Adaptive Traffic Signal Control based on Multi-Agent Reinforcement Learning. Case Study on a simulated real-world corridor

Dickens Kakitahi Kwesiga[1], Angshuman Guin[1], Michael Hunter[1]

***Abstract*—**Previous studies that have formulated multi-agent reinforcement learning (RL) algorithms for adaptive traffic signal control have primarily used value-based RL methods. However, recent literature has shown that policy-based methods may perform better in partially observable environments. Additionally, RL methods remain largely untested for real-world normally signal timing plans because of the simplifying assumptions common in the literature. The current study attempts to address these gaps and formulates a multi-agent proximal policy optimization (MA-PPO) algorithm to implement adaptive and coordinated traffic control along an arterial corridor. The formulated MA-PPO has a centralized-critic architecture under a centralized training and decentralized execution framework. Agents are designed to allow selection and implementation of up to eight signal phases, as commonly implemented in field controllers. The formulated algorithm is tested on a simulated real-world seven intersection corridor.

The speed of convergence for each agent was found to depend on the size of the action space, which depends on the number and sequence of signal phases. The performance of the formulated MA-PPO adaptive control algorithm is compared with the field implemented actuated-coordinated signal control (ASC), modeled using PTV-Vissim®-MaxTime® software in the loop simulation (SILs). The trained MA-PPO performed significantly better than the ASC for all movements. Compared to ASC the MA-PPO showed 2% and 24% improvements in travel time in the primary and secondary coordination directions, respectively. For cross streets movements MA-PPO also showed significant crossing time reductions. Volume sensitivity experiments revealed that the formulated MA-PPO demonstrated good stability, robustness, and adaptability to changes in traffic demand.

***Index Terms*— adaptive traffic control, Multi-agent reinforcement learning, proximal policy optimization, software in the loop simulation, traffic simulation**

Dickness Kakitahi Kwesiga, Angshuman Guin and Michael Hunter are with the school of Civil and Environmental Engineering, Georgia Institute of Technology, Atlanta GA, 30332 USA (email: dkwesiga3@gatech.edu; angshuman.guin@ce.gatech.edu; michael.hunter@ce.gatech.edu)

I. INTRODUCTION

With the rise of complex and large machine learning (ML) models, there is heightened interest in developing ML based adaptive traffic signal control systems. Compared to other ML approaches, reinforcement learning (RL) is a direct choice for control problems such traffic signal control. Once trained, model free RL requires less computational resources to implement in real time compared to existing adaptive signal control systems based on mathematical programming. Previous efforts to develop adaptive RL-based signal control systems have mainly been limited to single agent / single intersection formulations with only a few recent efforts extending the formulation to multiple agents /multiple intersections. Additionally, previous studies have almost universally considered very simplified signal timing plans including limiting the number of phases, fixing the phasing sequences and fixing the duration of each signal phase. Therefore, for most part, RL approaches remain untested for the more complex, coordinated and realistic field signal control scenarios. The current study formulates multi-agent reinforcement learning (MARL) based adaptive signal algorithms considering real-world signal timing and network complexities. The formulated algorithms are tested on a real-world simulated arterial corridor and their performance compared to the field deployed optimal actuated-coordinated signal control (ASC) plans.

A. Related Studies

In RL, the learner, also called the agent, learns by continuous interaction with the environment. RL seeks to find the optimal mapping of states to actions, to achieve the highest numerical rewards. By trying out different actions over time the agent learns to identify the actions that maximize rewards given the state of the environment. MARL involves multiple agents interacting with a common environment and with each other. Moving from single agent to MARL introduces new challenges that require modification or formulation of new learning frameworks. These challenges include equilibrium selection, non-stationarity of the environment, scaling to many agents, multi-agent credit assignment, and partial observability [1, 2].

Several studies have formulated and tested RL-based adaptive traffic signal control algorithms in simulation environments. These studies show that RL-based algorithms can potentially outperform conventional fixed and actuated signal timing [3-12]. A few recent efforts have attempted to extend RL-based traffic control from a single intersection to multiple intersections and networks [13-18].

Most of the reviewed studies assume simplified signal plans including limiting the number of allowable phases [4-7, 9, 10, 12, 14, 16, 19, 20], assuming fixed timing sequences [3-10, 12-16, 19-21], fixing the duration or intervals of green for

each selected phase or phase combination [4, 10, 13, 22], and utilizing single ring barrier configurations [4, 10, 13, 22-25]. Therefore, for most part, RL approaches remain untested for the more complex and realistic field signal control scenarios.

The biggest share of reviewed studies formulates and trains RL control agents based on deep q-networks (DQN) and its variations [9-12, 19, 20, 22, 25-29]. The reviewed MARL studies are almost universally based on DQN [13, 15-18, 29, 30]. However, recent literature shows that policy gradient based methods show better performance and convergence, especially in partially observable environments [31, 32]. In a study intriguingly entitled "the surprising effectiveness of PPO in cooperative multi-agent games", the authors compared the performance of multi-agent proximal policy optimization (PPO) and other centralized training decentralized execution (CTDE) algorithms, including Q-mix, for four multi-agent environments. PPO showed comparable or superior performance and required minimum hyperparameter tuning compared to other algorithms [32].

Signal coordination is at the heart of arterial traffic flow optimization. A key objective of MARL-based traffic control algorithms is to promote cooperation between individual intersection agents to progress traffic through adjacent intersections. There is a very limited number of studies for MARL-based traffic control. Of the few studies, some have developed q-mix based algorithms [13, 15-17], others have utilized value network decomposition (VDN) [29, 30], while others have formulated independent q-learning algorithms [18, 30]. However, as indicated above, policy gradient (PG) algorithms, such as MA-PPO, are likely to achieve better performance in partially observable multi-agent environments compared to value-based methods [2, 31].

B. Problem definition and objectives

The current study formulates a multi-agent proximal policy optimization (MA-PPO) adaptive signal control algorithm and tests the algorithm on a simulated real-world corridor with field traffic volumes and geometry. The study seeks to overcome limitations identified in the literature, such as: 1) most studies are limited to single intersection control, 2) the few multiple intersections studies have almost universally formulated value-based RL algorithms, and 3) most studies utilize highly simplified signal control. Reinforcement learning approaches remain largely untested for more complex and normally coordinated real-world signal timing plans.

The formulated MA-PPO has a centralized critic architecture in which each agent is formulated with an actor network that selects independent actions conditioning on local observations and a centralized critic that estimates the agent's value function conditioning on global observations. All agents are formulated to allow the selection and implementation of up to eight signal phases, as commonly implemented in field controllers. For the test corridor and field measured traffic volumes, the performance of the formulated MA-PPO is compared against field implemented actuated-coordinated signal timings modeled using software in the loop simulation (SILs). To test the robustness of the formulated algorithm, sensitivity experiments are performed with volumes adjusted (a) upwards by 5% and (b) downwards by 10% from the field measured volumes

II. METHODOLOGY

A. Proximal Policy Optimization (PPO) Overview

Central to much of RL theory and most RL algorithms are value functions. Value functions are used to define the expected returns of individual states and actions. The state value function, $V^{\pi}(s)$, gives the expected returns starting from state (s), and following the policy (π) thereafter. The action value function, $Q^{\pi}(s,a)$, gives the expected returns starting from state (s), taking action (a), and following the policy thereafter. Thus, the state value function is the expected value across all action value functions, for all possible actions under a given policy.

RL algorithms fall under broad categories of value-based methods and policy-based methods. Value-based methods learn value functions and to select actions, the agent follows a policy derived from the learned value function. In deep RL, value-based methods such as DQN learn parameterized value-functions using neural networks. Policy-based based methods directly learn parametrized policy functions. Policy-based methods can easily learn stochastic policies which comes with several advantages, including (1) better performance under partially observable environments, and (2) better convergence properties as the action probabilities change smoothly as a function of learned parameters because policies are parameterized with continuous values [31]. PG algorithms are policy-based algorithms in which gradient-based optimization is used to optimize the policy parameters.

PPO belongs to a family of PG algorithms called actor-critic. Actor-critic algorithms have an actor network that learns a parameterized policy function to select actions and a critic network that learns a parameterized value function to evaluate the actions selected by the actor network. For discrete action spaces, on the forward pass, the actor network takes the environment state as input and outputs a probability distribution over the discrete action set. During training PG methods compute gradients of the loss function and optimize the policy parameters to follow the direction of higher expected returns. Returns are evaluated by value functions, $V^{\pi}(s)$ and $q^{\pi}(s,a)$. The foundational actor-critic algorithm called advantage actor critic (A2C) computes estimates of advantage, $Adv(s,a)$ using

$$Adv(s,a) = Q(s,a) - V(s) = \begin{cases} r^t & if\ s^{t+1}\ is\ terminal \\ r^t + \gamma V(s^{t+1}) - V(s^t) & otherwise \end{cases} \quad (1)$$

to guide policy gradients. In (1), r^t is the reward at time step, t with all other parameters already defined. Equations (1) – (4), and (7) are adapted from [2] while Equations (5) and (6) are adapted from [33]. In both cases the notation is modified.

Gradients are computed from the gradient theorem as in

$$\nabla_{\varphi} J(\varphi) = E_{s\sim \Pr(.|\pi),a\sim\pi(.|s;\varphi)}\left[Q^{\pi}(s,a)\frac{\nabla_{\varphi}\pi(a|s;\varphi)}{\pi(a|s;\varphi)}\right] \quad (2)$$

where J is the function measuring the quality of policy, $Pr(.|\pi)$ represents the state visitation probability under the policy π, s is the state, and φ are the policy parameters [2].

The actor loss, $L(\varphi)$ is computed as in

$$\mathrm{L}(\varphi) = -Adv(s^t, a^t) log\pi(a^t|s^t; \varphi) \quad (3)$$

with all terms defined previously.

On a forward pass, the critic takes the state as input and outputs a single value of the state value function. The critic optimization loss $L(\vartheta)$ is the squared error of the value estimate and the bootstrapped target estimate y^t as in

$$L(\vartheta) = (y^t - V(s^t; \vartheta))$$
$$y^t = \begin{cases} r^t & if\ s^{t+1}\ is\ terminal \\ r^t + \gamma V^{\pi}(s^{t+1}; \vartheta) & otherwise \end{cases} \quad (4)$$

where ϑ are the parameters of the critic network.

A potential challenge of PG methods is a significant change in policy from any individual gradient update which can result in instability. Trust region policy Optimization (TRPO) formulated by [34] and PPO formulated by [33] try to mitigate this risk. PPO builds on TRPO to formulate a computationally efficient surrogate objective function based on importance sampling weights, ρ(s,a) which is defined as the fraction of probabilities of selecting an action (a) in state (s) for the new and old policies as in.

$$\rho(s, a) = \frac{\pi_{new}(a|s; \varphi)}{\pi_{old}(a|s)} \quad (5)$$

The loss function $L(\varphi)$ is computed by clipping ρ(s,a) to restrict big changes to the policy for a single gradient update as in

$$\mathcal{L}(\varphi) = -\min \begin{pmatrix} \rho(s^t, a^t) Adv(s^t, a^t) \\ clip(\rho(s^t, a^t), 1-\epsilon, 1+\epsilon) Adv(s^t, a^t) \end{pmatrix} \quad (6)$$

In (6), the hyperparameter ϵ is selected to restrict how much the policy is allowed to change in a single update. Using ρ(s,a), PPO is able to restrict the magnitude of policy updates and update the policy multiple times using the same data

1. Multi-agent PPO – Centralized Critic

PPO with centralized critic belongs to the CTDE paradigm of MARL. Each agent's actor network selects local actions conditioning on only local observations or local observation histories. The loss function of the actor remains as described earlier. The critic network is only utilized during training and can therefore utilize information that may not be accessible or available during execution. This information can include the full state of the environment and any external data.

Figure 1, adapted from [2], shows the architecture of centralized critic for agent i conditioning on local observations of agent i (s_i) and on observations of all other agents in the environment (s_{n-i}) while still approximating the value function of agent i's policy. In this centralized state value-based critic architecture, each agent i gets a local reward r_i and does not take actions from other agents as inputs, as in centralized action value-based critic architectures. Despite not taking action inputs, centralized state value-based critics can still approximate advantage functions that provide preference over particular functions [2] and come with much simplicity compared to centralized action value-based critic architectures.

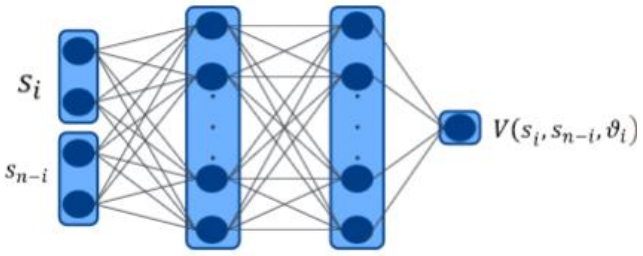

Figure 1: Architecture of Centralized critic for agent i, adapted from [2]

The loss function for agent i's critic is as in

$$\mathcal{L}(\vartheta_i) = (y_i - V(s_i^t, s_{n-i}^t; \vartheta_i))^2$$
$$y_i = r_i^t + \gamma V(s_i^t, s_{n-i}^t; \vartheta_i) \quad (7)$$

where ϑi are the network parameters, and r is the agent local reward with other variables as defined before.

B. Definition of state, action, and reward functions

1. State definition

The problem is formulated as a decentralized partially observable Markov decision process (DecPoMDP). At each intersection, the intersection actor network has access to only local observations. The local state has two component vectors, (1) vehicle state (Veh_{state}) and signal state (Sig_{state}). The vehicle state Veh_{state} is a vector of the number of vehicles (v) in each lane (n) on each approach (m) as in

$$Veh_{state} = \begin{bmatrix} v_{11} \\ v_{12} \\ . \\ . \\ v_{mn} \end{bmatrix}. \quad (8)$$

The study assumes the availability of connected vehicle (CV) data or video camera data at the intersection to deduce the total count of vehicles at all the approaches of the intersection. The considered approach distance is the distance to the upstream intersection.

The Sig_{state} has entries corresponding to the total number of intersection signal phases (n_p) as in

$$Sig_{state} = \begin{bmatrix} 10 \\ 0 \\ 12 \\ . \\ 0 \end{bmatrix}. \quad (9)$$

The entry for phase/s receiving green time are populated with the duration of green that has elapsed while all other entries are populated with zero. For all intersections of the corridor, the study assumes the availability of real time SPaT data which is increasingly ubiquitous.

The state for the centralized critic at intersection *i* (Sc_i) is defined as

$$Sc_i = [\left[Veh_{state_i} + Sig_{state_i}\right] +$$
$$\left[Veh_{state_1} + Sig_{state_1} + \cdots + \left[Veh_{state_n} + Sig_{state_n}\right]\right]. \quad (10)$$

This consists of (1) the local state of the agent which consists of Veh_{state} and Sig_{state} as defined above and (2) a concatenation

of local states at all *n* other intersections. The local state at the subject intersection *i* is placed in the first position.

It is perhaps worth reiterating that the critic network is only used during training and not during execution. During execution the trained algorithm does not access the global state defined in (10). This allows for flexibility in the information that can be included in the global state, including external data, as this data will not be required during execution.

2. *Action definition and implementation*

In this study, at each time *t*, an action is defined as whether the signal keeps the current phase or switches to any other phase in the set of valid phases. When the action changes from remaining in the same phase in time t-1 to selecting a different phase in t, the current phase yellow and red clearance intervals are displayed followed by the minimum green for the next selected phase. Every phase will have set minimum green that must be served, the phase green may be longer but not shorter than the minimum. Yellow and red clearance intervals allow safe termination of the current phase.

As indicated earlier, most of the reviewed studies assume very simplified signal plans. To more realistically replicate the field implemented signal plans, this study defines agent actions based on full dual ring barrier control, with the eight phase configurations as shown in Figure 2. In dual ring control one phase from each ring is active simultaneously, where Ring 1 consists of phases [1,2,3,4] and Ring 2 consists of phases [5,6,7,8]. Active phases must not conflict, thus the barriers are defined by the ends of phases [2,6] and [4,8]. When crossing a barrier both rings must change phases simultaneously. Accordingly, there are eight possible discrete actions [0,1,2,3,4,5,6,7] corresponding to eight paired phase combinations, [1,5],[1,6], [2,5], [2,6], [3,7], [3,8], [4,7], and [4,8]. Each action represents a compatible phase combination with one phase from each ring. For instance, action 0 calls for phase 1 in ring 1 and phase 5 in ring 2.

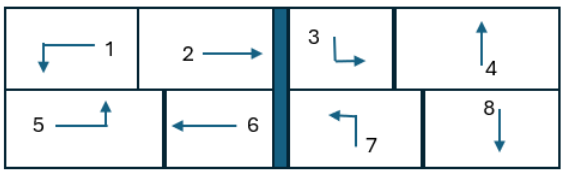

Figure 2: Dual ring barrier diagram

Kwesiga [25] describes action selection and implementation with a single ring barrier configuration. Action implementation for the dual ring barrier control in the current study is more complex as allowable actions may be limited at various points throughout the cycle or a phase as shown in Figure 3. To implement dual ring control the terminology “active phase” to refer to the current phase and “committed phase” to refer to the next selected phase is adopted for each ring. The benefit of tracking active and committed phase is that the only situation these differ is when a phase is in yellow or red clear. Thus, yellow and red clear may be timed without specifically implementing these indications as separate actions. Rather than this approach most recent studies include yellow and red clearance as separate actions in the action set [23, 29, 35]. However, this approach increases in the number of actions, potentially lengthening the training period.

Additionally invalid action masking (IAM) is used to force an agent to select the current active phase when required by signal control constraints. For example, when a phase is in it minimum green, yellow, or clearance intervals, the action must be to maintain the current active phase. IAM is also utilized to implement other constraints, such as implementing the ring barrier constraints. IAM is implemented by replacing logits corresponding to invalid actions with large negative numbers to preclude the selection of these actions.

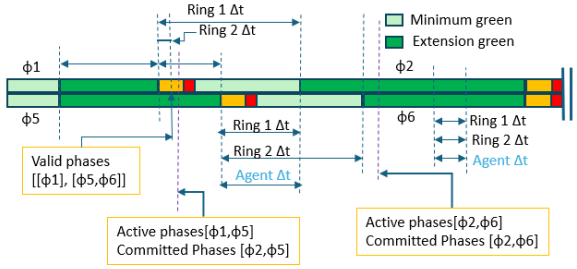

Figure 3: Action Implementation in dual ring barrier configuration

3. *Time step definition*

In implementing RL actions in a simulation environment, there are two time steps being implemented, (1) the simulation time step and (2) the RL algorithm time step. The simulation time step defines the frequency of updating the simulation environment. In the simulation the time step is set to 0.1 second as this provides reasonable flow models and is the fidelity of field signal timing. The MARL algorithm time step defines the frequency of actions by the MARL agents. The size of the MARL algorithm time step (Δt) is not constant as defined next. Unless otherwise stated, in this paper, the term time step or variable Δt refers to the MARL algorithm time step. The size of Δt determines the frequency of interaction with the simulation which largely determines the run time and the training duration [25]. It is thus critical to optimize Δt to improve run time efficiency.

To determine the MARL time step Δt, the time step for each agent, Δta_i, must be determined, where there are 1 to i agents in the MARL. To determine Δta_i for an agent, it is necessary to first determine the allowable time step for each agent ring $\Delta ta_i r_j$. where there are j rings in agent i. Next, $\Delta ta_i r_j$ for ring j depends on the state and duration of active phase and the committed phase in that ring. If the active phase is serving the clearance time, $\Delta ta_i r_j$ for ring j is equal to the remaining clearance time plus the minimum green of the committed phase. If the active phase is running minimum green, $\Delta ta_i r_j$ for ring j becomes the remaining minimum green duration. If the active green is running green extension, $\Delta ta_i r_j$ is set to 1 second. The agent Δta_i is determined as the minimum of the two ring time steps, i.e., min ($\Delta ta_i r_1$, $\Delta ta_i r_2$). See example

illustrations in Figure 3. Considering all agents in the environment, the overall Δt is selected as the minimum Δta_i for all agents, i.e., minimum (Δta_1, Δta_2, ... Δta_i). The more agents there are in the environment the likely smaller the Δt and consequently the larger the run time. The minimum allowable Δt is 0.1 second which corresponds to the simulation resolution of 10 steps per second.

4. *Reward definition*

The local reward r_i at each intersection i is taken as the negative sum of the normalized delay for all vehicles V on all approaches of the intersection as in

$$r_i^t = -\sum_{v \in V} \frac{d_v^t}{d_{max}}. \quad (11)$$

In (11), d_v^t is the individual vehicle delay at time t of each vehicle on the intersection approaches, while d_{max} is the maximum expected vehicle delay for that intersection. By using the ratio d_v^t/d_{max}, the delay is scaled between 0 and 1. The delay of each vehicle used in the computation of reward at the intersection excludes the delay encountered at previous intersections. This avoids penalizing the side streets in favor of the main line traffic with accumulated delay from other intersections. As shown by Lee [3], the summation of normalized delay in the reward tends to output the direct summation of delay values.

III. CASE STUDY

A. *Test Network*

The selected test network is a section of North Ave corridor in Midtown, Atlanta. Figure 4 is a google map snapshot showing the test corridor and the surrounding area. The section consists of seven signalized intersections marked with red circles on the figure. Moving from west to east the included intersections are North Ave with Tech Pkwy. NW, Techwood Dr. NW, I-75/I-85 off-ramp, Spring St. NW, W. Peachtree St. NW, Peachtree St. NE, and Jupiter St. NE.

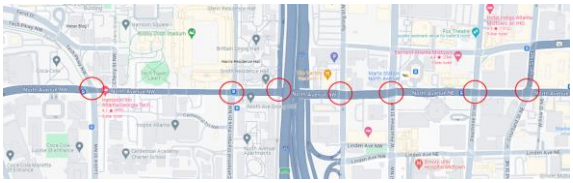

Figure 4: Test network extents

B. *Data*

The North Ave main street varies between a 4-lane and 6-lane facility, with turn bays at several intersections. Spring St. NW, W. Peachtree NW, and Jupiter St. NE are one-way streets. All the seven intersections run on Qfree Advanced Traffic Controllers (ATC), utilizing the MaxTime® traffic signal control software. For the signals, the Georgia Department of Transportation (GDOT), as part of an ongoing research project, provided detector layout maps and the signal timing databases. The signal timing databases include detector settings, controller settings, and signal timing plans. For this effort the PM peak was utilized. The existing corridor operates actuated-coordinated signal timing plans with a cycle length of 120 seconds in the PM peak. The coordinated movements are the North Ave. westbound and eastbound through movements, except for North Ave. @ Spring St. NW where the Spring St. southbound movement is coordinated.

Figure 5 shows the PM peak (04:30-05:30 pm) turning movement counts at the seven intersections collected on April, 04th 2023. The data was collected semi-automatically with videos cameras. The volumes in the figure are in vehicles per hour occurring over the peak hour.

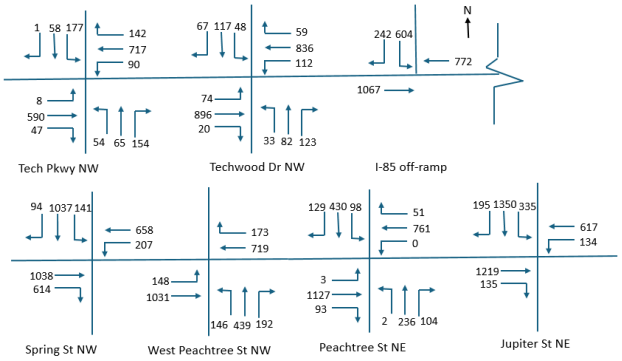

Figure 5: PM peak volumes

Additional trajectory data (i.e., vehicle location and speed) was drawn from drone videos as part of an ongoing GDOT research project. The drone data was collected on April 04th, 2023, in the PM peak from 3 pm to 7 pm. The data was collected with 6 drones to capture different angles and sections of the corridor. Field collected drone video data was processed using DataFromSky Software tools. The trajectory data was then utilized for simulation model calibration, discussed in section C below.

C. *Model development and calibration*

1. *Simulation Model development*

PTV-Vissim® is selected as the simulation environment. PTV-Vissim® is a microscopic, multi-modal traffic simulation software widely used in industry and research.

The network geometry is drawn from google maps and confirmed through site visits. Model construction, quality control, and quality assurance follow the guidance laid out in Hunter [36]. Figure 6 shows the network model in PTV-Vissim®.

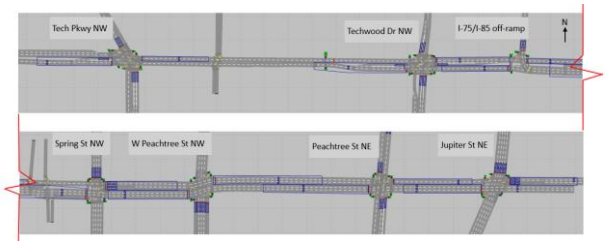

Figure 6: Network Model in PTV-Vissim®

For the existing conditions simulation model, the r emulator is utilized to control the intersections, forming the PTV-Vissim-MaxTime® SILs. The field controller MaxTime® databases with the field signal timing plans are loaded directly into the MaxTime® emulator which ensures that the existing conditions simulation runs the same signal timing plans and execution as the field. Each instance of MaxTime® (one for each intersection) is connected to PTV-Vissim through a unique transmission control protocol (TCP) port. A python event-based script embedded in Vissim is used to open and close MaxTime® instances at the beginning and ending of each simulation run, respectively.

The RL-based simulation similarly leverages the PTV-Vissim. However, the phase changes follow the agent's actions. Phase changes are implemented directly in Vissim with signal indication status controlled through COM.

2. Model calibration

Model calibration involves adjusting the model parameters so that the model performance matches field conditions [36]. Model calibration was performed utilizing the vehicle trajectories drawn from the drone videos. The annotated trajectories were post processed to extract vehicle speeds, headways, accelerations, etc.

Hunter [36] was used as guidance for the PTV-Vissim® model calibration. For this effort two sets of parameters are calibrated, (1) speed distribution and (2) car following model parameters. The desired speed distribution parameters in PTV-Vissim® are adjusted such that that the desired speed distribution matches the field observed speed distribution. For car following, the parameters of Wiedemann 74 flow model, that is, ax, bx_{Add}, bx_{Mult}, are adjusted such that the PTV-Vissim® simulated saturation headway distribution reasonably match field measured headway distribution. Figure 7 shows a comparison of cumulative distribution function (CDF) of field-measured headways vs calibrated model headways at North Ave. @ Spring St. The final calibrated model parameters are ax = 9, bx_{Add} = 2.2, and bx_{Mult} = 3.2.

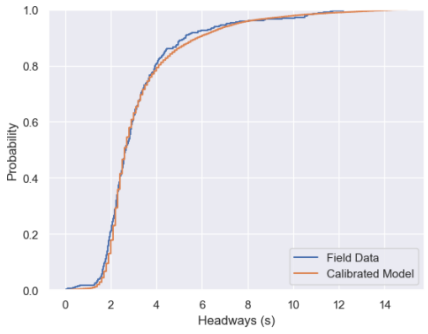

Figure 7: Field Vs. calibrated model headways at North Ave. @ Spring St.

D. RL and simulation architecture

Seven agents are formulated, one for each intersection. Figure 8 shows the interaction of agents with the PTV-Vissim® simulation environment. At each time step, each agent k pulls input from PTV-Vissim® including the local state S_k, and the global state Sc_k. Using the local state as input, each agent k selects action a_k, forming a set of global actions a_1 to a_7.

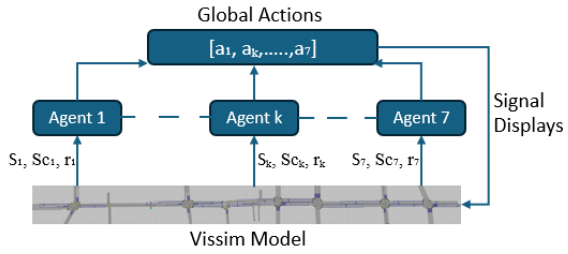

Figure 8: Interaction of agents with PTV-Vissim® Simulation environment

Figure 9 shows the detailed interaction between the agents (1-7) and PTV-Vissim® during the simulation run. To integrate RL into PTV-Vissim® the Component Object Model (COM), an application programming interface (API) for PTV-Vissim®, allows interaction with the simulation to extract data and to read and write to different objects during simulation runtime. This provides the means to extract vehicle data and performance measures to formulate state and rewards functions and to implement the selected actions in form of signal displays. The utilized architecture utilizes event-based scripts embedded directly in the PTV-Vissim® interface rather than external COM functions. This follows from a recent study by Kwesiga [25] that showed that executing PTV-Vissim® with event scripts is many fold faster than using external COM scripts. Three python scripts are embedded in PTV-Vissim®, the "State & Reward Script", the "Agent Script" and the "Model Runner Script". The Model Runner Script (MRS) is the central script tracking the simulation time and algorithm time step, initiating actions of the former two scripts and providing signal timings to PTV-Vissim® for implementation.

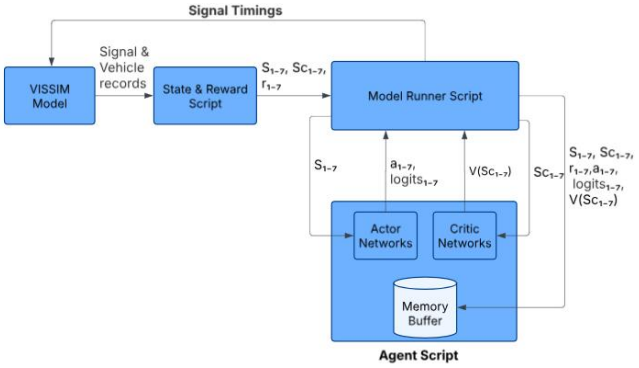

Figure 9:Agent-Vissim interaction during simulation run

At a timestep t, MRS passes the current local states (s_{1-7}) to the agents' actor networks (s_1 to agent 1 and so forth) to estimate the logits and actions. At the same time, MRS passes the global states Sck_{1-7} , one for each agent's critic network to

estimate the state values $V(Sck_{1-7})$ for that time step. With the selected actions, MRS determines Δt as described earlier. The selected actions for each agent are converted into signal timings and sent to PTV-Vissim® for display for a duration of Δt. The minimum green and yellow and clearance times are set to the same values present in the field signal timing plans. After Δt, MRS requests the next local state, global state and reward for each agent from the State & Reward Script. The State & Reward Script extracts vehicle records and signal states from PTV-Vissim® to formulate local state, global state, and reward functions as discussed in section II.B and Equations 8 through 11. A tuple of local state, global states, actions, rewards, logits and state values (s_{1-7}, Sck_{1-7}, a_{1-7}, $logits_{1-7}$, r_{1-7}, $V(Sck_{1-7})$ with 1to 7 corresponding to for each agent are saved in the memory buffer for each agent for training at episode end.

Insert

1. Agent architecture, training and Hyperparameter selection

All the seven agents are formulated with the same architecture for the actor and critic networks. Preliminary training is performed to fine tune the hyperparameters. The finally selected actor network has two hidden layers with 64 and 128 neurons, a clip ratio of 0.2, and a learning rate of 0.0003. The finally selected critic network has three hidden layers with 128, 256 and 256 neurons, a learning rate of 0.001 and a discount factor of 0.99.

Figure 10 shows the training architecture for each agent k. Training is performed at the end of each episode using data archived only that episode (on-policy learning). For a full trajectory over the entire episode using archived rewards and state values, advantage function values are estimated following Equation 1. Using the archived actions, logits, actions and local states, the actor network estimates the importance sampling weights as per Equation 5. Using importance sampling weights and estimated advantage values the actor clipped loss is determined per Equation 6. With the computed loss values, the parameters of the policy/actor network are updated using gradient accent. Using the archived global state values and rewards, the critic loss is computed as per Equation 7. The parameters of the critic network are then update via gradient descent.

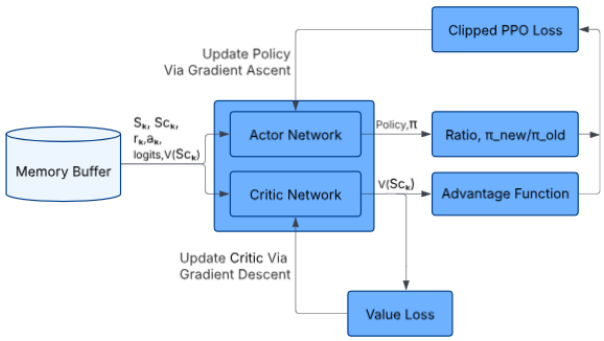

Figure 10: Algorithm architecture during training

2. MA-PPO Testing

Figure 11 shows the algorithm architecture during testing. As indicated, critic networks are only active during training but not during implementation/testing. During testing, agents 1-7, take local states (s_{1-7}) as input to predict the actions/signal timings that are implemented locally at each intersection. For each of the tests described in the section E, ten replicate simulation runs are performed, each with a unique random seed. Each simulation run lasts for one hour with data collected only in the last 45 minutes, with the first 15 minutes of the hour taken as the warm-up period.

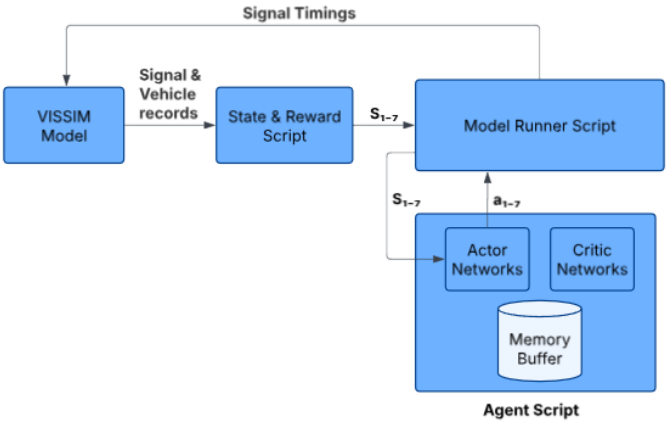

Figure 11: Algorithm architecture during testing

E. Results

1. Model training

Figure 12 shows the learning curves for the seven agents, one for each intersection. The x-axis plots the episode number while the y-axis plots the reward. Each data point is the average of rewards for all steps within the episode. Each episode lasts for 30 simulation minutes. Simulation runs and model training are performed on an x64-based PC equipped with 12th Gen Intel(R) Core i9-12900, 2400 MHZ, 16 Core(s), 24 Logical Processor(s),128 GB of RAM, Intel (R) UHD Graphics 770 GPU and Windows 11 operating system. It required approximately 60 hours to reach a fully trained model with the main limiting factor being PTV-Vissim®'s runtime efficiency as fully discussed in a recent study by Kwesiga [25]

The first four agents listed (North Ave. at Jupiter St., Peachtree St. NE, W. Peachtree St., and Spring St. NW) converge within the first 300 episodes. North Ave at I-75/I-85 off-ramp also converges within the first 300 episodes except for continued variability which eventually dies down. These intersections each have a maximum of four phases and despite the higher or comparable traffic volumes at these intersections, convergence occurs significantly faster than at intersections with additional phases.

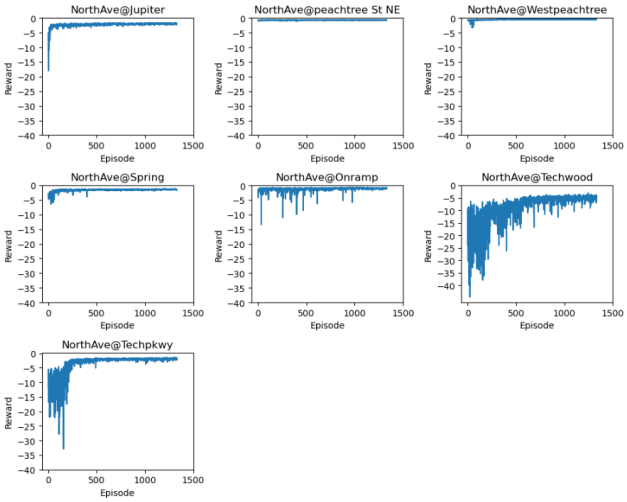

Figure 12. Learning Curves for different intersection agents

North Ave. at Tech Pkwy. NW (7 phases) converges after approximately 600 episodes while North Ave. at Techwood Dr. NW (8 phases) requires more than 1000 episodes to converge. The complexity of signal plans, including the number and sequence of phases appears to be the primary factor affecting the convergence. This is intuitively reasonable as the agents have larger action spaces.

2. MA-PPO vs field implemented coordinated ASC

The performance of the trained MA-PPO is evaluated against the field implemented signal timing plans using delay and travel time as performance measures. Figure 13 shows a comparison of main line movement travel time from the trained MA-PPO adaptive signal control and the field implemented coordinated ASC plans modeled using the MaxTime® emulator. The test volumes are the PM peak field volumes as shown in was then utilized for simulation model calibration, discussed in section C below.. The two movements are defined on the main street, from end to end of the model, in the eastbound (EB) and westbound (WB) direction. EB travel time is measured from a point upstream of North Ave @ Techpkwy to a point downstream of North Ave @Jupiter St NW while WB travel time is measured from a point upstream of North Ave @Jupiter St NW to a point downstream of North Ave @ Techpkwy. Each box plot is formed from ten data points, the average travel time for all vehicles in each of the ten replicate simulation runs.

In the WB which is the primary coordination direction for ASC, both MA-PPO and ASC show very comparable performance, with MA-PPO showing slightly lower travel time. However, in the WB, MA-PPO performed significantly better than ASC. Compared to the field implemented plans, MA-PPO showed a travel time improvement of 24% in the WB direction. Coordinated ASC is typically optimized to favor progression in a primary coordination direction, with the secondary direction likely receiving poorer service. This approach may be sub-optimal, especially if there is a high proportion of traffic in the secondary coordination direction. This is the case on North Ave. corridor. The superiority of MA-PPO comes from the ability to optimize performance in both directions.

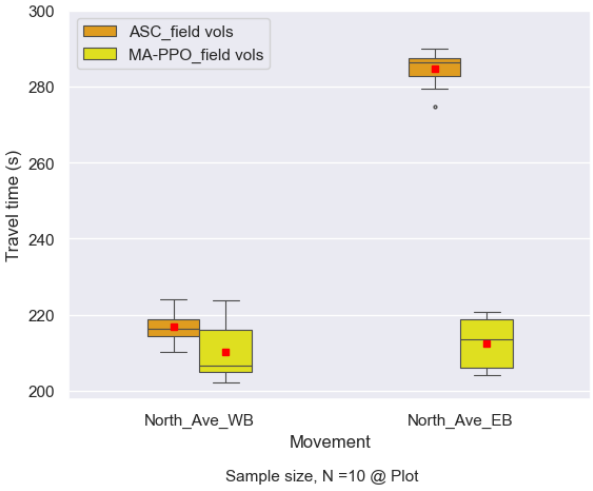

Figure 13: Main line movement travel time for ASC and MA-PPO for field measured traffic volumes

Figure 14 shows the performance of the two control systems for the cross street through movements using delay as the performance measure. MA_PPO shows significantly less delay for all the measured cross street movements (only through movements shown here for brevity). This includes Spring St. southbound which is the coordinated movement in the field ASC signal timings. Thus, the selected reward system and the centralized training leads to better performance for both main and cross streets at all intersections compared to the field implemented ASC.

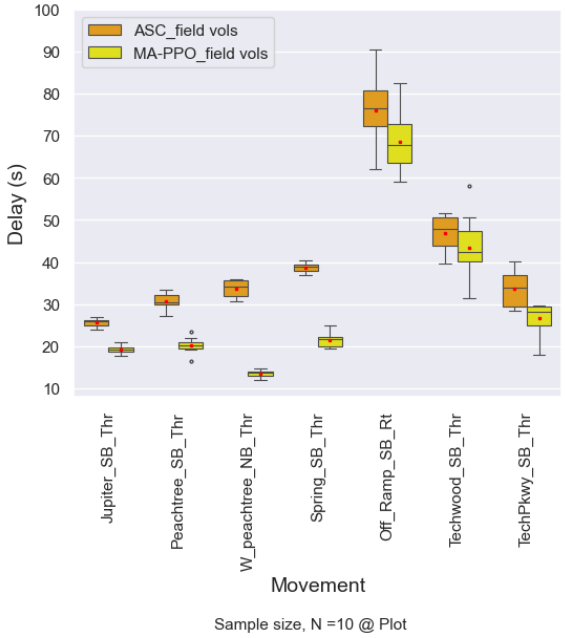

Figure 14: Cross street through movement Delays for MaxTime® and MA-PPO for field measured traffic volumes

3. Volume sensitivity and MA-PPO robustness

The above result is based on a single day field volume set and quite possibly may not reflect the typical day for which the ASC signal timings are optimized. To capture the impact of variability of traffic volumes and further test the robustness of the trained MA-PPO, sensitivity experiments are performed for two other volumes sets. Keeping the origin-destination (O-D) ratios fixed, the model entry volumes are adjusted (a) higher by 5% and (b) lower by 10% from the field measured volumes of was then utilized for simulation model calibration, discussed in section C below.. The limit of 5% higher is considered as some intersections are at or close to capacity for the field volumes.

Figure 15 shows the main street travel time from MA_PPO and ASC for the three volume levels. Travel time is measured from end to end of the model as described in section 2 above2. As expected, travel time increases as the volumes increase for both control systems. MA-PPO consistently outperforms ASC at all the three volume levels.

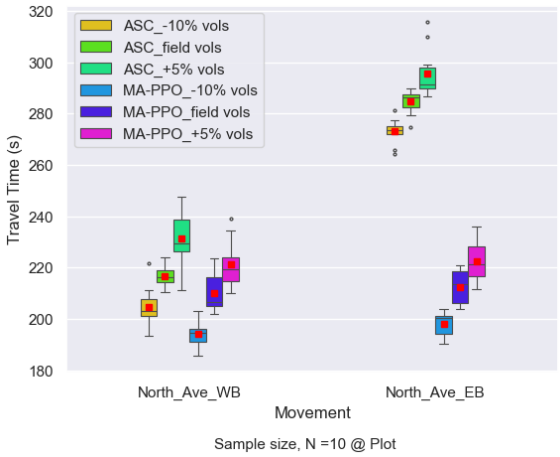

Figure 15: Main street Delay from MA-PPO vs coordinated ASC at three volume levels

Figure 16 shows a similar comparison for the cross streets, using delay as the performance measure. As with the North Ave. mainline, MA-PPO consistently outperforms ASC at the three volume levels. This demonstrates the robustness of the trained model to adapt to variations in traffic demand.

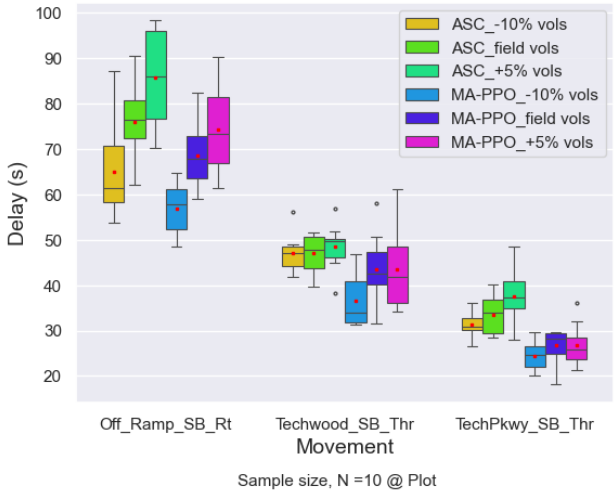

Figure 16: Cross street Delay from MA-PPO vs coordinated ASC at three volume levels

IV. CONCLUSION

This study formulates a multi-agent proximal policy optimization algorithm to implement adaptive and coordinated traffic control along an arterial corridor. The formulated MA-PPO has centralized critic architecture in which each agent has an actor network that selects independent actions conditioning on local observations and a centralized critic that estimates the agent's value function conditioning on global observations. All agents are formulated to allow selection and implementation of up to eight signal phases as commonly implemented in the field controllers. The formulated algorithm is tested on a simulated real-world corridor with seven intersections. The performance of the formulated MA-PPO adaptive control algorithm is compared with the field implemented ASC signal timing plans modeled using PTV-Vissim®-MaxTime® SILs.

The speed of convergence for each agent largely depended on the size of the action space which in turn depended on the number and sequence of signal phases. The intersections with 4 or fewer phases converged within 300 episodes each of 30 simulation minutes while the intersections with 7 and 8 phases required double to triple the number of episodes for convergence.

For the field measured traffic volumes, for all movements, the trained MA-PPO performed significantly better than the field implemented ASC signal timings. For the main street, MA-PPO optimizes performance in two directions compared to ASC which largely optimizes performance in the primary coordinated direction. Using travel time as the performance measure, the two control systems showed very comparable performance in the WB direction which is the PM peak direction and the primary coordination direction for ASC. In the EB direction, MA-PPO showed a 24% reduction in travel time compared to ASC. For cross streets, MA-PPO showed significantly reduced delay

To capture the variability of traffic volumes and further test the robustness of the trained MA-PPO, sensitivity experiments are performed for two other volumes sets, (a) 5%

above and (b) 10% below the field measured volumes. For both main street movements and cross streets movements, MA-PPO consistently outperformed ASC at the three volume levels. This demonstrated the robustness of the trained model to adapt to variations in traffic demand.

A key challenge of training of RL agents in a high-fidelity traffic microsimulation environment like VISSIM® is run time efficiency to reach hundreds to thousands of simulations runs required to fully train the agents. This study uses online serialized training for the formulated agents. Sixty hours were required to complete training of the agents. This limited the exploration and experimentation that could be achieved. Ongoing follow up studies are considering parallelized training architectures to reduce the required simulation run time.

REFERENCES

1. Zhang, K., Z. Yang, and T. Başar, *Multi-agent reinforcement learning: A selective overview of theories and algorithms.* Handbook of reinforcement learning and control, 2021: p. 321-384.
2. Albrecht, S.V., F. Christianos, and L. Schäfer, *Multi-agent reinforcement learning: Foundations and modern approaches*. 2024: MIT Press.
3. Lee, H., et al., *Effects analysis of reward functions on reinforcement learning for traffic signal control.* PLoS ONE, 2022. **17**(11 November).
4. Bálint, K., T. Tamás, and B. Tamás, *Deep Reinforcement Learning based approach for Traffic Signal Control.* Transportation Research Procedia, 2022. **62**: p. 278-285.
5. Li, Y., J. He, and Y. Gao. *Intelligent Traffic Signal Control with Deep Reinforcement Learning at Single Intersection*. in *ACM International Conference Proceeding Series*. 2021. Association for Computing Machinery.
6. Shabestary, S.M.A., et al. *Cycle-level vs. Second-by-Second Adaptive Traffic Signal Control using Deep Reinforcement Learning*. in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020.
7. Aslani, M., et al., *Developing adaptive traffic signal control by actor-critic and direct exploration methods.* Proceedings of the Institution of Civil Engineers: Transport, 2019. **172**(5): p. 289-298.
8. Casas, N., *Deep Deterministic Policy Gradient for Urban Traffic Light Control.* arXiv preprint arXiv:1703.09035, 2017.
9. Li, L., et al., *Traffic Signal Timing via Deep Reinforcement Learning*. 2016. p. 247-247.
10. Li, D., et al., *Adaptive Traffic Signal Control Model on Intersections Based on Deep Reinforcement Learning.* Journal of Advanced Transportation, 2020. **2020**.
11. Liu, S., G. Wu, and M. Barth. *A Complete State Transition-Based Traffic Signal Control Using Deep Reinforcement Learning*. in *2022 IEEE Conference on Technologies for Sustainability, SusTech 2022*. 2022. Institute of Electrical and Electronics Engineers Inc.
12. Bouktif, S., et al., *Deep reinforcement learning for traffic signal control with consistent state and reward design approach.* Knowledge-Based Systems, 2023. **267**.
13. Chang, A., et al., *CVDMARL: A Communication-Enhanced Value Decomposition Multi-Agent Reinforcement Learning Traffic Signal Control Method.* Sustainability (Switzerland), 2024. **16**(5).
14. Liu, D. and L. Li, *A traffic light control method based on multi-agent deep reinforcement learning algorithm.* Scientific Reports, 2023. **13**(1).
15. Fu, X., et al. *Research on Multi-Agent Reinforcement Learning Traffic Control*. in *2023 IEEE International Conference on Control, Electronics and Computer Technology, ICCECT 2023*. 2023. Institute of Electrical and Electronics Engineers Inc.
16. Bokade, R., X. Jin, and C. Amato, *Multi-Agent Reinforcement Learning Based on Representational Communication for Large-Scale Traffic Signal Control.* IEEE Access, 2023. **11**: p. 47646-47658.
17. Chen, X., et al. *A Collaborative Communication-Qmix Approach for Large-scale Networked Traffic Signal Control*. in *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*. 2021. Institute of Electrical and Electronics Engineers Inc.
18. Wu, C., Z. Ma, and I. Kim. *Multi-Agent Reinforcement Learning for Traffic Signal Control: Algorithms and Robustness Analysis*. in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. 2020.
19. Liu, X.-Y., et al., *Deep Reinforcement Learning for Traffic Light Control in Intelligent Transportation Systems.* 2023.
20. Bouktif, S., A. Cheniki, and A. Ouni, *Traffic signal control using hybrid action space deep reinforcement learning.* Sensors, 2021. **21**(7).
21. Wu, K. and S.I. Guler, *Estimating the impacts of transit signal priority on intersection operations: A moving bottleneck approach.* Transportation Research Part C: Emerging Technologies, 2019. **105**: p. 346-358.
22. Liang, X., et al., *A Deep Reinforcement Learning Network for Traffic Light Cycle Control.* IEEE Transactions on Vehicular Technology, 2019. **68**(2): p. 1243-1253.
23. Fan, W. and T. Yang, *Transit Signal Priority Control With Connected Vehicle Technology: Deep Reinforcement Learning Approach.* 2024.
24. Wang, S., et al., *Deep reinforcement learning-based traffic signal control using high-resolution event-based data.* Entropy, 2019. **21**(8).
25. Kwesiga, D., A. Guin, and M. Hunter. *Adaptive Transit Signal Priority Based on Deep Reinforcement Learning and Connected Vehicles in a Traffic Micro Simulation Environment*. in *2024 Winter Simulation Conference (WSC)*. 2024.

26. Pang, H. and W. Gao. *Deep Deterministic Policy Gradient for Traffic Signal Control of Single Intersection*. in *2019 Chinese Control And Decision Conference (CCDC)*. 2019.
27. Chanloha, P., et al., *Traffic Signal Control with Cell Transmission Model Using Reinforcement Learning for Total Delay Minimisation*, in *INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL ISSN*. 2015. p. 627-642.
28. El-Tantawy, S., B. Abdulhai, and H. Abdelgawad, *Design of reinforcement learning parameters for seamless application of adaptive traffic signal control.* Journal of Intelligent Transportation Systems: Technology, Planning, and Operations, 2014. **18**(3): p. 227-245.
29. Kwesiga, D., et al., *Integrating Transit Signal Priority into Multi-Agent Reinforcement Learning based Traffic Signal Control.* arXiv preprint arXiv:2411.19359, 2024.
30. Li, H., S. Li, and X. Zhang, *Coordination Optimization of Real-Time Signal Priority of Self-Driving Buses at Arterial Intersections Considering Private Vehicles.* Applied Sciences, 2023. **13**(19): p. 10803.
31. Morales, M., *Grokking deep reinforcement learning*. 2020: Manning Publications.
32. Yu, C., et al., *The surprising effectiveness of ppo in cooperative multi-agent games.* Advances in Neural Information Processing Systems, 2022. **35**: p. 24611-24624.
33. Schulman, J., et al., *Proximal policy optimization algorithms.* arXiv preprint arXiv:1707.06347, 2017.
34. Schulman, J., *Trust Region Policy Optimization.* arXiv preprint arXiv:1502.05477, 2015.
35. Long, M. and E. Chung. *Transit Signal Priority for Arterial Road with Deep Reinforcement Learning*. in *2023 8th International Conference on Models and Technologies for Intelligent Transportation Systems, MT-ITS 2023*. 2023. Institute of Electrical and Electronics Engineers Inc.
36. Hunter, M., et al., *VISSIM™ Simulation Calibration Procedure.* 2024.

DICKNESS KAKITAHI KWESIGA is a PhD student and a graduate research assistant in the School of Civil Engineering at Georgia Institute of Technology. His research interests include modeling, simulation and optimization of arterial corridor operations, traffic signal systems, Transit signal priority, emergency vehicle preemption, connected vehicles and AI based traffic control and AI/ML applications in traffic operations. His email is dkwesiga3@gatech.edu

ANGSHUMAN GUIN is a Senior Research Engineer in the School of Civil and Environmental Engineering at the Georgia Institute of Technology. His research attempts to find answers through innovations in the development of effective means of data collection, quality assurance, and processing to convert these data into informative metrics across a range of time-scales from near real time to decades. Dr. Guin's current research projects at Georgia Tech are broadly in Freeway Operations, Connected and Autonomous Vehicles, Intelligent Transportation Systems (ITS), Transportation Safety, Traffic Simulation and Data Management. His email is angshuman.guin@ce.gatech.edu

MICHAEL HUNTER is a Professor in the School of Civil and Environmental Engineering at Georgia Institute of Technology. His primary teaching and research interests are in transportation operations and design, specializing in emerging technology, adaptive signal control, simulation, and arterial corridor operations. His email is michael.hunter@ce.gatech.edu