applied sciences

MDPI

Article

# Extensible Hierarchical Multi-Agent Reinforcement-Learning Algorithm in Traffic Signal Control

**Pengqian Zhao, Yuyu Yuan * and Ting Guo**

Key Laboratory of Trustworthy Distributed Computing and Service, Ministry of Education, School of Computer Science (National Pilot Software Engineering School), Beijing University of Posts and Telecommunications, Beijing 100876, China

* Correspondence: yuanyuyu@bupt.edu.cn

**Abstract:** Reinforcement-learning (RL) algorithms have made great achievements in many scenarios. However, in large-scale traffic signal control (TSC) scenarios, RL still falls into local optima when controlling multiple signal lights. To solve this problem, we propose a novel goal-based multi-agent hierarchical model (GMHM). Specifically, we divide the traffic environment into several regions. The region contains a virtual manager and several workers who control the traffic lights. The manager assigns goals to each worker by observing the environment, and the worker makes decisions according to the environment state and the goal. For the worker, we adapted the goal-based multi-agent deep deterministic policy gradient (MADDPG) algorithm combined with hierarchical reinforcement learning. In this way, we simplify tasks and allow agents to cooperate more efficiently. We carried out experiments on both grid traffic scenarios and real-world scenarios in the SUMO simulator. The experimental results show the performance advantages of our algorithm compared with state-of-the-art algorithms.

**Keywords:** reinforcement learning; multi-agent system; traffic signal control; hierarchical reinforcement learning

check for updates

**Citation:** Zhao, P.; Yuan, Y.; Guo, T. Extensible Hierarchical Multi-Agent Reinforcement-Learning Algorithm in Traffic Signal Control. *Appl. Sci.* **2022**, *12*, 12783. https://doi.org/10.3390/app122412783

Academic Editors: Vicent Botti and Vicente Julian

Received: 17 November 2022
Accepted: 10 December 2022
Published: 13 December 2022

**Publisher's Note:** MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

**Copyright:** © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

## 1. Introduction

As the population grows and the number of vehicles increases, the problem of traffic jams is playing out in cities all over the world. Traffic congestion will reduce people's travel efficiency, increase fuel consumption and affect the normal operation of society. Therefore, experts came up with traffic signal control, which aims to find a strategy that can alleviate traffic jams based on real-time traffic conditions. Early TSC [1,2] simply designed the traffic signal program and switched the program according to the road information provided by the sensor. Such algorithms rely heavily on the manual formulation of programs. However, the traffic situation in the real world is very complex and changes all the time. Therefore, we need a more flexible and comprehensive strategy to solve this problem.

In recent years, with the development of neural networks and computational power resources, deep reinforcement learning (DRL) has been widely used in various serialization decision scenarios [3–6]. Compared with traditional methods, DRL abstracts TSC problem as a Markov decision process (MDP) without heuristic assumptions and rules made by experts. In the process of interacting with the environment, DRL searches for an optimal policy based on the feedback given by the environment. Classical reinforcement learning is divided into two categories: value based and policy based. Deep Q-Network [7] is a well-known value-based reinforcement-learning algorithm which uses neural networks to fit the action-state value function of an agent so as to find the optimal action. Policy gradient [8] is a classic policy-based algorithm which uses neural networks to directly fit agent policies to seek optimal actions. These classical reinforcement-learning algorithms have excellent performance in various scenarios, such as video games [9,10], robot control [11,12], and

autonomous driving [13,14], etc. However, single-agent RL is not suitable for solving the traffic problem. When the TSC scene is too large, the environment state can be very complex. In addition, when there are too many intersections, the action space will also face the problem of an exponential increase. Therefore, more and more people have begun to study multi-agent reinforcement-learning (MARL) algorithms in complex environments.

The development of multi-agent reinforcement learning has also formed two categories, namely, independent learning and centralized learning. Centralized learning [15,16] usually has a centralized network which receives information from other agents to make comprehensive judgments. However, this method also needs to solve the problem of the large joint action space when facing the large-scale TSC problem. The independent learning method [17,18] does not exist centralized network, and each agent is independent of the others. These agents treat other agents as part of the environment when they interact with it. However, this method is prone to non-stationary problems: the update of an agent's own policy relies on the other agents, but the policies of other agents also change dynamically. Based on this situation, [19] multi-agent advantage actor critic (MA2C) can be used to solve the TSC problem. The algorithm obtains the state information of neighbors through communication and directly joins in its decision-making process without establishing a centralized network. At the same time, MA2C sets a spatial discount factor according to the location relationship, focusing on the situation of neighbors that are closer to themselves.

Although MA2C solves the above problems, it does not guarantee effective cooperation in large-scale TSC problems and still suffers from local optima. In this kind of multi-agent environment, establishing effective collaboration is the most direct way to accomplish the task. Feudal multi-agent actor-critic (FMA2C) [20] proposes to combine hierarchical reinforcement learning with MA2C to establish collaboration. FMA2C stratifies the problem and uses the upper layer to obtain more environmental information to assign goals to the lower layer. The lower level makes decisions based on the goals and environmental information it receives. The guidance of the upper layer for the lower layer enables the agent to make long-term decisions and, thus, solve the TSC problem more efficiently. However, FMA2C does not take full advantage of the target. The lower layer only considers the goal and state at the same time when making decisions and does not establish relationships with other agents.

Therefore, in this paper, we propose a novel goal-based multi-agent hierarchical model (GMHM) to solve the large-scale TSC problem. We first divide the environment into independent regions, each containing multiple traffic lights. In this region, we abstract an upper agent and use those traffic lights as lower agents. The upper agent assigns specific goals to the lower agents according to the environment information and the mutual position of subordinate agents. After receiving a goal, the lower agent not only considers that when making decisions but also infers its neighbor's actions from the neighbor's goals. The lower agent uses the goal to gain a more comprehensive understanding of the environment and to take full account of its neighbors. In this way, our algorithm ensures effective cooperation between agents.

The main contributions of our paper are as follows: First, we investigate the development of multi-agent reinforcement-learning algorithms in TSC problems and propose a goal-based multi-agent hierarchical model for existing problems. Second, we describe our proposed model in detail. For the upper agent, we use a transformer structure to assign specific goals to each lower agent. For the lower agents, we added the concept of goals to the MADDPG algorithm to make it more efficient to cooperate. Finally, we verify the advantages of our proposed algorithm over the state-of-the-art algorithms through several experiments.

The rest of the paper is structured as follows: Section 2 briefly elaborates on the background knowledge of reinforcement learning. Section 3 introduces the related work of reinforcement learning in solving TSC problems. In Section 4, we describe our proposed GMHM algorithm in detail. In Section 5, the effectiveness of our proposed algorithm is verified by experiments. Section 6 carries on the simple summary.

## 2. Background

We define the interaction process between the agent and the environment as a Markov decision process (MDP) [21]. In a multi-agent system, MDP can be represented by a quintuple as $M = (S, A, R, T, \gamma)$. Among them, $S$ is the state space of the agent, $A$ is the action space, $R : S \times A \times S \rightarrow \mathbb{R}$ is the reward function, $T : S \times A \times S \rightarrow [0,1]$ is the state transition function, and $\gamma \in [0,1]$ is the discount factor. In a partially observable environment, each agent can only observe the environment around itself to obtain an observation $O : S \rightarrow O$. Each agent uses their policy to obtain an action by observing the environment $\pi_\theta : S \rightarrow A$. After performing the action, the environment will reward each agent. Accordingly, the purpose of the agent is to maximize the cumulative reward $R_\theta = E_{\tau \sim p_\theta(\tau)}[R(\tau)]$.

In the mainstream algorithm of reinforcement learning, the value-based DQN fits the state value function through the neural network, thereby solving the dimensional disaster problem of the state space. The loss function of the network is:

$$\mathcal{L}(\theta_i) = E_{s,a,r,s'}[(r + \gamma max_{a'} Q(s', a'; \theta_i^-) - Q(s, a; \theta_i))^2] \tag{1}$$

The policy-based PG algorithm [8] directly fits the policy, and the update function of the network is:

$$\nabla_\theta J(\pi_\theta) = E[\nabla_\theta \log \pi(a|s) Q^\pi(s, a)] \tag{2}$$

The proposal of DDPG [22] solves the continuous action space problem. Unlike DQN and PG output action distribution, DDPG directly outputs a deterministic action. At the same time, DDPG uses a generalized AC architecture [23], where the actor aims to find the action that maximizes the Q value, and the critic aims to estimate the action value. The loss function of the critic net is:

$$\mathcal{L}(\phi) = \nabla_\phi \frac{1}{|B|} \sum_{(s,a,r,s',d) \in B} (Q_\phi(s, a) - y)^2 \tag{3}$$

where

$$y = r + \gamma Q_\phi(s', \mu_\theta(s')) \tag{4}$$

MADDPG, on the other hand, extends DDPG to a multi-agent environment. The algorithm first predicts the opponent's actions by fitting their policies. Then, a centralized action value network is constructed, which combines the joint action and the environment state to obtain its action state value. The algorithm solves the non-stationary problem by introducing the actions of other agents. Critic can be updated by minimizing the loss:

$$\mathcal{L}(\theta_i) = \mathbb{E}_{x,a,r,x'}[(Q_i^{\boldsymbol{\mu}}(x, a_1, \ldots, a_N) - y)^2, y = r_i + \gamma Q_i^{\boldsymbol{\mu}'}(x', a_1', \ldots, a_N')|_{a_j' = \boldsymbol{\mu}_j'(o_j)} \tag{5}$$

and the actor's policy gradient is computed by:

$$\nabla_{\theta_i} J(\boldsymbol{\mu}_i) = \mathbb{E}_{x,a \sim D}[\nabla_{\theta_i} \boldsymbol{\mu}_i(a_i|o_i) \nabla_{a_i} Q_i^{\boldsymbol{\mu}}(x, a_1, \ldots, a_N)|_{a_i} = \boldsymbol{\mu}_i(o_i)] \tag{6}$$

Hierarchical reinforcement learning is very effective in solving complex or reward-sparse environments. This method uses the idea of divide and conquer to decompose the problem: the upper layer makes long-term decisions, and the lower layer makes real-time decisions. Hierarchical reinforcement learning includes option based [24,25] and goal based [26–28]. The former will pre-train multiple sub-policies, and then the upper layer will switch the sub-policies according to the environment state. The latter formulates long-term goals for the lower layer, and the lower layer achieves this goal through its policy.

## 3. Related Work

Reinforcement learning has been used in TSC for a long time. Early reinforcement-learning algorithms such as [1,2] are limited by computing power and technology, and can

only study a single traffic light. With the development of deep learning, reinforcement learning solves the high-dimensional state input problem through neural networks [7]. The development of traffic simulators has also given us a suitable environment to conduct research.

A more realistic TSC scenario contains multiple traffic lights. Therefore, the research of multi-agent reinforcement learning is also continuously advancing in this field. Ref. [29] uses mean-field theory to consider multiple neighbors as a whole. Ref. [19] extends the actor–critic algorithm to consider discounted neighbor agent policies and states. Ref. [30] proposes a reward function based on maximizing pressure to optimize policy. Ref. [20] combines hierarchical reinforcement learning with MA2C, using upper layers to guide the cooperation of semaphores. Ref. [31] uses graph attention networks to facilitate communication between agents. In terms of the experimental environment, ref. [30–33] use Cityflow, and [19,20] use SUMO.

Our algorithm is inspired by FMA2C, but has a different network structure. At the same time, our algorithm has been more fully considered in the design, generation, and utilization of the goals.

## 4. Method

In this section, we detail the architecture, definition and learning process of our proposed model. First, we give the concrete architecture of GMHM, and then introduce the essential elements of our proposed algorithm. Finally, the policy learning process of the upper and lower layers in GMHM is described in detail.

### 4.1. Hierarchical Architecture

First, we will define the environment. We abstract TSC environment as a gird network consisting of edges and intersections $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ represents an intersection, and edge $e = (v_i, v_j) \in \mathcal{E}$ represents a road between two intersections $v_i, v_j \in \mathcal{V}$. In this paper, a lower agent is defined as a traffic light located at a certain intersection. The agent decides to switch the traffic signal by observing the situation at the intersection, which is the direct unit for traffic regulation. We divide the whole environment into multiple disjoint regions $\mathcal{G} = \mathcal{G}_1 \times \cdots \times \mathcal{G}_K$, each region $\mathcal{G}_i = v_1, \ldots v_k$ contains k signal lights controlled by the agent, and roads connecting each intersection. Through this division of regions, we realize the simplification of the large-scale TSC problem

In each region of the two-layer architecture, the upper layer is a virtual manager agent, and the lower layer agent is called worker who controls the intersections. In each region, the task of the manager is to assign goals to the various workers under its command based on the traffic conditions in the region. This goal is the manager's desired future state for the worker-controlled intersection. Workers combine environmental states and assigned goals to make decisions and change environmental states by toggling traffic signals. In this way, we divided the entire mission at the strategic and tactical levels.

If we only consider the situation in one region, the overall traffic situation still cannot be effectively alleviated. Therefore, we need to incorporate consideration of neighbors at both levels. Neighbors of the region are defined as $\mathcal{N}$, so we ontain the manager's neighbor $\mathcal{N}^{\boldsymbol{M}}$ and worker's neighbor $\mathcal{N}^{\boldsymbol{W}}$. We established a two-layer communication channel to assist the operation of the layered architecture. Among them, the manager will communicate with the neighbor manager to obtain the traffic situation in the neighbor region. In addition to obtaining the environmental state of the neighbor worker, the worker also obtains the phased goals of the neighbor workers.

Therefore, the agent–environment interaction process in GMHM is as follows: the manager combines the state of the environment and the states of neighbors to generate goals for workers. Workers combine environment state, neighbor state, self-goal, and neighbor-goal to generate actions and submit them to the environment. The environment executes actions submitted by workers to move to the next state and gives feedback to workers and managers. In this way, after considering the state of neighbors, both manager and worker obtain more sufficient environmental information, and their decision making

is more assured. At the same time, it also solves the instability caused by the mutual adaptation of agents in a multi-agent environment. This also ensures that when multiple regions are combined into a larger traffic environment, managers and workers can work with neighbors to resolve traffic jams.

### 4.2. Markov Decision Process in TSC

The quintuple of the Markov decision process is an abstract overview of the reinforcement-learning process, which can be defined differently in different scenarios. In the hierarchical traffic-signal control problem, the MDP of managers and workers are: $\mathcal{M}^{\boldsymbol{M}} = (S^M, A^M, R^M, T, \gamma^M)$, $\mathcal{M}^{\boldsymbol{W}} = (S^W, A^W, R^W, T, \gamma^W)$, respectively. In particular, we adopt SUMO as the experimental environment in this paper. We use various parameters in SUMO to define $\mathcal{M}^{\boldsymbol{M}}$ and $\mathcal{M}^{\boldsymbol{W}}$ in detail.

$S^M$ and $S^W$ are the state spaces of manager and worker, respectively. However, TSC is a partially observable environment, which means that the manager and the worker can only obtain an observation $O^M$ and $O^W$ of their own surrounding environment. We define the worker's local observation as $O^W = (wait[l], pressure[l])$ where $l$ is each incoming lane for this worker. The *wait* is the accumulated waiting time of the vehicle closest to the intersection at the current moment. *Pressure* refers to max-pressure [30], which defines the traffic pressure in this direction. More specifically, *pressure* is equal to the number of vehicles on the target road minus the number of vehicles on the current road. For the manager, we define its local observation as $O^M = (pressure_1, pressure_2, pressure_3, pressure_4)$, which is the vehicle pressure situation in four directions in this region.

We define a manager's action as the goals assigned to each subordinate worker. In order to help workers to measure the completion of the goal, we keep the form of the goal consistent with $O^W$. At the same time, goals also represent the future state that managers expect workers to achieve. For workers, the action space includes the individual switching commands of the traffic lights. For example, red lights in the east–west direction, go straight in the north–south direction or turn left in the east–west direction.

Finally, the most important thing is the setting of the reward function. The reward function represents the task we expect the agent to solve. As a manager, we expect to assign the correct goals to each subordinate. Therefore, we can take the state change between two decisions as the reward $R_t^M = \sum_l pressure_{t+k,l} - pressure_{t,l}$, where $l$ is each direction. For workers, their reward function is divided into two parts: $R^M = R_e + R_g$. $R_e = -\sum_l (pressure_l + wait_l)$ is the environmental reward, which is the feedback of the agent's action. $R_g = \sigma(O^W, g)$ is the intrinsic reward, which represents the completion of the worker's goal and $\sigma(X, Y) = X^T Y / (|X||Y|)$ . $T$ is state transition functions, which are determined by the environment.

### 4.3. Learning Process

In ordinary reinforcement learning, the agent performs a complete round of observation, decision making, feedback, and learning at every moment. However, in hierarchical reinforcement learning, the goal given by the manager to the worker cannot be completed in one time step. The goal requires the worker to achieve through a period of hard work. Therefore, the decision frequency of managers and workers is different. The manager makes a decision at every $d$ time step, and the worker still makes an instant decision. Figure 1 shows the network structure of managers and workers.

We assign each manager a DDPG algorithm to make decisions. At time step t (if it is time to make a decision), manager $M_i$ observes the environment and obtains local state *LS*. Through communication, $M_i$ can obtain the states of its neighbors $N_i^M$ so that $M_i$ has a wider range of environmental conditions *LS+NS*. Since workers are distributed differently across regions, we expect managers to set specific goals for each worker. Therefore, we added the transformer structure to DDPG. According to the location relationship, we assign different attention weights to different workers. In this way, we can obtain the action output $A$ of the manager, which is the goal $LG$ of the worker.

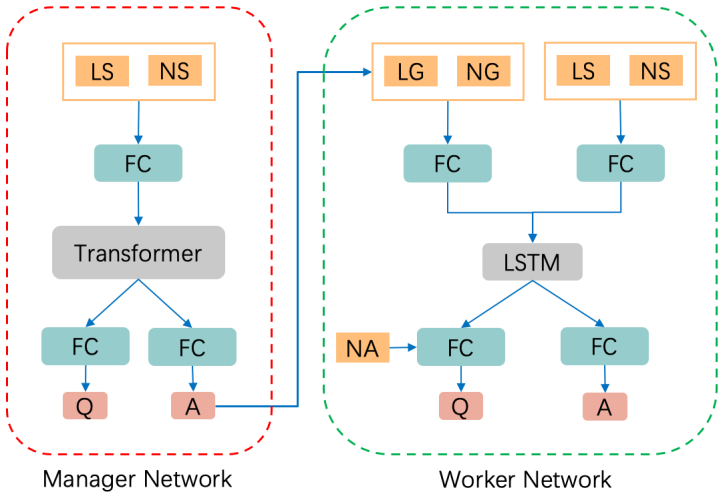


**Figure 1.** The architecture of our goal-based multi-agent hierarchical model. The red part is manager's network. The green part is worker's network. Workers make decisions based on goals passed on to them by managers.

DDPG is designed based on actor–critic architecture. The actor network is parameterized as $\mu_\theta^M$, the critic network is parameterized as $Q_\phi^M$. Therefore, the loss function of its actor network is:

$$\nabla_{\theta^M} J(\mu_{\theta^M}) = E[\nabla_{\theta^M} \mu_{\theta^M}(s) Q^\mu(s,a)|a = \mu_{\theta^M}] \tag{7}$$

The loss function of critic network is Equation (3), where the network's parameter is named $\phi^M$.

Moreover, DDPG uses soft-update to update the parameter of the target network:

$$\begin{aligned} \theta' &\leftarrow \tau\theta + (1-\tau)\theta' \\ \phi' &\leftarrow \tau\phi + (1-\tau)\phi' \end{aligned} \tag{8}$$

Likewise, we assign the MADDPG algorithm to each worker where we use $\pi_\theta^W$ and $Q_\phi^W$ to represent actor net and critic net. Workers decide how to switch traffic signals by observing the environment. As shown in Figure 1 at time t, workers will first collect information, including their local state *LS*, neighbors' states *NS*, their own goals *LG*, and their neighbors' goals *NG*. Using convolutional neural networks, we can extract features from these data. Since the instantaneous state *S* does not reflect the continuous change in the environment, we use long short-term memory (LSTM) to memorize the state of the environment to obtain the hidden state. In this way, we can make decisions based on the hidden state, and, thus, obtain the action. The updated formula of the actor network and critic network is

$$\nabla_{\theta_i} J(\boldsymbol{\pi}_i^M) = \mathbb{E}_{o,a,g\sim D}[\nabla_{\theta_i} \boldsymbol{\pi}_i^M(a_i|o_i,g_i) \nabla_{a_i} Q_i^{\boldsymbol{\pi}^M}(x,a_1,\ldots,a_N)|_{a_i} = \boldsymbol{\pi}_i^M(o_i,g_i)] \tag{9}$$

where $Q$ is a goal-based action-value function which uses state, goal and the actions of all agents as input and outputs the Q-value of $W_i$. The original MADDPG algorithm speculates on other agents' actions by fitting their policies. However, in the TSC problem, each worker has the same task. Therefore, we use our own policy to predict the actions $a_j = \pi_{\theta_i}^W(O_j^M)$ of other agents.

The update method of the action value function is

$$\mathcal{L}(\phi_i) = \mathbb{E}_D[(Q_i^{\boldsymbol{\phi}^M}(x,a_1,\ldots,a_N) - y)^2, y = r_i + \gamma Q_i^{\boldsymbol{\phi}'}(x',a'_1,\ldots,a'_N)|_{a'_j = \boldsymbol{\phi}'_j(o_j,g_j)} \tag{10}$$

We train managers to obtain a better task allocation policy and train workers to obtain a better traffic-signal switching policy. At the same time, the upper and lower layers realize

the cooperation between the agents by considering their neighbor states and neighbor goals. Through the division and splicing of regions, we can solve the large-scale TSC problem. The main procedure of our algorithm is shown in Algorithm 1.

**Algorithm 1** Goal-based multi-agent hierarchical model

Initialize the actor net of each manager and each worker $\pi^M$, $\pi^W$ with $\theta^M$ and $\theta^W$. Use $\phi^M$ and $\phi^W$ to represent the parameters of their critic network. Initialize learning rate $\alpha$.

**for each** episodes **do**
  **for** t=1,T **do**
    **for each** manager k $\in$ 1..m **do**
      sample $g^k$ from $\pi_k^M$
      **for each** worker $i \in \mathcal{G}_k$ **do**
        sample $a_i$ from $\pi_i^W$
        execute action $a_i$ and receive $r_{t,i}^W$ and $O_{t+1,i}^W$
      **end for**
      receive $r_{t,k}^M$ and $O_{t+1,k}^M$
    **end for**
  **end for**
  **for each** manager k $\in$ 1..m at every d steps **do**
    compute $\nabla_{\theta_k^M}$, $\nabla_{\phi_k^M}$ by Equation (7), Equation (3)
  **end for**
  **for each** worker $i \in \mathcal{G}_k$ **do**
    compute $\nabla_{\theta_i^W}$, $\nabla_{\phi_i^W}$ by Equation (9), Equation (10)
  **end for**
  update their actor network and critic network $\theta_{t+1} \leftarrow \theta_t - \alpha\nabla_\theta$, $\phi_{t+1} \leftarrow \phi_t - \alpha\nabla_\phi$
**end for**

## 5. Experiments and Results

In this section, we introduce the experimental environment and the baseline algorithm, respectively. Then, we enumerate the specific parameters involved in the experiment and analyze the experimental results at the end.

### *5.1. Baseline*

In this experiment, we use the MADDPG, MA2C and FMA2C as the baseline to compare with our algorithm. Among them, MADDPG is a classic algorithm in the multi-agent field. The algorithm can be applied in competitive or cooperative scenarios by setting different reward functions. The workers in our algorithm are improvements based on MADDPG. At present, variants of the MADDPG algorithm have been applied in many scenarios and achieved good results.

MA2C is an extension of the A2C algorithm in a multi-agent TSC environment. In this algorithm, each agent can obtain the policy and action information of other agents to improve its local observability. At the same time, a spatial discount factor is designed according to the location information between the agents to weaken the signals of the farther agents, so that they can focus more on their surroundings. FMA2C is a combination of hierarchical reinforcement learning and MA2C algorithm. The algorithm designs a hierarchical MDP suitable for the TSC environment and proposes a specific form of the goal. The MA2C is implemented based on the source code released in the paper [19]. FMA2C implements a hierarchical architecture based on the MA2C. Similar to FMA2C, our proposed algorithm is a combination of hierarchical reinforcement learning and the MADDPG algorithm.

### 5.2. Environment

At present, two kinds of simulators, SUMO and CityFlow, are widely used in the research of TSC problems. They can both customize the environment or simulate real-world road conditions. We chose SUMO as the simulator in our experiments. SUMO is an open-source, microscopic multi-modal traffic simulation software. It can be fine tuned for each vehicle, making it ideal for traffic-control research. SUMO realizes the customization of the environment by writing road-network files and vehicle files, and can also simulate the real environment by importing various types of map files. At the same time, SUMO comes with a variety of vehicle-following models and lane-changing models to simulate realistic vehicle-driving situations. SUMO provides a TraCI interface based on Python so that the algorithm and the environment can be easily connected. In conclusion, SUMO is a suitable environment to simulate and verify TSC problems.

### 5.3. Model Setting

In SUMO, we adopted the Krauss car-following model and the basic lane-changing model to make the vehicle as close to reality as possible while driving normally. We set the length of an episode to 720 steps, which is equivalent to 3600 s in the simulator. For each training, we ran four million seconds, about 1200 episodes. The region size was set to 4 (to include four workers).

For the manager, we used the structure of DDPG, including four neural networks. We used two fully connected layers with 128 units and 64 units for state extraction and transformer for attention assignment. We set $\gamma = 0.97$, $\alpha = 0.75$, batch size to 120, learning rate to 0.001 and decision frequency $d$ to 6 s.

For the worker, we used the structure of MADDPG, which also consists of four neural networks. We used two neural networks with 128 units and 64 units and long short-term memory models with 64 units to process the environmental state. For the Q function in it, we determined the input size of the function according to the number of neighbors of the agent. At the same time, we set $\gamma = 0.95$, $\alpha = 0.75$, batch size to 120, learning rate to 0.001 and decision frequency to 1 s.

### 5.4. Grid Network

As Figure 2 shows, we first conducted experiments in a symmetric-grid traffic environment. There are 16 ($4 \times 4$) traffic lights in the environment. The road connecting the signal lights has two lanes, a left turn lane, and a straight right turn lane. The maximum speed of vehicles is 20 m/s, the minimum clearance between vehicles is 2 m, and the road length is 200 m. At the same time, we set up multiple time-based traffic flows. We defined six kinds of traffic flow. We use F1–F6 to represent each traffic flow; they had different origin and destination (OD) pairs. The six types of traffic flow represent different real-world situations.

For example, F1 and F2 start generating vehicles first. At 900 s, the traffic flow of F1 peaks 660 veh/h and decreases to 0 at 1800 s. Meanwhile, at 900 s, F3 and F4 start to spawn vehicles in the environment. When F3 reaches the vertex, F5 and F6 start running. Finally, F3 ends at 2400 s and F5 ends at 3600 s. In this way, through different time settings and different OD settings, the traffic flow is overlapped in time and space. In addition, each traffic flow is coming from different directions, which causes moer easily traffic congestion.

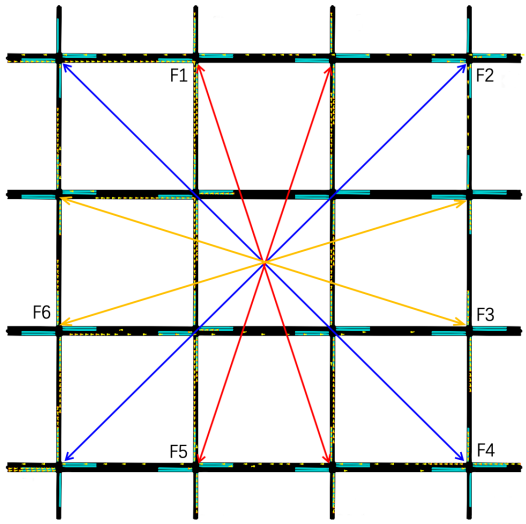


**Figure 2.** A traffic grid of 16 intersection with time-variant traffic-flow groups.

Taking Figure 3 as an example, the horizontal axis represents the number of training times, and the vertical axis represents the average cumulative reward. The thicker lines represent changes in workers' average reward. This reward represents the environmental reward of the worker so that our curve can more clearly measure the intuitive change traffic lights make to the environment. The shaded part refers to the standard deviation of the experimental results, which represents the fluctuation in the algorithm performance. We used five sets of random seeds to verify each algorithm to reflect the average performance of the algorithm.

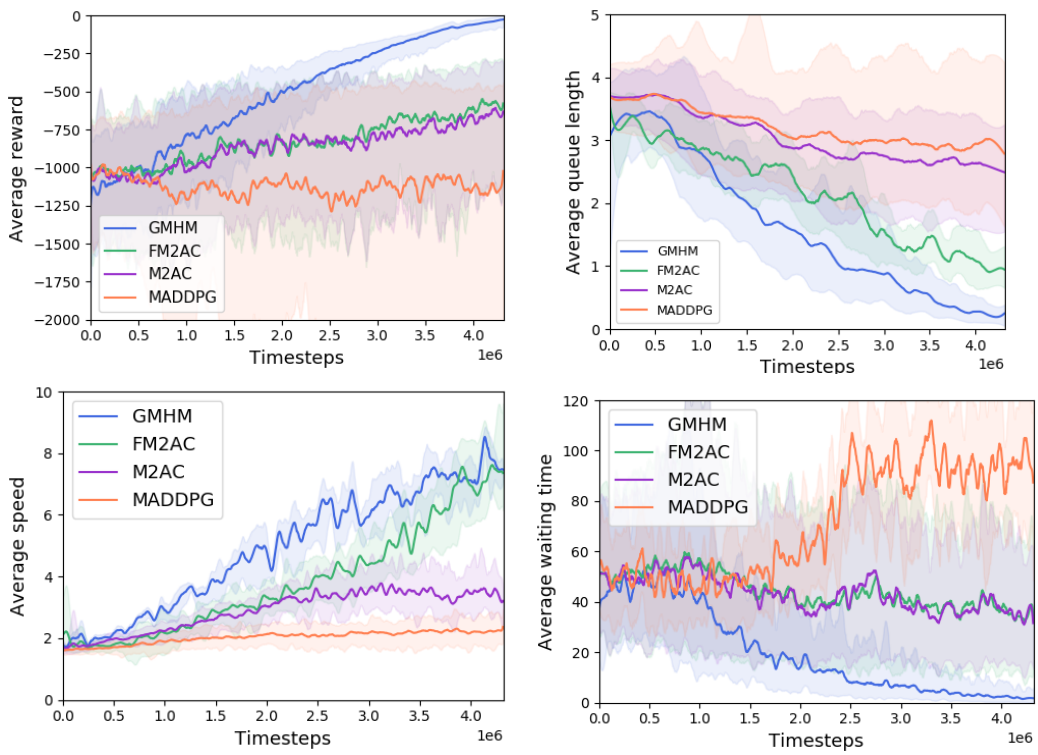


**Figure 3.** Experiment results in *grid network*. The curves in the figure show the performance of each algorithm on different indicators.

As can be seen from the figure, the effect of MADDPG is not very good. Both MA2C and FMA2C converge to a local optimal state along with the learning process. Our algorithm GMHM also achieves convergence and has the highest level of reward.

For MADDPG, each agent needs to maintain the policies of other agents and make predictions. When the number of agents is too large, the prediction of the joint actions will become more and more inaccurate. Considering other agents equally and ignoring the influence of the location factor also leads to poor performance of MADDPG. Both MA2C and FMA2C are algorithms that have been applied in the TSC field, so both can learn an effective policy. However, they did not obtain a good final result due to some problems in the algorithm itself. Our proposed algorithm comprehensively considers the relationship between the upper and lower layers. The attention mechanism is used by transformer to assign specific goals to workers at different locations while allowing workers to fully consider the goal information of neighbor workers. These tricks make our algorithm more stable and better than other algorithms.

### *5.5. Monaco*

In addition to grid environments, we also validated the robustness and scalability of our algorithm in real-world environments. As the Figure 4 shows, we sampled thirty controllable traffic lights in real Monaco city traffic and marked them in blue. We can also see that there are different types of intersections in this traffic network, which are not symmetrical like the grid traffic environment. An intersection may be connected by three roads, four roads, or even five roads. Therefore, different intersections have specific environmental states and action-space configurations. According to the positional relationship of these 30 traffic lights, we divided them into four regions, including 10, 8, 6, and 6 traffic lights, respectively.

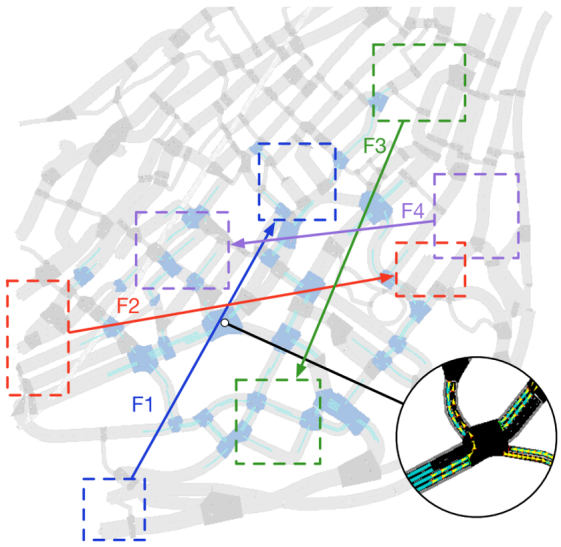


**Figure 4.** Monaco transport network with various intersections.

Different types of traffic lights and different sizes of regional settings reflect the scalability and adaptability of our algorithm. In this environment, we set up four sets of flows, F1–F4. Its origin and destination are rectangular areas located on the edge of the map, and each pair of OD passes through the center of the map.

The figures show the performance of our algorithm in the Monaco urban traffic scene, respectively. In the average reward, we can see that the GMHM performs the best, with fast convergence and a high final reward level. Figure 5 shows the average queue length and waiting time for GMHM in Monaco cities. The performance advantage of GMHM over other algorithms can also be seen.

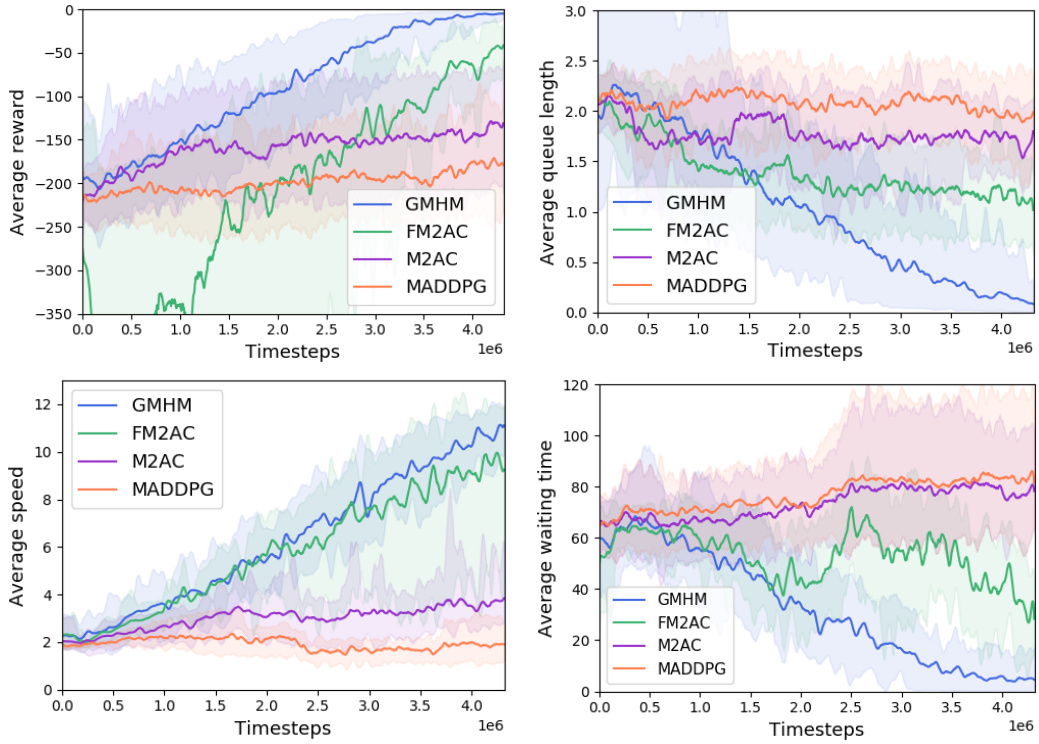


**Figure 5.** Experiment results in Monaco traffic network. The top two figures are the results of our method applied to A3C. The bottom two figures are the experiment results based on PPO.

### *5.6. Discussion*

This section elaborates on the experimental environment, baseline algorithms and experimental results in detail. Experiments in the SUMO simulator demonstrate the effectiveness of hierarchical-based multi-agent reinforcement-learning algorithms in solving traffic congestion problems. The level of cumulative average reward reflects the overall performance of the algorithm. The three indicators of average queue length, speed and waiting time measure the degree of congestion in the traffic system from different dimensions. Based on these four aspects, we make an overall comparison of different algorithms in Table 1. It can be seen from the results that the basic MADDPG algorithm performs generally in the face of specific environments. However, MA2C and FMA2C include additional considerations in multi-agent cooperation, and their effects weer also improved. Finally, our proposed GMHM algorithm fully considers the key element of the goal. The GMHM algorithm generates more reasonable goals through the transformer in the upper layer, and uses the goals in the lower MADDPG algorithm to speculate on the neighbors. In this way, we achieve collaboration between layers and within layers to adequately solve the TSC problem.

**Table 1.** Overall performance of different algorithms in the grid network and Monaco network.

| Metrics | Grid Network | | | | Monaco Network | | | |
|---|---|---|---|---|---|---|---|---|
| | **GMHM** | **FMA2C** | **MA2C** | **MADDPG** | **GMHM** | **FMA2C** | **MA2C** | **MADDPG** |
| Reward | **−137.6** | −692.7 | −714.5 | −1131.6 | **−16.3** | −53.7 | −158.4 | −221.9 |
| Queue length | **0.27** | 0.89 | 1.73 | 1.85 | **0.47** | 1.35 | 1.98 | 2.33 |
| Speed | **8.29** | 7.94 | 3.17 | 2.88 | **11.69** | 9.74 | 4.02 | 2.53 |
| Waiting time | **10.9** | 39.7 | 42.6 | 105.3 | **9.6** | 40.3 | 75.7 | 79.2 |

## 6. Conclusions

This paper proposes a novel goal-based multi-agent hierarchical model to solve large-scale TSC problems. We first divide the environment into disjoint regions, each containing several traffic lights. We abstract a manager as the upper agent in the area, and the actual traffic light as the lower agent called a worker. The purpose of the manager is to assign a long-term goal to workers in different locations by observing the environment. The worker will make actual action decisions based on the state of the environment and the goal received. In the decision-making process, the worker will fully consider the situation of the neighbor agents, and infer the neighbor's actions through its policy and the neighbor's goal. This not only solves the instability problem but also realizes the cooperation between the agents. Based on such an architecture, we can solve large-scale TSC problems by splicing regions.

We built a symmetrical mesh environment and an actual Monaco city environment in SUMO. The experimental results, respectively, verify the effectiveness of our algorithm and the excellent performance compared to SOTA in the real world. At the same time, the TSC problem can be abstracted as a flow control problem. Therefore, the algorithm can also be applied to network traffic [34], power dispatch [35,36], energy pipeline transportation [37], and so on.

**Author Contributions:** Conceptualization, Y.Y., P.Z. and T.G.; Formal analysis, P.Z.; Investigation, P.Z. and T.G.; Methodology, P.Z.; Project administration, Y.Y.; Resources, P.Z.; Software, P.Z.; Supervision, T.G.; Validation, P.Z.; Visualization, P.Z.; Writing—original draft, P.Z.; Writing—review & editing, Y.Y. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| RL | Reinforcement learning |
| TSC | Traffic signal control |
| GMHM | Goal-based multi-agent hierarchical model |
| MADDPG | Multi-agent deep deterministic policy gradient |
| SUMO | Simulation of urban mobility |
| SOTA | State of the art |
| DRL | Deep reinforcement learning |
| MDP | Markov decision process |
| MARL | Multi-agent reinforcement learning |
| MA2C | Multi-agent advantage actor critic |
| FMA2C | Feudal multi-agent actor-critic |
| DQN | Deep Q-network |
| DDPG | Deep deterministic policy gradient |
| LSTM | Long short-term memory |

## References

1. Hunt, P.; Robertson, D.; Bretherton, R.; Royle, M.C. The SCOOT on-line traffic signal optimisation technique. *Traffic Eng. Control* **1982**, *23*, 190–192.
2. Luk, J. Two traffic-responsive area traffic control methods: SCAT and SCOOT. *Traffic Eng. Control* **1984**, *25*, 14.
3. Yuan, Y.; Guo, T.; Zhao, P.; Jiang, H. Adherence Improves Cooperation in Sequential Social Dilemmas. *Appl. Sci.* **2022**, *12*, 8004. [CrossRef]

4. Yuan, Y.; Zhao, P.; Guo, T.; Jiang, H. Counterfactual-Based Action Evaluation Algorithm in Multi-Agent Reinforcement Learning. *Appl. Sci.* **2022**, *12*, 3439. [CrossRef]
5. Ibarz, J.; Tan, J.; Finn, C.; Kalakrishnan, M.; Pastor, P.; Levine, S. How to train your robot with deep reinforcement learning: Lessons we have learned. *Int. J. Robot. Res.* **2021**, *40*, 698–721. [CrossRef]
6. Chen, L.; Lu, K.; Rajeswaran, A.; Lee, K.; Grover, A.; Laskin, M.; Abbeel, P.; Srinivas, A.; Mordatch, I. Decision transformer: Reinforcement learning via sequence modeling. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 15084–15097.
7. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
8. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
9. Afsar, M.M.; Crump, T.; Far, B. Reinforcement learning based recommender systems: A survey. *ACM Comput. Surv. (CSUR)* **2021**. [CrossRef]
10. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef]
11. Rudin, N.; Hoeller, D.; Reist, P.; Hutter, M. Learning to walk in minutes using massively parallel deep reinforcement learning. In Proceedings of the Conference on Robot Learning, PMLR, London, UK, 8–11 November 2021; pp. 91–100.
12. Zhao, W.; Queralta, J.P.; Westerlund, T. Sim-to-real transfer in deep reinforcement learning for robotics: A survey. In Proceedings of the 2020 IEEE Symposium Series on Computational Intelligence (SSCI), Canberra, Australia, 1–4 December 2020; pp. 737–744.
13. Kiran, B.R.; Sobh, I.; Talpaert, V.; Mannion, P.; Al Sallab, A.A.; Yogamani, S.; Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 4909–4926. [CrossRef]
14. Chen, J.; Li, S.E.; Tomizuka, M. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. *IEEE Trans. Intell. Transp. Syst.* **2021**, *23*, 5068–5078. [CrossRef]
15. Lowe, R.; Wu, Y.I.; Tamar, A.; Harb, J.; Pieter Abbeel, O.; Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6379–6390.
16. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Stockholm, Sweden, 10–15 July 2018; pp. 4295–4304.
17. Tampuu, A.; Matiisen, T.; Kodelja, D.; Kuzovkin, I.; Korjus, K.; Aru, J.; Aru, J.; Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE* **2017**, *12*, e0172395. [CrossRef] [PubMed]
18. de Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv* **2020**, arXiv:2011.09533.
19. Chu, T.; Wang, J.; Codecà, L.; Li, Z. Multi-agent deep reinforcement learning for large-scale traffic signal control. *IEEE Trans. Intell. Transp. Syst.* **2019**, *21*, 1086–1095. [CrossRef]
20. Ma, J.; Wu, F. Feudal multi-agent deep reinforcement learning for traffic signal control. In Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Auckland, New Zealand, 9–13 May 2020; pp. 816–824.
21. Zhou, D.; Gu, Q.; Szepesvari, C. Nearly minimax optimal reinforcement learning for linear mixture markov decision processes. In Proceedings of the Conference on Learning Theory, PMLR, Boulder, CO, USA, 15–19 August 2021; pp. 4532–4576.
22. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. *arXiv* **2015**, arXiv:1509.02971.
23. Mnih, V.; Badia, A.P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; Kavukcuoglu, K. Asynchronous methods for deep reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, New York, NY, USA, 19–24 June 2016; pp. 1928–1937.
24. Bacon, P.L.; Harb, J.; Precup, D. The option-critic architecture. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
25. Tessler, C.; Givony, S.; Zahavy, T.; Mankowitz, D.; Mannor, S. A deep hierarchical approach to lifelong learning in minecraft. In Proceedings of the AAAI Conference on Artificial Intelligence, San Francisco, CA, USA, 4–9 February 2017; Volume 31.
26. Kulkarni, T.D.; Narasimhan, K.; Saeedi, A.; Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. *Adv. Neural Inf. Process. Syst.* **2016**, *29*, 3675–3683.
27. Vezhnevets, A.S.; Osindero, S.; Schaul, T.; Heess, N.; Jaderberg, M.; Silver, D.; Kavukcuoglu, K. Feudal networks for hierarchical reinforcement learning. In Proceedings of the International Conference on Machine Learning, PMLR, Sydney, NSW, Australia, 6–11 August 2017; pp. 3540–3549.
28. Andrychowicz, M.; Wolski, F.; Ray, A.; Schneider, J.; Fong, R.; Welinder, P.; McGrew, B.; Tobin, J.; Pieter Abbeel, O.; Zaremba, W. Hindsight experience replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5048–5058.
29. Wang, X.; Ke, L.; Qiao, Z.; Chai, X. Large-scale traffic signal control using a novel multiagent reinforcement learning. *IEEE Trans. Cybern.* **2020**, *51*, 174–187. [CrossRef]
30. Wei, H.; Chen, C.; Zheng, G.; Wu, K.; Gayah, V.; Xu, K.; Li, Z. 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, Anchorage, AK, USA, 4–8 August 2019; pp. 1290–1298.

31. Wei, H.; Xu, N.; Zhang, H.; Zheng, G.; Zang, X.; Chen, C.; Zhang, W.; Zhu, Y.; Xu, K.; Li, Z. Colight: Learning network-level cooperation for traffic signal control. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management, Beijing, China, 3–7 November 2019; pp. 1913–1922.
32. Chen, C.; Wei, H.; Xu, N.; Zheng, G.; Yang, M.; Xiong, Y.; Xu, K.; Li, Z. Toward a thousand lights: Decentralized deep reinforcement learning for large-scale traffic signal control. In Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020.
33. Zang, X.; Yao, H.; Zheng, G.; Xu, N.; Xu, K.; Li, Z. Metalight: Value-based meta-reinforcement learning for traffic signal control. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 1153–1160.
34. Yang, J.; He, S.; Xu, Y.; Chen, L.; Ren, J. A trusted routing scheme using blockchain and reinforcement learning for wireless sensor networks. *Sensors* **2019**, *19*, 970. [CrossRef]
35. Duan, J.; Shi, D.; Diao, R.; Li, H.; Wang, Z.; Zhang, B.; Bian, D.; Yi, Z. Deep-reinforcement-learning-based autonomous voltage control for power grid operations. *IEEE Trans. Power Syst.* **2019**, *35*, 814–817. [CrossRef]
36. Zhang, Z.; Zhang, D.; Qiu, R.C. Deep reinforcement learning for power system applications: An overview. *CSEE J. Power Energy Syst.* **2019**, *6*, 213–225.
37. Mason, K.; Grijalva, S. A review of reinforcement learning for autonomous building energy management. *Comput. Electr. Eng.* **2019**, *78*, 300–312. [CrossRef]