Coordinated Traffic Control Technical Report (RL and Beyond)

Parth Jaggi, Xiaoyu Wang and Nicolas Carrara.

September 1, 2020

1 Introduction

The transportation systems play an important role in the society by providing mobility and accessibility to people and goods. However, urban expansion and population growth in large metropolitan areas like Toronto produce significant traffic congestion, creating inefficiencies in the transportation system. The impact of traffic congestion is reflected in many aspects, including but not limited to traffic delay, pollution, fuel waste, and safety risks.

Intelligent Transportation Systems (ITS) manage traffic systems in real-time using advanced technologies to leverage the full potential of the transportation system and further alleviate congestion. As a fundamental and important part of ITS, Adaptive Traffic Signal Control (ATSC) aims to maximize the efficiency of the road network by adjusting the timing plan of traffic signals adaptively based on real-time traffic demand. The first ATSC was developed in the 1960s (Gordon et al., 2005). In the 1970s, famous ATSCs such as SCOOT (Split, Cycle and Offset Optimization Technique) and SCATS (Sydney Coordinated Adaptive Traffic System) were developed. These systems use the centralized closed-loop controlling framework and are widely deployed around the world (Zhao and Tian, 2012). However, increasingly more complicated road network topology and higher traffic demands impose stricter requirements on the control strategy. The performance of SCOOT or SCATS can hardly meet today's requirements (Petrella et al., 2005). Despite many years of traffic signal control research, new solutions are urgently needed. To handle this problem, there are ongoing research efforts to adopt more advanced control methods using decentralized strategies to reduce the system costs.

The rapid development of Reinforcement Learning (RL) provides a novel tool for decision-making which is exactly what traffic control needs. Many existing studies in other fields requiring control algorithms have shown the efficiency and reliability of RL applications (Silver et al., 2016). Earlier this decade, the possibility of using RL in traffic control has been demonstrated by research (El-Tantawy et al., 2013). More recent research focusing on the implementation of deep reinforcement learning and multi-agent reinforcement learning also

demonstrated good results.

The remainder of this paper is organized as follows: Section 2 discusses the preliminaries and the definition of the traffic control problem, Section 3 introduces approaches using traditional controllers, Section 4 details the application of reinforcement learning, and Section 5 discusses the multi-agent reinforcement learning approaches.

2 Preliminary and Problem Formulation

Traffic signal control is a problem involving two scales: single-intersection control and multi-intersection coordination. The two scales create two different control problems in some settings, but they largely share a common problem definition. This section outlines a generic definition and discusses the notable distinctions between the two levels of control. Due to the complexity of the traffic signal control problem, we further divide this section into 4 parts for clarity: definitions of traffic network, traffic dynamic and state, traffic signal control, and metrics.

2.1 Traffic Network

Fundamental concepts of traffic are important in formulating the traffic signal control problem; therefore, we first briefly describe traffic network composition.

We formulate the traffic network as a directed graph where intersections and road links are nodes and edges respectively. It is difficult to define the number of lanes on a link since a link connecting two intersections could be inconsistent, with different parts of a link containing different numbers of lanes, such as the presence of pocket lanes. Therefore, we further divide a link into contiguous road segments at locations where the number of lanes is changing. Since we focus more on signalized intersections in the traffic control task, we define the number of incoming lanes in a link as the number of lanes in the segment adjacent to the intersection. At some approaches, the right-turn movement uses a dedicated right turn lane which is commonly unrestricted by the traffic signal. Therefore, we exclude right turn lanes from the number of incoming lanes.

Intersections in the traffic network are defined as the intersecting point of two or more links, and they are represented as nodes in the graph. Incoming links of an intersection are also called approaches and are usually described by the direction of traffic flow, such as the north-bound (NB) approach. Within an intersection, available movements are prescribed by connecting an incoming lane to an outgoing lane. The different movements conflict with one another, thus traffic signals are needed to facilitate each movement efficiently. Traffic lights at an intersection assign colours to movements to prevent conflicting and divert traffic: green, yellow (ember in some countries), and red. Traffic light commonly assigns non-conflicting movements into groups (one movement could belong to multiple groups) to fully utilize the green time (conflicting movements can also be assigned in one group by designating some of them as priority movements).

The combination of colours on all traffic lights at an intersection forms a phase. A phase is divided into two types: green phase and intermediate phase (yellow phase and all-red phase). Each of the movement groups posses a green phase that assigns green lights to the movements inside this group and red lights to others. Between two consecutive green phases, intermediate phases are inserted to clear the traffic inside the intersection for safety concerns. The duration of a phase is called the phase time. In general, traffic light cycles through all predefined phases in either a fixed order or an unfixed order, known as the traffic light logic cycle, to assign green lights to all movements, and the duration of a cycle is called the cycle time. However, some traffic lights do not cycle through all phases, and the concept of the cycle is not applicable.

We summarize the traffic network related notations here:

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is the traffic network. $\mathcal{G}$ is a directed graph. Node $v_i \in \mathcal{V}$ represents the intersection i in the traffic network. And edge $e_i \in \mathcal{E}$ represents the road link i in the traffic network.
- $\mathcal{N}(v_i)$ is the immediate neighbor set of intersection v_i.
- Intersection $v_i \in \mathcal{V}$ is defined by tuple $(\mathcal{E}_i^{in}, \mathcal{E}_i^{out}, \mathcal{M}_i)$. $e_j \in \mathcal{E}_i^{in}(\mathcal{E}_i^{out})$ is the incoming (outgoing) link. An incoming link is also called an approach. Each incoming (outgoing) link e_j has multiple incoming (outgoing) lanes $l_{j,k}$. $m_j \in \mathcal{M}_i$ is the prescribed movement defined by a pair of incoming and outgoing lanes.
- $\mathcal{P}_i$: the set of all available phases of the intersection i. $p_{i,j} \in \mathcal{P}_i$ is a phase of traffic light i.
- $\mathcal{P}_i^g$: the set of all green phases of the intersection i. $\mathcal{P}_i^g \subset \mathcal{P}_i$.
- $\phi_{i,j}$: the length of the phase $p_{i,j}$.
- $\phi_{i,j}^{min}(\phi_{i,j}^{max})$: the minimum (maximum) available length of the phase $p_{i,j}$.
- Traffic light i's logic cycle $l_i \in \mathcal{L}$ is defined by tuple $(\vec{p_i}, \vec{\phi_i})$. $\vec{p_i}$ is a vector representing a list of phases. $\vec{\phi_i}$ is a vector consists of phase times corresponding to all the phases in the vector $\vec{p_i}$. Vectors $\vec{p_i}$ and $\vec{\phi_i}$ have the same length that equals to the number of phases in a traffic light cycle. The sum of phase times is the cycle length: $\Psi_i = \Sigma_j \phi_{i,j}$.

2.2 Traffic State

Traffic signal control algorithms are designed to control the traffic flow in the traffic network. And any control method relies on sensing traffic information. Hence, here we define traffic states, display traffic sensing approaches, and describe traffic dynamics.

Traffic signal control involves both scales of traffic analysis: microscopic and macroscopic. We focus on every single vehicle's behavior in microscopic

analysis, and focus on macro-level traffic flow's behavior in macroscopic. In microscopic, each vehicle moves through the traffic network from its origin to a destination following a prefixed or dynamic generated (traffic assignment) path. Every vehicle generates a trip once it departed. In each time step, a vehicle in the network is on a certain road link. The vehicle leaves one link when it goes over the stop-bar of the last link. A vehicle is treated as stopping when its velocity is below a given value (threshold speed).

Macroscopic fundamental diagrams (MFD) together with three related traffic states - density, speed, and volume - are commonly used to describe traffic dynamics. In the macroscopic analysis, we divide roads into sections. A section could be either a road link, a road segment in a link, or a specific part of a link, which depends on the exact model. The traffic states are used to describe sections' traffic dynamic within analysis time window: density measures the number of vehicles per unit length (veh/km); speed measures vehicles' space-mean velocity (km/h), and volume measures the number of vehicles per unit time (veh/h). Three MFDs build relations between any two of the three key states and describe how traffic states evolve. Based on the key traffic states, macroscopic flow models can be built to describe the macroscopic traffic dynamic and contribute to the designing of traffic control algorithms. Modeling traffic dynamics is not one of the main points of this article, but here are some generally used examples: Lighthill and Whitham (LW) model, cell transmission model (CTM), etc.

It is also important to know how to gather the traffic information used in the controller. In the real-world, limited by the technology and sensor deployment cost, we cannot access any information on the road we want. The most commonly used information source is the inductive loop (loop detector) (Gordon et al., 2005) which is deployed under the road pavement and produces a signal when there is a vehicle upon it. With a specific design (a set of two inductive loops close to each other), loop detectors can also provide the vehicle's speed. Despite loop detectors advantage in technology maturity and reliability, its fixed-position characteristic limits the information itself could provide. In the microscopic scale, loop detectors can only provide the occupancy (existing vehicle or not) and vehicle passing-by speed of certain positions, which is extremely less compared with the actual traffic state (all vehicles' position, speed, acceleration rate, etc.). In the macroscopic scale, loop detectors can provide the traffic volume of certain positions, but still, the rest two key states (density and speed) cannot be acquired.

With the development of image processing and computer vision (CV) technologies, the cameras deployed on the traffic signal pole (together with signal lights) may provide more information in the near future: we can identify vehicles and their position and speed from the images/videos captured by those cameras with image identification algorithms (Tang et al., 2019). Meanwhile, vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication technologies are also under rapid development. With V2V/V2I communication, vehicles could share there exact states (position and speed provided by GPS and odometer systems) to the road-side units (RSU) or traffic information center

(Dey et al., 2016) to support the traffic signal control algorithms. Finally, some literature build traffic dynamic model based on hypothetical traffic states like demand and turning rates which cannot be collected in the field instantly with existing technology.

We summarize the traffic state related notations here:

- Sec_i: macroscopic traffic model analysis section i.
- $k_i(veh/km)$: the traffic density of the analysis section Sec_i.
- $u_i(km/h)$: the mean speed of the analysis section Sec_i.
- $f_i(veh/h)$: the traffic volume (flow) of the analysis section Sec_i.
- c_i: vehicle i.
- $v_i(km/h)$: the speed of vehicle c_i.
- $\mathcal{C}_j$: the set contains all the vehicle on road link e_j. $c_i \in \mathcal{C}_j$ represents that vehicle c_i is on road link e_j.
- q_i^t: the queue length of link e_i at time t.
- $D_{i,j}^t$: the cumulative stopping time of vehicle $c_i \in \mathcal{C}_j$ on link e_j at time t.
- $T_{i,j}^t$: the cumulative travel time of vehicle $c_i \in \mathcal{C}_j$ on link e_j at time t.

2.3 Traffic Signal Control Problem

Second-based vs. Cycle-based

With the full background and preliminary of the traffic network and dynamic, we can now define the traffic signal control problem in detail. Traffic signal controllers could be very different in terms of the decision-making time period: second-based and cycle-based controller.

However, in general, we define a generic (single- or multi-intersection case) traffic signal control problem as: find a policy π mapping historical traffic state/observation to the joint action $\vec{u}$ (second- or cycle-based) to optimize the traffic system's performance. The rest of this paper covers and discusses approaches of how to find the policy; while the rest of this section takes care of the other two fundamental issues in the definition: how second/cycle-based controller works and how to evaluate the traffic system's performance.

Second-based traffic signal controllers make decisions in each second or time-step (few seconds): how will the traffic light be in the next second/time-step. In detail, for intersection i, the controller select an available green phase $p_{i,j} \in \mathcal{P}_i^g$ in each time-step and set it for the next time-step. And if the next green phase is different from the current one, the yellow phase and/or all-red phase will be inserted in order to clear the intersection up for safety concern. Some controllers have constraints on the order of green phases and restrict the available actions

to two: extend and change. This type of controllers is called the extend-change controller.

Cycle-based controllers make decisions at the end of each cycle or a certain time-period (fixed length) and determine the exact timing plan for the next cycle/period. A generic traffic light cycle with three main elements can be represented by a pie chart as shown in Figure 1 (a): slices with different colors representing phases, the area of the pie representing the cycle length, and angles of slices representing the proportion of cycle time each phase taking. Notice, to keep the chart clear, we let each slice in the chart as a small phase group begin with a green phase and including following intermediate phases. Mathematically, a generic cycle-based traffic signal controller which can operate all three elements has the action based formulated as a tuple $(\mathcal{P}_i, \vec{\phi}_i^{min}, \vec{\phi}_i^{max})$. The controller generates and applies a new cycle from the tuple based on the observation after a cycle finished. Besides the generic type of cycle-based controllers, two more types of controllers have constraints on the three changeable elements. Fixed-order cycle-based controllers fix the phase set $\mathcal{P}_i$ as a vector $\vec{p}_i$ and keep the cycle time and phase distribution flexible. Phase split controllers fixed the phase order as well as the cycle time $\Psi_i = \Sigma_j \phi_{i,j}$ and only keep the phase distribution flexible.

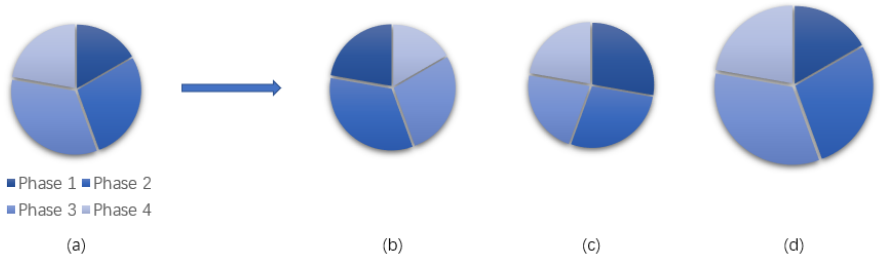


Figure 1: Traffic light cycles represented by pie charts have three changeable elements. (a) A traffic light cycle with 4 green phases. (b) Changed phasing order. (c) Changed phase distribution. (d) Changed cycle time.

Single-intersection vs. Multi-intersection

Despite the single-intersection control and multi-intersection coordination problems share most of the settings as we defined before, they do have some differences.

In the single-intersection control problem, we only consider the performance of that intersection and use intersection-level metrics as the goal of optimization. Compared with it, the multi-intersection control problem involves cooperation among all intersections within the analysis scope that makes the issue more complex. In the traffic network, an intersection's traffic state is influenced by their upstream and downstream neighbors: the spillback caused by down-

stream neighbors' heavy congestion limits the out-going flow of an intersection, and the traffic flow released from upstream neighbors increases the burden of this intersection. This characteristic is called the spatiotemporal coupling. An under-designed controller without considering the coupling and coordination may decrease the performance of the traffic network and even causes large-scale congestion like gridlock.

2.4 Metrics

Except for the algorithms finding the optimal policy, which we will cover in the rest of this paper, a multi-intersection controller is usually different from a single-intersection controller in the respect of evaluation metric.

Single-intersection level metrics commonly describe the traffic state directly. However, in the multi-intersection case, since intersections' traffic state is deeply coupled, a rise in a certain indicator of one intersection usually comes at the cost of a decline in neighbors' performance. Hence, for the multi-intersection scenario, besides metrics describing traffic state, some other indicators are designed for evaluating the effect of coordination.

Single-intersection Level Metrics

First, we introduce the intersection specific metrics. Apart from the metrics that will be introduced, another metric *delay* commonly used in the literature should be noted. The delay should be defined as the difference between the actual travel time and the ideal travel time (with free-flow speed) (Dowling, 2007). However, this definition is hard to evaluate and estimate in either the real-world or the simulation environment. The "delay" metric used in most relative literature should be referred to as the "stopping time" which indicates the time of vehicles driving below a certain speed. And in this case, the stopping time (or delay) in each second is equal to the number of stopped vehicles in quantity. Hence, we define the queue length and stopping time (delay) with the same equation.

- *Queue length (delay)*: the total queue length of an intersection:

$$Q^t = \Sigma_j q_j^t. \tag{1}$$

- *Cumulative delay* (El-Tantawy et al., 2013): the total cumulative delay (stopping time) of all vehicles on the incoming links:

$$CD^t = \Sigma_j \Sigma_i D_{i,j}^t. \tag{2}$$

  Not like the queue length, the cumulative delay increases rapidly when a vehicle waiting before a stop bar for a long period because of the integration of stopping time over all vehicles. This metric has two advantages: giving more penalty to the decision holding vehicles for a long period, and giving higher reward to the decision releasing vehicles with long delay.

- *Vehicle count*: the total number of vehicles on all incoming links:

$$VC^t = \Sigma_j |\mathcal{C}_j^t|, \tag{3}$$

where $|\cdot|$ means the size of the set.

- *Cumulative travel time*: the total cumulative travel time (stopping time) of all vehicles on the incoming links:

$$CTT^t = \Sigma_j \Sigma_i T_{i,j}^t. \tag{4}$$

- *Difference in cumulative delay*: the difference of the cumulative delay in two consecutive seconds:

$$DCD^t = CD^t - CD^{t-1}. \tag{5}$$

We can also define a metric as a combination of traffic states. For example, a commonly used cost function in the traffic field is:

$$Cost^t = a \cdot Q^t + b \cdot S^t, \tag{6}$$

where S^t is the number of stops of all vehicles within this intersection. a and b are two coefficients.

Network Level Metrics

For the multi-intersection scenario, one simple approach is to sum up a certain metric of all intersections. Besides, some other metrics defined above network level directly to evaluate the performance of global control.

- *Average travel time*: the average travel time of all completed trips:

$$ATT = \frac{1}{N} \sum_{i=1}^{N} T_i^{\mathcal{G}}, \quad c_i \in \mathcal{C}^{\mathcal{G}}, \quad |\mathcal{C}^{\mathcal{G}}| = N, \tag{7}$$

where $\mathcal{C}^{\mathcal{G}}$ is the set of vehicles that completed their trips. $T_i^{\mathcal{G}}$ is the travel time of the completed trip.

- *Average delay*: the average delay of all completed trips. Notice, here the delay also refers to the stopping time.

$$AD = \frac{1}{N} \sum_{i=1}^{N} D_i^{\mathcal{G}}, \quad c_i \in \mathcal{C}^{\mathcal{G}}, \quad |\mathcal{C}^{\mathcal{G}}| = N, \tag{8}$$

where $T_i^{\mathcal{G}}$: the stopping time (delay) of the completed trip.

- *Throughput* counts the number of completed trips:

$$Throughput = |\mathcal{C}^{\mathcal{G}}|. \tag{9}$$

Coordination Indicators

Finally, we introduce link-specific indicators designed to evaluate the performance of coordination between two adjacent intersections.

Arrival type (Manual, 2000) is an indicator with 6 levels describing the progression of a link **qualitatively**. All 6 levels' definitions are listed in the last two columns of Table 1. The definition of arrival type is intuitive, but it is hard to measure quantitatively: researchers have to observe the traffic flow in the field and assign arrival type to each link based on what they perceived. Hence, the platoon ratio is introduced so that the progression can be measured by sensors **quantitatively**.

Table 1: Relation between the arrival type and the platoon ratio

Platoon Ratio	Arrival Type	Description of Flow
0.333	1	Very poor progression
0.667	2	Unfavorable progression
1.000	3	Uncoordinated signals or random arrivals
1.333	4	Favorable progression
1.667	5	High favorable progression
2.000	6	Exceptional progression

Platoon ratio is defined as the ratio between two quantities: the proportion of vehicles arriving during the green phase in a full cycle, and the proportion of the green phase time in the cycle time. We can also derive the platoon ratio from the proportion of two flow rates:

$$R_p = \frac{P}{g/c} = \frac{f_g}{f}, \tag{10}$$

where P is the ratio of vehicles arriving during the green phase. g/c is the ratio of green phase time over the cycle time. f_g and f are the flow rates during the green phase period and the whole cycle respectively. The higher platoon ratio means more vehicles come from the upstream intersection could pass by the downstream one without stop, which indicates better coordination between the two intersections. The relation between the platoon ratio and the arrival type is given by Table 1.

3 Traditional Control Methods

In recent research, many advanced control strategies have been adopted into the field of traffic signal control. Dependent on the architecture of the control system, we can roughly separate them into two fields: optimization approaches finding the minimum cost of the whole system, and closed-loop controllers. In this section, these two branches will be discussed in detail separately.

3.1 Optimization Approach

In order to minimize the delay or travel time by traffic signal control, the most intuitive way is to form the signal timing plan as an optimization problem.

Dynamic Programming (DP) is a famous mathematical optimization strategy to find the optimal solution. DP breaks the problem into multiple solvable sub-problems and then finds the optimal solutions to the sub-problems recursively. (Chen and Sun, 2016) adopts DP in a isolated signalized intersection control problem. This work uses a microscopic model to estimate vehicle arrival at the end of the queue and considers three objective functions separately: the minimization of the delay, the minimization of the queue length, and maximization of the throughput. Under the circumstance of vehicle infrastructure integration (VII), more information can be used to design a more refined optimization method. In (Yao et al., 2019), with the prediction of traffic arrival using a dynamic platoon dispersion model, a dynamic programming algorithm with a rolling optimization scheme is adopted to minimize the intersection delay under the constraint of green time duration.

Considering the time-varying travel demand, (Chiou, 2019) proposes a two-stage model to simultaneously minimize the total travel delay over multiple periods with stochastic travel demand. And a period-dependent mathematical program with equilibrium constraints (PMPEC) was designed to mitigate the stochastic risk. Besides considering the required resources of classic dynamic programming, (Cai et al., 2009) introduces the approximate dynamic programming (ADP) method to find the approximation of the optimal solution by replacing the origin value function with a linear function. This approach reduces the computational burden of dynamic programming significantly.

3.2 Closed-loop Control

The closed-loop controller makes decisions based on feedback from the observed system outputs in order to achieve higher performance and stability. Most of the widely used controllers nowadays are in the closed-loop form.

Model predictive control (MPC) is a practical and powerful method to control a complex process with a set of constraints (Morari and Lee, 1999). MPC takes the future into consideration by generating the prediction based on the dynamic model of the controlled object. Like rolling optimization, MPC also works in a rolling horizon framework and updates control in real-time. (Hegyi, 2004) uses MPC for traffic signal control and discusses the necessary conditions for dynamic speed limits. Instead of studying an isolated intersection, (Lin et al., 2012) constructs a network-wide traffic controller based on the MPC theory. This research forms two macroscopic models as the predictor of the MPC. These two models focus on accuracy and computational efficiency separately to seek balance. Simulation results show that the MPC is capable of controlling unevenly distributed traffic flow in the network. Also for the network-level control task, (De Oliveira and Camponogara, 2010) decomposes a centralized control problem into coupled sub-problems and solves them with a distributed

multi-agent model predictive controller.

The max pressure controller is a distributed network control policy designed to solve the routing problem in the multi-hop communication network (Tassiulas and Ephremides, 1990). Given the knowledge of the traffic status in both upstream and downstream intersections, the max pressure controller gives priority to the phase which can serve the most demand in order to maximize the throughput of the network. (Anderson et al., 2018) proves the stability of the adoption of the max pressure controller in traffic signal control, while introducing a cycle-based max pressure control law to match the model's dynamics. This method was designed for distributed control and can handle the network-level traffic control task properly.

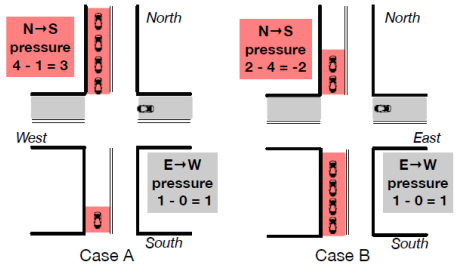


Figure 2: Illustration of the max pressure theory in two cases

All the methods discussed above focus on centralized or decentralized strategies. Hence, some researchers combined these two strategies together to form hierarchical control frameworks. (Ramezani et al., 2015) proposes a two-aggregated-model approach, which consists of regional and sub-regional macroscopic fundamental diagrams (MFD) to study the dynamics in different levels and minimize the network delay. In this approach, the higher-level control problem is solved by MPC, while the lower-level control problem is solved by a simple feedback controller. Further, (Kouvelas et al., 2018) incorporates the max pressure controller into the hierarchical control frameworks by replacing the lower-level feedback controller. For the upper-level control, a PI (proportional integral) regulator is introduced to control the transferring flows between regions.

The rule-based fuzzy controller was designed to process uncertainties in the system. To handle the uncertainty in the traffic system, (Bi et al., 2017) applies a type-2 fuzzy controller. This work also proposes a two-layer controller, the basic control layer and the system-level coordination layer. The basic controller allocates the green time for each intersection, while the upper-level controller coordinates the green time between two consecutive intersections. Based on this, (Srinivasan et al., 2006) introduces a fuzzy neural network and integrates the simultaneous perturbation stochastic approximation theorem to refine the controller.

4 Reinforcement Learning Approach

Reinforcement learning (RL) is a machine learning paradigm that learns how to act in an environment such that the expected sum of rewards is maximized.

4.1 Deep Reinforcement Learning

Traditional tabular Q-learning cannot generalize the learned experience to the state-action pairs that never visited. The non-linear neural network provides a feasible way to approximate the Q-function and alleviate the generalization issue by parameterization (Mnih et al., 2015). Genders et al. (Genders and Razavi, 2016) introduces deep Q-network (DQN) to the single isolated intersection control task with a discrete traffic state representation. The proposed Discrete traffic state encoding (DTSE) representation is shown as Figure 3. In the representation of a discrete traffic state, a convolutional neural network (CNN) can then be used by the agent as the non-linear approximation.

Deep Q-learning represents the learning of the action-value function which is parameterised by θ. Deep Q-Networks (DQNs) use a *replay buffer* to store the transition tuple $\langle s, u, r, s' \rangle$ where u is the action taken in state s, and r is the reward received, s' is the state reached upon taking that action. Batches are sampled from the replay memory and θ is updated by minimizing the squared *TD-error*:

$$\mathcal{L}(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{\mathrm{DQN}} - Q(s, u; \theta) \right)^2 \right] \tag{11}$$

where $y^{\mathrm{DQN}} = r + \gamma \max_{u'} Q(s', u'; \theta^-)$. θ^- represents the parameters of the *target network* that are kept constant some iterations and periodically copied from θ.

(b)

0	0	0	0	0	0	0	0	1	0	0	1
1	0	0	0	1	0	0	0	1	0	1	1
0	1	0	0	0	0	1	0	0	1	0	1

(c)

0	0	0	0	0	0	0	0	0.2	0	0	0
0.8	0	0	0	0.5	0	0	0	0	0	0	0
0	0.6	0	0	0	0	0.4	0	0	0	0	0

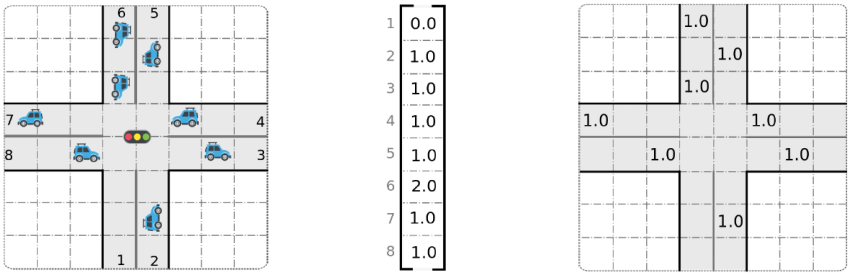
Figure 3: Illustration of the discrete traffic state representation with Boolean and real-valued vectors. (a) An approach if an intersection. (b) The matrix representing the occupancy. (c) The matrix representing the normalized vehicle velocity.

4.2 Traffic Control with Reinforcement Learning

RL can be used as a powerful control tool (Silver et al., 2016) that handles the traffic control task well since it does not need additional assumptions on the underlying traffic distributions. El-Tantawy et al. (El-Tantawy et al., 2013) adopts a multi-agent reinforcement learning algorithm for large-scale coordination control. In this approach, the agent (Q-learning agent) for each intersection shares their information with adjacent intersections to achieve coordinated control. For each agent, the state space consists of the elapsed time of the current phase and the maximum queue lengths of each phase, while the action space is defined by the phasing sequence. The reward of the whole system is given by the sum of the reward of each intersection which is the reduction in the total cumulative delay.

The reinforcement learning method can also be combined with the concept of max pressure theory. Wei et al. (Wei et al., 2019) uses pressure to represent the state space and chooses the negative pressure of intersection as the reward. Hence, this method aims to minimize the pressure at each intersection to maximize the throughput of the whole network.

Different from (Genders and Razavi, 2016), many other approaches tend to find better representations of the traffic state. (Nishi et al., 2018) meshes the snapshot of the intersection directly into a grid in order to adopt a CNN as shown in Figure 4. Following the similar concept, (Wei et al., 2018) designs a more complex state space with finer meshed intersection snapshot and applies a CNN with historical information to generate the control law.



(a) Snapshot of the intersection (b) Vector with the number of vehicles per lane (c) Matrix representation

Figure 4: Illustration of a matrix representation of a intersection

However, this meshing operation is not a proper approach to the represent the traffic state. Firstly, not all intersections have this symmetric appearance. Secondly, the matrix contains many cells without any meaning in the actual world, such as the cells that correspond to the space outside the road. Hence, how to design a well-defined traffic state representation is still a problem.

To avoid the meshing of intersection's snapshot and to seek a better approach of coordination, (Tan et al., 2019) still uses the queue lengths as the state space, and applies a multi-layer perceptron with ReLU function rather than a CNN.

This approach also forms a hierarchical control framework and replaces the DQN with a deep deterministic policy gradient (DDPG) method in the upper-level controller. Finally, a cross-regional coordinator is proposed in this article to explicitly learn a global Q-function. The architecture of the three-level controller is shown in Figure 5.

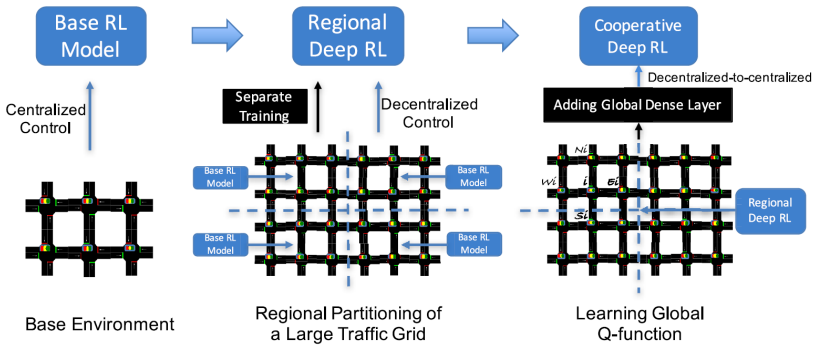


Figure 5: Illustration of the different levels of the Coder system

5 Multi-Agent Reinforcement Learning

Reinforcement Learning as a solution methodology that can be applied to single-agent systems as well as multi-agent systems. In similar settings, multi-agent systems are often much more complex as compared to single-agent systems. One of the reasons for the increase in complexity is the increase in size of the action-space as the number of agents increase. For example, if we consider a situation where a single global agent controls all the agents in a multi-agent system, for such a global agent the action-space increases exponentially in the number of agents. Another cause for increased complexity is increase in observation-space with increase in number of agents. This is especially true for partially observable scenarios, where global state isn't available and agents are only able to perceive their local observations. Solution methodology used in multi-agent systems also depends on the constraints imposed by the system. One such constraint is of information-sharing (or communication) between agents, which can vary from no information-sharing to complete information-sharing. Depending upon this constraint, the set of eligible reinforcement learning methodologies will also change.

Most single-agent reinforcement learning problems are formulated as Markov Decision Processes (MDPs). Under this framework, it is assumed that the agent has access to the Markov State of the system and also the agent's Policy is conditioned over this state. For the multi-agent system the above assumption translates to each agent possessing the global Markov State of the system. For most multi-agent systems this assumption will likely be false, as in such systems each agent generally has access to only its own observation. Furthermore, as

the number of agents increase in a multi-agent system it becomes increasingly difficult to satisfy the assumption that all agent would have access to the global State of the system. Therefore we need other formulations to consider problems under multi-agent systems.

Types of Multi-Agent Systems

- Cooperative:
  Represents those multi-agent systems where the agents work together towards a joint reward. The reward received is based on the actions performed by all the agents. This is the coordination problem where agents need to work together collectively.

- Competitive:
  Represents those multi-agent systems where agents compete with one another for the reward. These systems are also called as zero-sum games, and the agents have individual opposing rewards. We have solution concepts for these such as the Minimax Equilibria.

- Mixed:
  These systems are a combination of the above two systems. These are neither fully competitive nor neither fully cooperative. These are also called as general-sum games. Agents here receive individual rewards that are based on cooperative and competitive aspects of the environment.

Dec-POMDP

Dec-POMDP (Oliehoek et al., 2016) or Decentralized-Partially Observable Markov Decision Process is one of the ways to formulate a multi-agent system. All parameters corresponding to Dec-POMDP are part of the following tuple $G = \langle S, U, P, r, Z, O, n, \gamma \rangle$. $s \in S$ describes the true state of the environment. At each timestep, each agent $a \in A \equiv \{1, ..., n\}$ picks an action $u^a \in U$ which forms the joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$ for all agents. The transition in the environment corresponding to the joint action is modelled using state transition function $P(s'|s, a) : S \times \mathbf{U} \times S \to [0, 1]$. The reward defined by the reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to \mathbb{R}$ is shared amongst all agents. The discount factor is $\gamma = [0, 1)$.

We consider a *partially observable* scenario in which each agent draws individual observations $z \in Z$ according to observation function $O(s, a) : S \times A \to Z$. Each agent has an action-observation history $\tau^a \in T \equiv (Z \times U)^*$, on which it conditions a stochastic policy $\pi^a(u^a|\tau^a) : T \times U \to [0, 1]$. The joint policy π has a joint *action-value function*: $Q^\pi(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\inf}, \mathbf{u}_{t+1:\inf}}[R_t|s_t, \mathbf{u}_t]$, where $R_t = \sum_{i=0}^{\inf} \gamma^i r_{t+i}$ is the discounted return.

The assumptions of Dec-POMDP hold true for the traffic environment as well, since in the traffic environment we do not have access to the state of the intersection, but only an observation. In the following sub-sections we would discuss the various methodologies in which reinforcement learning can be used

with multi-agent systems. Most of them work under the Dec-POMDP formulation, while some use the MDP formulation. Further details to be expanded below.

5.1 Independent Agents

When used with Q-Learning this approach is also known as Independent Q-Learning (IQL) (Tan, 1993). In this approach to multi-agent RL, each agent acts independently by training over only its own local observations. An agent cannot view other agent's observations, so this is also the case of no information-sharing. For an agent in IQL, other agents and the environment together constitute the perceived environment. This means that as some agents learn, the environment perceived by other agents changes. This is why IQL training is generally unstable as the environment perceived by the agent is non-stationary. This is also the reason why there are no convergence guarantees for IQL as compared to Q-learning. But the advantage of the IQL approach is scalability: as each agent learns independently, the computation increases linearly in number of agents. Also, IQL serves as a surprisingly strong benchmark even in mixed and competitive games (Tampuu et al., 2017).

5.2 Global Agent

In this approach we have a single-agent that controls all the agents in our environment. This can be formulated as an MDP or POMDP depending upon the environment constraints: whether the environment allows access to the Markov State or only to the local observations. The major assumption made by this methodology is that perception information generated at each location is available at a centralized position, for the single global agent to utilize for decision-making. And this may not be true for most multi-agent scenarios due to communication constraints. The non-stationarity issue faced by IQL is not faced by Global Agent, as the environment perceived by the global agent is stationary. The downside faced by Global Agent is scalability as the action-space scales exponentially in number of agents. And in the case when local observations are made available in place of the global state, the observation space also increases exponentially in the number of agents.

5.3 VDN: Value Decomposition Networks

Value Decomposition Networks (VDN) (Sunehag et al., 2018) attempts to factorize the Q_{tot} which is the action-value for the joint-action $\mathbf{u}_t$. They represent Q_{tot} as a sum of individual agent action-values Q_a that condition only on individual actions and observations. Because of this setup, a decentralized policy arises simply from each agent selecting actions greedily with respect to its Q_a. VDN learns joint action-value function $Q_{tot}(\boldsymbol{\tau}, \mathbf{u})$, where $\boldsymbol{\tau} \in \mathbf{T} \equiv \mathcal{T}^n$ is joint

action-observation history and $\mathbf{u}$ is joint action.

$$Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \sum_{i=1}^{n} Q_i(\tau^i, u^i; \theta^i) \tag{12}$$

The VDN loss function is similar to (11), where Q is replaced by Q_{tot}. The advantage of this representation is that a decentralized policy arises from each agent performing greedily with respect to its action-value function Q_a.

5.4 QMIX: Monotonic Value Function Factorization

VDN allows us to train agents in a centralized setting, and deploy them in a decentralized manner. However, VDN assumptions severely limit the complexity of centralized action-value functions that can be represented through its architecture. QMIX (Rashid et al., 2018) tries to address this issue by claiming that the factorization need not be that constrained and that we only need to ensure that argmax performed on Q_{tot} yields the same results as the argmax performed on individual action-values Q_a of each agent:

$$\underset{\mathbf{u}}{\operatorname{argmax}}\, Q_{tot}(\boldsymbol{\tau}, \mathbf{u}) = \begin{pmatrix} \operatorname{argmax}_{u^1} Q_1(\tau^1, u^1) \\ \vdots \\ \operatorname{argmax}_{u^n} Q_n(\tau^n, u^n) \end{pmatrix} \tag{13}$$

This allows each agent to participate in decentralized execution by just choosing actions greedily based on Q_a. And the added benefit is that the argmax of Q_{tot} required for off-policy updates is also quite tractable.

VDN representation satisfies (13), but the constraint is also valid for a large number of representations which VDN cannot represent and QMIX is able to. The family of monotonic functions is a sufficient but not necessary condition to satisfy (13). And the monotonicity can be implemented via a constraint on the relationship between Q_{tot} and Q_a:

$$\frac{\partial Q_{tot}}{\partial Q_a} \geq 0, \ \forall a \in A \tag{14}$$

QMIX architecture consists of agent networks, mixing network, and a hypernetwork (Ha et al., 2016) to enforce (14). The QMIX architecture is shown in Figure 6. The mixing network here is a feed-forward neural network that takes agent action-values Q_a as input and outputs the joint action-value function Q_{tot} as shown in Figure 6a. The monotonicity constraint of (14) is to be applied on the weights (but not biases) of the mixing network, and the weights need to be positive.

Plots shown in Figure 7 show the mean test win rate across multiple games with varieties of algorithms, namely IQL, VDN and QMIX. As we can see from the plots, IQL is a good baseline that on certain environments even challenges VDN. This is true for the the 5 Marines (5m) environment as shown in Figure 7b. But we also see that IQL fails to beat the enemy consistently, as well as

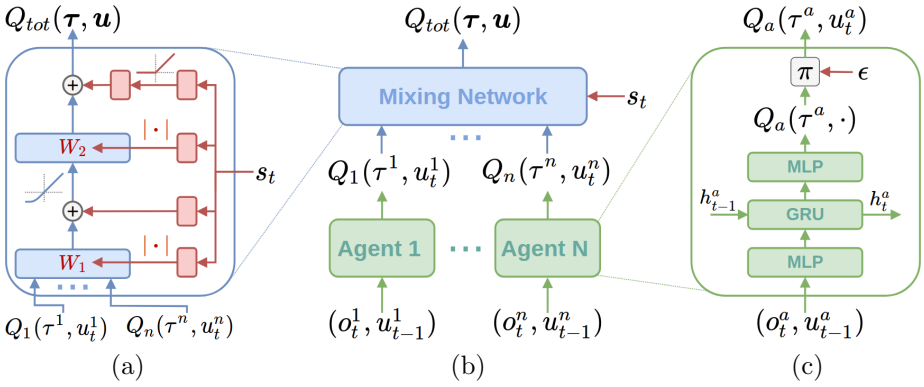
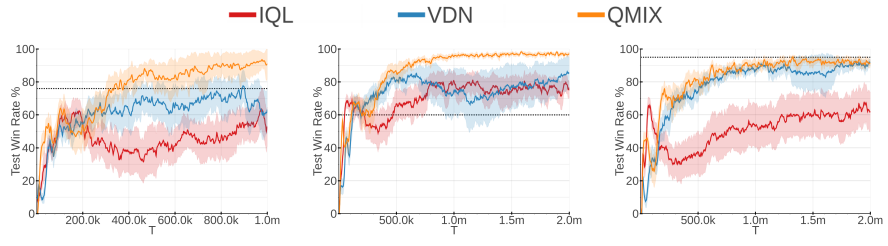


Figure 6: (a) Mixing Network structure. (b) Complete QMIX architecture. (c) Agent Network.

suffers from unstable training which becomes more apparent as the environments grows in complexity. The benefit of learning the joint action-value function can be seen by VDN beating IQL consistently on all environments. VDN starts to learn coordination activities such focus-firing which helps it achieve superior performance on many environments. But on environments such as 3m where there are fewer players and more fine-grained control is required, QMIX performs much better. To gauge the complexity between these environments it is useful to view the performance of the heuristic-based algorithm which is shown by a dotted line in Figure 7.



(a) 3m (b) 5m (c) 8m

Figure 7: Win rates for IQL, VDN, and QMIX on different team compositions in StarCraft II.

QMIX: Traffic Control Domain

In the traffic domain, there are multiple different design choices we have to make to use QMIX for traffic signal control tasks. The major design choices available are related to Observation definition, Agent Network architecture, Hypernetwork architecture and the State definition which is the input to the Hypernet-

work. When selecting the Observation definition we have the same options that were available during the training of IQL and Global Agents. The observations could be based on short-range detection (TDTSE, etc.), long-range detection or the different variations in each of the these categories. The Agent Network can be CNN-based agent that is fed the TDTSE observation or it can be RNN-based agent that uses detector activations at each timestep as the input. The design choices taken for the Hypernetwork design and the State design are interdependent as the State is the input of the Hypernetwork. There are three major variations:

- Concatenated Vectorized Observations:
  The TDTSE observations are vectorized and concatenated to form the State that is passed into the Hypernetwork.

- Concatenated Observations with CNN-based Hypernetwork:
  The TDTSE observations retain its image-like shape, and the Hypernetwork architecture is changed from using Fully-Connected layers to Convolutional layers. This reduces the number of the weights inside the Hypernetwork and should streamline the learning.

- Composition of Intersection Statistics:
  Instead of composing the State as a concatenation of TDTSE observations, here the State is composed of intersection statistics such as Lane Occupancy, Queue Length, Average Speed, and others. This allows ready-made information to be supplied to the Hypernetwork so it does not need to generate these things from the TDTSE observations as done by previous two configurations.

5.5 ROMA: Value Function Factorization with Emergent Roles

ROMA (Role Oriented Multi Agent, Wang et al. (2020)) is a CTDE algorithm built on top of QMIX. It starts with the claim that CTDE architectures might no be enough for complex tasks that can be divided into sub-tasks requiring a collection of skills, or as they call it, roles. A roles is a pattern of behavior, and agents sharing roles will share the same behaviors, or trajectories.

ROMA responds by introducing a mechanism to enforce the agents learn and adopt particular roles. Roles must fullfits the following properties:

- Identifiable: a role correspond to a specific behavior.

- Specialized: agent sharing roles should share similar responsibilities.

- Dynamic: agent's role changes over time, adapting to the current state of the environment.

- Versatile: sampled role should be diverse enough to solve the task at hand.

Those properties are instantiated through two regularizations of the QMIX loss function. But first, to enforce the dynamic property, each agent i samples at each time step a role ρ_i from the stochastic embedding roles-space, conditionned on the current observation o_i:

$$(\boldsymbol{\mu}_{\rho_i}, \boldsymbol{\sigma}_{\rho_i}) = f(o_i; \theta_\rho), \rho_i \sim \mathcal{N}(\boldsymbol{\mu}_{\rho_i}, \boldsymbol{\sigma}_{\rho_i}), \tag{15}$$

where θ_ρ is the role encoder parameters to learn.

To make a behavior identifiable, they minimize the conditional entropy $H(\rho_i|\tau_i, o_i)$, and to generate versatility they maximise $H(\rho_i|o_i)$. Those optimisation problems can be expressed as maximising the conditional mutual information (CMI) between a trajectory and a role, given the current observation. Because CMI is intractable, they use a variational estimator (Wainwright et al., 2008; Alemi et al., 2017) parametrized by ξ. They combine a GRU (Cho et al., 2014) trajectory encoder to give the following loss:

$$\mathcal{L}_I(\theta_\rho, \xi) = \mathbb{E}_{(\tau_i^{t-1}, o_i^t)\sim D}\left[D_{\mathrm{KL}}[p(\rho_i^t|o_i^t)\|q_\xi(\rho_i^t|\tau_i^{t-1}, o_i^t)]\right], \tag{16}$$

where $\mathcal{D}$ is a replay buffer, $\boldsymbol{\tau}^{t-1}$ is the joint trajectory, and $D_{\mathrm{KL}}[\cdot\|\cdot]$ is the KL divergence operator.

Specialisation is enforced by introducing a dissimilarity model for trajectories d_θ. Let $D_\phi = (d_{ij})$ where $d_{ij} = d_\phi(\tau_i, \tau_j)$ is the estimated dissimilarity between trajectories of agent i and j then ROMA solve the following optimisation problem:

$$\begin{aligned} &\underset{\theta_\rho, \xi, \phi}{\text{minimize}} \quad \|D_\phi^t\|_{2,0} \\ &\text{subject to} \quad I(\rho_i^t; \tau_j^{t-1}|o_j^t) + d_\phi(\tau_i^{t-1}, \tau_j^{t-1}) > U, \forall i \neq j, \end{aligned} \tag{17}$$

where U controls the compactness of the role representation and F is the Frobenius norm. Because this problem is untractable, the second regularizer is expressed as:

$$\begin{aligned} \mathcal{L}_D(\theta_\rho, \phi, \xi) = \mathbb{E}_{(\boldsymbol{\tau}^{t-1}, \boldsymbol{o}^t)\sim\mathcal{D}, \boldsymbol{\rho}^t\sim p(\boldsymbol{\rho}^t|\boldsymbol{o}^t)}\big[&\|D_\phi^t\|_F \\ &- \sum_{i\neq j} \min\{q_\xi(\rho_i^t|\tau_j^{t-1}, o_j^t) + d_\phi(\tau_i^{t-1}, \tau_j^{t-1}), U\}\big] \end{aligned} \tag{18}$$

where $\mathcal{D}$ is the replay buffer, $\boldsymbol{\tau}^{t-1}$ is the joint trajectory, $\boldsymbol{o}^t$ is the joint observation, and $\boldsymbol{\rho}^t = \langle \rho_1^t, \rho_2^t, \cdots, \rho_n^t \rangle$.

Finally ROMA final loss is the following:

$$\mathcal{L}(\theta) = \mathcal{L}_{TD}(\theta) + \lambda_I \mathcal{L}_I(\theta_\rho, \xi) + \lambda_D \mathcal{L}_D(\theta_\rho, \xi, \phi), \tag{19}$$

where lambdas are scaling factors and $\mathcal{L}(\theta)$ the QMIX total loss. The whole process is described Figure. 8. Through a Starcraft II experiment, they shown that ROMA outperforms QMIX on this specific problem, as plotted on Figure. 9

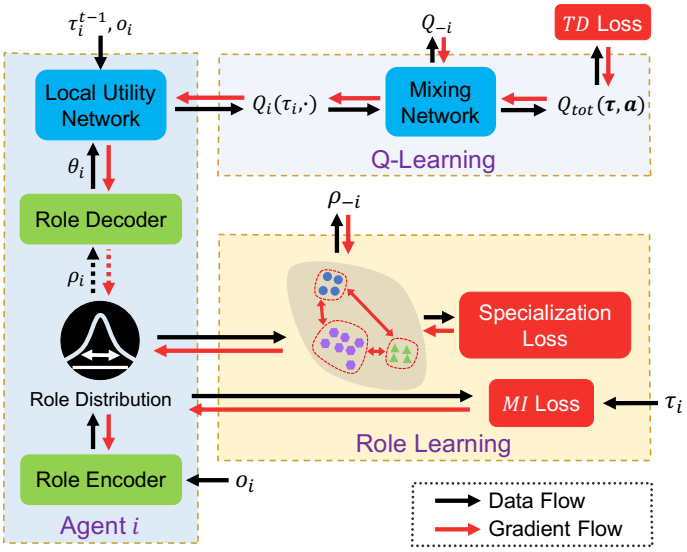


Figure 8: ROMA framework

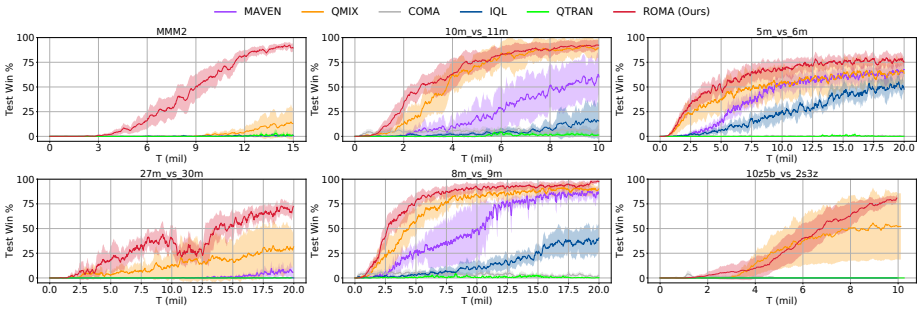


Figure 9: ROMA performances on Starcraft 2 environments.

ROMA and the traffic control problem. ROMA plays the same role as QMIX in the traffic control problem. However, it should be able to create specialized roles for some intersections. For example, it could discover how to protect a jammed perimeter by assigning "guarding" roles to the surrounding intersections. One must note that the practitioner does not specify himself the roles space. Roles space is directly learnt by the ROMA framework. With that in mind, one could use ROMA as a role discoverer. Indeed, it could find roles in simulation that we would not know about in real life. We could implement those roles in practice with an handcrafted solution (or directly with the ROMA agents). Please refer to Figure. 10 to see an example of a learnt roles space in Starcraft II. Those roles are typically used by professional players to micro-manage their units; They already know about those, but it is fair to assume that it is not straightforward and ROMA could have help discover them.

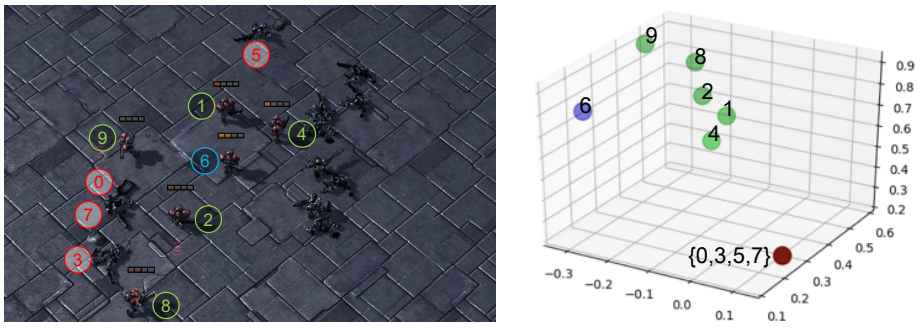


Figure 10: Learned roles. Red are dead marines, blue are marines with low life, and green are marines with high life. Usually marines with a lot a life remaining should "tank" the bullets to protect injured marines, ie they should be moved in front of other marines.

5.6 Graph Attention Layer: Enhancement with Graph Topology

One property distinguishes multi-intersection traffic control problem with other multi-agent systems is that the topology of intersections is clear and fixed by the road network. The road network causes the spatiotemporal coupling issue as we discussed in Section 2.3. It also informs us that each intersection is not connecting to others extensively but only connects to the nearest neighbors. Hence, we could utilize the structure of the graph and handle the information in a more efficient way with the help of graph neural networks (GNNs). GNNs under rapid development have multiple branches identified by their information-gathering methods (Zhou et al., 2018). Because of the local connection property of the traffic (an intersection's traffic state is barely affected by other intersections far from it), the graph attention network (GAT) (Veličković et al., 2017) which embeds node features with neighbors' information is the best candidate in solving the multi-agent traffic control problem.

The attention mechanism was designed to identify the importance of each part of the input features to the output in the machine learning field. The GAT utilizes multi-head attention on the graph to create node embeddings as the weighted sum of input features. Each GAT layer gather information from the nodes and their 1-hop neighbors. By overlaying multiple GAT layers, the GAT network could create higher-level node embeddings with more plentiful information from farther neighbors. Take a simple arterial traffic network with 5 intersections as an example: as shown in Figure 11, the node 3's embedding at the last GAT layer is determined by the information from the whole graph (the farthest neighbor of node 3 is its 2-hop neighbor).

Compared with other GNNs, the GAT advantages in several fields: the receptive field of each node is flexible, the relationships between node pairs are learnable, and the node embeddings only depend on neighbors' information

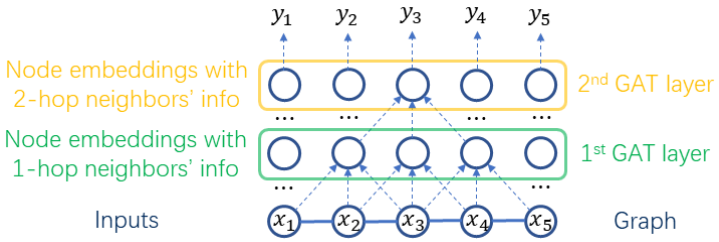


Figure 11: A 2-layer GAT network on an example graph representing an artery with 5 intersections. Dash arrows represent multi-head attention weights.

rather than global's. These advantages give the GAT flexibility and expressiveness. With the GAT, we can build a global DQN agent taking advantage of the graph topology. As shown in Figure 12, local CNNs pre-process the intersections' raw observation to high-level features and feed them to the global GAT. As a global information handler, the GAT layers on top of CNNs create node embeddings including the influence from neighbor intersections and generate the Q-values corresponding to the joint action.

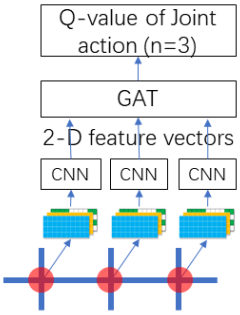


Figure 12: A global DQN agent with GAT layers handling the graph topology.

Despite we are still working on the GAT + RL approach for multi-intersection traffic coordination and didn't have a definite solution yet, some recent literature has shown that the capability of combining the GAT with RL. (Chen et al., 2020) combines the GAT with the actor-critic RL framework. Experiments based on a cooperation environment shows that the GAT enabled framework outperforms the MADDPG algorithm in both effectiveness and scalability. (Zhou et al., 2020) divides agents into groups by using the GAT to solve sub-tasks in a cooperation problem. This work uses the QMIX framework and embeds GAT networks into local DQNs. Experiments in the StarCraft II environment indicates that this approach performs better than the vanilla QMIX framework.

References

Robert L Gordon, Warren Tighe, ITS Siemens, et al. Traffic control systems handbook. Technical report, United States. Federal Highway Administration. Office of Transportation ..., 2005.

Yi Zhao and Zong Tian. An overview of the usage of adaptive signal control system in the united states of america. In *Applied Mechanics and Materials*, volume 178, pages 2591–2598. Trans Tech Publ, 2012.

Margaret Petrella, Stacey Bricka, Michael P Hunter, and Jane Lappin. Driver satisfaction with an urban arterial after installation of an adaptive signal system. In *12th World Congress on Intelligent Transport SystemsITS AmericaITS JapanERTICO*, 2005.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

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

Zheng Tang, Milind Naphade, Ming-Yu Liu, Xiaodong Yang, Stan Birchfield, Shuo Wang, Ratnesh Kumar, David Anastasiu, and Jenq-Neng Hwang. Cityflow: A city-scale benchmark for multi-target multi-camera vehicle tracking and re-identification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8797–8806, 2019.

Kakan Chandra Dey, Anjan Rayamajhi, Mashrur Chowdhury, Parth Bhavsar, and James Martin. Vehicle-to-vehicle (v2v) and vehicle-to-infrastructure (v2i) communication in a heterogeneous wireless network–performance evaluation. *Transportation Research Part C: Emerging Technologies*, 68:168–184, 2016.

Richard Dowling. Traffic analysis toolbox volume vi: Definition, interpretation, and calculation of traffic analysis tools measures of effectiveness. Technical report, 2007.

Highway Capacity Manual. Highway capacity manual. *Washington, DC*, 2, 2000.

Shukai Chen and Daniel Jian Sun. An improved adaptive signal control method for isolated signalized intersection based on dynamic programming. *IEEE Intelligent Transportation Systems Magazine*, 8(4):4–14, 2016.

Zhihong Yao, Luou Shen, Ronghui Liu, Yangsheng Jiang, and Xiaoguang Yang. A dynamic predictive traffic signal control framework in a cross-sectional vehicle infrastructure integration environment. *IEEE Transactions on Intelligent Transportation Systems*, 2019.

Suh-Wen Chiou. A two-stage model for period-dependent traffic signal control in a road networked system with stochastic travel demand. *Information Sciences*, 476:256–273, 2019.

Chen Cai, Chi Kwong Wong, and Benjamin G Heydecker. Adaptive traffic signal control using approximate dynamic programming. *Transportation Research Part C: Emerging Technologies*, 17(5):456–474, 2009.

Manfred Morari and Jay H Lee. Model predictive control: past, present and future. *Computers & Chemical Engineering*, 23(4-5):667–682, 1999.

András Hegyi. *Model predictive control for integrating traffic control measures*. Netherlands TRAIL Research School, 2004.

Shu Lin, Bart De Schutter, Yugeng Xi, and Hans Hellendoorn. Efficient network-wide model-based predictive control for urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 24:122–140, 2012.

Lucas Barcelos De Oliveira and Eduardo Camponogara. Multi-agent model predictive control of signaling split in urban traffic networks. *Transportation Research Part C: Emerging Technologies*, 18(1):120–139, 2010.

Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. In *29th IEEE Conference on Decision and Control*, pages 2130–2132. IEEE, 1990.

Leah Anderson, Thomas Pumir, Dimitrios Triantafyllos, and Alexandre M Bayen. Stability and implementation of a cycle-based max pressure controller for signalized traffic networks. *Networks & Heterogeneous Media*, 13(2):241, 2018.

Mohsen Ramezani, Jack Haddad, and Nikolas Geroliminis. Dynamics of heterogeneity in urban networks: aggregated traffic modeling and hierarchical control. *Transportation Research Part B: Methodological*, 74:1–19, 2015.

Anastasios Kouvelas, Dimitris Triantafyllos, and Nikolas Geroliminis. Two-layer hierarchical control for large-scale urban traffic networks. In *2018 European Control Conference (ECC)*, pages 1295–1300. IEEE, 2018.

Yunrui Bi, Xiaobo Lu, Zhe Sun, Dipti Srinivasan, and Zhixin Sun. Optimal type-2 fuzzy system for arterial traffic signal control. *IEEE Transactions on Intelligent Transportation Systems*, 19(9):3009–3027, 2017.

Dipti Srinivasan, Min Chee Choy, and Ruey Long Cheu. Neural networks for real-time traffic signal control. *IEEE Transactions on intelligent transportation systems*, 7(3):261–272, 2006.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

Wade Genders and Saiedeh Razavi. Using a deep reinforcement learning agent for traffic signal control. *arXiv preprint arXiv:1611.01142*, 2016.

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. ACM, 2019.

Tomoki Nishi, Keisuke Otaki, Keiichiro Hayakawa, and Takayoshi Yoshimura. Traffic signal control based on reinforcement learning with graph convolutional neural nets. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 877–883. IEEE, 2018.

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*, pages 2496–2505. ACM, 2018.

Tian Tan, Feng Bao, Yue Deng, Alex Jin, Qionghai Dai, and Jie Wang. Cooperative deep reinforcement learning for large-scale traffic grid signal control. *IEEE transactions on cybernetics*, 2019.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4):e0172395, 2017.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *AAMAS*, pages 2085–2087, 2018.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

David Ha, Andrew Dai, and Quoc V Le. Hypernetworks. *arXiv preprint arXiv:1609.09106*, 2016.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. Roma: Multi-agent reinforcement learning with emergent roles. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. Deep variational information bottleneck. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2017.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Haoqiang Chen, Yadong Liu, Zongtan Zhou, Dewen Hu, and Ming Zhang. Gama: Graph attention multi-agent reinforcement learning algorithm for cooperation. *Applied Intelligence*, pages 1–11, 2020.

Tianze Zhou, Fubiao Zhang, and Chenfei Wang. Multi-agent reinforcement learning with graph clustering. *arXiv preprint arXiv:2008.08808*, 2020.

Iain Guilliard, Scott Sanner, Felipe W Trevizan, and Brian C Williams. A non-homogeneous time mixed integer lp formulation for traffic signal control. In *Transportation Research Board 95th Annual Meeting*, 2016.

Table 2: Summary and comparison of approaches

Article	Method	Architecture	Second- vs. Cycle-based	State	Training Obj.	Evaluation Obj.
Guilliard et al. (2016)	Mixed integer linear programming	Centralized	Second-based?	Queue length, traffic flow	Delay	TTT, delay
Chen and Sun (2016)	Dynamic programming	Single intersection	Cycle-based	Vehicle arrivals (loop detector)	Throughput or queue length or delay	Traffic volumn, saturation rate, delay
Yao et al. (2019)	Dynamic programming	Single intersection	Second-based	Vehicle arrivals (VII)	Queue length	Queue length, delay
Chiou (2019)	Dynamic programming, PMPEC	Centralized	Cycle-based	Travel flow	Weighted delay and queue length	Model robustness
Lin et al. (2012)	Model predictive control	Centralized	Cycle-based	Queue length, vehicle arrivals	Total travel time	TTT, delay, throughput
Anderson et al. (2018)	Max pressure control	Decentralized	Cycle-based	Queue length, pressure	Pressure	Throughput, #stops, delay
Kouvelas et al. (2018)	Feedback (PI) + max pressure	Hierarchical	Cycle-based	Queue length, pressure	Pressure	Delay, speed, stop time, TTT, throughput
El-Tantawy et al. (2013)	Reinforcement learning	Decentralized	Second-based	Queue length	Delay	Throughput, delay, queue length, stop time, etc.
Wei et al. (2019)	Reinforcement learning	Decentralized	Second-based	Queue length, pressure	Pressure	Average travel time
Genders and Razavi (2016)	Deep RL	Single intersection	Second-based	Discrete traffic state encoding	Delay	Throughput, queue length, travel time, delay
Wei et al. (2018)	Deep RL	Decentralized	Second-based	Intersection snapshot, queue length, and waiting time	Weighted metrics	Reward, queue length, delay, Average travel time
Tan et al. (2019)	Deep RL	Hierarchical	Second-based	Queue length	Queue length	Queue length, delay