Traffic Signal Control using Reinforcement Learning

Chițu Irina, group 507, irina.chitu@s.unibuc.ro
Iordache Bogdan, group 512, ioan.iordache@s.unibuc.ro
Manghiuc Teodor, group 507, teodor.manghiuc@s.unibuc.ro
Marchitan Teodor, group 512, teodor.marchitan@s.unibuc.ro
Sotir Anca, group 507, anca.sotir@s.unibuc.ro

30 May 2022

1 Introduction

Traffic congestion has been affecting people's daily lives nowadays. A significant amount of time on commute can be spent due to bad traffic conditions. To alleviate this issue, one of effective ways is a more efficient traffic signal control system.

Previous traffic signal control systems relied heavily on oversimplified information and rule-based methods. Nevertheless, we now have richer data, more computing power and advanced methods to drive the development of intelligent transportation. The strength that RL learns an optimal policy by interacting with the environment makes it suitable for real-world traffic signal control. Currently, many RL methods have been proposed.

For terminology, the most important notions in traffic signal control are:

- **Movement signal**: a movement signal is defined on the traffic movement, with green indicating the corresponding movement is allowed

and red indicating the movement is prohibited

- **Phase**: a phase is a combination of movement signals (see figure 1 for an example of commonly used phases)
- **Interval**: The period of time during which all signal indications remain constant
- **Phase sequence**: a phase sequence is a sequence of phases which defines a set of phases and their order of changes
- **Signal plan**: a signal plan for a single intersection is a sequence of phases and their corresponding starting time.

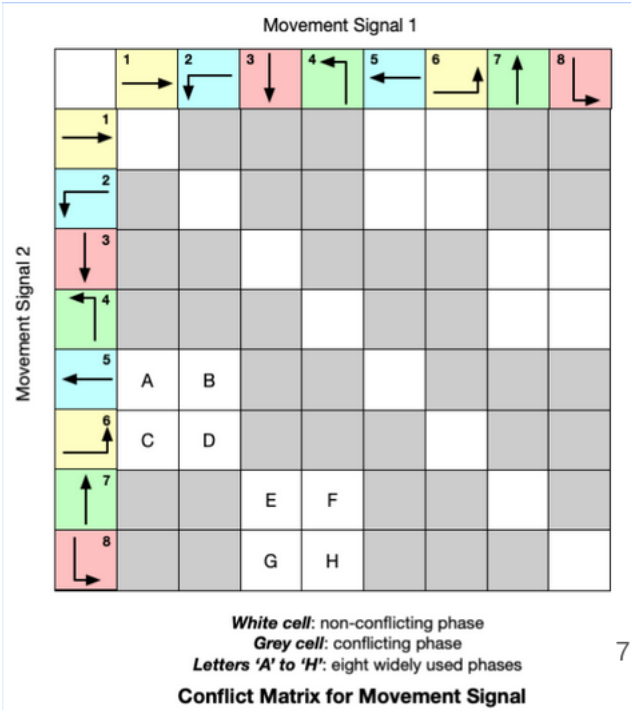

Figure 1: Various combinations of movement signals and commonly considered phases

2 Survey of Related Work

2.1 Overview

Before RL-based approaches, the best performing ones were based on various heuristics. Out of these we mention two: Self-Organizing Traffic Lights (SOTL) [2] and MaxPressure [9].

SOTL defines an algorithm in which traffic light phases are requested when the number of cars waiting on some incoming lane exceeds a threshold. Some deterministic rules are established in order to ensure what requests are satisfied at any given time.

MaxPressure uses the concept of pressure (a difference between the density of cars on incoming lanes and the density of cars from outgoing lanes, for a given phase). This algorithm tries at each step to enable the phase with the maximum pressure value.

When talking about RL approaches, of great importance are the definitions of the state of the system and of the reward. The actions that the agent can perform are the changes in the phase of one or more intersections.

For the state, the most common features are:

- **queue length**: number of waiting vehicles on a lane
- **waiting time**: time in which vehicles are not moving
- **volume**: total number of vehicles on a lane
- **delay**: actual travel time minus expected travel time
- **speed**: average speed of vehicles
- **phase duration**: in order to avoid flickering of traffic lights
- **positions of vehicles**
- **phase**: the actual phase may sometimes be also included in the state.

When defining the reward for the actions performed by the agent, some commonly used parameters are:

- **queue length**
- **waiting time**
- **change of delay**
- **speed** of vehicles
- **number of stops**: how many cars needed to stop due to the current action
- **throughput**: total number of vehicles that pass the intersection
- **frequency of signal/phase change**
- **pressure**.

Last, but not least, is the way in which methods deal with systems of multiple intersections. There are three distinguishable approaches:

1. **centralized control**: a single agent deals with all intersections at once, the action space can be huge (P^N, where P is the number of phases for an intersection and N is the number of intersections)
2. **individual RL without coordination**: train one for each of the intersections individually, only cares about local reward for the corresponding intersection; hard to converge because of the high number of agents (can be mitigated by using shared structures)
3. **individual RL with coordination** same as before, this time the state for a given intersection contains information about the state of the neighboring ones.

In the next sections we will present a number of recent approaches for traffic signal control using RL.

2.2 IntelliLight: A Reinforcement Learning Approach for Intelligent Traffic Light Control

Previous studies did not test their approaches on real-world traffic data and overlooked the importance of interpreting policies when analyzing the rewards. **Hua Wei, Guanjie Zheng, Huaxiu Yao and Zhenhui Li** [12] extend the **deep reinforcement learning** field by making several important new contributions:

- experiment with **real traffic** from 24 intersections (data collected from surveillance cameras in Jinan, China for a period of 31 days);
- interpret the **policies** (especially when the resulting reward is the same for several phases) and prove their method can intelligently adjust to different traffic conditions (such as peak hours vs. non-peak hours, weekday vs. weekend and major arterial vs. minor arterial);
- alter Deep Q-Network's structure by adding a "**Phase Gate**" which aids distinguishing the decision process for different phases (Figure 2).

Their model is designed for a **four way intersection** with **two phases** (Green, which includes a 3 second yellow light and Red). It is composed of two parts: **offline** and **online**. The target of the first stage is to **collect and train** on data samples from the traffic going through a system with a **fixed timetable** for the lights. During the second one, **rewards** are being computed. At every time interval Δt, the traffic light agent will analyse the state and find the most beneficial action in the long run by following the Bellman Equation (Equation 1)). Afterwards, it will observe the environment and extract the reward which will be stored into memory. The network is updated after several timestamps based on the logs from the memory.

$$q(s_t, a, t) = r_{a,t+1} + \gamma maxq(s_{a,t+1}, a', t+1) \tag{1}$$

The **state** (s) covers one intersection and for each lane (i) defines: queue length (L_i), number of vehicles (V_i), updated waiting time of vehicles (W_i)

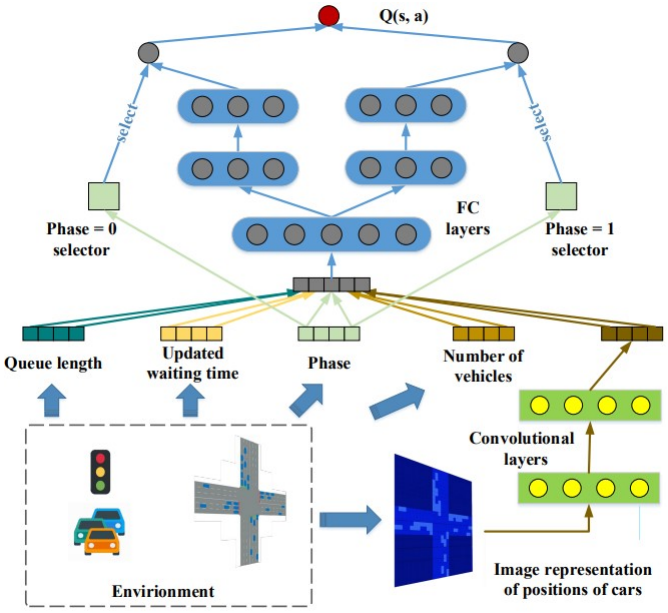

Figure 2: Network Structure [12]

and an image representation of vehicles' position (M), current phase (P_c) and next phase (P_n).

The **action** (a) can be either 1 (change the light to next phase) or 0 (keep the current phase).

The **reward** (r) is defined as a weighted sum (Equation 2), where D_i is the delay for lane i, C is the indicator of light switches (either 0 on 1 depending on whether or not the current phase is kept or not), N is the total number of vehicles and T is the total travel time of vehicles that passed the intersection during the time interval Δt after the last action.

$$r = w_1 * \sum_{i \epsilon I} L_i + w_2 * \sum_{i \epsilon I} D_i + w_3 * \sum_{i \epsilon I} W_i + w_4 * C + w_5 * N + w_6 * T \quad (2)$$

Figure 3 shows that the proposed model is able to learn and to adapt to different traffic trends without any prior knowledge (e.g. major arterial).

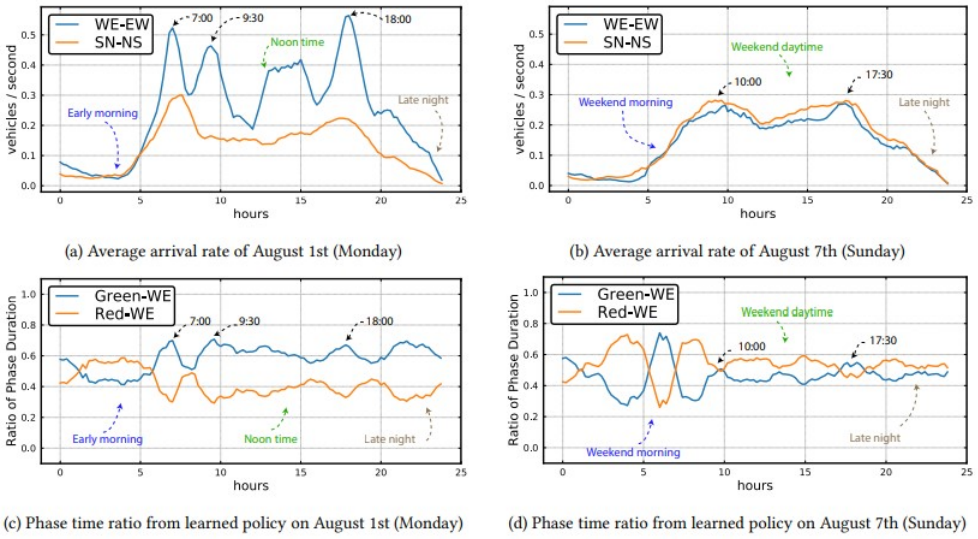

Figure 3: Average arrival rate on two directions (WE and SN) and time duration ratio of two phases (Green-WE and Red-WE) [12]

2.3 PressLight: Learning Max Pressure Control to Coordinate Traffic Signals in Arterial Network

One major issue of current RL-based traffic signal control approaches is that the setting is often heuristic and lacks proper theoretical justification from transportation literature. The authors of this paper examined two fundamental elements in RL setting: reward and state. The main contributions of the paper are theoretically grounded by max pressure (MP) control method [3, 9] and are as follows:

- the reward function can be set to minimize the pressure of an intersection
- by including the variables from the evolution equations into the state definition in RL, the state is a sufficient statistics for the system dynamics

The proposed solution focuses on the multi-intersection control scenario, where each intersection is controlled by an RL agent. At each time step

t, agent i observes from the environment as its state o_i^t. Given the vehicle distribution and current traffic signal phase, the goal of the agent is to give the optimal action a (i.e., which phase to set), so that the reward r (i.e., the average time of all vehicles) can be maximized.

- **State (Observation).** State is defined for one intersection. It includes the current phase p, the number if vehicles on each outgoing lane $x(m)$ ($m \in L_{out}$) and the number of vehicles on each segment of every incoming lane $x(l)_k$ ($l \in L_{in}, k = 1...K$) (in this paper $K = 3$)

- **Action.** At time t, each agent chooses a phase p as its action a_t from action set A, indicating the traffic signal should be set to phase p. Each action candidate a_i is represented as a one-hot vector

- **Reward.** The reward r_i is defined as

$$r_i = -P_i,$$

  where P_i is the **pressure** of intersection i. The pressure of intersection i is defined as the sum of the absolute pressures over all traffic movements (as shown in Figure 4):

$$P_i = |\sum_{(l,m)\in i} w(l,m)|,$$

  and the pressure of a movement is defined as the difference of vehicle density between the incoming lane and the outgoing lane. Thus the pressure of movement (l, m) is described as follows:

$$w(l,m) = \frac{x(l)}{x_{max}(l)} - \frac{x(m)}{x_{max}(m)}$$

Deep Q-Network (DQN) is used as function approximator to estimate the Q-value function. During training an experience replay memory is maintained by adding the new data samples in and removing old samples occasionally. Periodically, the agent will take samples from the memory to use them to update the network.

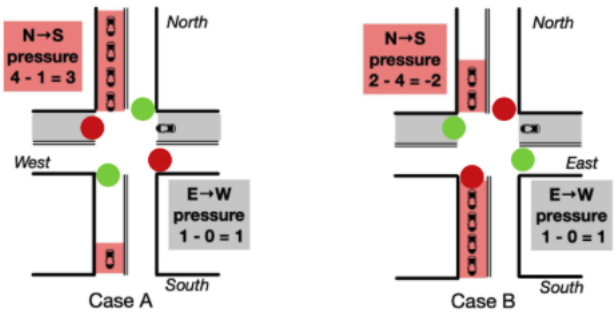

Figure 4: Illustration of max pressure control in two cases. In Case A, green signal is set in the North $\longrightarrow$ South direction; in Case B, green signal is set in the East $\longrightarrow$ West direction.

Experiments were conducted on CityFlow [15] using both synthetic and real-world traffic flow data. Figure 5 shows the results of the proposed method **PressLight**, which outperforms all other conventional transportation methods and learning methods.

	Synthetic traffic				Real-world traffic					
	LightFlat	LightPeak	HeavyFlat	HeavyPeak	Qingdao Rd., Jinan	Beaver Ave., State College	8th Ave., NYC	9th Ave., NYC	10th Ave., NYC	11th Ave., NYC
FixedTime	93.29	109.50	325.48	246.25	317.40	336.29	432.60	469.54	347.05	368.84
GreenWave	98.39	124.09	263.36	286.85	370.30	332.06	451.98	502.30	317.02	314.08
MaxPressure	74.30	82.37	262.26	225.60	567.06	222.90	412.58	370.61	392.77	224.54
GRL	123.02	115.85	525.64	757.73	238.19	455.42	704.98	669.69	676.19	548.34
LIT	65.07	66.77	233.17	258.33	58.18	338.52	471.30	726.04	309.95	340.40
PressLight	**59.96**	**61.34**	**160.48**	**184.51**	**54.87**	**92.00**	**223.36**	**149.01**	**161.21**	**140.82**

Figure 5: Performance comparison between all the methods in the arterial with 6 intersections w.r.t. average travel time (the lower the better). Top-down: conventional transportation methods, learning methods, and the proposed method in this paper.

2.4 Learning Traffic Signal Control from Demonstrations

Most of the RL-based approaches for traffic signal control suffer from a key challenge: large exploration space. The authors of this paper proposed a

method for integrating expert knowledge into RL for this particular task [14]. The authors seek expert knowledge from classical methods in the transportation field, such as Self-Organizing Traffic Light control (SOTL) [2]. As to utilization of expert knowledge, it was leveraged in the form of demonstrations, (i.e. expert-like trajectories in decision-making tasks).

The main contributions of the paper are, as follows:

- it is the first work that tries to integrate demonstrations into RL for traffic signal control
- it exploits demonstrations collected from SOTL to accelerate an actor-critic RL algorithm
- actor and critic are trained with demonstrations respectively in order to provide expert-like initialization
- provide ablation studies in order to assess the role of demonstrations in overall performance.

The model used for this approach is an Advantage Actor Critic (A2C) [13]. The actor and the critic are initialized using expert knowledge inferred from demonstrations provided by SOTL. The advantage is defined using the following equation:

$$A^{\pi}(s,a) = Q^{\pi}(s,a) - V^{\pi}(s) = Q^{\pi}(s,a) - \sum_{a'} \pi(a'|s) Q^{\pi}(s,a')$$

For **the actor** training using demonstrations, the sampled actions are drawn from the categorical distribution provided by the policy π, using:

$a_{\text{soft}} = \text{softmax}((g+\pi)/\tau)$, where g corresponds to random re-parametrization.

The loss is defined as:

$L_{pre}(\theta_{\pi}) = \text{Cross-Entropy}(a_{\text{soft}}, a_D)$, where a_D is the action of the demo.

For **the critic** training using demonstrations, four losses are used when comparing the output to the demonstrations: 1-step TD loss, n-step TD loss, a large margin classification loss and L2 regularization loss.

The equation for the 1-step TD loss is the following:

$$L_{TD}(\theta_Q) = \frac{1}{2}(R(s,a) + \gamma Q(s',a') - Q(s,a|\theta_Q))^2$$

While for the large margin we have:

$$L_{\text{margin}}(\theta_Q) = \max_a[Q(s,a) + l(a_D,a)] - Q(s,a_D)$$

where $l(a_D, a) = 0.8$ if $a_D \neq a$ else 0.

During the experiments, the actor-critic model initialized using expert knowledge achieved better performance when compared to other approaches. The tests are run on independent intersections from three cities. Using SOTL for demonstration collection, the method can quickly learn a better initialization, i.e., to allocate much more time of green signal to movements with higher traffic. Figure 6 shows an ablation study of how actor and/or critic initialization using demonstrations improve the performance of vanilla A2C.

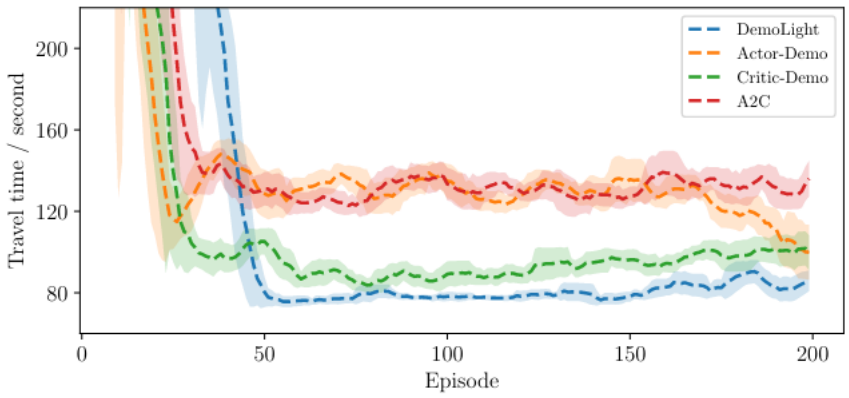

Figure 6: Ablation study for learning traffic signal control through demonstrations.

2.5 Learning Phase Competition for Traffic Signal Control

The authors of this paper [16] introduce a novel design called FRAP, which is very powerful for traffic signal control in the single intersection case, but can also be extended for multi-intersection cases. FRAP is especially designed to capture the competition between phases and, most importantly, it ensures invariance to symmetrical cases (flipping the traffic flow). A common pattern of the traffic, for example, would be that people make their morning commute to work and return home in the evening. RL models should easily capture and adapt the equivalence between these flipped cases. Furthermore, using FRAP greatly reduces the exploration space, which leads to finding better solution for difficult traffic patterns and also to faster convergence.

Contributions

- Propose a novel model design FRAP (invariant to flipping and rotation because it captures relations between traffic movements);
- Demonstrate that FRAP converges faster than other RL methods;
- Demonstrate generalizability of FRAP (handle complex road structures, traffic flows, multi-intersection environments).

A single RL agent observes an intersection and changes the signal accordingly. The goal of this agent is to learn a policy which optimizes travel time. The problem can be formulated as a Markov Decision Process $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$. The goal of this problem is to maximize $G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + \dots$.

The states are defined by the number of vehicles waiting for each traffic movement, along with the current signal phase. Actions consist of choosing the phase for the next time interval. The reward is based on the average queue length for each traffic movement (optimizing queue length has been proved to be equivalent to optimizing travel time in [17]).

The design of the FRAP network is based on the following principles:

- **Competition:** larger traffic movement indicates higher demand for green signal; if two signals conflict, priority is given to the one with higher demand.
- **Invariance:** the model should be invariant to symmetries (like rotation

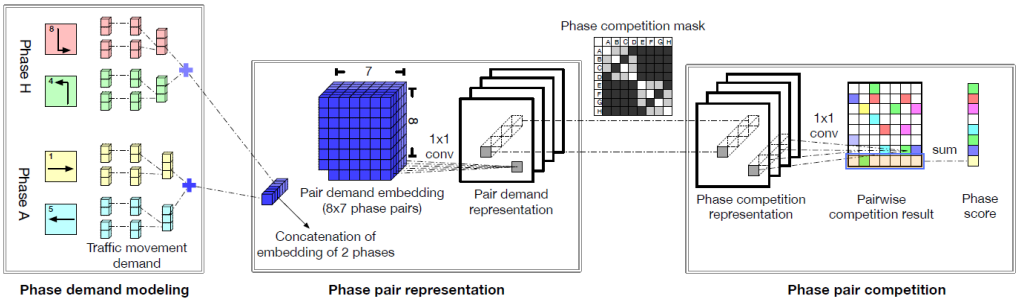

Figure 7: The network design of FRAP signal control.

and flipping).

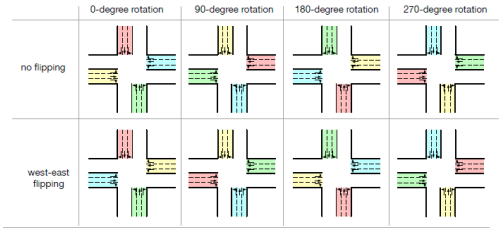

Figure 8: For each four way intersection configuration, we can define seven other configurations just by applying rotations and flipping.

The authors have used the Ape-X Deep Q-Network, a distributed framework which is efficient for large search space problems. The deep reinforcement learning is composed of two parts:

1. **Acting:** assigns multiple actors with exploration policies to interact with the environment and store observed data in the replay memory;
2. **Learning:** sample training data from the memory to update the model.

Note that the two parts of the learning framework can run concurrently, similarly to a generator and a consumer.

Phase invariant signal control design
The challenges which were addressed in this paper are working with a large search space and with intersections of varying geometry (with 3, 4, 5 entering approaches) or with different combinations of traffic movement. Thus, it would be inefficient to learn different agents for each intersection type. The

network is made up of multiple embedding and convolutional layers. The model takes as input the state features and predicts the score (Q value) for each action as described in the following Bellman Equation: $Q(s_t, a_t) = R(s_t, a_t) + \gamma \cdot maxQ(s_{t+1}, a_{t+1})$. The agent can then choose the phase with the highest score. Predicting the Q value is done in three stages:

1. **Phase demand modeling:** represent the demand for each signal phase; the input features are passed through two fully connected layers, using ReLU activations.
2. **Phase pair representation:** build a volume of phase pair embeddings and obtain a representation by applying convolutional layers.
3. **Phase pair competition:** predict the score (Q-value) for each phase taking into account its competition with other phases.

Some learning details would be that they used Adam optimizer with a learning rate of $1e-3$. Also, three actors run in parallel to implement the Ape-X DQN framework.

Experiments and interesting results
They have used CityFlow for their experiments. As for datasets, they have used two private real-world datasets (Jinan and Hangzhou) and a public dataset (Atlanta).

For evaluation, the average time spent by vehicles on approaching lanes was used. The numbers show that FRAP outperforms all other compared models.

- The authors have shown that FRAP achieves good results even without explicit coordination between agents.
- While other models need re-training when the traffic patterns drastically change, FRAP is much less affected. Thus, it spares us the re-training time.
- The model adapts faster than IntelliLight to different traffic volumes.
- The experiments done on different intersection structures show best results compared with the other models. The standard one is the four-way intersection, so the authors have made the following changes to their model: for 3-way intersections, disable some neurons and the

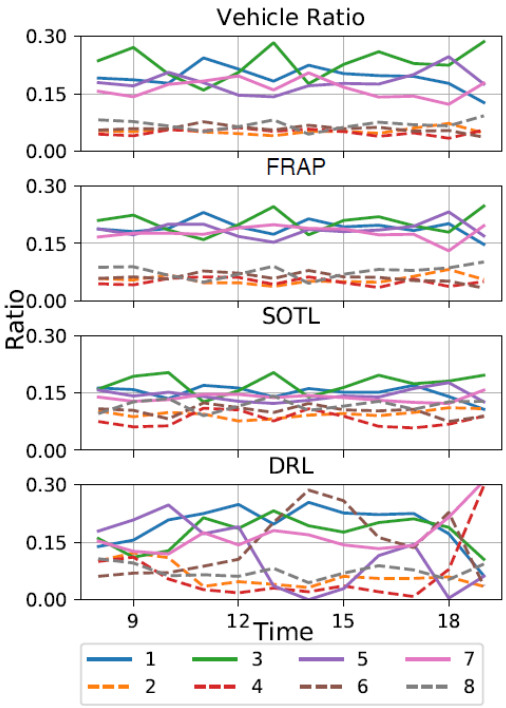

Figure 9: An interesting manner to visualize the policy learned with FRAP. The first graph represents the volume of vehicles through time per traffic movement; the other three represent the green time allocated per traffic movement. Only FRAP is correlated with the actual volume of vehicles.

input is zero-padded; for 5-way intersections, an additional phase is added.

2.6 CoLight: Learning Network-level Cooperation for Traffic Signal Control

This paper focuses on the cooperation of traffic signals between intersections.

The authors are the first ones to facilitate their communication by using **graph attentional networks** which are able to learn the dynamics of temporal and spatial changes. Moreover, none of the previous studies that use reinforcement learning methods to cooperate traffic signals evaluate their

methods on a **large-scale road network** with hundreds of traffic signals.

Another key contribution is the **index-free model learning with parameter sharing**. Similar to the "mean-field" idea, the influences of all neighboring intersections is averaged over with the learned attention wrights. A consequence of the parameter sharing strategy is that the overall parameters of the learning method are largely reduced.

The major components are defined as:

- **system state space** $\mathcal{S}$ and **observation space** $\mathcal{O}$, where the observation $o \epsilon \mathcal{O}$ of a state $s \epsilon \mathcal{S}$ for agent i at time t (o_i^t) consists of its current phase (which direction is in green light) represented by a one-hot vector and the number of vehicles on each connected lane;
- **set of actions** $\mathcal{A}$: from time t to t + Δt, each intersection would be in the chosen phase p as its action $a_i^t \epsilon \mathcal{A}_i$;
- **transition probability** $\mathcal{P}$: the system arrives at the next state s^{t+1} according to the state transition probability $\mathcal{P}(s^{t+1}|s^t, a^t) : \mathcal{S} \times \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \rightarrow \Omega(\mathcal{S})$, where $\Omega(\mathcal{S})$ denotes the space of state distributions;
- the **reward** of each agent i at time t ($r_i^t : \mathcal{A}_1 \times \ldots \times \mathcal{A}_N \rightarrow \mathbb{R}$) is defined as $r_i^t = -\sum_l u_{i,l}^t$ (where $u_{i,l}^t$ is the queue length on the approaching lane l at time t) and aims to minimize the travel time for all vehicles in the system
- each agent chooses an action based on a **policy** (π) in order to maximize the total reward $G_i^t = \sum_{t=\tau}^{T} \gamma^{t-\tau} r_i^t$, where T is the total time steps of an episode and $\gamma \epsilon [0, 1]$ is the **discount factor** which differentiates the rewards in terms of temporal proximity.

The structure of the proposed cooperated reinforcement learning network is (Figure 10):

- an observation layer which embeds a k-dimensional (raw) data into an m-dimensional latent space using a Multi-Layer Perceptron with ReLU as the activation function; the generated hidden state $h_i \epsilon \mathbb{R}^m$ represents the current traffic condition of the i-th intersection.

- interior neighborhood cooperation layers (GAT layers with 5 attention heads) which focus on the interactions between intersections and learn the traffic trend
- a Q-value prediction layer with the following loss function: $L(\theta) = \frac{1}{T}\sum_{t=1}^{t=T}\sum_{i=1}^{i=N}(q(o_i^t, a_i^t) - \tilde{q}(o_i^t; a_i^t, \theta))^2$, where $\tilde{q}$ is the predicted q-value, N is the number of intersections in the whole road network and θ represents all the trainable variables in the proposed model

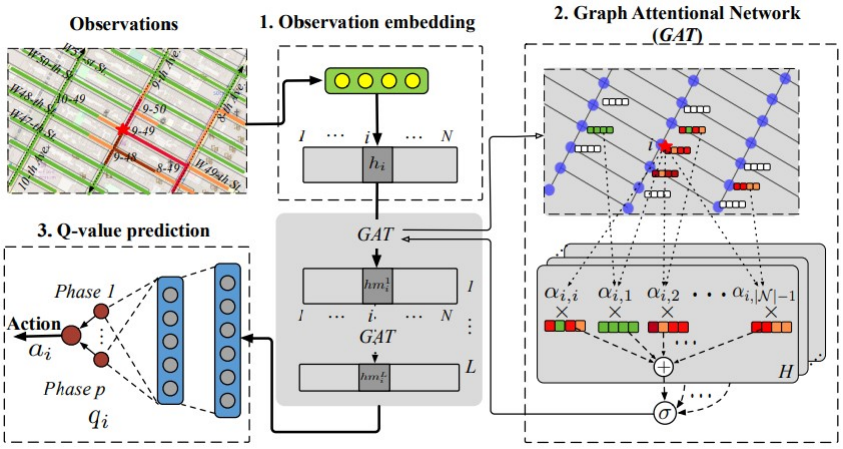

Figure 10: Model Framework [11]

CoLight is more scalable and performs consistently better than other RL-based methods. Moreover, it is far more efficient than most of them.

2.7 MetaLight: Value-based Meta-reinforcement Learning for Traffic Light Control

The training mechanism of DRL follows a trial-and-error manner and thus the superior performance is conditioned on a large number of training epochs. If the traffic condition is complicated, traditional DRL models need long time to generate enough samples and to have models well-trained. Thus the agent for traffic signal control should be able to learn quickly with few

samples. The authors of this paper propose a novel meta-reinforcement learning framework, **MetaLight**, which is build upon the gradient-based meta-reinforcement learning line. The key contributions are as follows:

- improved a structure-agnostic DQN-based traffic signal control model called FRAP [16] which enables heterogeneous scenarios sharing the same parameters and, based on the meta-reinforcement learning paradigm, a well-generalized initialization from various traffic signal control tasks is learnt
- proposed two types of adaptation mechanisms: individual-level adaptation and global-level adaptation

The FRAP network consists of several embedding layers and convolutional layers. The former parameters are shared across lanes which means the number and type of approaching lanes only affect the network structure rather than the parameters of embedding layers. Furthermore, FRAP uses fixed number of 1×1 filters in convolutional layers, thus they are also independent of the number and type of phase. In summary, the structure of FRAP depends on the number of lanes and phases in the intersection but the network parameters are sharing in different intersections.

In order to improve the flexibility of FRAP on different lanes combination, the authors of this paper proposed an improved model named FRAP++ (Figure 11), which enhance FRAP from two folds: (1) The FRAP++ represents the phase demand by averaging each lane's demand instead of adding the demand in order to remove the influence of difference in the lane number under each phase and make FRAP widely applicable. (2) FRAP updates parameters only after each whole episode, which violates DQN one-step updating mechanism. Instead, FRAP++ improves the updating frequency by undertaking mini-batch updating each step in one episode.

- **State** of FRAP++ consists of the number of vehicles and signal phase on each approaching lane
- **Action** for RL agent is defined as choosing the phase for the next time interval

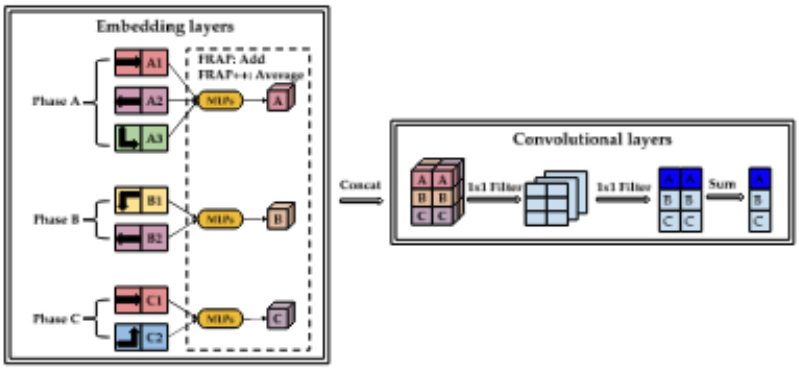

Figure 11: The illustration of FRAP and FRAP++. FRAP uses the sum of lanes' representation to represent phase while FRAP++ uses the mean of them. Yellow multi-layer perceptrons (MLPs) are shared by each phase.

- **Reward** is defined as the average queue length on approaching lanes

MetaLight follows the traditional gradient-based meta-reinforcement learning framework, MAML (Model-Agnostic Meta-Learning). However, traditional design of MAML mainly focuses on policy-based DRL problems. Empirically, on value-based DRL models like FRAP++, MAML only slighty outperforms random initialization. Thus, the authors improved MAML by alternatively utilizing individual-level adaptation and global-level adaptation. MetaLight takes advantage of fast learning in DQN by updating parameters at each time-step and extracting the common knowledge in MAML by gradient descent.

Individual-level Adaptation DQN uses a neural network to represent the action-state function $Q(s, a)$. FRAP++ follows the standard design of DQN with experience replay and target value network. In each intersection I_i, the agent's experiences $e_i(t) = (s_i(t), a_i(t), r_i(t), s_i(t+1))$ at each timestep t are stored in set D_i. In individual-level adaptation, the parameters θ_i of each task T_{i_s} are updated at each timestep by gradient descent. In particular, in value-based reinforcement learning, individual-level adaptation is taken at each timestep to speed up the learning process on source intersections.

Global-level Adaptation After the adaptation in individual-level, global-

level adaptation aims to aggregate the adaptation of each intersection I_i, and then update the initialization θ_0 of meta-learner using newly sampled transitions $D_i^{'}$

Transfer Knowledge of New Intersections In the meta-training process of MetaLight, a well-generalized initialization of parameters in f is learnt. Then the initialization θ_0 is applied to a new target intersection I_t in order to compute the optimal parameters θ_t.

Experiments were all conducted in the CityFlow [15] simulation platform. The authors used four real-world datasets from two cities in China: Jinan (JN) and Hangzhou (HZ), and two cities in the United States: Atlanta (AT) and Los Angeles (LA). In order to build enough heterogeneous scenarios, they added some new phase settings (Figure 12).

Phase Setting	A	B	C	D	E	F	G	H
4a	✓			✓	✓			✓
4b		✓	✓			✓	✓	
4c	✓			✓		✓	✓	
4d		✓	✓		✓			✓
6a	✓	✓	✓	✓	✓			✓
6b	✓			✓	✓	✓	✓	✓
6c	✓	✓	✓		✓	✓	✓	
6d		✓	✓	✓		✓	✓	✓
6e	✓	✓	✓	✓	✓		✓	
6f	✓		✓	✓		✓	✓	✓
8	✓	✓	✓	✓	✓	✓	✓	✓

Figure 12: Eleven phase settings in experiments are composed of different phases from A to H. Red represents PS1 and blue denotes PS2.

The testing set is classified into three types and introduced as follows: Task-1 is a set of homogeneous tasks in which testing sets are similar with training sets except traffic flow. Task-2 represents heterogeneous tasks which means testing datasets are different from training datasets in both traffic flow and phase setting. Task-3 consists of both homogeneous and heterogeneous tasks from different cities (Jinan, Atlanta and Los Angeles).

The most representative results are the one conducted on the task-3 (as shown

in Figure 13), where MetaLight outperforms all baselines, adapts much faster and is more stable.

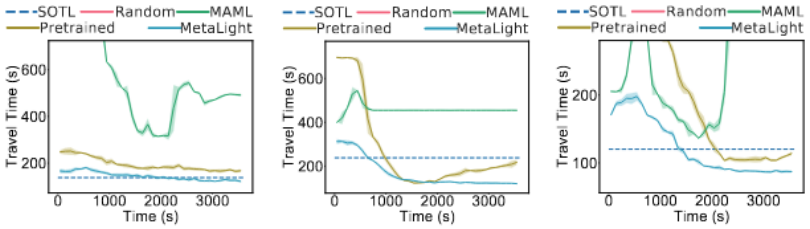

Figure 13: Meta-testing results for task-3. The random baseline curves are excluded since their results are much worse.

2.8 Toward A Thousand Lights: Decentralized Deep Reinforcement Learning for Large-Scale Traffic Signal Control

As suggested by the title of this paper [1], the focus shifts to traffic control at a larger scale. No other method of Reinforcement Learning had been tested until then on traffic networks of more than one thousand traffic lights.

The contribution of the authors is that they define a decentralized RL algorithm that can handle large-scale traffic signal control and make the first experiments on real-world environment containing thousands of traffic lights.

The authors introduce three key aspects must be addressed by an effective method for traffic signal control:

- **Scalability:** the model should be able to handle a large-scale network.
- **Coordination:** optimizing the signal timings should be done jointly; they are often in close proximity of each other and failure to synchronize might deteriorate the state of the other intersection.
- **Data feasibility:** the method should not use data which is hard to acquire in a real world scenario.

There can be found multiple problems with the previous RL approaches, based on the three issues presented above.
Firstly, in the case of centralized optimization methods, scalability cannot be satisfied due to the large space that must be analyzed.
Secondly, decentralized RL methods can easily scale up, but it is hard to assign the global-wise reward function to each intersection. Thus, they do not satisfy coordination. For example, the average waiting time and delay might be minimized per intersection, but this does not guarantee objective optimization for the whole network, especially at a larger scale.
Finally, some methods use data such as aerial view of each intersection in the network. This is unrealistic in real life so they do not satisfy data feasibility.

The following aspects are relevant for the definition of MPLight:

- The authors chose a **decentralized** approach in order to ensure **scalability**. A single intersection would be managed by a RL agent.
- **Parameter sharing** is enabled by choosing **FRAP** [16] as a base model. FRAP was especially designed for traffic signal control and is invariant to transformations such as rotation and flipping. This means that agents who control the traffic flow of differently structured intersections **essentially follow the same logic and should benefit from each other's knowledge**. This process also enhances learning speed.
- The state and reward design is based on **PressLight** [10]. Essentially, using the concept of pressure will lead to **balancing the distribution of vehicles** through the network and finally **maximizing throughput of the system**. Thus, this chosen strategy addresses **coordination between agents**.
- The pressure of an intersection is **derived from simple features like queue length** which is easily available in real-world scenarios, which ensures **data feasibility**.

The Reinforcement Learning Environment
The **observations** for each interssection are composed of the currently active signal phase and the pressure of all traffic movements *(the pressure of a*

traffic movement is defined as the absolute difference between the number of vehicles on the entering lane and the number of vehicles on the exiting lane of that movement).
The **action** at a given time consists of the agent choosing one of the possible signal phases.
The **reward** is defined as minus the pressure of the intersection *(note: the pressure of the intersection is the absolute difference between the sum of vehicles on entering lanes and the sum of vehicles on exiting lanes).*

Using FRAP as a base network ensures the best performance (compared with the other previous models) and faster training process. Similarly to the paper in which FRAP is proposed, **deep Q-learning** is used to predict the score (i.e. Q value) for each possible signal phase, and then the agent chooses the phase with the highest score.

The **replay memory is also shared** between agents which facilitates the learning process.

Their experiments
The authors use CityFlow for their experiments, similarly to the other papers. The model is analysed on both synthetic and real-world datasets.

- **Using the synthetic dataset:** The network is a four by four grid of four-way intersections, with 300 meter road segments between them. Based on statistical analysis on real data, the vehicles have a 10% chance to take a left turn, a 60% chance to go straight through and 30% chance to take a right turn at an intersection. Also, four configurations were considered as following (two patterns per arrival rate): two different vehicle arrival rates and two patterns (flat - small variance, peak - high variance).
- **Using a real-world dataset:** The road network of Manhattan, New York. The road net is obtained from OpenStreetMap, while vehicle data is taken from an open source taxi trip dataset. This network is composed of approximately 2.5 thousand street lights.

The evaluation was made using **travel time** (which should be minimized) and the **throughput** (which should be maximized).

The results show that MPLight outperformed all the other models until that point in time, in both syntethic and real cases.

Model	Travel Time				Throughput			
	Config 1	Config 2	Config 3	Config 4	Config 1	Config 2	Config 3	Config 4
FixedTime	573.13	564.02	536.04	563.06	3555	3477	3898	3556
MaxPressure	361.17	402.72	360.05	406.45	4702	4324	4814	4386
GRL	735.38	758.58	771.05	721.37	3122	2792	2962	2991
GCN	516.65	523.79	646.24	585.91	4275	4151	3660	3695
NeighborRL	690.87	687.27	781.24	791.44	3504	3255	2863	2537
PressLight	354.94	353.46	348.21	398.85	4887	4742	5129	5009
FRAP	340.44	298.55	361.36	598.52	5097	5113	5483	4475
MPLight	**309.33**	**262.50**	**281.34**	**353.13**	**5219**	**5213**	**5652**	**5060**

Figure 14: The results on the four synthetic experiment configurations.

Model	Travel Time	Throughput
FixedTime	974.23	1940
MaxPressure	497.76	2143
GRL	-*	-*
GCN	653.45	5045
NeighborRL	-*	-*
PressLight	600.42	3447
FRAP	512.70	6346
MPLight	**472.51**	**6932**

Figure 15: The results on the large-scale real Manhattan dataset.

2.9 Comparing methods

Comparing the presented approaches is not a trivial task, due to some limitations:

- differences in synthetic data generation

Model	Travel Time
GCN	653.45
GCN + pressure	**646.47**
PressLight- pressure	654.04
PressLight	**600.42**
FRAP	512.70
FRAP + pressure	**472.51**

Figure 16: The impact of using the pressure concept for three different models: An interesting experiment was to analyze the impact of pressure-based design for multiple model architectures. All models which were considered for that experiment show a better result with pressure rather than without pressure.

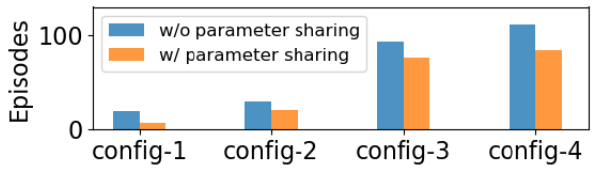

Figure 17: The impact of parameter sharing in the synthetic data experiments. The results show that the number of episodes until convergence was smaller when using parameter sharing.

- single intersection performance vs. multi-intersection networks
- using various subsets of a roadnet (just a main road, or the whole network)
- different simulation environments (SUMO vs. CityFlow).

In figure 18 the FRAP algorithm is compared, among others, with the IntelliLight method, achieving better performance on both single-intersection and multi-intersection environments.

CoLight was also compared with IntelliLight, and it achieves better average travel time on multi-intersection settings, as seen in figure 19.

In figure 20, MPLight is displayed with better performance than PressLight

Model	Jinan	Hangzhou	Atlanta
Fixedtime	880.18	823.13	493.49
Formula	385.46	629.77	831.34
SOTL	1422.35	1315.98	721.15
DRL	1047.52	1683.05	769.46
IntelliLight	358.83	634.73	306.07
A2C	316.61	591.14	244.10
FRAP	**293.35**	**528.44**	**124.42**

Figure 18: Comparison of performance between FRAP and other methods

Model	$Grid_{6\times6}$-Uni	$Grid_{6\times6}$-Bi	$D_{NewYork}$	$D_{Hangzhou}$	D_{Jinan}
Fixedtime [15]	209.68	209.68	1950.27	728.79	869.85
MaxPressure [24]	186.07	194.96	1633.41	422.15	361.33
CGRL [23]	1532.75	2884.23	2187.12	1582.26	1210.70
Individual RL [30]	314.82	261.60	-*	345.00	325.56
OneModel [5]	181.81	242.63	1973.11	394.56	728.63
Neighbor RL [1]	240.68	248.11	2280.92	1053.45	1168.32
GCN [18]	205.40	272.14	1876.37	768.43	625.66
CoLight-node	178.42	176.71	1493.37	331.50	340.70
CoLight	**173.79**	**170.11**	**1459.28**	**297.26**	**291.14**

*No result as *Individual RL* can not scale up to 196 intersections in New York's road network.

Figure 19: Comparison of performance between CoLight and other methods and FRAP on multi-intersection environments using synthetic data.

Model	Travel Time				Throughput			
	Config 1	Config 2	Config 3	Config 4	Config 1	Config 2	Config 3	Config 4
FixedTime	573.13	564.02	536.04	563.06	3555	3477	3898	3556
MaxPressure	361.17	402.72	360.05	406.45	4702	4324	4814	4386
GRL	735.38	758.58	771.05	721.37	3122	2792	2962	2991
GCN	516.65	523.79	646.24	585.91	4275	4151	3660	3695
NeighborRL	690.87	687.27	781.24	791.44	3504	3255	2863	2537
PressLight	354.94	353.46	348.21	398.85	4887	4742	5129	5009
FRAP	340.44	298.55	361.36	598.52	5097	5113	5483	4475
MPLight	**309.33**	**262.50**	**281.34**	**353.13**	**5219**	**5213**	**5652**	**5060**

Figure 20: Comparison of performance between MPLight and other methods

3 Experiments

3.1 CityFlow

CityFlow [15] is a simulation environment designed for large-scale traffic signal control bench-marking.

Unlike previous environments such as SUMO [4], CityFlow leverages multi-threaded computations in order to efficiently simulate a large amount of traffic. Due to this fact, it performs well on multi-agent settings with many intersections, and it is able to simulate even road networks of entire cities.

The system is complex and provides a mean of storing road network structure along with logic for car following, intersection logic, and even a model for how cars may switch lanes.

In order to define a simulation setting, a couple of files must be provided:

- **roadnet description**, which contains among other things:
  - roads, and their corresponding lanes
  - intersections, with corresponding road-links and lane-links
  - traffic light phases description for each intersection
- **flow configuration**, which describes the traffic in the simulation as a list of vehicles for which it specifies:
  - dimensions
  - maximum speed
  - maximum acceleration
  - intersection related speed.

CityFlow also provides a front-end for displaying the steps of the simulations on a web interface using WebGL, as can be seen in figure 21.

When compared to SUMO in the setting of multiple intersections, it can achieve even a 25 times reduction of simulation speed, as shown in figure 22.

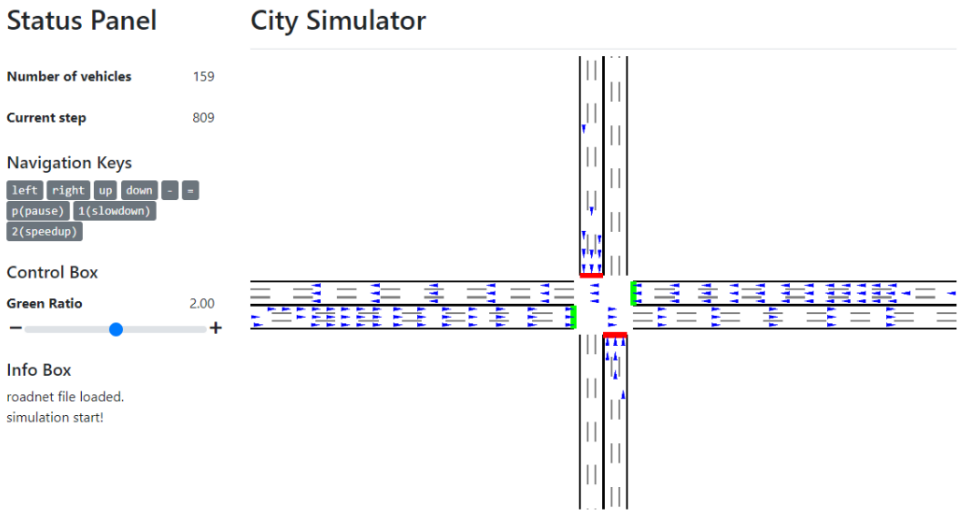

Figure 21: CityFlow WebGL front-end

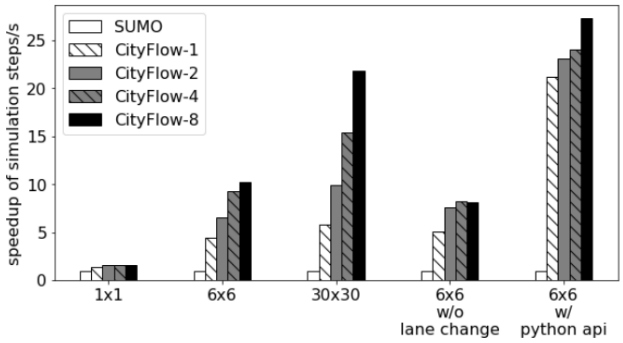

Figure 22: Speedup of CityFlow compared to SUMO

In order to be able to train various models more easily, we have wrapped the simulation environment inside an OpenAI Gym environment. The setting in our experiments corresponds to a 1x1 two-lane intersection with one hour of traffic recorded in Hangzhou, China.

3.2 Problem Representation

For the representation of the state for one intersection we consider:

- the number of waiting vehicles for each incoming lane
- the total number of vehicles on each lane, both incoming and outgoing (inspired from the definition of pressure)

The actions performed by the agent can change the current phase of the intersection, at each time-step. The reward is computed as the total number of waiting vehicles on incoming lanes. We also tried to penalize phase changes that are performed too often (to make the model avoid flickering), but with no sensible improvements.

3.3 Stable Baselines 3

We looked at several RL algorithms listed in table 1. Out of these, we propose running experiments on DQN [6], A2C [5], and PPO [8].

Method	**Policy**	**Observations**
Q-learning	off-policy	discrete observation space
SARSA	on-policy	discrete observation space
Deep Q-network (DQN)	off-policy	
Deep Deterministic Policy Gradient (DDPG)	off-policy	only continuous action space
Advantage Actor Critic (A2C)	on-policy	
Proximal Policy Optimization (PPO)	on-policy	

Table 1: Considered methods for RL experiments

For the implementation of the selected methods, we employed Stable Baselines 3 [7] an up-to-date version of a pyTorch-based library of RL algorithms.

3.4 Results

For the chosen algorithms, through hyper-parameter tuning, we have decided on the following settings:

- **DQN**: batch size = 128, learning rate of 0.0005
- **A2C**: num. steps per update = 15, learning rate of 0.0005
- **PPO**: batch size = 256, learning rate of 0.0007

The training is done for 200 episodes, each episode consisting of 3.600 steps.

Figure 23 contains the evolution of the total reward computed for each training episode. Best models are stored based on reward at the end of each episode. Running the best models on the simulation we achieve the evolution of the average waiting time described in figure 24.

Figure 23: Evolution of total reward computed for each training episode

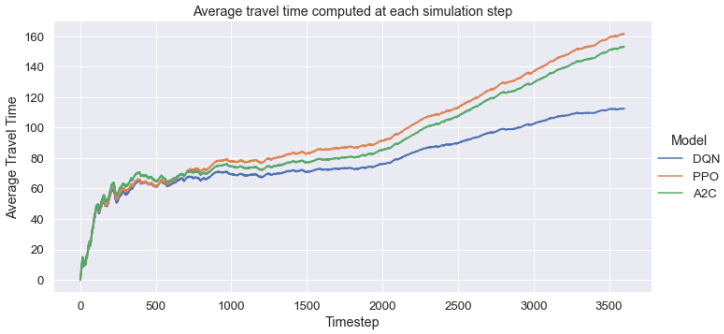

Figure 24: Average vehicle waiting time at each step, using best performing models

4 Conclusions

Through this work, we provide a brief overview of most recent RL methods for traffic signal control. We have seen how various formulations of state and reward can improve traffic.

Also, we provide a comparison of popular RL algorithms on a minimal representation of the state.

References

[1] Chacha Chen, Hua Wei, Nan Xu, Guanjie Zheng, Ming Yang, Yuanhao Xiong, Kai Xu, and Zhenhui Li. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3414–3421, 2020.

[2] Seung-Bae Cools, Carlos Gershenson, and Bart D'Hooghe. Self-organizing traffic lights: A realistic simulation. In *Advances in applied self-organizing systems*, pages 45–55. Springer, 2013.

[3] Jennie Lioris, Alex Kurzhanskiy, and Pravin Varaiya. Adaptive max pressure control of network of signalized intersections. *IFAC-PapersOnLine*, 49(22):19–24, 2016.

[4] Pablo Alvarez Lopez, Michael Behrisch, Laura Bieker-Walz, Jakob Erdmann, Yun-Pang Flötteröd, Robert Hilbrich, Leonhard Lücken, Johannes Rummel, Peter Wagner, and Evamarie Wießner. Microscopic traffic simulation using sumo. In *2018 21st international conference on intelligent transportation systems (ITSC)*, pages 2575–2582. IEEE, 2018.

[5] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[6] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[7] Antonin Raffin, Ashley Hill, Maximilian Ernestus, Adam Gleave, Anssi Kanervisto, and Noah Dormann. Stable baselines3, 2019.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[9] Pravin Varaiya. Max pressure control of a network of signalized intersections. *Transportation Research Part C: Emerging Technologies*, 36:177–195, 2013.

[10] Hua Wei, Chacha Chen, Guanjie Zheng, Kan Wu, Vikash Gayah, Kai Xu, and Zhenhui Li. Presslight: Learning max pressure control to coordinate traffic signals in arterial network. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1290–1298, 2019.

[11] Hua Wei, Nan Xu, Huichu Zhang, Guanjie Zheng, Xinshi Zang, Chacha Chen, Weinan Zhang, Yanmin Zhu, Kai Xu, and Zhenhui Li. CoLight. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, nov 2019.

[12] Hua Wei, Guanjie Zheng, Huaxiu Yao, and Zhenhui Li. Intellilight: A reinforcement learning approach for intelligent traffic light control. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '18, page 2496–2505, 2018.

[13] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3):229–256, 1992.

[14] Yuanhao Xiong, Guanjie Zheng, Kai Xu, and Zhenhui Li. Learning traffic signal control from demonstrations. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 2289–2292, 2019.

[15] Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The world wide web conference*, pages 3620–3624, 2019.

[16] Guanjie Zheng, Yuanhao Xiong, Xinshi Zang, Jie Feng, Hua Wei, Huichu Zhang, Yong Li, Kai Xu, and Zhenhui Li. Learning phase competition for traffic signal control. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 1963–1972, 2019.

[17] Guanjie Zheng, Xinshi Zang, Nan Xu, Hua Wei, Zhengyao Yu, Vikash Gayah, Kai Xu, and Zhenhui Li. Diagnosing reinforcement learning for traffic signal control. *arXiv preprint arXiv:1905.04716*, 2019.