trained on a small number of intersections and transferred to larger-scale road networks of hundreds of intersections. However, training such transferable approaches still requires interacting with the real world. On the other hand, offline reinforcement learning refers to methods that rely on historical data instead and so avoid interacting with the environment. Offline reinforcement learning has only been studied in single-intersection road networks and without any transfer capabilities. In this work, we introduce an inductive offline RL (IORL) approach based on a recent combination of model-based reinforcement learning and graph-convolutional networks to enable offline learning and transferability.

IORL yields policies that can 1) be trained from limited data without requiring online real-time collection nor interaction with the environment, making it significantly more practical, and 2) be zero-shot transferred to new intersections and traffic distributions, enabling immediate hazard-free scalability. We evaluate several instantiations of IORL, obtained from different amounts of offline data and different policies, on a zero-shot transfer setting involving both road networks and traffic settings never experienced during training, and in a larger transfer experiment on the central part of Manhattan. We demonstrate that IORL is both robust to the observed policy and data efficient (like its online equivalent, MuJAM) and that it outperforms domain-specific baselines and performs close to its online equivalent. Further, it can easily be fine-tuned, avoiding catastrophic initialization and still reaching the level of performance of its online equivalent.

3.1 Introduction

A lack of space and funding combined with ever-increasing vehicle-usage-rates aggravate traffic congestion whose social, economical, and environmental tolls are well known (Schrank et al., 2012; Barth and Boriboonsomsin, 2008; Schrank et al., 2015). Optimizing the use of existing road infrastructures via adaptive traffic signal control (ATSC) based on the real-time state of the road network has a crucial role to play in mitigating said congestion in a cost-effective fashion (Mannion et al., 2016). The task of manually designing efficient heuristic-based adaptive policies is nontrivial. Machine learning offers an appealing self-optimizing alternative to move beyond such approaches. Recent developments and successes of deep online[1] reinforcement learning (deep RL or DRL), in particular, enable learning optimal policies from high dimensional observations of trajectories, gathered via interaction with the environment (i.e. actual control of TSCs) in a trial-and-error spirit. Among these methods, considering that real-world ATSC involves the simultaneous control of a large number of intersections, multi-agent RL (MARL) has gained the most traction.

One of the curses of online RL is the need for continual interaction with the environment (to collect data) while training. Combined with notorious data inefficiency, this broadly curbs real-world applicability and social acceptability of RL as random exploration from untrained policies might cause both safety hazards and catastrophic performance (Rasheed et al., 2020; Wei et al., 2021).

Recent contributions (Devailly et al., 2021, 2022) have lessened the need to retrain new policies for any change in the setting (e.g. any modification to the road network, traffic distribution, or constraints) on which

[1] Online RL approaches rely on iteratively collecting data from the environment by interacting with it and learning from this data to improve policies.

ATSC is to be applied by focusing on the generalizability of learnings and enabling zero-shot transfer[2] (to new intersections, traffic distributions, and controller constraints) once policies have already been trained on a diverse group of settings. Transferring policies to new settings addresses in part the data efficiency challenge. However, the original training of these transferable policies still relies on exploratory online interactions. The burgeoning field of *offline RL* (ORL) (see § 3.2.5) aims to address this challenge by enabling the training of policies from already available data (i.e. without interacting with the environment). The challenge is, therefore, to improve upon the good parts of a historical behavior (i.e. learn generalizable dynamics of the environment and promising behaviors) without observing anything else than the historical behavior itself. The challenge of ORL lies in the distribution shift between states observed under the historical policy (during data collection) and states the trained policy will end up in according to its behavior. ORL has barely been mobilized in RL-ATSC so far, and only in restricted scenarios (unique isolated intersections), but may prove to be critical to its expansion (Mannion et al., 2016).

3.1.1 Contribution

In this work, we introduce Inductive Offline Reinforcement Learning (IORL), a new graph-based method that leverages MBRL to yield policies that are trained without interacting with the environment and transferred to new settings, translating into immediate scalability. Fig. 3.2 and 3.3 decompose recent advances in inductive RL-ATSC and illustrates how IORL builds upon them to enable inductive-offline learning. IORL benefits from the following new capabilities:

- *Offline Generalization & Scalability:* IORL is the first inductive-offline RL-ATSC method. This means that not only can it be trained without interacting with the environment (therefore bypassing safety and inefficiency concerns involved with online training of RL agents), it can then be immediately used (via zero-shot transfer) on a large variety of new scenarios involving new road-networks, new traffic distributions and new controller constraints. IORL is the first Multi-Agent ORL-ATSC method (more generally the first used on more than a single intersection). Zero-shot transfer translates into immediate scalability as the same model can be applied with a constant set of shared parameters on any new setting, independently of the size of the road network. Furthermore, while most MARL methods do not enable coordination, or at most enable communication (see § 3.3.1), IORL enables explicit coordination of TSCs via joint action modeling *at scale*.

[2]Immediate transfer with no additional training required.

- *Flexibility:* IORL displays stable performance that seems robust to the policy that is observed in offline data. This means that whether the TSCs that are to be equipped with RL currently work with fixed-time policies or dynamic heuristic-based approaches, IORL can exploit the data generated from the observation of these policies equally well. Even though we only represent demand at its finest level of granularity (vehicle level) and experiment only with cyclic control (traffic signal control must respect a predefined cycle of connectivity at a given intersection) in this work, by inheriting from the ability of IGRL (Devailly et al., 2021) and MuJAM (Devailly et al., 2021), IORL is the first ORL-ATSC method that can 1) use both lane-level and vehicle-level state information, and 2) be trained for constraint agnosticism.

- *Data efficiency (Offline & Online):* Benefiting from MBRL, IORL can be trained with a relatively small amount of offline data, meaning that even in the offline learning setting, it could be applied faster after the installation of sensors. After training IORL by exclusively using offline data, it is possible to fine-tune the policy online. This way, safety and inefficiency concerns of early training can still be bypassed as IORL only interacts with the environment after being trained, and IORL can still reach the same level of performance as its online counterpart with little online interaction (online data). We refer to this technique as Offline Warm-starting(OW).

To test these claims, we evaluate various instantiations of IORL, trained with offline datasets of varying sizes and generated observing different policies on zero-shot transfer learning tasks involving synthetic road networks and traffic distributions both never experienced during training. We also evaluate an instantiation of IORL in the large real road network of central Manhattan. In both experiments, we compare the performance of our offline approach with its online equivalent (i.e. MuJAM), as well as fixed-time and dynamic heuristic-based baselines. We show 1) that with limited data and with both policies observed offline, IORL outperforms other transferable baselines, and 2) that its performance is close to that of MuJAM. We also demonstrate that OW is a viable and straightforward way to keep the benefits of IORL training and still reach MuJAM's performance.

3.2 Background

3.2.1 Graph Convolutional Networks

To model complex relations between objects in a network (e.g. road-network), it is natural to represent them as a graph. To exploit such a complex structure, a popular neighborhood-information aggregation framework is the GCN (Kipf and Welling, 2016). Both nonlinear transformations of semantic and structural information can

be exploited with this architecture. To do so, messages (parameterized transformations of node representations) are propagated along edges recursively. When dealing with heterogeneous networks (i.e. graphs with different types of nodes and edges), Heterogeneous GCNs or HGCNs (Schlichtkrull et al., 2018) (referred to as GCNs for the rest of the paper for simplicity) consist in the use of different parameters for different types of edges. The aggregation to be performed on every node, at every layer is as follows:

$$n_i^{(l)} = f\left(\sum_{e \in E} \sum_{j \in N_e(i)} W^{l_e} \cdot n_j^{(l-1)}\right) \tag{3.1}$$

where n_i^l is the embedding at layer l of node i, f is a non-linear differentiable function, $N_e(i)$ are the immediate neighbors of node i in the graph of relation type e, E is the set of relation types, and W^{l_e} is the l^{th} layer's weight matrix for message propagation corresponding to a relation of type e. All transformations of this deep learning architecture are differentiable and training of its parameters is performed via gradient-based optimization.

3.2.2 Markov Decision Process

A Markov Decision Process (MDP) is composed of a set of states S, a set of actions A, and a transition function $\mathscr{T}$, defining probabilities of transitioning from a state to another state given an action (i.e.
dynamics of the environment), and a reward function R defining performance with respect to the task (Puterman, 1990). It is used as a framework to solve sequential decision-making tasks. Attempting to solve an MDP means finding a mapping from S to A (i.e. a policy) maximizing cumulative future rewards (i.e. value): $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$ where r_t represents the reward experienced at time step t, and γ is a temporal discount factor.

3.2.3 Reinforcement Learning

Reinforcement learning is a popular way to solve MDPs by trial-and-error (Sutton and Barto, 2018). Transitions (observations gathered from the environment) are used to train models (usually parametric estimators such as deep neural networks) in an empirical dynamic programming fashion to either 1) learn to estimate state-action *value functions* (Q functions) which implicitly define an optimal policy over choosing actions maximizing them, or 2) explicitly learn an optimal mapping from S to A (often referred to as actor networks).

3.2.4 Model-Based Reinforcement Learning

Model-Free RL directly learns from value estimates. Model-Based RL (MBRL), on the other hand, leverages knowledge of environmental dynamics (i.e. the ability to accurately simulate the evolution of parts or all of

the state space based on action-state pairs). MBRL empirically yields better data efficiency and improved performance in both the contexts of complex planning tasks (Schrittwieser et al., 2020; Silver et al., 2017).

One of the most popular ways to use this knowledge is by planning upstream of action selection. To do so, trajectories are simulated, and the results of these rollouts determine the action with the highest expected value. Among planning-based approaches, Tree Search (TS) based MBRL methods have reached superhuman performance in complex planning tasks (Silver et al., 2016, 2017). One of the main limitations of such approaches is that in most real-world applications, no reliable simulator of the environment is readily available. Environmental dynamics must therefore be learned, and doing so in high-dimensional state spaces where most features may be unrelated to the task at hand is nontrivial. In environments with these attributes, MBRL significantly lagged behind MFRL in terms of performance (Schrittwieser et al., 2020). Planning in a latent value-equivalent MDP Schrittwieser et al. (2020) enables implicit feature selection and dimensionality reduction for model learning in MBRL. For this, an autoregressive model must simply be able, given a starting state and a sequence of actions, to predict value-related metrics at every simulated step. This contribution propelled MBRL to the level of MFRL (with improved data efficiency) on tasks with high dimensional state spaces while keeping its lead for complex planning tasks.

In the context of ATSC however, MBRL has been largely ignored in favor of MFRL since the early 2010s (see §3.3.3) (Mannion et al., 2016; Devailly et al., 2022). Moreover, many state-of-the-art RL-ATSC methods, including transferable RL-ATSC, rely on parameter-sharing schemes and inductive abilities of graph convolutional networks (GCNs)(see § 3.3.2 & § 3.2.1). Learning a model of the dynamics (evolution) of such complex and high dimensional objects (graphs) is particularly challenging and it is only recently that MuJAM (Devailly et al., 2022) was able to combine latent space planning with inductive GCN-based learning for ATSC.

3.2.5 Offline Reinforcement Learning

Given a fixed dataset collected under a historical policy Π_Ω, ORL aims to learn an improved policy Π_A without interacting with the environment (to limit cost and safety concerns) (Levine et al., 2020; Prudencio et al., 2022). Generalizing 1) learned environmental dynamics and 2) promising behaviors of Π_Ω to unvisited parts of the state space is the core challenge of ORL. ORL must address the distributional shift that exists between 1) regions of the state and actions spaces that have historically been explored and are represented in available datasets, and 2) regions of the state and action spaces that a policy trained on this data will actually interact with. In other words, when applying Π_A that was obtained using Π_Ω, it is likely that the agent will either end up in parts of the state space that were never seen by Π_Ω, which can lead to degeneration if the estimates

corresponding to these states are unreliable. Some approaches prevent degeneration by constraining the policy to the historical behavior. In the particular case of MBRL, an explicit measure of uncertainty can also be used to constrain the policy (Yu et al., 2020). On top of their advantages with respect to planning and data efficiency, MBRL methods tend to be less prone to overestimation and overfitting as they do not solely rely on bootstrapping value estimates and typically learn from complementary supervised signals. As ORL implies dealing with out-of-distribution regions of the state space, it can be seen as a transfer learning problem. For this reason, MBRL tends to outperform MFRL methods in offline RL settings (Yu et al., 2020).

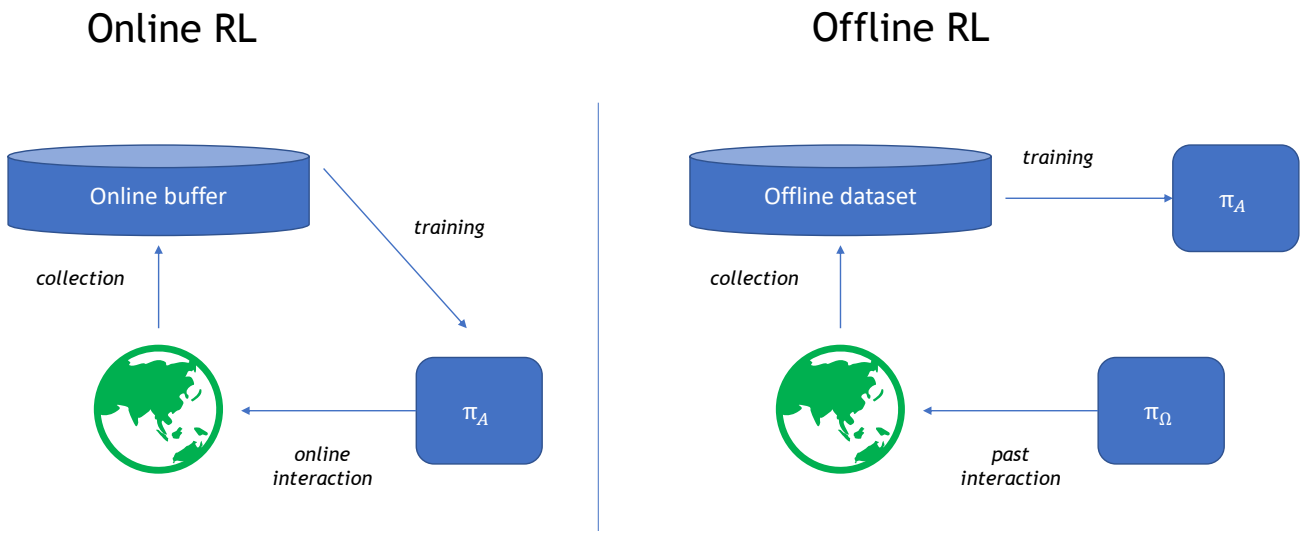

Figure 3.1: Online Vs. Offline learning paradigms

Traditional online reinforcement learning is illustrated on the left side. Π_A is typically initiated randomly (randomly parameterized) and interacts with the environment following the corresponding random behavior. Observations of these interactions are collected in a memory buffer that is used for training Π_A (i.e. find new values for the parameters which are expected to improve performance). The improved model (i.e. with the latest parameters) is used again to interact with the environment and observe new interactions corresponding to the new behavior. This loop of incremental improvement is repeated many times. By contrast, in offline RL (illustrated on the right side), data used for training Π_A correspond to interactions of a historical policy Π_Ω with the environment. Often, this historical data is readily available as it was collected as part of other processes. If not, it only needs to be collected (i.e. observed) without modifying any existing behavior.

3.3 Related Work

3.3.1 Multi-Agent RL-ATSC

Action spaces grow exponentially with the number of intersections. The number of available actions $\prod_{i=1}^{n} |A_i|$ with n the number of intersections and $|A_i|$ the number of possible actions for intersection i according to

its current state. For this reason, centralized approaches (Casas, 2017; Prashanth and Bhatnagar, 2011) to multi-intersections RL-ATSC do not enable significant scalability. Decentralization and distribution of control via MARL is a critical component to the success of modern RL-ATSC approaches (El-Tantawy and Abdulhai, 2010; Abdoos et al., 2011; El-Tantawy et al., 2013; Khamis and Gomaa, 2012; Steingrover et al., 2005; Salkham et al., 2008; Van der Pol and Oliehoek, 2016; Arel et al., 2010; Balaji et al., 2010; Aslani et al., 2018; Pham et al., 2013; Kuyer et al., 2008; Zheng et al., 2019; Xiong et al., 2019; Nishi et al., 2018; Wei et al., 2019; Xu et al., 2013; Khamis et al., 2012; Aslani et al., 2017; Chu et al., 2019; Iša et al., 2006; Wiering et al., 2004b; Wiering, 2000; Zhang et al., 2019; Mannion et al., 2016; Wiering et al., 2004a). Coordination between a large number of agents involves the same explosion of the action space as using a centralized controller. For this reason, most MARL-ATSC methods include no form of coordination between agents. Some include communication ability (El-Tantawy and Abdulhai, 2010; Nishi et al., 2018; Zhang et al., 2019; Wei et al., 2019; Devailly et al., 2021; Arel et al., 2010), and a small proportion introduce explicit coordination (El-Tantawy et al., 2013; Kuyer et al., 2008; Van der Pol and Oliehoek, 2016; Xu et al., 2013; Devailly et al., 2022).

3.3.2 Parameter Sharing & Transfer Learning

Parameter sharing between agents has enabled further scalability (Devailly et al., 2021; Zang et al., 2020; Zheng et al., 2019; Wei et al., 2019; Chen et al., 2020; Wang et al., 2019) by keeping the number of parameters constant (and therefore keeping memory requirements in check) at both training and inference time. Zang et al. (2020) use meta-learning to limit the quantity of experience required to adapt to a new setting, and Devailly et al. (2021) achieve zero-shot transfer of learned policies on road networks with orders of magnitude more controllers than the ones used during training.

3.3.3 Model-Based

Whether its dynamics model (i.e.

simulator) is learned or provided, MBRL typically offers higher data efficiency than MFRL and improved performance in complex planning tasks (see §3.2.4). Most MBRL methods for RL-ATSC are from the early 2010s or before (Salkham and Cahill, 2010; Khamis et al., 2012; Wiering, 2000; Wiering et al., 2004a; Steingrover et al., 2005; Khamis and Gomaa, 2012; Wiering et al., 2004b; Kuyer et al., 2008) and do not enable scaling. In the context of high dimensional graph representations of road networks, Devailly et al. (2022) decentralizes modeling at the lane level and plans in a latent value-equivalent MDP using MuZero (Schrittwieser et al., 2020) to enable inductive MBRL with GCNs.

3.3.4 Offline RL-ATSC

Offline RL has barely been mobilized in ATSC. Dai et al. (2021) use MFRL to perform offline learning while Kunjir and Chawla (2022) use MBRL. In both cases, a single intersection is controlled (see Figure 3.3).

3.4 IORL: Inductive-Offline-Reinforcement-Learning

This section first introduces the MDP for ATSCs. It then describes 1) how the road network is represented (with a GCN), 2) the modeling approach used to train a simulator of the environment, and 3) the way this simulator is used to act optimally in the environment. Finally, it details both the new offline learning paradigm and procedure, as well as the optional online-fine-tuning procedure.

3.4.1 ATSC MDP

The MDP for ATSC involves the simultaneous and coordinated control of all traffic signal controllers in a given road network.

State space S

The state must include all relevant information for decision-making, including:

- The structure of the road network. Different networks have different structures, but the structure is fixed per network and so across the timesteps of an episode.
- The state of connectivity of the road network defines how vehicles are allowed to move from one lane to another lane at a given intersection at the current timestep. This part is immediately influenced by actions and indirectly influences traffic. It changes within a given episode but typically involves a finite number of possible combinations as each controller has a finite number of phases[3] defining connectivity.
- The state of traffic is the state of all vehicles within the road network. It changes within a given episode.

In this work, a GCN(see § 3.2.1) is used to represent the state. Its structure and features are the same as those described in Devailly et al. (2022).

[3] A phase defines the full state of connectivity at a given intersection.

Action space A

In the cyclic setting, the local action space of a controller is binary and consists in choosing to pursue the current phase or switch to the next phase in the cycle. The joint (coordinated) action space consists of the intersection of all local action spaces. All phases involving yellow lights which can exist between two consecutive selectable phases of the cycle are enforced for a fixed duration. Additionally, two consecutive local changes in the phase must be separated by a minimum of 5 seconds.

Reward function R

The reward function penalizes the number of stopped vehicles as follows:

$$R(s) = -\sum_{l \varepsilon L} \sum_{v \varepsilon V_l} q_v,$$
$$q_v = \begin{cases} 1, & \text{if } \sigma_v < 0.1 \textit{ and } \delta_v \leq 50 \\ 0, & \text{otherwise} \end{cases} \tag{3.2}$$

where L is the set of lanes, V_l is the set of vehicles currently on lane l, σ_v is the current speed of vehicle v, and δ_v is the distance of vehicle v to the end of lane l (i.e. to the next intersection). In Equation 3.4.1, $q_v = 1$ when a vehicle is at a full stop near the next intersection and 0 otherwise. This popular (Wei et al., 2021) queue length reward is dense in the sense that it is quickly influenced by a change in connectivity at an intersection. This eases the attribution of rewards to actions and the choice of this particular reward function is further motivated by the fact that the corresponding metrics are simple counts that can be captured using detectors installed either on the road network (e.g. at the level of lanes) or on vehicles.

Steps & Episodes

An episode either lasts until all trips are completed or for a fixed number of steps (i.e. a fixed duration).

3.4.2 Simulator and model-based planning

IORL relies on decentralized MBRL in a latent space introduced and detailed in Devailly et al. (2022). The (local) simulator (or dynamics model) part of our model learns to predict, locally for every lane, the evolution of traffic in a latent space based on the current state of the environment and actions taken by the controllers (agents). The (local) actor (or prior) part learns to predict, for every intersection, the local action that will be part of the best joint action. The simulator is applied on every lane of the network as many times as required

to simulate rollouts and compare their value to identify the most promising joint (and therefore also local) actions. In practice, the planning is achieved with a tree search (see §3.2.4). The actor serves as a prior to only simulate promising actions in the tree search. It is used at every intersection as many times as required to sample joint actions to simulate the corresponding rollouts. The representation for the state of the road network (architecture and features of the GCN), the training procedure of our decentralized (per lane) latent-space simulator and decentralized actor (or prior) network Φ, and the planning algorithm (tree search) used to act upon simulated rollouts, are the same as those used in Devailly et al. (2022).

Continual training & simulation

To learn using an offline dataset, IORL performs the following iterative procedures in parallel.

- States are continuously sampled from existing observations to refresh their corresponding training targets by running TSs (i.e. simulating rollouts starting from these states) with current parameters, This enables 1) updating values estimates for all lanes and 2) identifying the best joint action among the ones which were sampled in the root node of the TS in order to replace target actions (those used to train the actor) by the local actions corresponding to the best joint action.

- Observations (i.e. a sequence of transitions), which are composed of an initial state, a sequence of actions per intersection, a sequence of observed rewards per lane, a sequence of target long term values per lane, and a sequence of target local actions per intersection, are continuously sampled and used to train both the simulator and Φ (actor). Given the initial state and the sequence of actions, the model is trained to predict all other provided quantities by gradient-based optimization (backpropagation of the corresponding errors).

As the learned simulator improves, reward-related estimates corresponding to different trajectories become more reliable. As Φ improves, the TS samples actions that correspond to more promising trajectories, which provide improved value estimates. In turn, these improved estimates update prior targets. This defines the positive feedback loop of continuous and parallel training and simulation.

Out-of-distribution model-based extrapolation

ORL consists in dealing with out-of-distribution (OOD) regions of the state space (see § 3.2.5). In MFRL, if the value of an OOD state-action pair is mistakenly evaluated as promising with current parameters (i.e. it

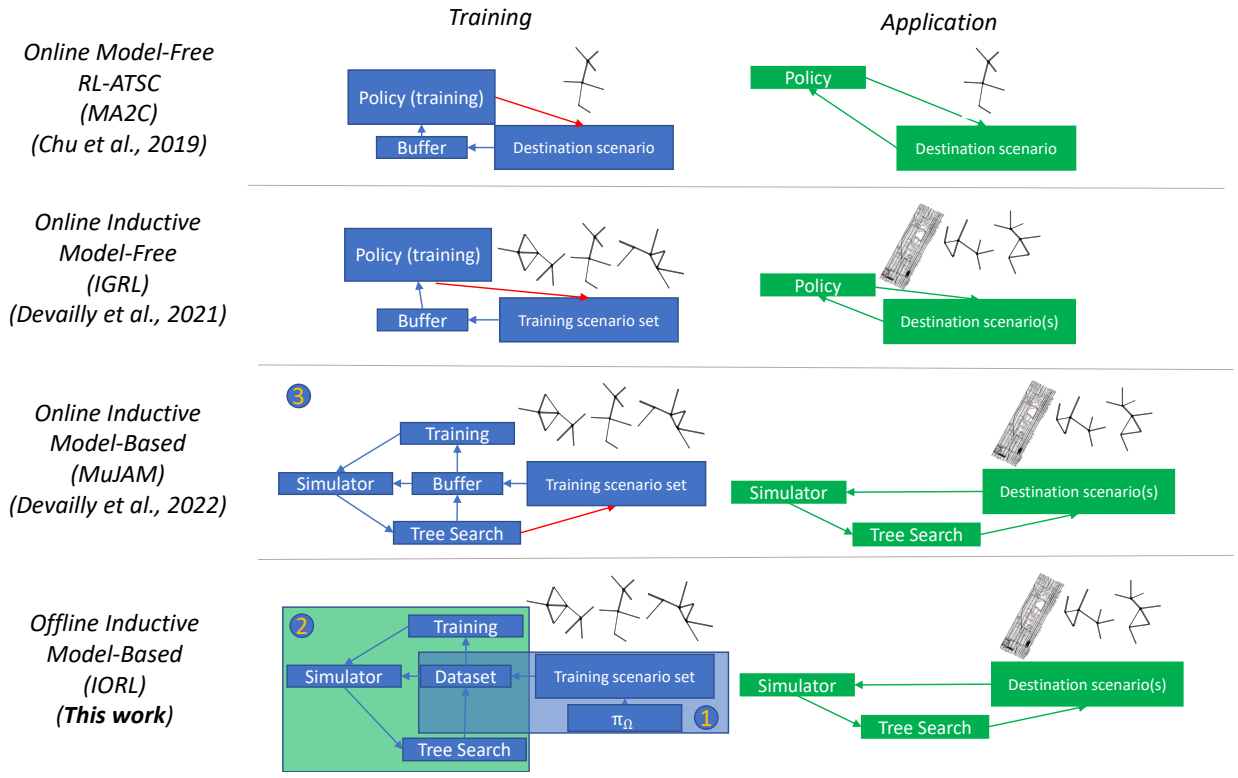

Figure 3.2: Evolution of generalization

The 1st row corresponds to the traditional RL-ATSC framework for which the training and application settings (network architecture, traffic distribution, policy constraints) must be identical. The 2nd row corresponds to the introduction of model-free inductive capabilities with IGRL (Devailly et al., 2021) which enables transferability to new network architectures and traffic distributions as well as scalability. The 3rd row corresponds to the introduction of model-based inductive learning (MuJAM (Devailly et al., 2022)) which builds upon IGRL to enable better performance, improved sample efficiency, coordination at scale, and a new dimension of generalization to policy constraints, and constitutes the online equivalent of IORL. The final row illustrates how IORL combines model-based inductive learning and offline learning paradigms. Note that the main contribution of IORL is to completely remove online interaction with the environment during training (represented as a red arrow in the other rows). Offline learning phases (see § 3.4.3) are numbered and highlighted in orange.

provides the best state-action value estimate among all available actions), it will immediately derail the state-action estimates of the rest of the state space by bootstrapping, By contrast, if an OOD state-action pair seems promising, IORL explicitly simulates the corresponding transition (and possibly subsequent transitions) using its dynamics model to assess its value before it influences training targets.

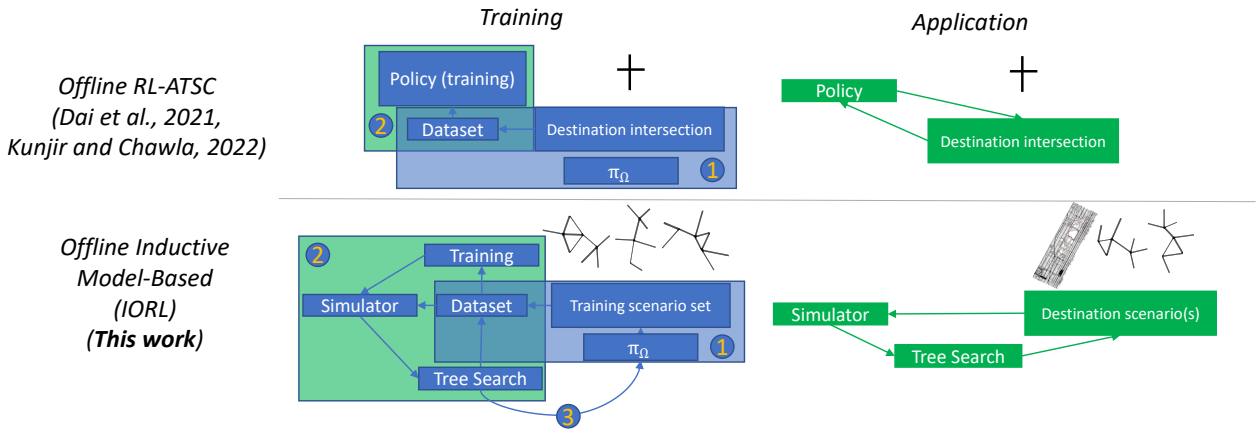

Figure 3.3: Offline scaling

The 1st row corresponds to recent offline RL-ATSC efforts which enable offline learning on a single intersection. The 2nd row illustrates how IORL leverages inductive learning to enable the transferability and scalability of offline RL. Offline learning phases (see § 3.4.3) are numbered and highlighted in orange. The 3rd step refers to fine-tuning and is optional for IORL.

3.4.3 Offline Learning Formulation

Phase 1: Data collection

In the first phase, observations are collected under the historical TSC policy Π_Ω. In real-world ATSC the historical policy refers to the policy which was used until now (e.g. the ubiquitous fixed-time policy). These observations are transitions between consecutive timesteps and include both the consecutive states, action(s) linking these consecutive steps, and rewards corresponding to these transitions. Unless specified, we use a fixed-time policy as the historical policy Π_Ω in this work. The pseudocode corresponding to this phase is provided as Algo. 8

Phase 2: Offline Training

In the second phase, observations are continually sampled from the collected data by two processes, running in parallel: the first process updates training targets by running TSs, while the second performs gradient-based optimization using these targets. As the exploration of simulated trajectories improves (i.e. more promising actions are sampled) the model simulates unobserved trajectories which are evaluated as better than the ones generated by Π_Ω. Under the assumption that the dynamics model and Φ are reliable in such OOD states, the policy is expected to enable improved efficiency (i.e. higher cumulative long-term reward) compared to Π_Ω. The pseudocode corresponding to this phase is provided as Algo. 9

Phase 3 (optional): Fine-Tuning

Under the assumption of unlimited trial-and-error, online RL can always validate assumptions by exploring the state space. For this reason, it does not require dealing with OOD parts of the state space. Offline RL is therefore not expected to reach the same level of performance as online RL *ceteris paribus*. In this section, we propose a way to combine the advantages of both approaches. Once IORL is trained offline, even if it might not be as efficient as its online equivalent, it is still expected to perform reasonably well. Online interaction of this trained policy with the environment is therefore expected to be quite efficient already and therefore avoid catastrophic performance. We introduce a training strategy, OW, which consists in using IORL's trained parameters as an initialization to train an online version of IORL (i.e. MuJAM).[4]. The pseudocode corresponding to this phase is provided as Algo. 10

3.4.4 Algorithms (pseudocodes)

This section provides pseudocodes used in IORL for both training and inference.

Data collection

This algorithm defines how data is collected from the historical policy Π_Ω.

Algorithme 8 : Data Collection

Input : N : Number of observations to be gathered

Input : T : Episode duration

Initialize empty replay buffer (offline dataset) D_{π_Ω} and j = 0

do in parallel

- **Self-play processes (gathering experiences by interacting with the environment):**
  - **while** $j<N$ **do**
    - Generate and start episode (road network and trips)
    - **for** $t=0,\ldots,T$ **do**
      - Build the graph G_{s_t} corresponding to s_t
      - Select local actions using the historical policy : $a_t = \pi_\Omega(s_t)$
      - Execute joint action A_t, observe reward r_t and new state s_{t+1}
    - Add all transitions from the episode $\{(G_{s_t}, A_t, r_t, G_{s_{t+1}})\}$ to the shared buffer D_{π_Ω}
    - $j = j + T$
  - Terminate self-play process

Save D_{π_Ω}

[4] As in Devailly et al. (2022), for online training, we use noisy layers in Φ to enable coherent exploration.

Offline Training

This algorithm defines how the offline dataset is used to perform offline training of IORL.

Algorithme 9 : Offline Training (IORL)

Input : T : Episode duration

Input : N : Batch size

Input : H : Training depth

Input : $Stop$: Stopping criterion (e.g. no improvement over a number of training or evaluation steps)

Initialize model (GCN & MLPs weights : θ)

(Φ designates the actor MLP, while ϕ designates the local multinomial distribution whose probabilities are obtained by applying Φ on selectable phases' nodes of a given intersection (Devailly et al., 2022))

Load replay buffer (offline dataset) D_{π_Ω}

do in parallel

Centralized process (enables asynchronous access to D_{π_Ω} from the other processes):

Simulation processes (continuous update of training targets with tree searches):

while *not Stop* **do**

Sample a random episode e from D_{π_Ω}

Sample a random step (i.e. transition) t from e

Add/Update $(value_t, A_t^*)$ to e_t by re-running the Tree Search (Devailly et al., 2022) with fresh parameters

Training process (training the simulator and actor) :

while *not Stop* **do**

Sample a random minibatch of N sequences of transitions of length H from D

Gather initial representations G_{init} from G_{s_t} (Devailly et al., 2022)

loss = 0

for $t \rightarrow t+H$ **do**

$r_t^{pred}, v_t^{pred} = Infer_{value}(G_t)$ (Devailly et al., 2022)

$\phi_t = Infer_{prior}(G_t)$ (Devailly et al., 2022)

$G_{t+1} = Simulate(G_t, A_t)$ (Devailly et al., 2022)

$loss += l1(r_t^{pred}, r_t) + l1(v_t^{pred}, v_t) + cross\_entopy(\phi_t, A_t^*)$

Backpropagate the loss to compute gradients and perform gradient descent based optimization

Save updated model (GCN & MLPs weights : θ)

Fine-Tuning

This algorithm defines how OW is obtained by fine-tuning the policy yielded by IORL, online.

Algorithme 10 : Fine-tuning (OW)

Input : T : Episode duration

Input : N : Batch size

Input : H : Training depth

Input : $Stop$: Stopping criterion (e.g. no improvement over a number of training or evaluation steps)

Load model (GCN & MLPs weights : θ)

(Φ designates the actor MLP, while ϕ designates the local multinomial distribution whose probabilities are obtained by applying Φ on selectable phases' nodes of a given intersection (Devailly et al., 2022))

Initialize empty replay buffer D

do in parallel

- **Centralized process (enables asynchronous access to D from the other processes):**
- **Self-play processes (gathering experiences by interacting with the environment) :**
  - **while** *not Stop* **do**
    - Generate and start episode (road network and trips)
    - Observe initial state s_1
    - **for** t=1,T **do**
      - Build the graph corresponding to s_t, G_{s_t}
      - Get $value_t, A_t^*$ from running the tree search on G_{s_t} (Devailly et al., 2022)
      - Get ϕ_{noisy} using parameter noise in $\Phi(G_{t_{init}})$
      - Sample an exploratory joint action A_t from ϕ_{noisy}
      - Execute A_t and observe (per-lane) rewards r_t and new state s_{t+1}
    - Add all transitions $(G_{s_t}, A_t, A_t^*, r_t, G_{s_{t+1}}, value_t)$ to D
- **Simulation processes (continuous update of training targets with tree searches):**
  - **while** *not Stop* **do**
    - Sample a random episode e from D
    - Sample a random step (i.e. transition) t from e
    - Add/Update $(value_t, A_t^*)$ to e_t by re-running the Tree Search (Devailly et al., 2022) with fresh parameters
- **Training process (training the simulator and actor) :**
  - **while** *not Stop* **do**
    - Sample a random minibatch of N sequences of transitions of length H from D
    - Gather initial representations G_{init} from G_{s_t} (Devailly et al., 2022)
    - loss = 0
    - **for** $t \rightarrow t+HH$ **do**
      - $r_t^{pred}, v_t^{pred} = Infer_{value}(G_t)$ (Devailly et al., 2022)
      - $\phi_t = Infer_{prior}(G_t)$ (Devailly et al., 2022)
      - $G_{t+1} = Simulate(G_t, A_t)$ (Devailly et al., 2022)
      - $loss+ = l1(r_t^{pred}, r_t) + l1(v_t^{pred}, v_t) + cross\_entopy(\phi_t, A_t^*)$
    - Backpropagate the loss to compute gradients and perform gradient descent based optimization

Save updated model (GCN & MLPs weights : θ)

3.5 Experiments

3.5.1 General Setup

The general setup is the same as the one introduced by Devailly et al. (2021). SUMO (Lopez et al., 2018) is used to perform all the experiments in this section.

Synthetic road networks

The structure and connectivity of the road networks are randomly generated using SUMO. Between 2 and 6 intersections form a typical road network. Lanes are between 100 and 200 meters long and roads are composed of 1 to 4 lanes.

Traffic generation

Every 2 minutes, trajectory distributions (probabilities for a trip to start and end on given lanes) are randomly sampled. Traffic generation is therefore nonstationary and asymmetric.

Hyperparameters

Unless specified, we use the same hyperparameters as the online equivalent of our approach (see Devailly et al. (2022)).

Evaluation

We use instantaneous delay $d_t = \sum_{v \varepsilon V}(m_v^* - \sigma_v)/m_v^*$ as the measure of performance, where $m_v^* = min(m_v, m_l)$, V is the current set of all vehicles in the road network, m_v is the maximum speed a vehicle can physically reach, m_l is, according to its current position, the maximum speed a vehicle can legally reach.

Robustness

Random seeds influence the generation of road networks and traffic as well as the initialization of parametric methods and exploratory noise (only for approaches trained online). All experiments are run 5 times to ensure the robustness of the presented conclusions.

Fixed Time

By default, SUMO defines cyclic programs based on the structure of a given intersection. This approach allows immediate applicability to any road network as it does not involve training, and is therefore compatible with this zero-shot transfer setting.

Max-moving-car heuristic (Greedy)

This approach dynamically maximizes the number of moving vehicles at an intersection by switching to the next phase if more vehicles are stopped than moving on incoming lanes. It also benefits from applicability without training on any new road network and is therefore compatible with this zero-shot setting.

MuJAM

MuJAM is the online version of IORL. It continuously gathers data by interacting with the environment in an exploratory fashion.[5]

IORL

IORL is the main method introduced in this work as described in § 3.4. Unless specified otherwise, we use a dataset of only 30K, designated as *D3* in Table 3.5.4, to train IORL. Compared to usual model-free RL algorithms which are notoriously inefficient, including in the ATSC setting (Wei et al., 2021), and typically require millions or even billions of observations for efficient training (Schrittwieser et al., 2020), this amount a data is fairly small. An observation is an observed transition of the MDP. A transition consists in a tuple that includes: 1) two consecutive states (current and next) of the road network, 2) the joint action taken at the current state, and 3) the reward corresponding to this transition. The default policy which is observed (used to collect observations) in the offline dataset (Π_Ω) is the Fixed-Time baseline as it is the most commonly used in the real-world [6] Prashanth and Bhatnagar (2010). IORL-Greedy refers to a variant where the Fixed-Time Π_Ω is replaced with the Greedy dynamic baseline.

OW

OW refers to the optional online fine-tuning of IORL described in 3.4.3.

[5] Coherent exploratory behavior is enforced using noisy layers as described in Devailly et al. (2022).
[6] Adaptive TSC represents less than 3% of traffic signals according to Florida Atlantic University

Training and hyperparameters

Training episodes all last 600 seconds (10 minutes). During training, performance is continuously monitored on the same set of validation road networks and early stopping is performed if aggregated performance on this set does not improve for a given number of gradient descent iterations. All trainable methods are trained using the same sets of training road networks and trips and are evaluated using the same sets of validation road networks and trips. Different random seeds are used for training sets and validation sets of a same experiment, and every time experiments are run (see § 3.5.1)

3.5.2 Experiment 1: Inductive Learning & Zero-Shot Transfer

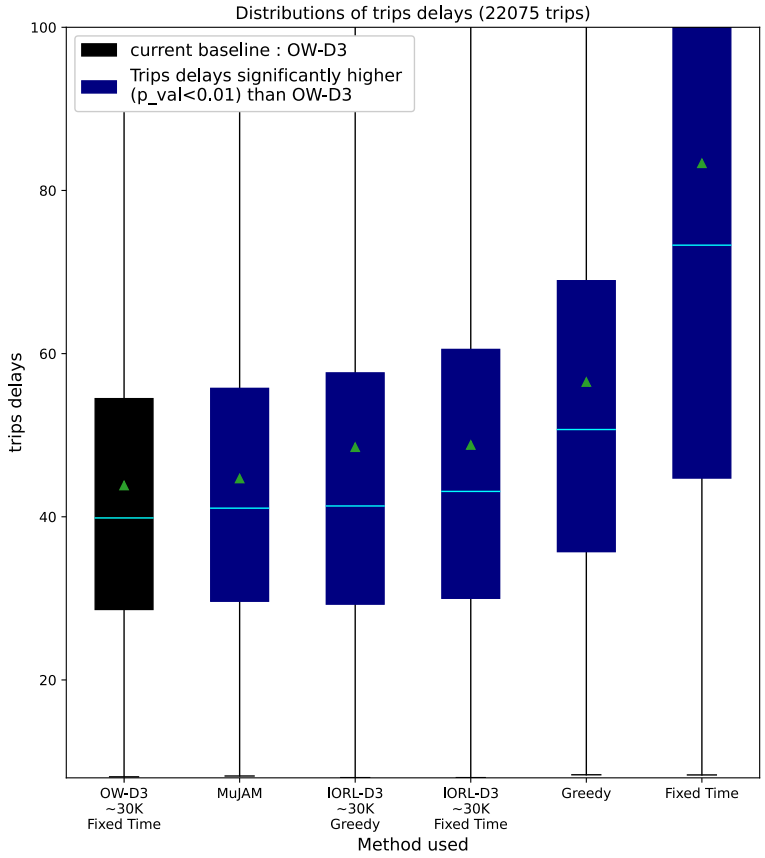

Figure 3.4: Distributions of total delay for the main baselines and proposed methods (OR and IORL)

The total delay for a given trip is the sum of all instantaneous delays (see §3.5.1) experienced by the corresponding vehicle during its trip. OW-D3 refers to the online-fine-tuned version (see § 3.4.3) of IORL-D3. T-tests are paired (per trip).

In this experiment, we evaluate the performance of zero-shot transfer to a set of randomly sampled road

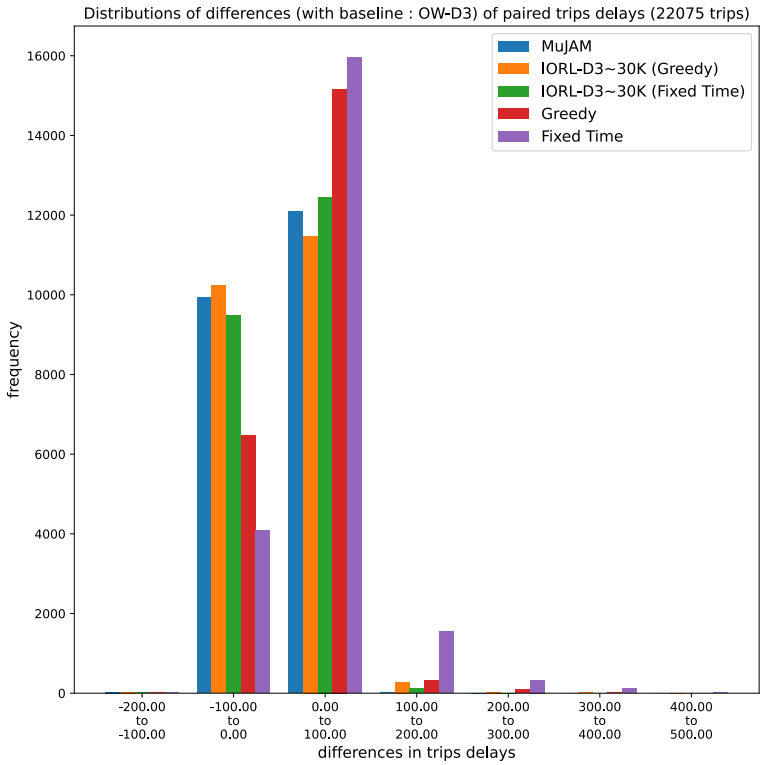

Figure 3.5: Distributions of paired (per trip) differences in delays between any given method and IORL (30K).

networks and traffic distributions[7] never seen in training. The random sets of road networks and trips used for evaluation are shared between all methods, trained or not. This enables comparison of paired trip delays across methods. Distributions of delays and paired t-tests are reported in Fig. 3.4 and distributions of differences of paired trip delays are reported in Fig. 3.5.

Fully Offline Learning

A first observation is that IORL, even though it never had the opportunity to validate assumptions by interacting with the environment, outperforms the fixed-time policy used to generate the offline dataset (i.e. the policy it observed during training). Lower trips delays (see Fig. 3.4) suggest that even trained with a very restricted policy that does not take the state of demand (i.e. traffic) into account, the learned simulator (or dynamics model) is able to generalize to OOD parts of the state space (i.e. combinations of intersections, traffic regimes and controller behaviors not observed in the training data), and that IORL does not degenerate or undesirably exploit inaccuracies of its learned simulator. In fact, by only observing the fixed-time policy, IORL is even able to significantly outperform the dynamic baseline and even reach a level of performance that is close to the one

[7] On average, every 4 seconds, a vehicle is added to a given road network,

of its online counterpart. This demonstrates that model-based ORL is a viable way to train inductive MARL policies.

Robustness to the observed policy

Another key observation is that training IORL with a dataset that was generated with the dynamic baseline enables reaching a similar level of performance (a similar distribution of trip delays) as reported in Fig. 3.4. As the dynamic baseline is drastically different from the fixed-time one and therefore represents a different region of the state space, this suggests IORL is robust to the policy and region of the state space it observes as part of offline training. As building the offline dataset using the dynamic baseline as Π_Ω yields better results than using the fixed time baseline as Π_Ω, it also suggests that IORL is able to exploit the more valuable regions of the state space reached by the latter policy. Applicability of offline RL to real-world scenarios might therefore not depend much on the historical policy Π_Ω.

Performance Vs. Fairness

Similarly to Devailly et al. (2021, 2022), we observe that methods that improve performance in this transfer setting also tend to distribute delays in a more equitable way. Fig. 3.5 shows that while domain-specific baselines may cause significant delays to some trips when compared to IORL, no trips are significantly delayed by IORL when compared to these baselines.

3.5.3 Experiment 2: Online fine-tuning from offline initialization

As described in 3.4.3, we propose a third (optional) step in the training of IORL. We initialize the online training of its online counterpart with parameters obtained by offline learning.

Avoiding catastrophic performance

As we can see in Fig. 3.6, the training of MuJAM (i.e. the online model-based equivalent of IORL) involves catastrophic initial performance caused by a random initialization of its parameters. In comparison, the online fine-tuning of IORL starts at a level that is already significantly above that of the policy it is trained from and can then improve its learned simulator and policy to reach the level of a fully-trained online instance of MuJAM. This suggests that OW is a promising way to further ease the social acceptability of the training of RL-ATSC policies in the real world, as it combines the best of both offline and online training paradigms.

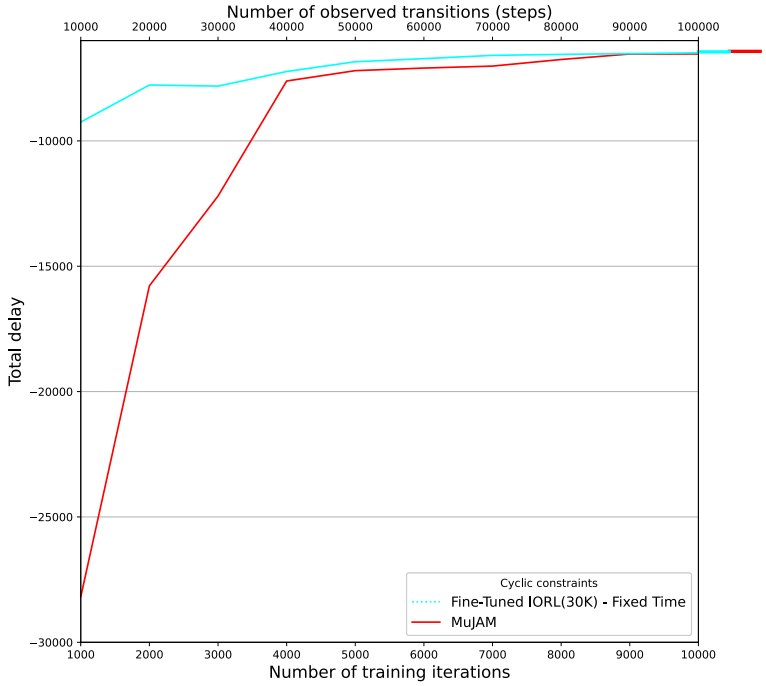

Figure 3.6: Average total delay (cumulative delay for all trips and steps in a given episode)

Thick lines at the right of the figure represent final performance (when training is completed). To evaluate online data efficiency and ensure a fair comparison between initializations, the ratio of observed transitions to gradient descent iterations is fixed, and performance is reported every 1K training steps.

3.5.4 Experiment 3: Ablation study

In this experiment, we study the influence of the size of the offline dataset on the performance of IORL. The main objective is to report on IORL's data efficiency. Information regarding the composition (size and variety) of all datasets is reported in Table. 3.5.4

Dataset	# observations	# networks	# intersections	Observed Policy
D1	1.2K	2	8.6(0.8)	Fixed Time
D2	5.4K	9	38.6(3.8)	Fixed Time
D3	30.0K	50	212.2(6.8)	Fixed Time
D4	100.2K	167	712.4(14.26)	Fixed Time

Table 3.1: Description of the datasets used for offline learning

All observed episodes last 10 minutes (600 steps). A given road network (and therefore a given intersection) is therefore observed for exactly this duration.

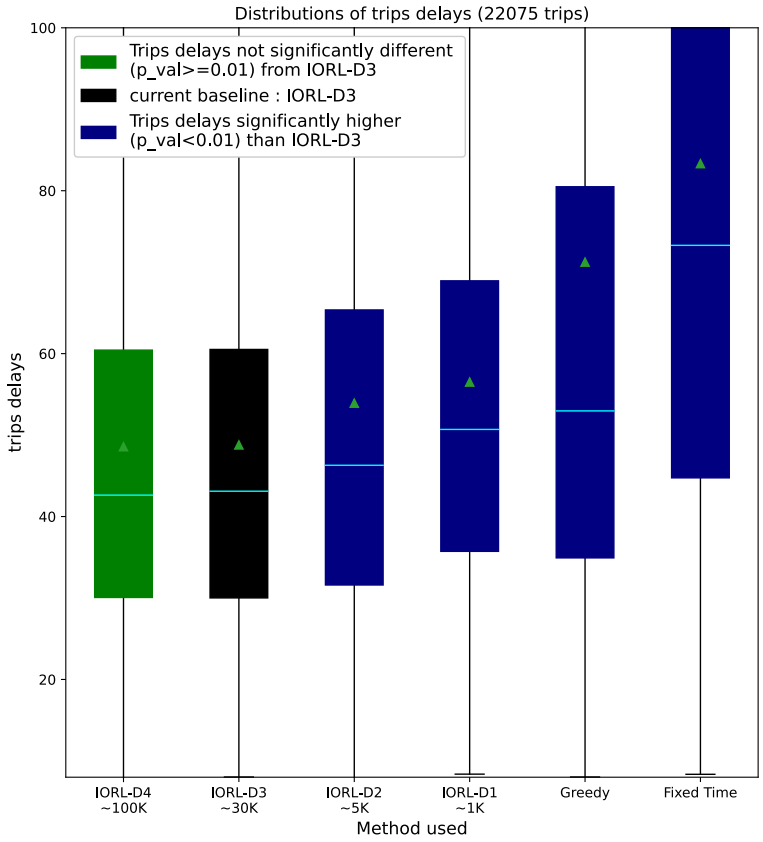

Figure 3.7: Distributions of total delay for IORL using different offline datasets. T-tests are paired (per trip).

Influence of the buffer size

As larger offline datasets include more transitions and a larger variety of settings, they should provide a more varied signal, which should represent a larger part of the state space *ceteris paribus* for IORL to exploit. As expected, the larger the offline dataset is, the better IORL performs in evaluation (the lower the delays) as reported in Fig. 3.7. However, a plateau seems to be reached at around 30K transitions as using a significantly larger buffer does not reduce delays significantly.

Data Efficiency

RL is notoriously data inefficient. In RL-ATSC, training of MFRL often requires millions of transitions. A key advantage of MuJAM, as an MBRL method, is its ability to improve data efficiency. We observe that this advantage is transferred to its offline counterpart, IORL, as 1) using only 1.2K transitions, IORL is able to reach lower delays than the Fixed-Time baseline 2) using only 5.4K transitions, IORL is able to reach lower delays than both domain-specific baselines (Fixed Time and Greedy), and 3) our default instantiation or IORL which uses only 30K transitions is able to further close the gap of performance between IORL and MuJAM (see

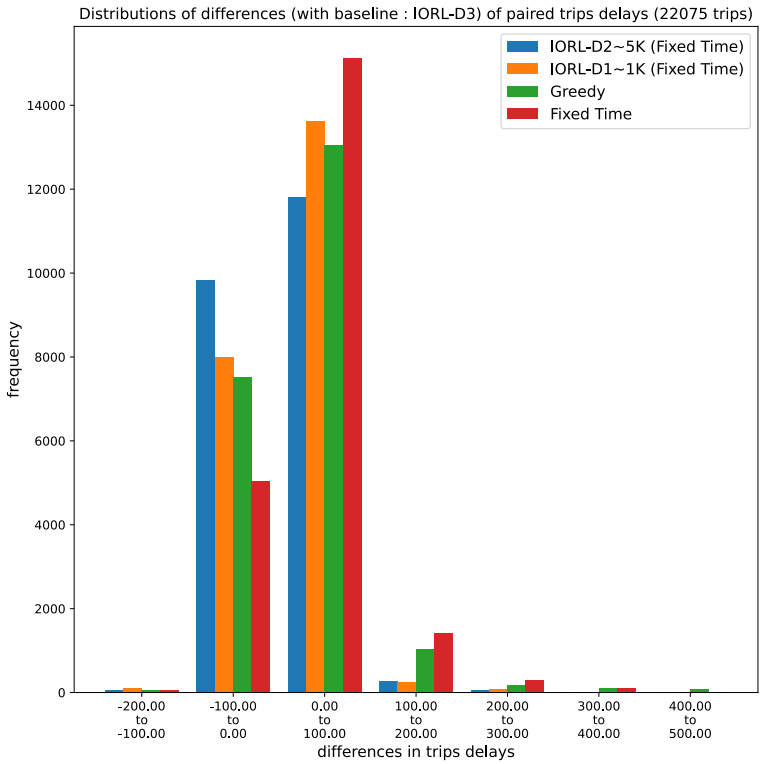

Figure 3.8: Distributions of paired (per trip) differences in delays between any given method and IORL(30K).

Fig. 3.4 & Fig. 3.7). MuJAM was proven to be more data efficient (in the online setting) than IGRL (Devailly et al., 2022) which was not surprising considering the fact that MuJAM is a model-based approach and IGRL is a model-free approach. Comparing the data efficiencies of MuJAM and IORL can be considered in two ways:

- Regarding online data efficiency (i.e. performance under a given number of interaction with the environment)

- Regarding offline data efficiency (i.e. performance under a given amount of historical data)

For online data efficiency,the comparison of OW and MuJAM reported in Fig. 3.6 shows that an improved initialization of parameters via OW enables reaching a higher level of performance faster than a random one. For offline data efficiency, training MuJAM on a dataset created using its own randomly initialized policy as the historical policy Π_Ω does not seem reasonable as 1) it would defeat the original purpose of offline learning (i.e. avoiding catastrophical performance during data collection), and 2) only low value states and behaviors (corresponding to the randomly initialized parameters) would be represented in the dataset making it hard to find and leverage any meaningful reinforcement signal.

3.5.5 Experiment 4: Real road networks

In this experiment, we evaluate the performance of IORL on zero-shot transfers to a large real road network to further asses robustness (generalizability to road network, traffic distribution, and OOD states) and scalability of offline training, in the same spirit as similar experiments performed by Devailly et al. (2021, 2022). Fig. 3.9 provides a visualization of the real road network used in this experiment, which corresponds to central Manhattan. Considering the very large number of intersections and the high computational cost of using a GCN on such a large network, in this particular scenario, β is set to 0. In other words, we perform no explicit planning and instead rely immediately on Φ to sample the joint action which will be applied in the road network at every step (as done in Devailly et al. (2021, 2022)). Considering the higher computational cost compared to small synthetic networks used in previous experiments, we evaluate traffic generated using 6 different seeds. As experiments are repeated 5 times (see § 3.5.1), the total number of evaluation instances per method is 30. Contrary to previous experiments, however, episodes now last 30 minutes (i.e. 3 times as long).

Model-based coordination at scale

Looking at Fig. 3.10, we first observe that MuJAM outperforms both domain-specific baselines which can scale to the same extent, as the cumulative delay experience by all vehicles in the network is drastically inferior. Similarly to Devailly et al. (2022), we, therefore, conclude that with the help of learned coordinated priors (i.e. Φ), MBRL is scalable and involves the same computational requirements as MFRL when no explicit planning is performed.

Offline robustness and scalability

We also observe that IORL outperforms both domain-specific baselines, with a performance (cumulative delay) somewhere in between the dynamic baseline and MuJAM which validates (as its ranking does not differ from the results of the previous experiments) the robustness and scalability of offline learnings in this zero-shot context.

Viability of fine-tuning

Similarly to what is observed in experiment 3: § 3.5.4, OW (i.e the online-fine-tuned version of IORL) performs similarly to MuJAM. We note that while in experiment 3: § 3.5.4, OW seems to slightly outperform MuJAM, in this evaluation MuJAM seems to slightly outperform OW. Given enough online interaction with the envi-

ronment, the initialization provided by OW aims to help with initial performance only. We do not expect any significant differences in asymptotic performances (i.e. after training) between the two methods.

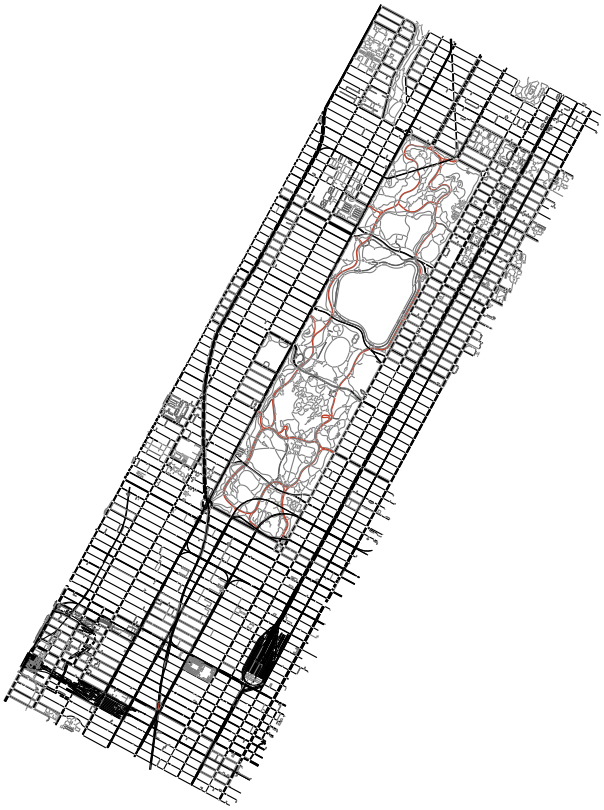

Figure 3.9: Central Manhattan road network.

3.6 Conclusion

We introduce IORL, the first MARL-ATSC method which alleviates the need for interaction with the environment during training. IORL is a model-based reinforcement learning approach that relies on learning a robust value-equivalent latent model of the environment from a dataset of observed transitions to simulate trajectories and plan ahead of decision-making. We demonstrate that, without ever interacting with the road-network environment, strictly observing either fixed-time or adaptive policies, IORL outperforms domain-specific baselines in zero-shot transfer tasks involving both road networks (real and synthetic) and traffic distributions never experienced in training and that it approaches the level of performance of its online equivalent. Such transferability translates into immediate scalability as trained policies can be applied to road networks of arbitrary sizes. Moreover, we show that this approach can be used as an initialization for online-fine-tuning to combine the advantages of both approaches (i.e. reaching the online level of performance while avoiding catastrophic and

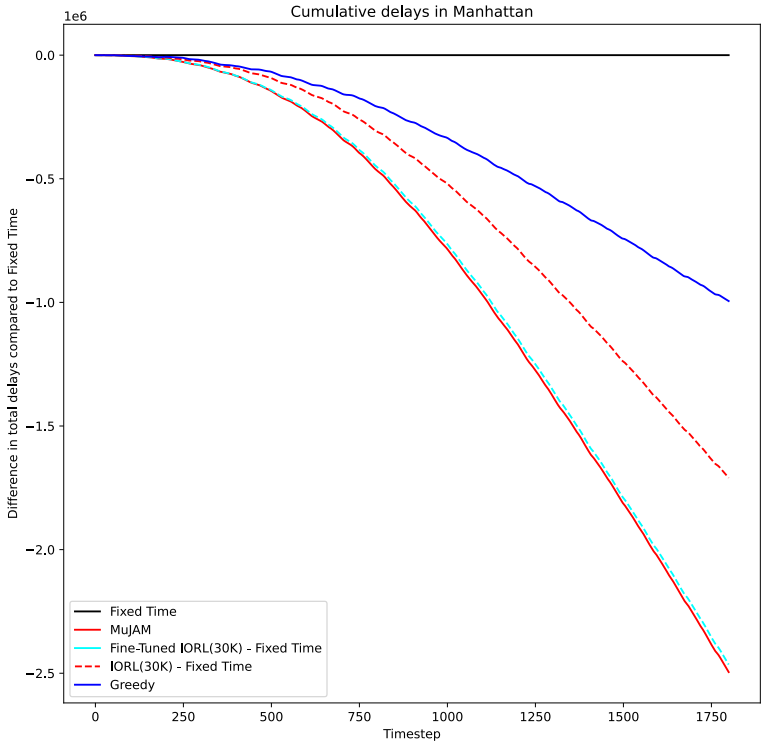

Figure 3.10: Manhattan - Cumulative difference in total delay (compared to the fixed-time policy)

The cumulative difference between the total delay experienced under a given method and the cumulative delay experienced under the fixed time policy.)

socially unacceptable initial performance). Finally, like its online counterpart, IORL also enables exploiting available data at its finest level of granularity as traffic can be represented at the level of individual vehicles. In line with recent developments, IORL pushes the generalization of RL-ATSC policies further. Inductive model-based graph RL methods can now not only generalize to the road network architecture, traffic distributions, and policy constraints but also generalize to OOD regions of the state space as demonstrated by the offline learning setting. We hope that this additional step toward fully-agnostic RL-ATSC contributes to its real-world applicability.

3.6.1 Future Work

Some ORL approaches explicitly compensate for uncertainty (Yu et al., 2020; Kidambi et al., 2020) by penalizing transitions for which the learned model of the dynamics is less likely to be reliable. An interesting avenue for future research would be to design an uncertainty-aware version of IORL to restrain model exploitation[8] and hopefully reach a fully-offline-performance that is closer to the level of its online equivalent, MuJAM.

[8]Model exploitation refers to the detrimental use of the dynamics model to reach OOD parts of the state space for which our value estimates might be overoptimistic.

Acknowledgment

This work was partially funded by the Natural Sciences and Engineering Research Council (NSERC), Fonds de Recherche du Québec: Nature et Technologies (FRQNT), and Fondation HEC Montréal.

References

Abdoos, M., Mozayani, N., and Bazzan, A. L. (2011). Traffic light control in non-stationary environments based on multi agent q-learning. In *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*, pages 1580–1585. IEEE.

Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135.

Aslani, M., Mesgari, M. S., and Wiering, M. (2017). 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*, 85:732–752.

Aslani, M., Seipel, S., Mesgari, M. S., and Wiering, M. (2018). Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran. *Advanced Engineering Informatics*, 38:639–655.

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

Barth, M. and Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171.

Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. (2020). Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Dai, X., Zhao, C., Li, X., Wang, X., and Wang, F.-Y. (2021). Traffic signal control using offline reinforcement learning. In *2021 China Automation Congress (CAC)*, pages 8090–8095. IEEE.

Devailly, F.-X., Larocque, D., and Charlin, L. (2021). Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Devailly, F.-X., Larocque, D., and Charlin, L. (2022). Model-based graph reinforcement learning for inductive traffic signal control. *arXiv preprint arXiv:2208.00659*.

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

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150.

Iša, J., Kooij, J., Koppejan, R., and Kuijer, L. (2006). Reinforcement learning of traffic light controllers adapting to accidents. *Design and Organisation of Autonomous Systems*, pages 1–14.

Khamis, M. A. and Gomaa, W. (2012). Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In *2012 11th international conference on machine learning and applications*, volume 1, pages 586–591. IEEE.

Khamis, M. A., Gomaa, W., and El-Shishiny, H. (2012). Multi-objective traffic light control system based on bayesian probability interpretation. In *2012 15th International IEEE conference on intelligent transportation systems*, pages 995–1000. IEEE.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kunjir, M. and Chawla, S. (2022). Offline reinforcement learning for road traffic control. *arXiv preprint arXiv:2201.02381*.

Kuyer, L., Whiteson, S., Bakker, B., and Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pages 47–66. Springer.

Nishi, T., Otaki, K., Hayakawa, K., and Yoshimura, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International conference on intelligent transportation systems (ITSC)*, pages 877–883. IEEE.

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. (2013). Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, volume 10, pages 1196–1201. IEEE.

Prashanth, L. and Bhatnagar, S. (2010). Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421.

Prashanth, L. and Bhatnagar, S. (2011). Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1640–1645. IEEE.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. (2022). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Rasheed, F., Yau, K.-L. A., Noor, R. M., Wu, C., and Low, Y.-C. (2020). Deep reinforcement learning for traffic signal control: A review. *IEEE Access*, 8:208016–208044.

Salkham, A. a. and Cahill, V. (2010). Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th international IEEE conference on intelligent transportation systems*, pages 531–538. IEEE.

Salkham, A. a., Cunningham, R., Garg, A., and Cahill, V. (2008). A collaborative reinforcement learning approach to urban traffic control optimization. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 560–566. IEEE.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Schrank, D., Eisele, B., Lomax, T., and Bak, J. (2015). 2015 urban mobility scorecard.

Schrank, D., Lomax, T., and Eisele, B. (2012). 2012 urban mobility report. *Texas Transportation Institute,[ONLINE]. Available: http://mobility. tamu. edu/ums/report*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B., et al. (2005). Reinforcement learning of traffic light controllers adapting to traffic congestion. In *BNAIC*, pages 216–223.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Van der Pol, E. and Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*.

Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., and Xiong, H. (2019). STMARL: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. *arXiv preprint arXiv:1908.10577*.

Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. (2019). Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2021). Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18.

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158.

Wiering, M., Veenen, J. v., Vreeken, J., and Koopman, A. (2004a). Intelligent traffic light control.

Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004b). Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 453–458. IEEE.

Xiong, Y., Zheng, G., Xu, K., and Li, Z. (2019). Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2289–2292.

Xu, L.-H., Xia, X.-H., and Luo, Q. (2013). The study of reinforcement learning for traffic self-adaptive control under multiagent markov game environment. *Mathematical Problems in Engineering*, 2013.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142.

Zang, X., Yao, H., Zheng, G., Xu, N., Xu, K., and Li, Z. (2020). Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1153–1160.

Zhang, Z., Yang, J., and Zha, H. (2019). Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651*.

Zheng, G., Xiong, Y., Zang, X., Feng, J., Wei, H., Zhang, H., Li, Y., Xu, K., and Li, Z. (2019). Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972.

General Conclusion

Reinforcement Learning for ATSC used to typically require a combination of 1) millions of observations (transitions) for training, 2) training from scratch for any modification of the application scenario, and 3) interaction with the real environment with socially questionable efficiency.

In this thesis, we have tackled data inefficiency in large scale RL-ATSC by introducing and enforcing increasing levels of generalization. By leveraging the flexible computational graph and parameter-sharing scheme of GCNs, the model discussed in the 1st chapter enables both generalization to network architectures and traffic distributions (which translates into massive scalability) as well as the exploitation of the finest level of granularity in available data (i.e. representing traffic at the vehicle level). With modern model-based RL and decentralized simulation, the model discussed in the 2nd chapter further improves performance and sample efficiency, enables a new dimension of generalization to policy constraints, and makes explicit coordination (in the form of joint action modeling) scalable for the first time. Finally, with offline RL, the method discussed in the 3rd chapter bypasses the need for interaction with the environment altogether and proposes a training scheme that enables reaching the performance of online training while avoiding catastrophic initial performance. With this last level of generalization (to regions of the ATSC state space which were not visited under the historical policy and therefore unseen during training), RL-ATSC enables the learning, with no interaction with the environment, of policies which immediately transfer (and therefore scale) to any new setting.

The combination of model-based RL and offline RL may greatly ease the contribution of third parties to ATSC as historical datasets (observations) can be used to both learn to simulate the road network environment and train new ATSC policies without requiring any actual control (which cities rightfully restrict).

In practice, thinking of the potential target user of such approaches, either a person in charge of deciding whether or not to try new ATSC approaches for a given destination scenario (e.g. intersection, district, city), or someone willing to provide the corresponding service, we suggest proceeding as follows: First, start by collecting data on a few intersections (ideally intersections with a variety of structures and traffic distributions) to build a rich training set. Then, the corresponding data can be used to train generalist simulators and policies

as described in Chapters 2 & 3. Finally, before applying a transferable policy to the full destination scenario, we suggest leveraging the generalist simulator beforehand to estimate the performances of both the previous (historical) and new policies and compare them. As it can help provide an estimated return on investment and reduce uncertainty, the goal of this additional step is to help key decision-makers (speed up adoption). After initial deployment and on top of monitoring, we then suggest to continue gathering observations under the new policy and use them to fine-tune the policy in a continual learning fashion.

In this work, we have demonstrated that policies can generalize and transfer on a variety of dimensions. It would be interesting to evaluate transferability from training in simulated environments to real-world environments. Also, as all the presented methods exploit vehicular data instead of lane level aggregations, when the expensive installation of sensors is not an option (as is the case in most parts of the world), geolocation data of vehicles or even individuals (e.g. using phones) may constitute promising alternatives. Studying the availability and reliability of such data in real-world applications of RL-ATSC would also be interesting. Such a level of granularity additionally enables flexible objective definition as policies can now be informed of the types of vehicles (e.g. ambulance, public transportation) and could learn to favor certain types of vehicles based on local preferences. Additionally, while only vehicles are represented as demand in this work, the flexibility of GCNs offers a straightforward way to include other types of entities as new types of nodes. Including alternative types of nodes for other road users such as pedestrians and cyclists to represent them at their *finest level of granularity* may help develop joint optimization of heterogeneous demand. Finally, while we implicitly focus on training and applying ATSC policies in distinct scenarios for which real-time data collection is readily available, offline evaluation of the potential of equipping a new intersection for which only structural (architecture) information is currently known (no demand information is available and no sensor has been installed, which is the case for the vast majority of intersections in the world) would provide a valuable prioritization tool for municipalities.

Bibliography

Abdoos, M., Mozayani, N., and Bazzan, A. L. (2011). Traffic light control in non-stationary environments based on multi agent q-learning. In *2011 14th International IEEE conference on intelligent transportation systems (ITSC)*, pages 1580–1585. IEEE.

Abdulhai, B., Pringle, R., and Karakoulas, G. J. (2003). Reinforcement learning for true adaptive traffic signal control. *Journal of Transportation Engineering*, 129(3):278–285.

Akkaya, I., Andrychowicz, M., Chociej, M., Litwin, M., McGrew, B., Petron, A., Paino, A., Plappert, M., Powell, G., Ribas, R., et al. (2019). Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*.

Arel, I., Liu, C., Urbanik, T., and Kohls, A. G. (2010). Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4(2):128–135.

Aslani, M., Mesgari, M. S., and Wiering, M. (2017). 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*, 85:732–752.

Aslani, M., Seipel, S., Mesgari, M. S., and Wiering, M. (2018). Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran. *Advanced Engineering Informatics*, 38:639–655.

Bakker, B., Whiteson, S., Kester, L., and Groen, F. C. (2010). Traffic light control by multiagent reinforcement learning systems. In *Interactive Collaborative Information Systems*, pages 475–510. Springer.

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

Barth, M. and Boriboonsomsin, K. (2008). Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 2058(1):163–171.

Bellman, R. (1957). A markovian decision process. *Journal of mathematics and mechanics*, pages 679–684.

Bingham, E. (2001). Reinforcement learning in neurofuzzy traffic signal control. *European Journal of Operational Research*, 131(2):232–241.

Busoniu, L., De Schutter, B., and Babuska, R. (2006). Decentralized reinforcement learning control of a robotic manipulator. In *2006 9th International Conference on Control, Automation, Robotics and Vision*, pages 1–6. IEEE.

Casas, N. (2017). Deep deterministic policy gradient for urban traffic light control. *arXiv preprint arXiv:1703.09035*.

Chen, C., Wei, H., Xu, N., Zheng, G., Yang, M., Xiong, Y., Xu, K., and Li, Z. (2020). Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421.

Chu, T., Wang, J., Codecà, L., and Li, Z. (2019). Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Dai, X., Zhao, C., Li, X., Wang, X., and Wang, F.-Y. (2021). Traffic signal control using offline reinforcement learning. In *2021 China Automation Congress (CAC)*, pages 8090–8095. IEEE.

Devailly, F.-X., Larocque, D., and Charlin, L. (2021). Ig-rl: Inductive graph reinforcement learning for massive-scale traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*.

Devailly, F.-X., Larocque, D., and Charlin, L. (2022). Model-based graph reinforcement learning for inductive traffic signal control. *arXiv preprint arXiv:2208.00659*.

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

El-Tantawy, S., Abdulhai, B., and Abdelgawad, H. (2013). Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (marlin-atsc): methodology and large-scale application on downtown toronto. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1140–1150.

Fortunato, M., Azar, M. G., Piot, B., Menick, J., Osband, I., Graves, A., Mnih, V., Munos, R., Hassabis, D., Pietquin, O., et al. (2017a). Noisy networks for exploration. *arXiv preprint arXiv:1706.10295*.

Fortunato, M., Blundell, C., and Vinyals, O. (2017b). Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

Genders, W. and Razavi, S. (2016). Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034.

Hasselt, H. V. (2010). Double Q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621.

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., Botvinick, M., Blundell, C., and Lerchner, A. (2017). Darla: Improving zero-shot transfer in reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1480–1490. JMLR. org.

Houli, D., Zhiheng, L., and Yi, Z. (2010). Multiobjective reinforcement learning for traffic signal control using vehicular ad hoc network. *EURASIP journal on advances in signal processing*, 2010(1):724035.

Hubert, T., Schrittwieser, J., Antonoglou, I., Barekatain, M., Schmitt, S., and Silver, D. (2021). Learning and planning in complex action spaces. In *International Conference on Machine Learning*, pages 4476–4486. PMLR.

Hunt, P., Robertson, D., Bretherton, R., and Winton, R. (1981). Scoot-a traffic responsive method of coordinating signals. Technical report.

Iša, J., Kooij, J., Koppejan, R., and Kuijer, L. (2006). Reinforcement learning of traffic light controllers adapting to accidents. *Design and Organisation of Autonomous Systems*, pages 1–14.

Khamis, M. A. and Gomaa, W. (2012). Enhanced multiagent multi-objective reinforcement learning for urban traffic light control. In *2012 11th international conference on machine learning and applications*, volume 1, pages 586–591. IEEE.

Khamis, M. A., Gomaa, W., and El-Shishiny, H. (2012). Multi-objective traffic light control system based on bayesian probability interpretation. In *2012 15th International IEEE conference on intelligent transportation systems*, pages 995–1000. IEEE.

Kidambi, R., Rajeswaran, A., Netrapalli, P., and Joachims, T. (2020). Morel: Model-based offline reinforcement learning. *Advances in neural information processing systems*, 33:21810–21823.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Koseki, S., Shazade, J., Golnoosh, F., David, R., Catherine, R., and Jean-Louis, D. (2022). Ai & cities risks, applications and governance. Technical report, UN-Habitat.

Kunjir, M. and Chawla, S. (2022). Offline reinforcement learning for road traffic control. *arXiv preprint arXiv:2201.02381*.

Kuyer, L., Whiteson, S., Bakker, B., and Vlassis, N. (2008). Multiagent reinforcement learning for urban traffic control using coordination graphs. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 656–671. Springer.

Levine, S., Kumar, A., Tucker, G., and Fu, J. (2020). Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Liang, X., Du, X., Wang, G., and Han, Z. (2019). A deep reinforcement learning network for traffic light cycle control. *IEEE Transactions on Vehicular Technology*, 68(2):1243–1253.

Liang, X., Guler, S. I., and Gayah, V. V. (2020). Traffic signal control optimization in a connected vehicle environment considering pedestrians. *Transportation Research Record*, 2674(10):499–511.

Lopez, P. A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Flötteröd, Y.-P., Hilbrich, R., Lücken, L., Rummel, J., Wagner, P., and Wießner, E. (2018). Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

Lowrie, P. (1990). Scats-a traffic responsive method of controlling urban traffic. *Sales information brochure published by Roads & Traffic Authority, Sydney, Australia*.

Mannion, P., Duggan, J., and Howley, E. (2015). Parallel reinforcement learning for traffic signal control. *Procedia Computer Science*, 52:956–961.

Mannion, P., Duggan, J., and Howley, E. (2016). An experimental review of reinforcement learning algorithms for adaptive traffic signal control. In *Autonomic Road Transport Support Systems*, pages 47–66. Springer.

Mikami, S. and Kakazu, Y. (1994). Genetic reinforcement learning for cooperative traffic signal control. In *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*, pages 223–228. IEEE.

Nishi, T., Otaki, K., Hayakawa, K., and Yoshimura, T. (2018). Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International conference on intelligent transportation systems (ITSC)*, pages 877–883. IEEE.

OECD (2021). General government spending (indicator),. Technical report, OECD.

Oh, J., Singh, S., and Lee, H. (2017a). Value prediction network. *Advances in neural information processing systems*, 30.

Oh, J., Singh, S., Lee, H., and Kohli, P. (2017b). Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2661–2670. JMLR. org.

Omidshafiei, S., Pazis, J., Amato, C., How, J. P., and Vian, J. (2017). Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2681–2690. JMLR. org.

Papoudakis, G., Christianos, F., Rahman, A., and Albrecht, S. V. (2019). Dealing with non-stationarity in multi-agent deep reinforcement learning. *arXiv preprint arXiv:1906.04737*.

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. (2013). Learning coordinated traffic light control. In *Proceedings of the Adaptive and Learning Agents workshop (at AAMAS-13)*, volume 10, pages 1196–1201. IEEE.

Prabuchandran, K., AN, H. K., and Bhatnagar, S. (2014). Multi-agent reinforcement learning for traffic signal control. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 2529–2534. IEEE.

Prashanth, L. and Bhatnagar, S. (2010). Reinforcement learning with function approximation for traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 12(2):412–421.

Prashanth, L. and Bhatnagar, S. (2011). Reinforcement learning with average cost for adaptive control of traffic lights at intersections. In *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 1640–1645. IEEE.

Prudencio, R. F., Maximo, M. R., and Colombini, E. L. (2022). A survey on offline reinforcement learning: Taxonomy, review, and open problems. *arXiv preprint arXiv:2203.01387*.

Puterman, M. L. (1990). Markov decision processes. *Handbooks in operations research and management science*, 2:331–434.

Rasheed, F., Yau, K.-L. A., Noor, R. M., Wu, C., and Low, Y.-C. (2020). Deep reinforcement learning for traffic signal control: A review. *IEEE Access*, 8:208016–208044.

Sadeghi, F., Toshev, A., Jang, E., and Levine, S. (2018). Sim2real viewpoint invariant visual servoing by recurrent control. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4691–4699.

Salkham, A. a. and Cahill, V. (2010). Soilse: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning. In *13th international IEEE conference on intelligent transportation systems*, pages 531–538. IEEE.

Salkham, A. a., Cunningham, R., Garg, A., and Cahill, V. (2008). A collaborative reinforcement learning approach to urban traffic control optimization. In *2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, pages 560–566. IEEE.

Schlichtkrull, M., Kipf, T. N., Bloem, P., Van Den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer.

Schrank, D., Eisele, B., Lomax, T., and Bak, J. (2015). 2015 urban mobility scorecard.

Schrank, D., Eisele, B., Lomax, T., et al. (2019). Urban mobility report 2019.

Schrank, D., Lomax, T., and Eisele, B. (2012). 2012 urban mobility report. *Texas Transportation Institute,[ONLINE]. Available: http://mobility. tamu. edu/ums/report*.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. (2020). Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al. (2016). Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. (2017). Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815*.

Steingrover, M., Schouten, R., Peelen, S., Nijhuis, E., Bakker, B., et al. (2005). Reinforcement learning of traffic light controllers adapting to traffic congestion. In *BNAIC*, pages 216–223.

Sutton, R. S. and Barto, A. G. (2018). *Reinforcement Learning: An Introduction*. The MIT Press, second edition.

Szepesvári, C. (2010). Algorithms for reinforcement learning. *Synthesis lectures on artificial intelligence and machine learning*, 4(1):1–103.

Van der Pol, E. and Oliehoek, F. A. (2016). Coordinated deep reinforcement learners for traffic light control. *Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016)*.

Varaiya, P. (2013). The max-pressure controller for arbitrary networks of signalized intersections. In *Advances in Dynamic Network Modeling in Complex Transportation Systems*, pages 27–66. Springer.

Wang, X., Ke, L., Qiao, Z., and Chai, X. (2019a). Large-scale traffic signal control using a novel multi-agent reinforcement learning. *arXiv preprint arXiv:1908.03761*.

Wang, Y., Xu, T., Niu, X., Tan, C., Chen, E., and Xiong, H. (2019b). STMARL: A spatio-temporal multi-agent reinforcement learning approach for traffic light control. *arXiv preprint arXiv:1908.10577*.

Wang, Z., Schaul, T., Hessel, M., Van Hasselt, H., Lanctot, M., and De Freitas, N. (2015). Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.

Wei, H., Xu, N., Zhang, H., Zheng, G., Zang, X., Chen, C., Zhang, W., Zhu, Y., Xu, K., and Li, Z. (2019a). Colight: Learning network-level cooperation for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1913–1922.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2019b). A survey on traffic signal control methods. *arXiv preprint arXiv:1904.08117*.

Wei, H., Zheng, G., Gayah, V., and Li, Z. (2021). Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation. *ACM SIGKDD Explorations Newsletter*, 22(2):12–18.

Wiering, M. (2000). Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pages 1151–1158.

Wiering, M., Veenen, J. v., Vreeken, J., and Koopman, A. (2004a). Intelligent traffic light control.

Wiering, M., Vreeken, J., Van Veenen, J., and Koopman, A. (2004b). Simulation and optimization of traffic in a city. In *IEEE Intelligent Vehicles Symposium, 2004*, pages 453–458. IEEE.

Xiong, Y., Zheng, G., Xu, K., and Li, Z. (2019). Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2289–2292.

Xu, L.-H., Xia, X.-H., and Luo, Q. (2013). The study of reinforcement learning for traffic self-adaptive control under multiagent markov game environment. *Mathematical Problems in Engineering*, 2013.

Yu, T., Thomas, G., Yu, L., Ermon, S., Zou, J. Y., Levine, S., Finn, C., and Ma, T. (2020). Mopo: Model-based offline policy optimization. *Advances in Neural Information Processing Systems*, 33:14129–14142.

Zang, X., Yao, H., Zheng, G., Xu, N., Xu, K., and Li, Z. (2020). Metalight: Value-based meta-reinforcement learning for traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 1153–1160.

Zhang, J., Shi, X., Xie, J., Ma, H., King, I., and Yeung, D.-Y. (2018). Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*.

Zhang, Z., Yang, J., and Zha, H. (2019). Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization. *arXiv preprint arXiv:1909.10651*.

Zheng, G., Xiong, Y., Zang, X., Feng, J., Wei, H., Zhang, H., Li, Y., Xu, K., and Li, Z. (2019). Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972.