Model-Based Graph Reinforcement Learning for Inductive Traffic Signal Control

FRANÇOIS-XAVIER DEVAILLY, DENIS LAROCQUE, AND LAURENT CHARLIN

Department of Decision Sciences, HEC Montreal, Montreal, QC H3T 2A7, Canada

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

This article has supplementary downloadable material available at https://doi.org/10.1109/OJITS.2023.3376583, provided by the authors.

ABSTRACT We introduce MuJAM, an adaptive traffic signal control method which leverages model-based reinforcement learning to 1) extend recent generalization efforts (to road network architectures and traffic distributions) further by allowing a generalization to the controllers’ constraints (cyclic and acyclic policies), 2) improve performance and data efficiency over related model-free approaches, and 3) enable explicit coordination at scale for the first time. 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 a recent transferable approach.

INDEX TERMS Adaptive traffic signal control, transfer learning, multi-agent reinforcement learning, joint action modeling, model-based reinforcement learning, graph neural networks.

I. INTRODUCTION

ADAPTIVE-TRAFFIC-SIGNAL-CONTROL (ATSC) aims at minimizing traffic congestion, which gives rise to a plethora of environmentally and socially harmful outcomes [1], [2], [3]. 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 [4], [5], [6]. Reinforcement learning techniques have exhibited greater efficiency and improved generalization ability compared to conventional traffic control methods [7]. 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 Sections II-A1 and II-A2). This lack of transferability, combined with the notorious data inefficiency of these methods (i.e., the need to gather a massive amount of experience to train RL policies) [8] 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 the coordination ability between agents (see Section II-A2). Regarding representation of the state of the road network, most neural network architectures used in RL do not enable dealing with changing numbers 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 a detailed (granular) representation of all entities. 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) [9], [10]. The RL approaches introduced in this work leverage inductive capabilities of GCNs to enable immediate transfer (i.e., with no additional training) to new intersections and traffic distributions, which translates into massive scalability, as introduced by [9]. This inductive learning and transfer setting is illustrated in Figure 1.
enable learning policies that generalize for unseen road-network topologies and traffic distributions [9], 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 [8] and these transferable methods are only compatible with cyclic constraints.

B. CONTRIBUTION

Model-based RL (MBRL), which explicitly models the dynamics of the environment, tends to both outperform model-free RL (MFRL) in complex planning tasks and enable better sample efficiency [20], [21]. We introduce joint action modeling with MuZero [21] (MuJAM), a new method for RL-ATSC which learns to simulate the ATSC environment (i.e., the road-network). Simulation is performed in a latent space and decentralized at the lanes’ level. The ability to predict the evolution of the road network’s state based on the joint action of all controllers is used to plan ahead of decision making in a coordinated fashion.

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

1) ACYCLIC-INDUCTIVE-GRAPH RL

RL-ATSC remains divided between learning cyclic and acyclic policies [8]. This comes at a cost for urban mobility planning as experimentation with different constraints systemically requires retraining using a different approach. Even transferable approaches are limited in this regard. The action space remains binary and identical across intersections only when the policy consists of switching to the next phase or remaining in the current phase (using cyclic constraints). For this reason, methods that aim to generalize over road-network architectures and traffic distributions are limited to using cyclic constraints [9].

MuJAM enables the use of transferable approaches with acyclic constraints. Policies under the latter, looser type of constraints, typically perform better.

2) 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.

3) GENERALIZATION TO ATSC CONSTRAINTS

Most reinforcement learning methods for adaptive-traffic-signal-control require training from scratch 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...
translates into better data efficiency as we gather more relevant signals 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. As this paper focuses on transferable policies, in line with the inductive evaluation setting introduced in [9], methods that require any form of tuning (e.g., parameter or hyperparameter) before being applied to a new context cannot be used in this setting. This disqualifies most trainable methods (except for IG-RL) and most heuristic-based methods. As a result, the chosen baselines in this paper are those that can be applied in this specific context.

C. OUTLINE

This work begins by connecting contributions to previous work in the field and providing background information on the machine learning methods used in the proposed methodology and the definition of the ATSC decision process. Next, the methodology's architecture, inference, and training procedure are explained in detail. Finally, the work presents numerical results from a set of inductive (transfer) experiments supporting our claims.

II. BACKGROUND

A. RELATED WORK

1) SCALABILITY

Even though some works propose to control a few intersections using a single agent [22], [23], 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 [11], [14], [15], [16], [17], [19], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33], [34], [35], [36], [37], [38], [39], [40]. A subset of these MARL approaches propose learning architectures specific to the ATSC setting. Reference [32] introduce a phase invariant design of the value function per phase using a shared model between phases, and [15] decentralize the value function per vehicle instead and leverages the estimates of a model of probabilities for a vehicle to locally transition to the next place in a queue or to the queue of a next road-lane. Other MARL approaches aim to represent and address more specific challenges such as varying weather conditions [17] or accidents [38]. One of the challenges of decentralization is the nonstationarity, for a given agent, of the dynamics of the environment as behaviors of other agents evolve during training. Reference [37] suggest adding a representation of neighbor policies to the state of a controller to marginalize this effect while [39] demonstrate that hybrid approaches can merge centralization and decentralization by enforcing consistency between global and local value functions.

Traditional MARL methods can control up to a few dozen intersections as demonstrated in [25]. Experiments of the corresponding scale have been conducted on real road network architectures corresponding, for instance, in Singapore [29] and Tehran [36] using simulators, and Toronto in the real world [26].

However, an increase in parameters leads to an explosion of costs (computational power, memory, and data requirements) with the number of intersections, which hinders further scalability. Parameter sharing in MARL methods [9], [32], [34], [41], [42], [43], [44] enables far greater scalability (up to a thousand controllers). Such methods usually rely on GGCNs [9], [34], [42], [43], [45]. Transfer learning is a promising way to scale further while drastically diminishing the training requirements. To limit the quantity of experience required to train under a new setting, [41] uses meta-learning, and [9], [45] achieves zero-shot transfer from small synthetic networks to massive real-world networks using the inductive capabilities of GGCNs.

2) COORDINATION

References [16], [27] enable communication between neighboring intersections, a form of implicit coordination (i.e., extending the local state spaces of agents). This is still the most common form of coordination used in the literature [9], [24], [33], [34], [39], [46]. Joint action modeling (i.e., explicit coordination) is challenging because of the resulting explosion of the action space. Some approaches do tackle joint action modeling on small road networks [19], [26], [28], [35]. For instance, [35] frames ATSC as a cooperative Markov game between adjacent controllers to address it.

B. 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, $T$, 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, $\pi$, which maximizes the expected accumulation of future value (discounted by a temporal factor $\gamma$): $V_{t} = \sum_{k=0}^{\infty} \gamma^{k}r_{t+k+1}$ where $r_{t}$ represents the reward experienced at time step $t$.

RL is a popular framework for solving MDPs via trial and error. When the learning of expected value ($V_{t}$) 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, when the dynamics are known in advance, they can be provided to the RL method. This, combined with
tree search (TS) planning has enabled AlphaZero to reach superhuman performance in complex planning tasks such as Chess, Shogi, and Go [20], [47]. However, in most real-world environments, dynamics are complex and unknown and must be learned. Learning transition functions (i.e., $s_{t+1} = T(s_t, a_t)$) in high dimensional state spaces is expensive and challenging. Until recently, the performance of MBRL approaches significantly lagged that of MFRL approaches in corresponding tasks [21].

Recent approaches include implicit feature selection and dimensionality reduction by planning in a value-equivalent-latent space instead of planning in the original state space of the MDP [21], [48]. 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 two main advantages:

- The dynamics model can ignore all dynamics unrelated to the task at hand (dynamics that do not influence value-related metrics). In the ATSC setting, for instance, one could imagine that to reduce congestion, simulating the general evolution of the flow and density of traffic on a lane is enough for optimal decision-making and that simulating the exact movements of individual cars might be excessive.
- The computations are projected into a latent vectorized space instead of predicting (usually reconstituting) high dimensional states such as images.

With these advances, MuZero [21] matches the performance of MBRL algorithms in their favored domains (complex planning tasks) and outperforms MFRL algorithms in their favored domains (problems involving high dimensional state spaces).

C. HETEROGENEOUS GRAPH CONVOLUTIONAL NETWORKS (GCNs)

Graph convolutional networks consist of stacking convolutional layers to recursively aggregate transformations of neighborhood information in a graph [49]. As a message-passing framework, GCNs enable learning and exploiting nonlinear patterns involving structural and semantic (nodes and edges features) signals. Their extensions to heterogeneous graphs (with multiple types of nodes and edges), Heterogeneous GCNs or HGCNs [50], work by applying the following message-passing equation on every node at every layer:

$$n_i^{(l)} = f \left( \sum_{e \in E, j \in N_e(i)} W_{le} \cdot n_j^{(l-1)} \right) \quad (1)$$

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 $W_{le}$ is the $l$th layer’s weight matrix for message propagation corresponding to a relation of type $e$. Parameters are typically learned by backpropagation of an error obtained using node embeddings. For simplicity, we will refer to HGCNs as GCNs for the rest of the paper. In RL environments for which states can be represented as graphs, GCNs can be used as state encoders. In the particular case of MBRL, however, predicting in a differentiable (learnable) way the evolution of a high-dimensional graph object whose structure and features change remains challenging.

D. ATSC DECISION PROCESS

Below, we describe modeling the ATSC problem as an MDP. The ATSC decision process defines the joint control of all TSCs in a given road network. We follow the formalism developed in [9].

1) STATES

Each state of the process comprises 1) connectivity: the status of all existing connections between lanes at intersections and the status of their respective controllers’ constraints, and 2) demand: the positions and speeds of vehicles. State information is encoded using vector representations that qualify nodes and edges of the GCN (see Section III-D). Every state has its own adjacency matrix.

2) ACTIONS

The joint action at a given time step is a combination of local actions, each consisting in the selection of a legal2 phase for each TSC. Once a phase is selected for the next phase, one of two things happens: If the selected phase is the current one, 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.

3) TRANSITION CONSTRAINTS

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

---

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

2A legal phase is a phase which can be selected by a TSC at a given time step according to this TSC’s constraints.
FIGURE 2. Training MuJAM - our proposed model-based RL approach—From top to bottom: The state ($s_t$) of the road network is represented as a graph object ($G_t$), and observations (transitions) are gathered via interaction with the environment (we use SUMO [51]) and stored in a replay buffer. The latent space dynamics model is trained via end-to-end backpropagation to predict value-related quantities (rewards, values and distributions over actions). Optimal actions $A^*$ and state values $v$ are determined via a tree search as shown in Figure 3.

4) REWARD
The reward is the negative sum of local queue lengths. 3

5) STEP
The duration of each step is one second.

6) EPISODE
Episodes can end after all trips are completed or after a given time.

III. MUJAM
We propose a method for RL-ATSC that builds upon inductive-graph-reinforcement learning (with GCNs). It models the dynamics of the ATSC environment using a latent space to combine MBRL with 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). Figures 2 & 3 provides a high-level description of the proposed method.

A. DYNAMICS MODEL
We now define the simulator model, which, given the current state of the road network and a sequence of actions, can simulate corresponding trajectories (in the latent space) and predict value-related metrics (see Fig. 3). The components of the dynamics model are listed in Section III-D.

1) ROAD-NETWORK STATE ENCODING
Our dynamics model is learned at the lane level. We obtain an initial state representation for all lanes (a vector obtained using a GCN aggregating information about structure, connectivity, traffic, etc.) for planning. The structure of a given state representation (i.e., the rules for building nodes, edges and their corresponding features) are detailed in Section III-D1.

$^3$A vehicle is counted in a queue if it is stopped at a maximum of 50 meters of the intersection.

2) TRAFFIC VS. CONNECTIVITY DYNAMICS
For the simulation of a given transition, we distinguish two parts of environmental dynamics in the ATSC MDP.

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

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

Traffic dynamics have to be learned to be simulated. MBRL with a learned dynamics model usually involves learning to predict the next state conditioned on the current state and a given action. Simulating the evolution of a complex graph object is nontrivial II-C. The task is even more challenging as planning involves simulating many rollouts on large and complex (heterogeneous and dynamic) graphs and must be performed in real time for ATSC. For this reason, we use MuZero [21], 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 provided to the model (learned simulator) 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).

3) SIMULATING TRANSITIONS
After encoding the current state of the road network, to model a transition given an action, we 1) manually update features related to connectivity dynamics in the
GCN (see Section III-A2), 2) use the GCN to update the representations of all lanes 3) predict local rewards \( r \) and local 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 with a given transition for the entire road network are then simply obtained by summing all lane-level estimates.

**B. PLANNING WITH A TREE SEARCH**

The following part describes how, using the simulator model at any given state/time step, the TS is performed to select a joint action and evaluate the expected future value of the state. The TS (see Fig. 3) can be performed according to a predefined search budget \( \beta \) (a hyperparameter). If \( \beta = 0 \), we sample an action per intersection using local priors (distributions) obtained from the representation of the current state (i.e., original node)(see Section III-C). The combination of these local actions constitutes the joint action. Alternatively, if \( \beta > 0 \), \( \beta \) trajectories will be simulated as part of the TS (trajectories must not exceed the maximum search depth \( \delta \)).

1) GROWING THE TREE

Inspired by the approach proposed in [52], we perform guided sampling of the joint action space. As the number of possible actions 1) varies dramatically across time steps (because of constraints) and across road networks and 2) explodes exponentially with the number of intersections, we propose the following progressive widening strategy for the TS: If \( \text{samples}_n < \text{binom}(n/C1^2, \text{visits}_n^3) \) (where \( n \) is the current node, \( \text{samples} \) is the number of sampled joint actions at the current node, \( \text{actions} \) is the number of selectable actions at the current node, \( \text{visits} \) is the number of times the current node has been visited in the current TS, and \( C1, C2, C3 \) are hyperparameters (see Section III), then a new joint action is sampled (see Section III-C). 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 ensures that the search width increases (i.e., more actions are sampled) with the action space’s size, ceteris paribus.

2) 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 [47]:

\[
\arg\max_a \left( Q(s, A) + c(s) \cdot \Phi(s, A) \right) = \frac{P_{\text{visits}}}{1 + C_{\text{visits}}} [\sqrt{P_{\text{visits}}} - 1] + C_{\text{visits}}
\]

with \( c(s) = \log(P_{\text{visits}} + C_{\text{base}}) / C_{\text{base}} + C_{\text{init}} \) if \( A \) is the joint action, \( P_{\text{visits}} \) is the number of times the parent node was visited, \( C_{\text{visits}} \) is the number of times the child node (i.e., corresponding action) was visited, \( \Phi(s, A) \) is the joint prior (i.e., the probability of sampling the given joint action obtained by the product of local probabilities), and \( C_{\text{base}}, C_{\text{init}} \) are hyperparameters. Additionally, as done in [21], we add Dirichlet noise to local priors of the source node to favor exploration at the root of the TS.

**C. PRIORITIZED SAMPLING OF SIMULATED ROLLOUTS**

In a coordinated setting, the action space becomes intractable as the number of intersections increases. The total number of actions at a given time step is \( \prod_{u=1}^{n} |A_u| \) where \( n \) is the number of intersections and \( |A_u| \) is the size of the action space for intersection \( u \) 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. To illustrate, this would mean that the first node in the planning part of Fig. 3 could be immediately followed by (linked to) more than a million children nodes. We instead sample the action space parsimoniously. We adapt the TS sampling strategy proposed in Sampled MuZero [52] to a coordinated setting. Namely, we train a prior multilayer perceptron (MLP) \( \phi \) 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).

1) LOCAL PRIORS

To sample local actions, we use local (for each TSC) multinomial distributions with parameters:

\[
p_{u,i} = \exp(\logit_i) / \sum_{j \in A_u} \exp(\logit_j),
\]

where \( p_i \) is taken to be the probability of sampling action \( i \). \( A_u \) is the set of actions (i.e., legal phases) for the TSC (or intersection) \( u \) at the current state, and \( \logit_i \) is a logit corresponding to action \( i \) and obtained via our model (see Section III-D2).

2) BEST COORDINATED SET

A TS is used to identify the best-coordinated set of actions \( A^* \) 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:

\[
\sum_{u \in \text{Lanes}} r_i + \gamma v_i,
\]

where \( L \) is the set of all Lanes, \( r_i \) is the predicted reward for the transition \( T(s, A) \rightarrow s' \), and \( v_i \) is the estimated (via TS) long term value for state \( s' \) (i.e., the state reached after taking joint action \( A \)).

3) IMPROVING PRIORS

To train \( \phi \), local actions are extracted from the best-coordinated set \( A^* \) and become training targets. As training progresses, local priors increasingly favor sampling 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 neighboring information (i.e., using representations obtained from the GCN) makes our model-based approach scalable, as it can be used with an arbitrary planning budget. In the extreme case of a null budget (i.e., no TS), actions can still be selected by acting greedily for
TABLE 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                              |

D. ARCHITECTURE
With $G_t$, the graph representation of state $s_t$, the simulator (dynamics) model consists of the following parametric models:

- **Representation:** $e(z_t | G_t)$
- **Transition:** $g(z_t | z_{t-1}, A_{t-1})$
- **Reward:** $w(r_t | z_t)$
- **Value:** $q(v_t | z_t)$
- **Prior:** $\phi(\Phi_t | z_t)$

Road-network objects are modeled as nodes in the GCN. The four types of nodes are vehicle, lane, connection and phase. The main difference between the GCN architectures of IG-RL [9] 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 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.

1) GRAPH FEATURES
Tables 1,2 summarize node and edge features. **Current speed** represents the normalized current speed of a vehicle (between 0 and 1, expressed as a proportion of the lane). **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** indicates 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** indicates if the connection will be opened on the next phase of the cycle. **Number of switches to open** indicates, in the case of cyclic constraints, how many switches the connection will be opened (i.e., green). **Next opening has priority** 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.

2) COMPUTATIONAL GRAPH & PROPAGATION RULES
Algorithms (pseudocodes) detailing elements in the rest of the current section are provided as supplementary material.

**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, to contextualize this representation, messages are propagated $K$ (a hyperparameter) times along L-to-L edges.

**Dynamics:** Based on a latent representation $z_t$ of the road network, and given a joint action $A_t$, to obtain the 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. 1, 2). This informs the model of the evolution of connectivity dynamics (i.e., the

---

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.

8 Every GCN layer uses its own set of parameters.
during training, instead of using independent noise at each transition. Furthermore, to enforce exploration in the environment, we continuously reanalyze episodes in the replay buffer to refresh TS estimates using updated parameters. This last step aims to model traffic dynamics between lanes based on the previous representation $z_{t}$ and updated connectivity features.

**Value-related-metrics computation:** For all lanes, local reward ($r$) and local value ($v$) estimates for a given time step are obtained by feeding the 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 an MLP ($\phi$) and then normalized (per intersection) using a softmax to obtain probabilities (i.e., the predicted local action distribution, used as prior during planning).

Fig. 4, 5, and 6 illustrates and describes, under a cyclic policy for simplicity, a slice of the computational graph (i.e., how MuJAM models dynamics and predicts value-related quantities and coordinated priors).

### 3) 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.

### E. TRAINING

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

#### 1) IMPROVING DATA EFFICIENCY

Targets for $v$ and $\Phi$ are built using the result of TS when interacting with the environment. However, as proposed in [21], 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 [53] 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. Consistent with [9], coherent exploration improved training speed and asymptotic performance in our experiments.

### IV. EXPERIMENTS

In our experiments, we use a zero-shot transfer setting proposed by [9] and inspired by other transfer settings in RL [54], [55].

We compare the performance 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.
A. GENERAL SETUP

1) NETWORK GENERATION
Road networks are randomly generated (i.e., random connectivity and structure) using SUMO [51]. 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. TSC Programs are formulated through the SUMO software. Phases of TSC programs correspond to permissible states at a specific intersection. Unlike being confined to simplified axis links like North-South or East-West, programs can encompass various configurations as described in [9].

2) TRAFFIC GENERATION
Traffic generation (i.e., the generation of trips) follows asymmetric 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.

3) EVALUATION
We use the same networks and trips to evaluate all methods. Performance is evaluated using the instantaneous delay, defined as $d_t = \sum_{v \in V} (s_v - s^*_v)/s^*_v$, where $s_v^*$ is the maximum speed a vehicle can legally reach considering the lane it is on, $s_v$ is the vehicle speed at time step $t$, $V$ 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 $t$.

4) ROBUSTNESS
We run all experiments five 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.

B. BASELINES

1) FIXED TIME
The Fixed Time baseline follows SUMO default cyclic programs, which are defined based on the structure of a given intersection.

2) MAX-MOVING-CAR HEURISTIC (GREEDY)
This dynamic baseline aims to enable the movement of a maximum number of vehicles at any given time. To do so, it switches the 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.

3) IG-RL
This is the only learnt baseline for which zero-shot transfer to new road architectures and traffic distributions is achievable. It consists of a GCN, which gathers demand at the vehicle level and is trained via RL using noisy parameters for exploration [9].

4) MUJAM
For MuJAM, the default and constraint-agnostic instantiation of our approach, 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, all intersections used in training have cyclic constraints, while for MuJAM-A, all intersections used in training have acyclic constraints. 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 [21], which does not periodically reanalyze transitions to refresh training targets with updated parameters. Finally, for MuIM (independent modeling), a TS is performed independently for each intersection instead of performing a joint TS for the entire road network (i.e., joint-action-modeling is removed).

5) TRAINING AND HYPERPARAMETERS
Training episodes last 10 minutes (simulation time). Performance is continuously evaluated during training on a separate set of 10 road networks. If a method does not reach a higher average reward on this set for $\omega$ steps, early stopping is enforced (i.e., training ends). For IG-RL, hyperparameters are the same as those used in the original paper [9]. We now refer to all methods based on MuZero as Mu methods. For Mu methods, hyperparameters are listed in Tables 3 and are either chosen according to other works or based on computational constraints.

C. 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 compute differences in delays. We report paired t-tests when compared to the best-performing method (MuJAM with acyclic constraints) in Fig. 7.
1) MODEL-BASED INDUCTIVE LEARNING

*Mu methods* enable lower trip delays (lower means, medians and quartiles) compared to all baselines, as shown in Fig. 7. These methods also lead to higher asymptotic performance (i.e., lower average total delay after training is completed) (see Fig. 8). With the only notable exception of MuJAM-NR-C, 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, and Atari games) can also be leveraged in ATSC’s challenge.

2) CONSTRAINT AGNOSTICISM

First, we observe that methods under acyclic constraints outperform methods under cyclic constraints as 1) the distribution of trip delays (means, medians and quartiles) is the lowest for MuJAM with acyclic constraints (see Fig. 7) and 2) both data efficiency and asymptotic performance are better for MuJAM-A than MuJAM-C (see Fig. 8). Although this result is not surprising as the acyclic setting is less constraining than the cyclic setting, it is the first time an acyclic approach is transferable (with zero-shot transfer) to new road-network and traffic distributions. Furthermore, evaluating our hybrid approach, MuJAM, under cyclic and acyclic constraints, yields similar asymptotic performance compared to corresponding specialist formulations (MuJAM-A and MuJAM-C). As shown in Fig. 8, 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 a variety of road-network architectures, traffic distributions, and behavioral constraints simultaneously.

3) COORDINATION

The ablation of coordination ability (i.e., MuIM-C) leads to a lower asymptotic performance when compared to MuJAM-C (see Fig. 8). 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).

4) 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. 8). This ablation makes MuJAM-NNL-C the worst performing *Mu method*. This demonstrates that exploration coherence is key to data efficiency and performance in RL-ATSC, confirming what was reported in [9]. Removing the ability to refresh training targets with updated parameters (i.e., MuJAM-NR-C) negatively impacts asymptotic performance (see Fig. 8). However, 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.

D. 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 heavy traffic\(^9\) (tens of thousands of vehicles with asymmetric traffic distributions) to evaluate the extent of generalizability and scalability of approaches studied in IV-C. Combining new network structures (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 enables running this large-scale experiment with learned methods, as training on such a large network (particularly if using a different set of parameters 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 traffic is already dense\(^10\) before evaluation starts for all approaches. At this point, an episode lasts one hour (density continuously increases during that 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 Section IV-C, we only evaluate methods on six instances of randomly generated traffic per run (for a total of 30 instances as we repeat all experiments five times as described in Section IV-A4).

1) RESULTS

A first observation is that with MuJAM, we can use—for the first time—a coordinated RL-ATSC approach trained with explicit joint action modeling (not only the ability to communicate) in such a large setting. To enable this, we only use coordinated priors (see Section III-C) to select an action and ignore planning. In other words, we use a null search budget for planning \(\beta=0\). 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. 9 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 outperforms all other approaches. We also note that IG-RL underperforms the dynamic baseline after a certain time. We hypothesize that IG-RL suffers from a gradual shift in distribution. As density increases in the road network with time, observed traffic distributions get further away from distributions used during training.

V. 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 type 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 [9], 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.

\(^9\)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. The network used in this section can be found at github.com/FXDevailly/IGRL/Manhattan.net.xml.

\(^{10}\)At this point, 18,000 vehicles have been inserted on average.
MuJAM opens a path for 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 of 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 be less dependent on the online control of TSCs and easier to train using offline data. Transferable-Offline RL-ATSC would enable safe and cost-efficient training on offline data and zero-shot transferability across networks, traffic, and constraints.

REFERENCES

[1] D. Schrank, T. Lomax, and B. Eisele, 2012 Urban Mobility Report, Texas Transp. Inst., Bryan, TX, USA, 2012. [Online]. Available: http://mobility. tamu. edu/ums/Rep.

[2] M. Barth and K. Boriboonsomsin, “Real-world carbon dioxide impacts of traffic congestion,” Transp. Res. Rec., vol. 2058, no. 1, pp. 163–171, 2008.

[3] D. Schrank, B. Eisele, T. Lomax, and J. Bak, 2015 Urban Mobility Scorecard, Texas A&M Transp. Inst., Bryan TX, USA, 2015.

[4] P. Hunt, D. Robertson, R. Bretherton, and R. Winton, SCOOT-A Traffic Responsive Method of Coordinating Signals, Transp. Res. Lab, Private Co., Bracknell, U.K., 1981.

[5] P. Lowrie, Scots-a Traffic Responsive Method of Controlling Urban Traffic: Sales Information Brochure, Roads Traffic Authority, Sydney, NSW, Australia, 1990.

[6] A. Haydari, M. Zhang, and C.-N. Chuah, “Adversarial attacks and defense in deep reinforcement learning (DRL)-based traffic signal controllers,” IEEE Open J. Intell. Transp. Syst., vol. 2, pp. 402–416, 2021.

[7] M. Noaen et al., “Reinforcement learning in urban network traffic control: A systematic literature review,” Expert Syst. Appl., vol. 199, Aug. 2022, Art. no. 116830.

[8] H. Wei, G. Zheng, V. Gayah, and Z. Li, “A survey on traffic signal control methods,” 2020, arXiv:1904.08117.

[9] F.-X. Devailly, D. Larocque, and L. Charlin, “IG-RL: Inductive graph reinforcement learning for massive-scale traffic signal control,” IEEE Trans. Intell. Transp. Syst., vol. 23, no. 7, pp. 7496–7507, Jul. 2022.

[10] Y. Y. A. Carballo, R. Li, and K. Takeda, “RSG-GCN: Predicting semantic relationships in urban traffic scene with map geometric prior,” IEEE Open J. Intell. Transp. Syst., vol. 4, pp. 244–260, 2023.

[11] Y. Xiong, G. Zheng, K. Xu, and Z. Li, “Learning traffic signal control from demonstrations,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 2289–2292.

[12] A. Salkham and V. Cahill, “SoIsie: A decentralized approach to optimization of fluctuating urban traffic using reinforcement learning,” in Proc. 13th Int. IEEE Conf. Intell. Transp. Syst., 2010, pp. 531–538.

[13] M. A. Khamis, W. Gomaa, and H. El-Shishiny, “Multi-objective traffic light control system based on Bayesian probability interpretation,” in Proc. 15th Int. IEEE Conf. Intell. Transp. Syst., 2012, pp. 995–1000.

[14] M. Wiering, “Multiagent reinforcement learning for traffic light control,” in Proc. 17th Int. Conf. Mach. Learn. (ICML), 2000, pp. 1151–1158.

[15] M. Wiering, J. V. Veenen, J. Vreeken, and A. Koopman, “Intelligent traffic light control,” Inst. Inf. Comput. Sci., Univ. Utrecht, Utrecht, The Netherlands, Rep. UI-CS-2004-029, 2004.

[16] M. Steingrover, R. Schouten, S. Peelan, E. Nijhuis, and B. Bakker, “Reinforcement learning for traffic light controllers adapting to traffic congestion,” in Proc. BNAIC, 2005, pp. 216–223.

[17] M. A. Khamis and W. Gomaa, “Enhanced multiagent multi-objective reinforcement learning for urban traffic light control,” in Proc. 11th Int. Conf. Mach. Learn. Appl., 2012, pp. 586–591.

[18] M. Wiering, J. Vreeken, J. Van Veenen, and A. Koopman, “Simulation and optimization of traffic in a city,” in Proc. IEEE Intell. Veh. Symp., 2004, pp. 453–458.

[19] L. Kuyer, S. Whiteson, B. Bakker, and N. Vlassis, “Multiagent reinforcement learning for urban traffic control using coordination graphs,” in Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases, 2008, pp. 656–671.

[20] D. Silver et al., “Mastering chess and shogi by self-play with a general reinforcement learning algorithm,” 2017, arXiv:1712.01815.

[21] J. Schrittwieser et al., “Mastering Atari, go, chess and shogi by planning with a learned model,” Nature, vol. 588, no. 7839, pp. 604–609, 2020.

[22] N. Casas, “Deep deterministic policy gradient for urban traffic light control,” 2017, arXiv:1703.09035.

[23] L. Prashanth and S. Bhatnagar, “Reinforcement learning with average cost for adaptive control of traffic light at intersections,” in Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC), 2011, pp. 1640–1645.

[24] S. El-Tantawy and B. Abdulhai, “An agent-based learning towards decentralized and coordinated traffic signal control,” in Proc. 13th Int. IEEE Conf. Intell. Transp. Syst., 2010, pp. 665–670.

[25] M. Abdoos, N. Mozayan, and A. L. C. Bazzan, “Traffic light control in non-stationary environments based on multi agent Q-learning,” in Proc. 14th Int. IEEE Conf. Intell. Transp. Syst. (ITSC), 2011, pp. 1580–1585.

[26] S. El-Tantawy, B. Abdulhai, and H. Abdelgawad, “Multiagent reinforcement learning for integrated network of adaptive traffic signal controllers (MARLIN-ATSC): Methodology and large-scale application on downtown Toronto,” IEEE Trans. Intell. Transp. Syst., vol. 14, no. 3, pp. 1140–1150, Sep. 2013.

[27] A. A. Salkham, R. Cunningham, A. Garg, and V. Cahill, “A collaborative reinforcement learning approach to urban traffic control optimization,” in Proc. IEEE/WIC/ACM Int. Conf. Web Intell. Agent Technol., 2008, pp. 560–566.

[28] E. van der Pol and F. A. Oliehoek, “Coordinated deep reinforcement learners for traffic light control,” in Proc. Learn., Inference Control Multi-Agent Syst. (NIPS), 2016, pp. 21–38.

[29] P. Balaji, X. German, and D. Srivivasan, “Urban traffic signal control using reinforcement learning agents,” IET Intell. Transp. Syst., vol. 4, no. 3, pp. 177–188, 2013.

[30] M. Aslani, S. Seipel, M. S. Mesgari, and M. Wiering, “Traffic signal optimization through discrete and continuous reinforcement learning with robustness analysis in downtown tehran,” Adv. Eng. Inform., vol. 38, pp. 639–655, Oct. 2018.

[31] T. T. Pham et al., “Learning coordinated traffic light control,” in Proc. Adaptive Learn. Agents Workshop (AAMAS), 2013, pp. 1196–1201.

[32] G. Zheng et al., “Learning phase competition for traffic signal control,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 1963–1972.

[33] T. Nishi, K. Otaki, K. Hayakawa, and T. Yoshimura, “Traffic signal control based on reinforcement learning with graph convolutional neural nets,” in Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC), 2018, pp. 877–883.

[34] H. Wei et al., “CoLight: Learning network-level cooperation for traffic signal control,” in Proc. 28th ACM Int. Conf. Inf. Knowl. Manage., 2019, pp. 1913–1922.

[35] L.-H. Xu, X.-H. Xia, and Q. Luo, “The study of reinforcement learning for traffic self-adaptive control under multiagent Markov game environment,” Math. Problems Eng., vol. 2013, Jan. 2013, Art. no. 962869.

[36] M. Aslani, M. S. Mesgari, and M. Wiering, “Adaptive traffic signal control with actor-critic methods in a real-world traffic network with different traffic disruption events,” Transp. Res. Part C, Emerg. Technol., vol. 85, pp. 732–752, Dec. 2017.

[37] T. Chu, J. Wang, L. Codecà, and Z. Li, “Multi-agent deep reinforcement learning for large-scale traffic signal control,” IEEE Trans. Intell. Transp. Syst., vol. 21, no. 3, pp. 1086–1095, Mar. 2020.

[38] J. Isa, J. Kooij, R. Koppejan, and L. Knijjer, “Reinforcement learning of traffic light controllers adapting to accidents,” Design Org. Auton. Syst., Univ. Amsterdam, Amsterdam, The Netherlands, 2006.

[39] Z. Zhang, J. Yang, and H. Zha, “Integrating independent and centralized multi-agent reinforcement learning for traffic signal network optimization,” 2019, arXiv:1909.10651.

[40] P. Mannion, J. Duggan, and E. Howley, “An experimental review of reinforcement learning algorithms for adaptive traffic signal control,” in Autonomic Road Transport Support Systems. Berlin, Germany: Springer, 2016, pp. 47–66.
[41] X. Zang, H. Yao, G. Zheng, N. Xu, K. Xu, and Z. Li, “MetaLight: Value-based meta-reinforcement learning for traffic signal control,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 1153–1160.

[42] C. Chen et al., “Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control,” in Proc. AAAI Conf. Artif. Intell., 2020, pp. 3414–3421.

[43] Y. Wang, T. Xu, X. Niu, C. Tan, E. Chen, and H. Xiong, “STMARL: A spatio-temporal multi-agent reinforcement learning approach for traffic light control,” 2020, arXiv:1908.10577.

[44] H. Wang, H. Chen, Q. Wu, C. Ma, and Y. Li, “Multi-intersection traffic optimisation: A benchmark dataset and a strong baseline,” IEEE Open J. Intell. Transp. Syst., vol. 3, pp. 126–136, 2022.

[45] T. Shi, F.-X. Devailly, D. Larocque, and L. Charlin, “Improving the generalizability and robustness of large-scale traffic signal control,” IEEE Open J. Intell. Transp. Syst., vol. 5, pp. 2–15, 2024.

[46] I. Arel, C. Liu, T. Urbanik, and A. G. Kohls, “Reinforcement learning-based multi-agent system for network traffic signal control,” IET Intell. Transp. Syst., vol. 4, no. 2, pp. 128–135, 2010.

[47] D. Silver et al., “Mastering the game of go with deep neural networks and tree search,” Nature, vol. 529, no. 7587, pp. 484–489, 2016.

[48] J. Oh, S. Singh, and H. Lee, “Value prediction network,” in Proc. 31st Adv. Neural Inf. Process. Syst., 2017, pp. 1–11.

[49] T. N. Kipf and M. Welling, “Semi-supervised classification with graph convolutional networks,” 2017, arXiv:1609.02907.

[50] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling, “Modeling relational data with graph convolutional networks,” in Proc. 15th Int. Conf. Eur. Semantic Web Conf., 2018, pp. 593–607.

[51] P. A. Lopez et al., “Microscopic traffic simulation using SUMO,” in Proc. 21st IEEE Int. Conf. Intell. Transp. Syst., 2018, pp. 2575–2582. [Online]. Available: https://elib.dlr.de/124092/

[52] T. Hubert, J. Schrittwieser, I. Antonoglou, M. Barekatain, S. Schmitt, and D. Silver, “Learning and planning in complex action spaces,” in Proc. Int. Conf. Mach. Learn., 2021, pp. 4476–4486.

[53] M. Fortunato et al., “Noisy networks for exploration,” 2019, arXiv:1706.10295.

[54] J. Oh, S. Singh, H. Lee, and P. Kohli, “Zero-shot task generalization with multi-task deep reinforcement learning,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 2661–2670.

[55] I. Higgins et al., “DARLA: Improving zero-shot transfer in reinforcement learning,” in Proc. 34th Int. Conf. Mach. Learn., 2017, pp. 1480–1490.