applied sciences

Review

Applications of Multi-Agent Deep Reinforcement Learning: Models and Algorithms

Abdikarim Mohamed Ibrahim [1], Kok-Lim Alvin Yau [1,*], Yung-Wey Chong [2,*] and Celimuge Wu [3]

[1] Department of Computing and Information Systems, School of Engineering and Technology, Sunway University, Selangor 47500, Malaysia; abdikar.i@imail.sunway.edu.my
[2] National Advanced IPv6 Centre, Universiti Sains Malaysia (USM), Penang 11800, Malaysia
[3] Graduate School of Informatics and Engineering, The University of Electro-Communications, Tokyo 182-8585, Japan; celimuge@uec.ac.jp
* Correspondence: koklimy@sunway.edu.my (K.-L.A.Y.); chong@usm.my (Y.-W. C.)

Abstract: Recent advancements in deep reinforcement learning (DRL) have led to its application in multi-agent scenarios to solve complex real-world problems, such as network resource allocation and sharing, network routing, and traffic signal controls. Multi-agent DRL (MADRL) enables multiple agents to interact with each other and with their operating environment, and learn without the need for external critics (or teachers), thereby solving complex problems. Significant performance enhancements brought about by the use of MADRL have been reported in multi-agent domains; for instance, it has been shown to provide higher quality of service (QoS) in network resource allocation and sharing. This paper presents a survey of MADRL models that have been proposed for various kinds of multi-agent domains, in a taxonomic approach that highlights various aspects of MADRL models and applications, including objectives, characteristics, challenges, applications, and performance measures. Furthermore, we present open issues and future directions of MADRL.

Keywords: multi-agent deep reinforcement learning; reinforcement learning; multi-agent reinforcement learning; deep Q-network; applied reinforcement learning

check for updates

Citation: Ibrahim, A.M.; Yau, K.-L.A.; Chong, Y.-W.; Wu, C. Applications of Multi-Agent Deep Reinforcement Learning: Models and Algorithms. *Appl. Sci.* **2021**, *11*, 10870. https://doi.org/10.3390/app112210870

Academic Editors: Andrea Omicini and Stefano Mariani

Received: 1 October 2021
Accepted: 25 October 2021
Published: 17 November 2021

1. Introduction

Multi-agent deep reinforcement learning (MADRL) is a group of agents (or decision makers) that interact with each other and their operating environment to achieve goals in a cooperative or competitive manner. MADRL extends the functions of the traditional reinforcement learning (RL) and multi-agent reinforcement learning (MARL) with deep learning (DL), which is the recent advancement of artificial intelligence. DL enables efficient representation and storage of high-dimensional state, action, and reward using artificial neural networks (ANN) [1]. Hence, DL addresses the main challenges of RL and MARL, namely, the curse of dimensionality whereby the exponential increase in the state and action spaces (or the number of states and actions) increases the storage requirement and the convergence time (or the number of iterations) to the optimal action due to the large state and action spaces.

MADRL enables multiple agents to: (a) observe their states (or decision making factors); (b) exchange knowledge (e.g., immediate rewards [2], Q-values, value functions, and optimal policies [3]) with neighboring agents; (c) interact with their operating environment; and (d) learn knowledge on their own and select appropriate actions in the absence of external critics (or teachers) to supervise the learning process. This enables the agents to coordinate their actions [4] and achieve system-wide performance enhancement [5].

MADRL has made a breakthrough in recent years due to its ability to solve complex real-world problems, where traditional RL is unable to cope up with. MARL, in which multiple agents cooperate or compete with each other, has also shown limitations in resource allocation in wireless networks, traffic signal control, flood monitoring, and network routing [6]. With the rising popularity of MADRL, several surveys have studied MADRL from different perspectives. Nguyen et al. [7] present the technical challenges in MADRL, such as moving-target, partial observability, continuous state and action spaces, and transfer learning. Hernandez-Leal et al. [8] examine the technical challenges of MADRL from the emergent behaviors, communication, and cooperative learning perspectives. Oroojlooyjadid and Hajinezhad [9] review the cooperative setting of MADRL. Da Silva et al. [10] review knowledge reuse in MADRL. Some literature focuses on theoretical analyses, including Zhang et al. [11] and Gronauer and Diepold [12], who reviewed MADRL's technical challenges from the mathematical perspective.

This paper extends the existing literature by providing a survey of MADRL algorithms applied to various state-of-the-art applications, and presenting a taxonomy of MADRL attributes, including objectives, characteristics, challenges, applications, and performance measures. The MADRL algorithms are classified, analyzed, and discussed based on the taxonomy, and their open issues are explained. To the best of our knowledge, our explanations from these perspectives have not been presented in the literature. Table 1 presents a summary of recent surveys of MADRL in terms of foci, approach, and the targeted multi-agent environment. The table also shows details about our paper. Ultimately, this paper aims to outline active and recent research areas in MADRL applications and motivate future research.

The rest of this paper is organized as follows. Section 2 presents RL and its multi-agent approaches. Section 3 presents the theory behind MADRL and its representation. Section 5 presents the attributes of MADRL in a taxonomy. Section 6 presents various applications of MADRL based on the taxonomy. Section 7 presents open issues and future directions. Section 8 concludes the paper.

Table 1. Summary of the foci, approaches, and targeted multi-agent environments of recent and our surveys.

Reference, Year	Foci	Approach	Multi-Agent Environment
Nguyen et al. (2020) [7]	The use of transfer learning in MADRL approaches for non-stationarity and partial observable multi-agent environments	Deep Learning	• Collaborative • Competitive
Hernandez-Leal et al. (2018) [8]	An overview of the MADRL literature on the use of RL and MARL in multi-agent environments, and the computational complexity of MADRL	Deep Learning	• Collaborative
Oroojlooyjadid and Hajinezhad (2019) [9]	Enhancement of MADRL approaches and algorithms (i.e., independent learners, fully observable critic, value function factorization, consensus, and learning to communicate)	Deep Learning	• Collaborative
Da Silva et al. (2020) [10]	Transfer learning in MADRL	Deep Learning	• Collaborative • Competitive

Table 1. *Cont.*

Reference, Year	Foci	Approach	Multi-Agent Environment
Zhang et al. (2021) [11]	Theoretical analysis of MADRL convergence guarantees	Game Theory	• Collaborative • Competitive • Mixture
Gronauer and Diepold (2021) [12]	Theoretical analysis of MADRL approaches in terms of training schemes and the emergent patterns of agent behavior	Deep Learning	• Collaborative • Competitive • Mixture
This survey	MADRL models and algorithms applied to various multi-agent environments	Deep Learning	• Collaborative • Competitive • Mixture

2. Background of Multi-Agent Deep Reinforcement Learning: Reinforcement Learning and Multi-Agent Reinforcement Learning

This section explains RL and MARL, which serve as the fundamentals for MADRL.

2.1. Reinforcement Learning

The traditional RL approach [13], which is formulated based on Markov decision process (MDP) [8,12,14,15], enables a *single* agent (or a decision maker) to interact with its operating environment in a trial and error manner, learn a policy (e.g., a control policy), and perform sequential decision making for optimizing system performance. One of the popular SARL (single-agent reinforcement learning) (called RL for simplicity henceforth) approaches is Q-learning.

In Q-learning, an agent learns action-value functions, and it has three main advantages in that an agent: (a) models the system performance, instead of individual factors affecting the performance; (b) does not require prior knowledge about the dynamic operating environment; and (c) does not require transition probability. Q-learning enables an agent i to observe state $s_t^i \in S$ and select the best possible action $a_t^i \in A^i$ at time t, and then receive an immediate reward $r_{t+1}^i(s_{t+1}^i)$ at time $t+1$. Subsequently, the agent i updates Q-value $Q_t^i(s_t^i, a_t^i)$ that represents the appropriateness of taking action a_t^i under state s_t^i for the state–action pair (s_t^i, a_t^i) as follows:

$$Q_{t+1}^i(s_t^i, a_t^i) \leftarrow (1-\alpha)Q_t^i(s_t^i, a_t^i) + \alpha[r_{t+1}^i(s_{t+1}^i) + \gamma \max_{a \in A} Q_t^i(s_{t+1}^i, a)] \tag{1}$$

where $\alpha \in [0,1]$ represents the learning rate, $\gamma \max_{a \in A} Q_t^i(s_{t+1}^i, a)$ represents the discounted reward, and $\gamma \in [0,1]$ represents the preference for the discounted reward compared to the immediate reward. A two-dimensional look-up Q-table is used to store $|S^i| \times |A^i|$ entries of agent i's Q-values; hence, the Q-table size increases exponentially with increasing number of states $|S^i|$ and actions $|A^i|$, particularly in complex operating environments, resulting in the disadvantages of the curse of dimensionality and longer convergence time to the optimal action.

Algorithm 1 shows the Q-learning algorithm. An agent i observes the current state s_t^i (step 2) and chooses an action a_t^i (step 3) at time t. The agent i selects one of the two types of actions: (a) the exploitation action $a_t^i = \text{argmax}_{a \in A} Q_t^i(s_t^i, a)$ has the maximum Q-value, and it is chosen to maximize value function $v_\pi^i(s_t^i)$; and (b) the exploration is a random action, and it is chosen to discover better actions in order to adapt to the operating environment. The agent can use techniques such as ε-greedy [16] and the softmax [13] approach, to achieve a balanced trade-off between exploration and exploitation based on the dynamicity of the operating environment. The agent i receives immediate reward $r_{t+1}^i(s_{t+1}^i)$ (step 4) and updates Q-value $Q_{t+1}^i(s_t^i, a_t^i)$ using Equation (1) at time $t+1$ (step

5). The steps are repeated as time goes by. The algorithm assumes a single agent in the operating environment, and so it optimizes the local performance regardless of the global performance. Q-learning converges to the optimal Q-value Q^* when: (a) the environment has discrete and finite state and action spaces; (b) each state–action pair is observed infinitely; (c) the sum of the learning rate goes to infinity; and (d) the sum of the squares of the learning rates is finite [17–19].

There are two main types of RL, namely, temporal difference (TD) [20] (i.e., Q-learning) and Monte Carlo (MC). Both TD and MC are model-free and learn based on experience without the need for prior knowledge or the transition probability, and each state and action must be visited often to ensure convergence. In an MC method, such as the policy gradient method, the value function $v_\pi(s_t)$ is estimated based on the average return received at the end of each episode. In the TD method, the value function is estimated based on the reward received at each step, and so it achieves faster convergence [7].

Algorithm 1: Q-learning algorithm.

```
1: **Procedure**
2:     Observe current state $s_t^i$
3:     Select action
           $a_t^i = \begin{cases} \text{random}, & \text{if exploration} \\ \text{argmax}_{a \in A} Q_t^i(s_t^i, a), & \text{if exploitation} \end{cases}$
4:     Receive immediate reward $r_{t+1}^i(s_{t+1}^i)$
5:     Update Q-value $Q_{t+1}^i(s_t^i, a_t^i)$ using Equation (1)
6: **End Procedure**
```

2.2. *Multi-Agent Reinforcement Learning*

The traditional MARL approach [21], which is formulated based on a Markov game [22], enables multiple agents to interact with its operating environment and neighboring agents in a trial and error manner, learn a policy (e.g., a control policy), and perform sequential decision making for optimizing system-wide performance. A Markov game is the generalization of the MDP in the multi-agent and shared environment; specifically, it represents the I-agent MDP problem $(S, A^1 \times \ldots \times A^i, R^1, \ldots, R^i, P)$, where $I = 1, \ldots, i, \ldots |I|$ is a set of interacting agents solving a cooperative task [23], S is the global state represented by the set of states observed by the $|I|$ agents, and $a^i \in A^i$ is an action, as part of the set of joint action $A^1 \times \cdots \times A^i$, selected by agent i. The transition probability function $P : S \times A^1 \times \cdots \times A^i \to P(S)$ describes the state transition from $s_t \in S$ to $s_{t+1} \in S$. Each agent i receives its own reward, given the reward function $R^i : S \times A^1 \times \cdots \times A^i \to R$ that provides the immediate reward $r_{t+1}^i(s_{t+1}^i) \in R^i$ as a feedback signal for each agent i. Specifically, (a) each agent $i \in I$ observes state $s_t^i \in S$, and selects and executes action $a_t^i \in A^i$ based on individual policy $\pi^i : S \to P(A^i)$ at time step t; (b) the environment transits from the current state $s_t \in S$ to next state $s_{t+1} \in S$; and (c) each agent $i \in I$ receives immediate reward $r_{t+1}^i(s_{t+1}^i) \in R^i$ based on the joint action $A^1 \times \cdots \times A^i$ and the transition probability P. Unlike RL, the aim of the agents in MARL is to change their policies in a way that maximizes expected long-term system-wide rewards in the future.

In Markov games, it is assumed that each agent i observes state s_t^i that captures all decision-making information of the system. However, in practice, agents may have partial observations about the complex environment only, and so they make sequential decisions under uncertainty. This is referred to as the partially observable Markov game [24] (commonly called partially observable Markov decision process), which is a generalization of both Markov game and MDP, represented by $(S, O^1 A^1 \times \ldots \times O^i A^i, R^1, \ldots, R^i, P)$, where $o^i \in O^i$ is the observation of a local state, as part of the global state S, observed by agent i. Hence, agent i learns individual policy $\pi^i : O_t^i \to P(A^i)$ that maps local observations to individual actions at time step t. For simplicity, an agent i observation o_t^i is represented by a local state s_t^i. To solve a Markov game, MARL enables multiple agents to interact with the operating environment and exchange knowledge (e.g., Q-value $Q_t^i(s_t^i, a_t^i)$ and immediate reward $r_{t+1}^i(s_{t+1}^i)$) among themselves, to learn the best possible joint policy $\pi(a_t^1, \ldots, a_t^i | s_t)$ in order to achieve global optimization in a collaborative manner as time goes by.

While the state transition in RL is attributed to the environmental dynamics of an agent only, the *local* state transition in MARL is attributed to both environmental dynamics and the non-stationary policies of neighboring agents learning at the same time. Hence, each agent's actions can cause further dynamicity in the operating environment. This problem is called moving target [25], whereby an agent's best possible policy can affect other agents' performances, hence their policies, since all agents select and take their respective actions simultaneously in a shared operating environment. In this case, the convergence property of MDP no longer applies, so the single-agent RL approach may not converge when it is applied in a multi-agent environment. Knowledge exchange enables agents to consider their own and others' performances in order to address the moving target problem. The knowledge sharing feature of MARL has the advantage of achieving higher scalability, because a complex system-wide optimization problem can be decomposed into a set of distributed problems solved by individual agents in a distributed manner with faster learning [26,27].

Algorithm 2 shows the MARL algorithm. MARL enables an agent $i \in I$ to observe its state $s_t^i \in S^i$ (step 2), exchange knowledge (steps 3 and 4), and select the best possible action $a_t^i \in A^i$ at time t in an independent manner (step 5), and then receive an immediate reward $r_{t+1}^i(s_{t+1}^i)$ at time $t+1$ (step 6). Subsequently, agent i updates its Q-value $Q_{t+1}^i(s_t^i, a_t^i)$ using the Q-function (step 7) as follows:

$$Q_{t+1}^i(s_t^i, a_t^i) \leftarrow (1-\alpha) Q_t^i(s_t^i, a_t^i) + \alpha [r_{t+1}^i(s_{t+1}^i) + \gamma \sum_{j \in N} \eta^{i,j} \max_{a^j \in A} Q_t^j(s_{t+1}^j, a^j)] \quad (2)$$

where $\sum_j \eta^{i,j} = 1$, $\eta^{i,j}$ represents the importance of neighboring agent $j \in J^i$ at agent i, and J^i represents a set of agent i's neighboring agents. The joint action $a_t = (a_t^1, \ldots, a_t^i, \ldots, a_t^N)$ generates immediate reward $r_{t+1}(s_{t+1}) = (r_{t+1}^1(s_{t+1}^1), \ldots, r_{t+1}^i(s_{t+1}^i), \ldots, r_{t+1}^N(s_{t+1}^N))$ and converges to the optimal joint action as time goes by.

Algorithm 2: MARL algorithm.

```
1: Procedure
2:     Observe current state $s_t^i \in S$
3:     Send Q-value $Q_t^i(s_t^i, a_t^i)$ to neighboring agents $J^i$
4:     Receive $\max_{a^j \in A^j} Q_t^j(s_t^j, a^j)$ from agent $j \in J^i$
5:     Select action
           $a_t^i = \begin{cases} \text{random}, & \text{if exploration} \\ \text{argmax}_{a \in A} Q_t^i(s_t^i, a), & \text{if exploitation} \end{cases}$
6:     Receive immediate reward $r_{t+1}^i(s_{t+1}^i)$
7:     Update Q-value $Q_{t+1}^i(s_t^i, a_t^i)$ using Equation (2)
8: End Procedure
```

There are two types of MARL systems [12]: (a) the cooperative environment in which each agent maximizes $r_{t+1}^i(s_{t+1}^i) \in R$ as part of the global reward R in order to maximize the system performance; and (b) the competitive environment in which each agent maximizes its own local reward, which may minimize neighboring agents' rewards, rather than the global reward, and so the sum of rewards of all agents equal to zero $R = \sum_{i=1}^N R^i\left(s^i, a^i, s^{i'}\right) = 0$ [28]. This is also known as the zero-sum Markov game [29].

3. Multi-Agent Deep Reinforcement Learning

This section explains MADRL, which serve as the fundamentals for the discussion in Section 6.

3.1. Deep Reinforcement Learning

DRL addresses the shortcomings of RL and MARL—particularly the curse of dimensionality, whereby storing knowledge in a tabular manner is unable to cater for large scale and complex problems—by using neural networks as function approximators to generalize and approximate the value function. Deep Q-network (DQN), which is a value-based method is a popular single-agent DRL approach [30]. Using DQN, a single agent: (a) addresses the curse of dimensionality using DNN to represent and store states, actions, and knowledge; and (b) ensures convergence and increases the convergence rate to the optimal joint action (or increases the learning speed) using distinguishing features, particularly replay memory and target network. This enables DQN to solve complex problems. For instance, in [31], DQN was embedded in a centralized agent to gather global information from distributed agents, learn, select optimal actions, and then send the selected actions back to the distributed agents that performed the selected actions.

The MADQN approach, as shown in Figure 1, consists of multiple agents interacting with the operating environment and with each other. Each agent is based on the single-agent DQN approach. Hence, each agent in the multi-agent environment has: (a) a main network with a fully connected DNN to provide Q-value $Q_t^i(s_t^i, a_t^i; \theta^i)$; (b) a target network, which is a duplicate of the main network, to provide a stable target for learning; and (c) a replay memory to store the agent's experiences. The fully connected DNN has: (a) the input layer which consists a set of neurons to receive a state that consists of substates; (b) the hidden layer; and (c) the output layer which consists a set of neurons to provide the Q-value $Q_t^i(s_t^i, a_t^i; \theta^i)$ of each possible action $a_t^i \in A^i$. Each link between neurons k^i and j^i has a weight $w_k j^i$, and the output of neuron k^i is as follows:

$$y_k^i = \varphi(\sum_{j=0}^{m} w_{kj}^i . x_j^i) \tag{3}$$

where φ is an activation function (e.g., sigmoid function); the input x_j^i is assigned a weight w_{kj}^i representing its importance compared to other inputs $(x_0^i, \ldots, x_j^i, \ldots, x_m^i)$. DQN has two types of networks that use the fully connected neural network architecture, namely, the main network and the target network. The main network produces Q-values for all the possible actions A^i in its output layer, and the action a_t^i with the maximium Q-value is selected during exploitation. The target network, which is a duplicate of the main network every C steps, produces a stable target for learning.

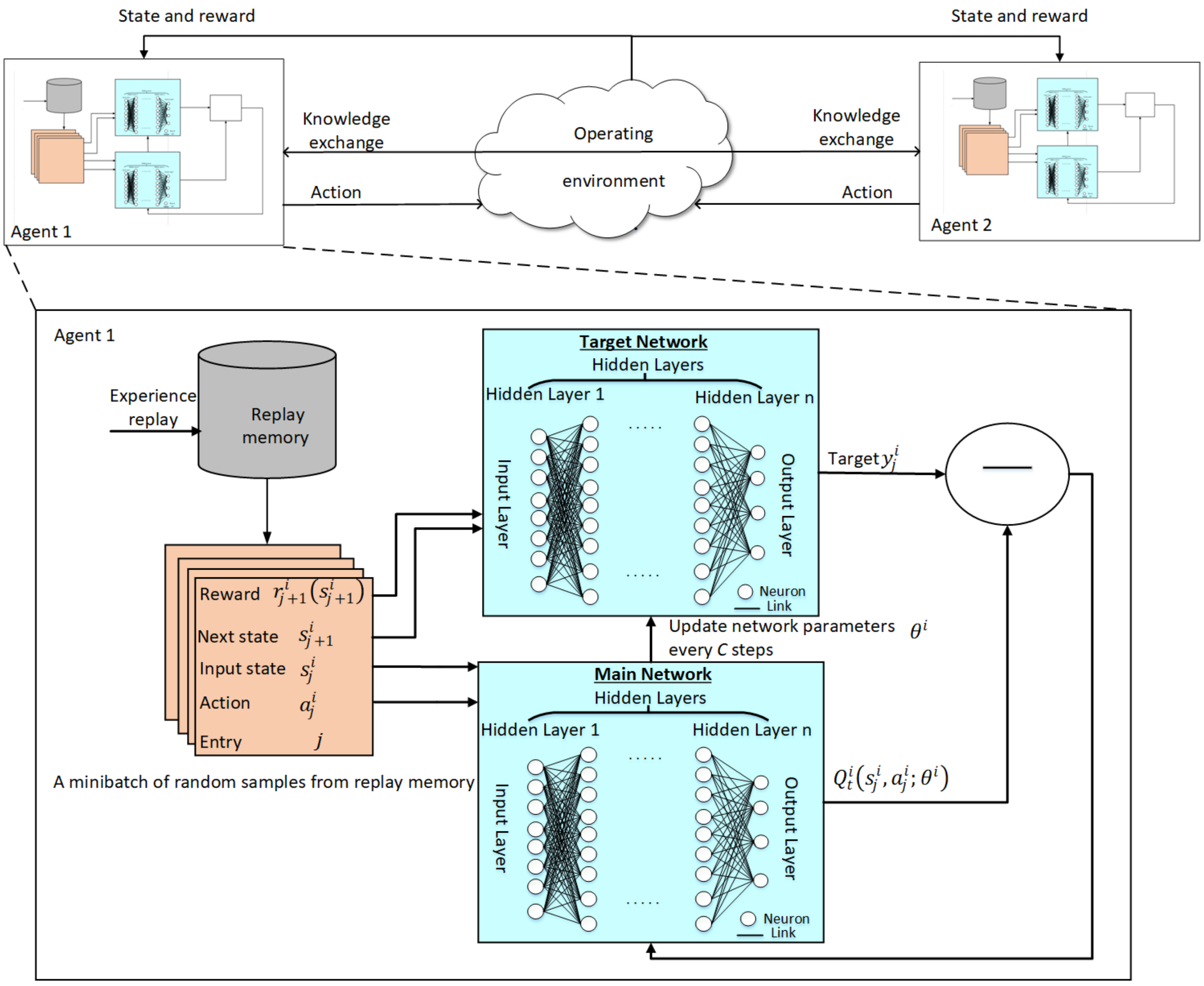

Figure 1. An example of a MADQN architecture. Two agents exchange information (e.g., Q-values) used to update their respective Q-values. Each agent observes and stores its state and reward from the operating environment in replay memory for training, and selects its own action. Agent 1 is enlarged to show: (a) the replay memory; (b) the main network that produces Q-values for all possible actions given a particular state; and (c) the target network that produces a stable target used to compute the loss of the selected action.

Algorithm 3 shows the DQN algorithm. There are two main phases. During the action selection phase, the agent i observes the current state s_t^i (step 3) and chooses an exploitation or exploration action a_t^i (step 5) at time t. The agent i receives immediate reward $r_{t+1}^i(s_{t+1}^i)$ and next state s_{t+1}^i (step 6), and stores the transition (or experience) $(s_t^i, a_t^i, r_{t+1}^i(s_{t+1}^i), s_{t+1}^i)$ in its replay memory D_t^i (step 7). During the training phase, the agent i selects a mini-batch of samples randomly from its replay memory D_t^i to train the main network (step 8). The agent i calculates a target y_j^i using a target Q-value $Q_j^i(s_j^i, a_j^i; \theta^{i,-})$, which is generated using the target network $\theta^{i,-}$, over several steps $j = 1, \ldots, N$ (step 10). The target network $\theta^{i,-}$ is a duplicate of the main network θ^i. The target y_j^i is used to compute the loss $L_j^i(\theta^i)$ of the selected action a_j^i; and the loss $L_j^i(\theta^i)$, which is expected to reduce as time goes by, is used to train the main network (step 11). The loss $L_j^i(\theta^i)$ is calculated as follows:

$$L_j^i(\theta^i) = \mathbb{E}_{s_j^i, a_j^i \sim p(.)}[(y_j^i - Q_j^i(s_j^i, a_j^i; \theta^i))^2] \tag{4}$$

where $p(s, a)$ is the probability distribution of a state–action pair. The agent i performs gradient descent on the loss $L^i_j(\theta^i)$ and updates the weight of its main network θ^i. The gradient of the loss function $\nabla_{\theta^i} L^i_j(\theta^i)$ is as follows:

$$\nabla_{\theta^i} L^i_j(\theta^i) = \mathbb{E}_{s^i_j, a^i_j \sim p(.):s^i_{j+1} \sim \epsilon}[(y^i_j - Q^i_j(s^i_j, a^i_j; \theta^i)\nabla_{\theta^i} Q^i_j(s^i_j, a^i_j; \theta^i)] \tag{5}$$

Algorithm 3: DQN algorithm.

```
1: Procedure
2:   for episode = 1 : $M$ do
3:     Observe current state $s^i_t$
4:     for time $t$ = 1 : $T$ do
5:       Select action
         $a^i_t = \begin{cases} \text{random}, & \text{if exploration} \\ \text{argmax}_{a \in A} Q^i_t(s^i_t, a; \theta^i), & \text{if exploitation} \end{cases}$
6:       Receive immediate reward $r^i_{t+1}(s^i_{t+1})$ and next state $s^i_{t+1}$
7:       Store experience $(s^i_t, a^i_t, r^i_{t+1}(s^i_{t+1}), s^i_{t+1})$ in replay memory $D^i_t$
8:       Sample a minibatch of $N$ experiences $(s^i_t, a^i_t, r^i_{t+1}(s^i_{t+1}), s^i_{t+1})$ from replay memory $D^i_t$
9:       for step $j$ = 1 : $N$ do
10:        Set target
           $y^i_j = \begin{cases} r^i_{j+1}(s^i_{j+1}), & \text{if episode } m \text{ terminates} \\ r^i_{j+1}(s^i_{j+1}) + \gamma \text{max}_a Q^i_t(s^i_{j+1}, a; \theta^{i,-}), & \text{otherwise} \end{cases}$
11:        Perform gradient descent on $(y^i_j - Q^i_t(s^i_j, a^i_j; \theta^i))^2$ with respect to main network parameters $\theta^i$ using Equation (5)
12:      end for
13:      Perform $\theta^{i,-} = \theta^i$ every C steps
14:    end for
15:  end for
16: End Procedure
```

3.2. Multi-Agent Deep Q-Network

The multi-agent deep Q-network (MADQN) is based on DQN, and so it uses DNN to address the curse of dimensionality, and replay memory and target network to increase the convergence rate to the optimal joint action. Most importantly, MADQN extends the single-agent DQN approach. Multiple agents exchange knowledge among themselves in order to achieve global optimization in a collaborative manner, and so it addresses the moving target problem. Knowledge exchange enables agents to consider their own and neighboring agents' performances [32].

Algorithm 4 shows the MADQN algorithm. MADQN enables an agent $i \in N$ to observe its state $s^i_t \in S^i$ (step 3), exchange knowledge (steps 4 and 5), perform steps 5 to 13 in Algorithm 3, and update its Q-value $Q^i_{t+1}(s^i_t, a^i_t)$ using the Q-function (step 9). The agent uses the exchanged knowledge (i.e., Q-value) to calculate the Q-value of each state–action pair as follows [33]:

$$\begin{aligned} Q^i_{t+1}(s^i_t, a^i_t) = (1-\alpha)Q^i_t(s^i_t, a^i_t; \theta^i) + \alpha[r^i_{t+1}(s^i_{t+1}) \\ +\gamma \max_{a \in A^i} Q^i_t(s^i_{t+1}, a; \theta^{i,-}) - Q^i_t(s^i_t, a^i_t; \theta^i)] \\ + \sum_{j \in N} \eta^{i,j} Q^j_{t-1}(s^j_{t-1}, a^j_{t-1}; \theta^j) \end{aligned} \tag{6}$$

The traditional DQN approach and MADQN approaches have been applied in a multi-agent environment with centralized and distributed agents. The traditional DQN approach can be embedded in the centralized agent to: (a) gather data, information, or observations from distributed agents; (b) learn from the gathered data and update Q-values to maximize the expected long-term system-wide rewards; and (c) send optimal actions to the distributed agents, which execute the optimal actions accordingly. For achieving scalability, DQN can be embedded in the distributed agents with two assumptions [34]: (a) each agent ignores the presence of the rest of the agents' actions and assumes that they are part of the operating environment; and (b) the changes of the rest of the agents' actions in response to each agent's actions being part of the dynamicity of the operating environment. Unfortunately, these assumptions can cause instability, particularly when the rest of the agents start learning and their actions become non-stationary, causing the agents to fail to converge on the optimal joint action. Hence, MADQN enables agents to exchange knowledge with each other and learn the best possible actions in order to achieve scalability (O.5).

MADQN can be embedded in distributed agents so that each of them is aware of the rest of the agents. To enable this, agents exchange knowledge and use it to update their respective Q-values, $Q^i_{t+1}(s^i_t, a^i_t; \theta^i)$, based on a weighted sum of delayed rewards or value functions from themselves and neighboring agents using Equation (6), where J^i is a set of agent i's neighboring agents, and $\eta^{i,j}$ represents the importance of neighboring agent j at agent i. For instance, with $\sum_j \eta^{i,j}/|J^i|$, the Q-value for neighboring agent j has an equal effect on an agent i's Q-value $Q^i_{t+1}(s^i_t, a^i_t; \theta^i)$. Therefore, the agents select their respective actions while ensuring that the global Q-value converges to a unique and optimal equilibrium. The agents may exchange knowledge via direct communication or through prediction. For instance, the agents create the models of the operating environment and neighboring agents to predict their Q-values. As an example of an extension to DRL, MADRL uses actor–critic networks [35,36]. Each agent has unique actor and critic networks: (a) the critic network corrects the immediate reward using temporal difference; and (b) the actor network updates the Q-values using the temporal difference. Table 2 highlights the differences between RL methods and their convergence criteria.

Table 2. A summary of comparisons between MADQN and other preceding approaches.

Method	**Description**	**Advantages**	**Disadvantages**	**Convergence Criteria**
RL	A single agent interacts with the operating environment to learn the optimal policy, such as the action-value function (or Q-value) in Q-learning	• Models the system performance instead of individual factors affecting the performance • Does not require prior knowledge about the dynamic operating environment • Does not require transition probability	• Not suitable for distributed agents in multi-agent environments • Suffer from the curse of dimensionality	• Discrete and finite action state spaces • Each state–action pair is observed infinitely • The sum of the squares of the learning rates is finite

Table 2. *Cont.*

Method	Description	Advantages	Disadvantages	Convergence Criteria
MARL	Multiple agents exchange knowledge with each other and interact with the operating environment to learn the optimal policy	• Agents exchange knowledge to solve complex tasks • Ensures global optimization in the multi-agent environment	• Higher complexity compared to RL	• Knowledge must be exchanged and processed with a certain level of recurrence to ensure stability among agents
DQN	A single agent interacts with the operating environment to learn the optimal policy using DNN with experience replay and target network, which addresses the curse of dimensionality (or high-dimensional state–action spaces)	• Has efficient representation of states, actions, and knowledge • Uses target network to provide stable Q-value for learning in order to ensure convergence • Uses replay memory to provide experiences for learning in order to improve convergence rate	• Not suitable for distributed agents in the multi-agent environment • Higher memory and computation requirements compared to RL and MARL	• Replay memory must be of large size to ensure independent transitions among samples, and break sample correlations [37] • Small network parameters so that agents focus on important aspects and achieve a faster convergence [38]
MADQN	Multiple agents exchange knowledge with each other and interact with the operating environment to learn the optimal policy using DNN with experience replay and target network, which addresses the curse of dimensionality (or high-dimensional state–action spaces)	• Agents exchange knowledge to solve complex tasks • Ensures global optimization in the multi-agent environment • Has efficient representation of states, actions, and knowledge • Uses target network to provide stable Q-value for learning in order to ensure convergence • Uses replay memory to provide experiences for learning in order to improve convergence rate	• Higher complexity compared to RL, MARL, and DQN.	• Replay memory must be of large size to ensure independent transitions among samples, and break sample correlations [37] • Small network parameters so that agents focus on important aspects and achieve a faster convergence [38] • Knowledge must be exchanged and processed with a certain level of recurrence to ensure stability among agents

Algorithm 4: MADQN algorithm.

```
1: Procedure
2:  for episode = 1 : $M$ do
3:      Observe current state $s_t^i$
4:      Send Q-value $Q_t^i(s_t^i, a_t^i)$ to neighboring agents $J^i$
5:      Receive $\max_{a^j \in A} Q_t^i(s_t^j, a_t^j)$ from agent $j \in J^i$
6:        for $t = 1 : T$ do
7:          Preform steps 5 to 12 in Algorithm 3
8:          end for
9:      Update $\theta^{i,-} = \theta^i$ every C steps
10:     Update Q-value $Q_{t+1}^i(s_t^i, a_t^i)$ using Equation (6)
11:    end for
12: End Procedure
```

The convergence criteria of RL are not similar to those in MADRL, due to the non-stationarity of multi-agent environments [12]. Nevertheless, some works [21,39,40] have shown that the convergence criteria of RL can also be applied to a multi-agent environment. However, the underlining assumption of these works is that agents learn equally (i.e., actions and rewards of all agents are observable), which may not be possible practically [41]. Due to the complexity of multi-agent environments, most MADRL models and algorithms show convergence empirically. Convergence in cooperative MADRL using a single learning algorithm has been shown in [42,43]. In practice, there is lack of investigation on the convergence of MADQN [18,44,45], although the convergence rate has been shown to be higher for policy gradient methods (e.g., proximal policy optimization [46]) and actor–critic methods (e.g., asynchronous advantage actor–critic (A3C) [47]) as compared to value-based methods (i.e., DQN). The traditional DQN approach has been extended to several variants. For instance, double DQN reduces the overestimation issues [48] of the traditional DQN approach, and the dueling-DQN architecture decomposes the Q-value into two learning parts (i.e., state value and advantage value) and combines them in the final layer to estimate Q-value [49]. Moreover, in continuous environments (e.g., physical control), policy gradient methods such as deep deterministic policy gradient (DDPG) enhance the traditional DQN approach by slowly updating the target network rather than directly copying the weights (or network parameters), as seen in the traditional DQN approach. Additionally, A3C improves the training efficiency of the traditional DQN by enabling parallelized asynchronous training, in which multiple tasks can be solved simultaneously [50].

4. Research Methodology

The literature search was conducted using relevant search keywords shown in Table 3. There are three main topics, namely, general topics related to this research area, single-agent approaches applied in multi-agent environments, and multi-agent approaches. Our search identified 28 related papers in four widely used literature databases: Web of Science, ScienceDirect, Multidisciplinary Digital Publishing Institute (MDPI), and IEEE Xplore Digital Library. The selected papers presented DRL models and algorithms, including multi-agent approaches and single-agent approaches applied in multi-agent environments. The selected papers had to present the DRL designs and applications, along with a simulation and empirical results. Based on the related papers, there are 12 categories of DRL approaches that have been applied in a multi-agent environment. As shown in Figure 2, nine papers explain the use of the SADRL approach in a multi-agent environment; three papers explain the SADRL approach in a hierarchical multi-agent environment; three papers explain the MADRL approach with actor–critics applied to distributed entities; two papers explain the SADRL approach with actor–critics applied to centralized entities in a hierarchical multi-agent environment; and one paper explains each of the rest of the categories. Subsequently, we review the enhancements and performance of the traditional SADRL and MADRL approaches with a taxonomic approach to highlight various aspects of the enhancements, including objectives, characteristics, challenges, applications, and performance measures.

Table 3. Search keywords and their topics

Topic	Keyword
General	• Markov decision process (or MDP) • Applied reinforcement learning
Single-agent approaches applied in multi-agent environments	• Single-agent deep reinforcement learning (or SADRL) • Deep reinforcement learning (or DRL) • Deep Q-network (DQN) • Q-learning • Actor–critic network
Multi-agent approaches	• Multi-agent learning • Multi-agent system • Multi-agent optimization • Multi-agent deep reinforcement learning (or MADRL) • Multi-agent deep Q-network (MADQN) • Knowledge sharing • MADRL applications • MADRL convergence • Cooperative MADRL • Competitive MADRL

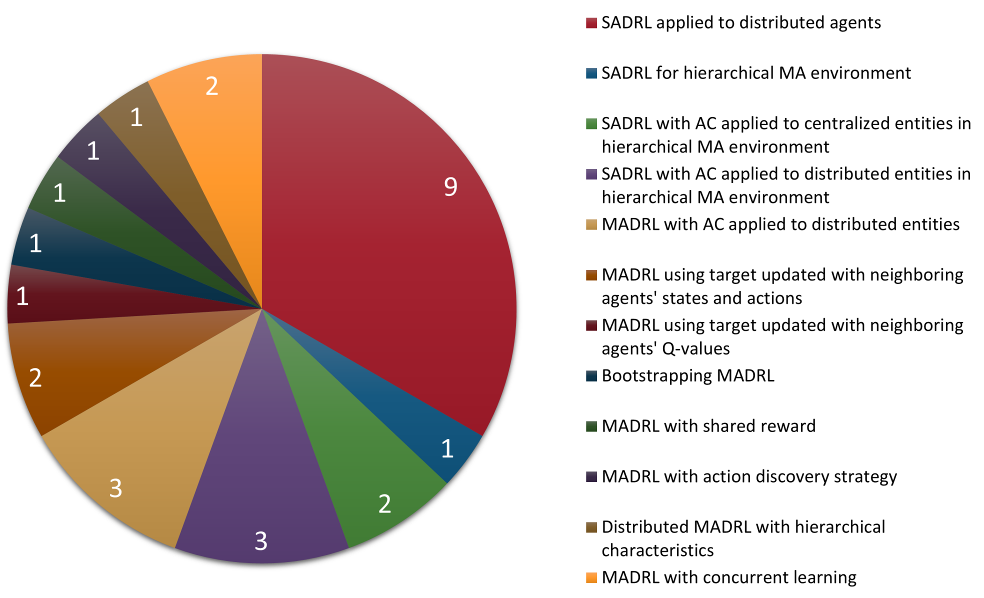

Figure 2. Categories of related papers in the literature.

5. Attributes of Multi-Agent Deep Reinforcement Learning

This section presents the taxonomy of MADRL attributes, as shown in Figure 3.

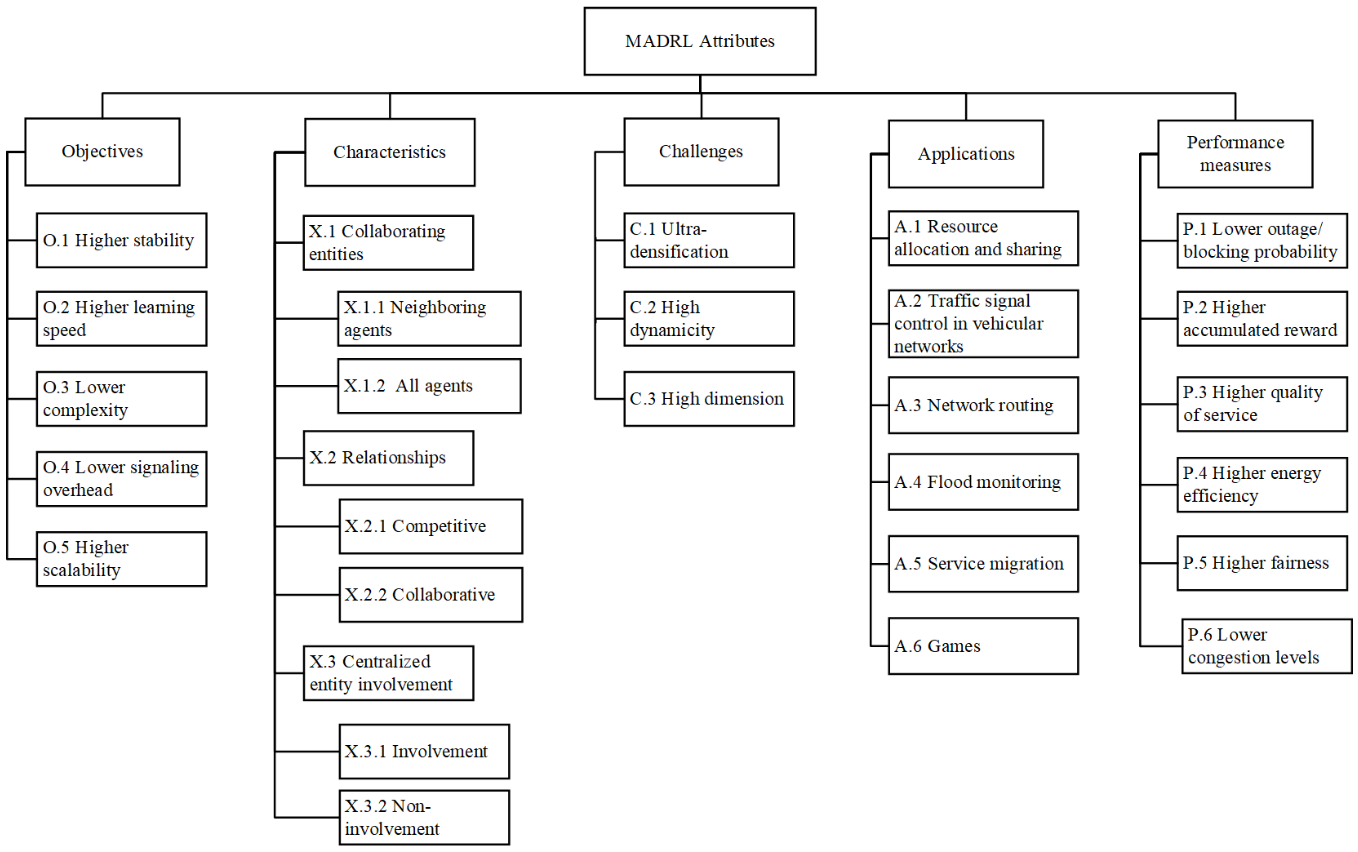

Figure 3. Taxonomy of MADRL attributes.

5.1. Objectives

MADRL has been applied to achieve the following objectives in the literature:

O.1 *Higher stability* ensures the convergence to the optimal joint action. It mitigates instability caused by several factors: (a) the dynamic and unpredictable operating environment; and (b) the dynamic and unpredictable actions are taken in a simultaneous (or nonsequential) manner by multiple agents. As an example, in order to improve the stability of the multi-agent environment: (a) distributed agents apply knowledge from the centralized entity to select their respective actions [35]; and (b) a common reward is given to distributed agents to encourage cooperation among themselves [51].

O.2 *Higher learning (or convergence) speed* reduces the number of iterations required to converge to the optimal joint action in the presence of large state and action spaces. As an example, in order to improve the learning speed in a multi-agent environment: (a) distributed agents select samples based on their priorities from the replay memory, whereby learned samples are prioritized, and redundant samples are removed [52,53]; and (b) distributed agents prioritize knowledge received from adjacent agents and scale down knowledge received from non-adjacent agents in order to increase the effects of local knowledge which is more correlated with their local states.

O.3 *Lower complexity* reduces the computational and hardware implementation complexities of DNN in multi-agent environments. It addresses the complexity of learning among distributed agents and the large number of iterations required for training (e.g., computing the value function). As an example, in order to reduce complexity in a multi-agent environment: (a) distributed agents segregate and distribute a complex task among themselves according to their respective capabilities [35]; and (b) the loss function of each training step is sampled and differentiated with respect to network parameters in order to update the Q-network with reduced complexity [52].

O.4 *Lower signaling overhead* reduces the frequency of message exchange between centralized and distributed entities. It addresses the increase of message exchange when the system size grows. A lower signaling overhead has been shown to reduce co-channel interference and improve fairness among distributed agents in wireless networks [52]. As an example, in order to reduce signaling overhead in a multi-agent environment: (a) distributed agents transfer the complex training process to the centralized entity [52]; and (b) the centralized entity gathers data from selected distributed agents rather than all of them [35].

O.5 *Higher scalability* enhances a system with a large number of agents. It addresses the challenges of segregating a centralized scheme and the need of exchanging a large amount of knowledge or information [35]. For example, distributed agents select a subset of states from a large complex set of states [35]. Higher scalability helps to improve various aspects of a system with a large number of distributed agents, including achieving a higher learning speed, a lower complexity, and a lower signaling overhead.

5.2. Characteristics

MADRL possesses the following characteristics in the literature:

X.1 *Collaborating entities* represent the agents that share knowledge in a multi-agent environment as follows:

- X.1.1 *Neighboring agents* indicates that an agent shares knowledge with neighboring agents only.
- X.1.2 *All agents* indicates that an agent shares knowledge with all agents in the environment.

X.2 *Relationship* indicates the relationship between agents in a multi-agent environment as follows:

- X.2.1 *Competitive* indicates that agents compete with each other for network services or resources (e.g., transmission opportunities).
- X.2.2 *Collaborative* indicates that agents collaborate with each other to achieve a common goal.

X.3 *Centralized entity* represents the centralized controller with a higher amount of resources (e.g., higher computing power) to perform centralized and complex tasks in a multi-agent environment. Thus, distributed agents perform distributed and simple tasks.

- X.3.1 *Involvement from the centralized entity* indicates that the centralized controller performs complex tasks in a centralized manner.
- X.3.2 *Non-involvement from the centralized entity* indicates that distributed agents perform complex tasks in a distributed manner.

5.3. Challenges

MADRL has been applied to address the following challenges in the literature:

C.1 *Ultra-densification* is attributed to the presence of a large number of agents (e.g., user devices) in the operating environment. Consequently, it affects the capability of multiple agents to exchange knowledge among themselves in a shared operating environment. For instance, in [35], the centralized entity performs learning by gathering data from a predefined number of neighboring distributed agents, and distributes knowledge to them so that they can select actions independently.

C.2 *High dynamicity* is attributed to the rapid changes of the operating environment (e.g., due to high mobility of agents), including states and rewards. Consequently, it affects learning and decision making. For instance, in [6], historical information (i.e., actions) related to distributed agents are part of the input information (i.e., state) of the DNN in order to adapt to the highly dynamic operating environment.

C.3 *High dimension* is attributed to the presence of a large state and action spaces. For instance, in [54], distributed agents receive knowledge about each other (e.g., delayed rewards and Q-values), use a function approximator to store and keep track of a smaller number of features instead of a large number of state–action pairs, and then use them to select their respective actions.

5.4. Applications

The main applications of MADRL in the literature are as follows:

A.1 *Resource allocation and sharing* enable agents to allocate and manage resources, such as distributed renewable energy sources (e.g., power grids) [55], and meet system demands in a distributed manner. In [35], in a wireless network, a node pair, which is a distributed agent, selects its operating channel for device-to-device (D2D) communication in a distributed manner. D2D enables neighboring nodes to communicate with each other directly without passing through a base station. MADRL can mitigate the instability of a multi-agent environment, whereby a centralized entity learns and shares learned knowledge with the node pairs, who subsequently make decisions independently. This helps to increase throughput. In [56], in a home network, a controller switches on or off home appliances based on their activities in the past 24 h and the user's presence to meet the user's resource (i.e., energy) demands and reduce resource consumption. MADRL addresses ultra-densification in which resources are limited in the presence of a large number of distributed agents. The centralized agent learns and performs action selection rather than the distributed agents, which also helps to increase energy efficiency.

A.2 *Traffic signal control in vehicular networks* enables traffic signal controllers at intersections to manage and control traffic, such as selecting traffic phases (e.g., the north–south and west–east bounds) following traffic rules (e.g., vehicles cannot make a right turn) for reducing congestion at intersections [57,58]. In [33], traffic signal controllers select their respective traffic phases in a distributed manner. MADRL addresses the ultra-densification and high dynamicity challenges using actors and critics that share knowledge (i.e., historical actions) with neighboring distributed agents cooperatively in order to select their respective actions. This helps to reduce the congestion level.

A.3 *Network routing* enables source nodes to establish routes and send packets to destination nodes in networks with dynamically changing traffic patterns. In [6], MADRL addresses the ultra-densification, high dynamicity, and high dimension challenges by learning based on historical knowledge while making optimal decisions in route selection. This helps to reduce latency.

A.4 *Flood monitoring* enables distributed agents to sense and monitor water levels based on real-time data to predict the complex flood levels [59,60]. In [61], multiple aircraft update their respective positions in a distributed manner, enabling them to predict and report water levels at different locations. MADRL addresses the challenge of high dimension by enabling the distributed agents to share a common reward in a cooperative manner. This helps to increase the accumulated reward.

A.5 *Service migration* enables intelligent devices, such as connected vehicles, to offload computing-intensive tasks (e.g., video and data processing, route planning, and traffic management in networks) from resource-limited distributed entities (e.g., vehicles and local servers) to centralized entities (e.g., edge servers at the edge of the network, and data centers) with extensive computing resources in order to achieve the expected service level. In [62], a service entity chooses whether or not to offload tasks to edge servers based on the utility (e.g., latency and energy) of a vehicle. MADRL addresses the ultra-densification and high dynamicity challenges by enabling distributed agents to share knowledge (i.e., historical actions) with neighboring distributed agents cooperatively in order to select their respective actions. This helps to reduce latency.

A.6 *Games* enable agents to achieve the goals of games, such as winning the games or increasing human players' engagement. In [63], two players select their movements (i.e., up or down) in a pong game. MADRL addresses the challenge of high dynamicity using a common reward received after winning a game to encourage cooperative behavior among the players. This helps to increase the accumulated rewards.

5.5. Performance Measures

MADRL has been applied to improve the following performance measures in the literature:

P.1 *Lower outage (or blocking) probability* reflects the reliability of a system to sustain its normal operation (e.g., achieving a required data rate in networking) based on: (a) internal factors (e.g., a higher amount of internal resources reduces the outage probability [31]); and (b) external factors (e.g., a communication channel with a lower interference level reduces the outage probability [35]).

P.2 *Higher accumulated reward* increases the total immediate rewards received by agents over time [35].

P.3 *Higher quality of service* reflects better system performance, such as higher throughput [35], lower end-to-end delay (or shorter packet delivery time [6]), and lower blocking probability (or the probability of an agent being denied of services due to poor system performance) [31].

P.4 *Higher energy efficiency* reduces energy consumption of devices, such as sensors and smart sockets embedded in wireless modules [64]. A balanced adjustment of on-time and off-time during the peak and non-peak times, respectively, increases energy efficiency [56].

P.5 *Higher fairness* reflects an equal distribution of network resources among distributed agents. A centralized entity may pool together network resources from distributed agents to provide shared network resources [52,65].

P.6 *Lower congestion levels* reduces the congestion caused by the high traffic volume during peak hours or due to disturbances (e.g., rainfalls) in traffic networks [66,67].

6. Application of Multi-Agent Deep Reinforcement Learning: State of the Art

This section presents the state-of-the-art MADRL approaches proposed for a diverse range of applications. This section also presents an application of the traditional DRL approach (or the single-agent deep reinforcement learning (SADRL) approach), in a multi-agent environment. Table 4 presents a summary of MADRL attributes applied to the state-of-the-art applications. In Table 4, the non-involvement of centralized entity attribute is not shown because MADRL has been applied to all distributed agents in multi-agent environment. Table 5 presents a summary on how the MADRL models in Section 6 extend the traditional MADRL.

6.1. SADRL Applied to Distributed Agents in a Multi-Agent Environment

The traditional SADRL approach (see Section 3.1) has been applied in distributed agents to handle large state and action spaces. For instance, the resource allocation scheme (A.1) in [51,68–70] address the challenge of high dynamicity (C.2) and enhance the throughput performance (P.3) of distributed agents in 4G networks. Using the SADRL approach, distributed agents: (a) do not exchange local information among themselves, and so the signaling overhead is reduced (O.4); and (b) use a lesser amount of data, including states and actions, for learning, and so it increases scalability (O.5) with a lower computational complexity. Nevertheless, applying SADRL in distributed agents has two main shortcomings, whereby distributed agents: (a) are unstable and unable to converge to the optimal joint action; and (b) are unable to solve complex tasks due to limited information. Hence, various approaches have been proposed. For instance, Nan et al. enable distributed

SADRL agents to use historical knowledge to ensure stability (O.1) in [69], and Arjit et al. detects missing data for improved reliability of medical image analysis in [71]. The rest of this section presents the SADRL approaches applied to multi-agent environments.

6.1.1. Chen's SADRL Approach in a Multi-Agent Environment

In [31], Chen et al. embedded SADRL in brokers to select routes across multiple domains from source nodes to destination nodes based on the local operating environment of the domains, and reserve resources (i.e., bandwidth) along the routes in multi-domain optical networks (A.3). The brokers maximize their individual rewards in a competitive (X.2.1) manner. Each domain is an independent network with its own local operations, rules, and management. The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the network, whereby the bandwidth availability changes with the demands from the domains and inter-domain routes.

Each broker i represents state s_t^i with a four-tuple information, including: (a) the source and destination node pair; (b) bandwidth request of the route; (c) the number of routes served by a domain in the current episode; and (d) the available bandwidth of each candidate route. The action a_t^i represents a selected route. The reward $r_{t+1}^i(s_{t+1}^i) = 1$ is awarded when the selected route is established successfully; otherwise, $r_{t+1}^i(s_{t+1}^i) = -1$. The traditional SADRL approach has been shown to increase accumulated reward (P.2) and reduce blocking probability (P.3).

6.1.2. Vila's SADRL Approach in a Multi-Agent Environment

In [72], Vila et al. embedded SADRL in base stations to select physical resource blocks, particularly the available radio resources, for sharing with other base stations efficiently (A.1). The base stations maximize the global reward in a collaborative (X.2.2) manner. The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the environment, whereby the amount of traffic load varies with the transmission power and frequency of the operating channel.

Each base station i represents state s_t^i with a two-tuple information, including the number of occupied and available physical resource blocks, respectively. The action a_t^i represents selected resource blocks. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the degree to which the network performance requirements are satisfied. The traditional SADRL approach has been shown to increase throughput (P.3).

6.1.3. Shilu's SADRL Approach in a Multi-Agent Environment

In [63], Shilu et al. embedded SADRL in mobile nodes to offload compute-intensive tasks (A.5) to a centralized entity (i.e., the base station) (X.3.1) based on their computational resources and the delay incurred in traffic offload. The mobile nodes maximize the global reward based on the shared information from base stations in a collaborative (X.2.2) manner. The proposed approach addresses the challenges of ultra-densification (C.1) caused by a large number of mobile nodes, and high dynamicity (C.2) due to the high mobility of mobile nodes.

Each mobile node i represents state s_t^i with a four-tuple of information, including: (a) computational tasks; (b) the delay incurred in traffic offload; (c) the available computational capacity of the base station; and (d) the energy consumption of the base station, which has limited energy, for processing computational tasks uploaded from mobile nodes. The action a_t^i represents whether to perform computational tasks locally or offload them to base stations. The reward (or cost) $r_{t+1}^i(s_{t+1}^i)$ represents the delay incurred. The traditional SADRL approach has been shown to reduce delay (P.3).

Table 4. Summary of MADRL approaches applied in various state-of-the-art applications.

MADRL Approach	Reference, Year	Objective					Characteristic					Challenge			Applications						Performance Measure					
		O.1 Higher stability	O.2 Higher learning speed	O.3 Lower complexity	O.4 Lower signaling overhead	O.5 Higher scalability	X.1.1 Neighboring agents	X.1.2 All agents	X.2.1 Competitive	X.2.2 Collaborative	X.3.1 Involvement from the centralized entity	C.1 ultra-densification	C.2 High dynamicity	C.3 High dimension	A.1 Resource allocation	A.2 Traffic signal control in vehicular networks	A.3 Network routing	A.4 Flood monitoring	A.5 Service migration	A.6 Games	P.1 Lower outage/ blocking probability	P.2 Higher accumulated reward	P.3 Higher Quality of Service	P.4 Higher energy efficiency	P.5 Higher fairness	P.6 Lower congestion levels
SADRL applied to distributed agents	Chen et al. (2019) [31]								×				×				×					×	×			
	Vila et al. (2020) [72]									×			×		×								×			
	Shilu et al. (2020) [63]									×	×	×	×						×				×			
	Ruoyun et al. (2020) [73]									×			×						×					×		
	Shing's et al. (2019) [56]					×				×		×			×									×		
	You et al. (2019) [6]					×			×			×	×	×			×						×			
	Zhao et al. (2019) [54]			×	×				×		×		×	×	×								×			
	Luis et al. (2021) [74]					×	×			×			×	×				×				×				
	Chen et al. (2021) [75]		×							×	×	×	×			×										×
SADRL for hierarchical MA environment	Donghan et al. (2020) [76]	×	×			×			×				×	×		×										×
SADRL with AC applied to centralized entities in hierarchical MA environment	Li et al. (2019) [35]								×		×	×	×		×						×	×	×			
	Ishan et al. (2020) [52]								×		×	×	×		×					×						

Table 4. *Cont.*

MADRL Approach	Reference, Year	Objective				Characteristic					Challenge			Applications				Performance Measure			
SADRL with AC applied to distributed entities in hierarchical MA environment	Chen et al. (2019) [77]				×				×		×	×		×					×		
	Xu et al. (2020) [70]		×						×		×	×		×					×		
	Zou et al. (2019) [78]										×	×		×					×		
MADRL with AC applied to distributed entities	Chu's et al. (2019) [79]	×	×			×	×			×		×	×		×						×
	Hurmat et al. (2020) [80]								×			×		×					×		
	Li et al. (2021) [81]							×		×		×			×						×
MADRL using target updated with neighboring agents' states and actions	Wu et al. (2020) [36]	×	×			×	×			×		×	×		×						×
	Qingyong et al. (2020) [82]	×	×				×			×		×		×						×	
MADRL using target updated with neighboring agents' Q-values	Ge et al. (2019) [33]	×	×			×	×			×		×	×		×						×
Bootstrapping MADRL	Tian et al. (2020) [83]	×	×				×			×		×			×						×
MADRL with shared reward	Baldazo et al. (2019) [61]	×	×					×		×			×			×		×			
MADRL with action discovery strategy	Lei et al. (2019) [55]	×	×							×		×		×						×	
Distributed MADRL with hierarchical characteristics	Yu et al. (2020) [84]	×		×					×		×			×						×	
MADRL with concurrent learning	Elhadji et al. (2017) [85]	×								×		×					×	×			
	Zhang et al. (2020) [86]	×	×					×		×		×		×						×	

Table 5. Summary of MADRL extensions and their advantages.

MADRL Approaches	**Description**	**Advantages**
Actor–critic	• Agent has unique actor–critic networks, in which: (a) the actor network predicts Q-values; and (b) the critic network provides a stable target for learning • Suitable for environments with centralized entity and distributed agents, in which: (a) the centralized entity gathers experiences from distributed agents, stores them in a replay memory, learns, and distributes the weights of the actor networks to distributed agents	• Overcomes the shortcomings of separate centralized and distributed entities • Achieving higher scalability (O.5) and stability (O.1), and a lower signaling overhead (O.4) and less complexity (O.3)
MADRL with knowledge exchange	• Distributed agents exchange knowledge with neighboring agents (X.1.1), estimate Q-values, and select their own actions	• Provides a partial observation of the operating environment since the agents collaborate • Achieving a higher learning speed (O.2) and stability (O.1)
Bootstrapped MADRL	• Distributed agents perform multiple actions and learn at the end of an episode rather than right after taking an action • Suitable for distributed agents to share the critic network for learning and use the main network for action selection	• Achieving a higher learning speed (O.2)
MADRL with action discovery	• Distributed agents explore, evaluate, and store explored actions for training • Distributed agents use a cost function to explore more actions when seeking optimal actions, and use potential functions to replace inappropriate actions with newly explored ones	• Achieving a higher learning speed (O.2)
MADRL with concurrent learning	• Distributed agents with different decision-making capabilities select their own actions to achieve a common global goal, in which they receive joint states and rewards	• Achieving a higher stability (O.1)

6.1.4. Ruoyun's SADRL Approach in a Multi-Agent Environment

In [73], Ruoyun et al. embedded SADRL in a service function chain (SFC), which is a chain of connected network services, to migrate virtual machines (VMs) to a data center based on the energy consumption and delay requirements of SFC users. The data center possesses virtual network functions (VNFs), including virtualized routers, firewalls, and network address translation services (A.5). The SFCs maximize the global reward that directs an SFC to a common target in a collaborative (X.2.2) manner. The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the traffic.

Each SFC i represents state s_t^i with a two-tuple information, including the amount of resources required from VNF and the amount of VNF resource remaining. The action a_t^i represents whether or not to migrate VMs to VNF, and if yes, which VNF to migrate to.

The reward (or cost) $r^i_{t+1}(s^i_{t+1})$ represents the delay incurred by the nodes. The traditional SADRL approach has been shown to increase energy efficiency (P.4).

6.1.5. Shing's SADRL Approach in a Multi-Agent Environment

In [56], Shing's et al. embedded SADRL in controllers to serve resource-limited distributed entities (e.g., home appliances, sensors, and smart sockets). There are two main mechanisms. Firstly, the controllers gather data from some distributed entities (i.e., based on the activities of home appliances), rather than all distributed entities, to reduce signaling overheads (O.4) and resource consumption of the resource-limited distributed entities. This helps to increase scalability (O.5). Secondly, the controllers perform adaptive learning since they receive updates (i.e., rewards) over a long time period (i.e., after one day from action selection and execution), and subsequently switch on/off home appliances based on their activities to improve the comfort of user experience in home area networks (A.1). The controllers maximize the global reward to ensure a comfortable level in a collaborative (X.2.2) manner. The proposed approach addresses the challenge of ultra-densification (C.1) caused by a large number of sensors, including smart sockets [64], motion sensors, temperature sensors, and door sensors.

Each controller i represents state s^i_t with a four-tuple information, including: (a) the high-conflict state, whereby switching an appliance on or off can cause a major conflict to user experience; (b) the possible conflict state can cause a moderate conflict to user experience; (c) the low-conflict state can cause a minor conflict to user experience; and (d) the battery state, including the residual energy and the energy consumption in the past 24 h. The action a^i_t represents: (a) whether or not to switch on a home appliance when its state is either high-conflict or possible conflict; (b) when to switch on or off a home appliance when its state is low-conflict; and (c) whether to charge or discharge the battery based on its battery state. The reward $r^i_{t+1}(s^i_{t+1})$ represents: (a) the idle time when the home appliance is switched off in the high-conflict and possible conflict states; and (b) the energy cost, the peak value of energy consumption, and the peak-to-average ratio of energy consumption based on the battery state when the home appliance is switched on in the low-conflict state. The traditional SADRL approach has been shown to increase energy efficiency (P.4).

6.1.6. You's SADRL Approach in a Multi-Agent Environment

In [6], You et al. embedded SADRL in distributed agents, which are the source and intermediate routers, to find the optimal route to the destination router independently based on routing metrics (i.e., the number of hops and reliability of a route) (A.3). The agents maximize their individual rewards in a competitive (X.2.1) manner. The key feature is that distributed agents observe local states and learn from historical knowledge (e.g., previous actions), rather than exchanging knowledge with neighboring agents, which reduces the signaling overhead (O.4), and hence increasing scalability (O.5). The proposed approach addresses the challenges of: (a) ultra-densification (C.1) caused by a large number of node pairs; (b) high dynamicity (C.2) due to the dynamic condition of the operating channels (i.e., the changes of the traffic patterns); and (c) high dimension (C.3) with large state and action spaces.

At each transmission opportunity, the source router i represents state s^i_t with a two-tuple information: (a) the destination router of the current packet and the next packet; and (b) the historical actions (i.e., the selected action executed right before the current action) of the source router i. The action a^i_t represents a selected neighboring (or next-hop) router j. The reward $r^i_{t+1}(s^i_{t+1})$ represents the total queuing time at router i and the transmission time to the selected next-hop router. Each agent i find the optimal route to minimize the average packet delivery time. The traditional SADRL approach has been shown to reduce end-to-end delay (P.3).

6.1.7. Zhao's SADRL Approach in a Multi-Agent Environment

In [54], Zhao et al. embedded SADRL in user equipment (UEs) to select and associate itself with one of the base stations (A.1) that provides the best possible transmission opportunities (i.e., operating channels and time slots) for data transmission based on channel quality (i.e., the signal-to-interference-plus-noise ratio (SINR) levels). The UEs measure and report the SINR level of their selected operating channels to centralized entities (X.3.1), which are the associated base stations. The associated base station gathers data from neighboring distributed agents and distributes it to UEs. This enables UEs to learn and select actions based on network-wide data. The UEs maximize their individual rewards in a competitive (X.2.1) manner. The proposed approach addresses the challenges of high dynamicity (C.2) due to the dynamic condition of the operating channels, and large state spaces (C.3) with the large number of UEs in a heterogeneous environment.

For each UE i, the state s_t^i represents whether or not the QoS requirements are met. The action a_t^i represents a transmission opportunity, which includes the base station, operating channel, and time slots. The reward $r_{t+1}^i(s_{t+1}^i)$ includes: (a) utility, which is a positive value covering the selected base station and the transmission power; and (b) an action selection cost, which is a negative value. The reward is applicable whenever the SINR of agent i meets the minimum QoS requirements; otherwise, the reward is a negative value. The traditional SADRL approach has been shown to increase throughput (P.3), and to reduce computational complexity (O.3) and the communication overhead (O.4).

6.1.8. Luis's SADRL Approach in a Multi-Agent Environment

In [74], Luis et al. embedded SADRL in a centralized agent, which is the centralized controller, to coordinate the trajectories of distributed agents, which are autonomous surface vehicles (ASV), based on whether an area has been visited or not within a time period in order to monitor the contamination level of an area of water resource (A.4). The agents maximize the global reward in a collaborative (X.2.2) manner. The ASVs select the next movement and report it to the centralized controller (X.3.1). The centralized controller guides the ASVs by rejecting possible disturbances using the experiences of all ASVs. The key feature is that distributed agents observe local states and learn from common experiences rather than exchanging knowledge with neighboring agents, which reduces the signaling overhead (O.4), and hence increasing scalability (O.5). The proposed approach addresses the challenges of high dimension (C.3) due to the large state–action spaces with a large number of ASVs in the multi-agent environment, and the dynamic condition of the operating channels, and the large state spaces (C.3) with a large number of ASVs in a heterogeneous environment.

For each ASV i, the state s_t^i represents a 3-channel (RGB) image of the area of water resource. The action a_t^i represents the ASV movements, which include perpendicular (i.e., N, S, E, W) and diagonal (i.e., NE, SE, SW, NW) movements. The reward $r_{t+1}^i(s_{t+1}^i)$ represents a positive value when the trajectory of the ASV covers the area of water resource. Higher reward is received by the agent i when it chooses action a_t^i that: (a) visits areas of water resource which have not been visited within a long time period; (b) does not cause collision with other ASVs; and (c) does not move out of the monitoring area.

The proposed approach extends the traditional SADRL approach so that the decoupled and fully connected output layer of the DNN of the centralized agent can represent the action $a_t^i \in A^i$ where $i \in A^N$. Each distributed agent has the same sets of actions and rules (or constraints), and each of its experience $(s_t^i, a_t^i, r_{t+1}^i(s_{t+1}^i), s_{t+1}^i)$ is shared with the centralized agent, contributing to the training process equally. The centralized agent uses batches of experiences shared by all agents during memory reply. The proposed approach has been shown to increase accumulated reward (P.2).

6.1.9. Chen's SADRL Approach in a Multi-Agent Environment

In [75], Chen et al. embedded SADRL in a centralized agent, which is the roadside units (RUs), to control the mobility of distributed agents, which are autonomous vehicles connected to the RUs, based on traffic conditions in order to manage the congestion level in vehicular networks (A.2). The agents maximize their individual rewards in a collaborative (X.2.2) manner. The autonomous vehicle sends local information (e.g., speed, location, lane position, and intention) to the RUs (X.3.1). Each RU gathers data from a set of autonomous vehicles connected to it and neighboring RUs, and then distributes global data back to the vehicles. This enables autonomous vehicles to learn based on local and network-wide data. The key feature is the use of dense rewards, which is received at each time step instead of when a vehicle exits a ramp, contributing to an increased learning speed (O.2). The proposed approach addresses the challenges of ultra-densification (C.1) caused by a large number of autonomous vehicles connected to a RU, and high dynamicity (C.2) due to the dynamic condition of the road traffic pattern.

Each RU i represents state s_t^i with three information: (a) node features containing speed, position, location, and moving direction; (b) the vehicular network topology; and (c) the set of autonomous vehicles connected to the RU. The action a_t^i represents a lane change (i.e., change to the left or right lane, or keep in the current lane). The reward $r_{t+1}^i(s_{t+1}^i)$ represents a positive value when vehicle moves to appropriate lanes. The proposed approach has been shown to reduce congestion levels (P.6).

6.2. Donghan's SADRL Approach for a Hierarchical Multi-Agent Environment

In [76], Donghan et al. embedded SADRL in a centralized agent, which is the centralized controller, to coordinate the traffic phases of distributed agents, which are traffic lights (A.2), based on traffic conditions in order to manage the congestion level. The agents maximize their individual rewards in a competitive (X.2.1) manner. The key features are that: (a) distributed agents use neighboring agents' knowledge from the last time step rather than the current time step, which increases stability (O.1) and learning speed (O.2); and (b) the centralized agent forwards knowledge to distributed agents according to their roles and positions in the multi-agent environment, which ensures higher scalability (O.5). The proposed approach addresses the challenges of high dynamicity (C.2) due to the dynamic condition of the road traffic pattern, and high dimension (C.3) with a large number of state–action pairs characterizing the large numbe of vehicles at an intersection.

At an intersection, the traffic light i, which is a distributed agent, represents state s_t^i with a four-tuple information, including: (a) the cumulative delay of the first vehicle in each incoming lane; (b) the total number of vehicles along each incoming lane within a distance (e.g., 50 m) from an intersection; and (c) the historical Q-values. The action a_t^i represents a selected traffic phase. The reward (or cost) $r_{t+1}^i(s_{t+1}^i)$ represents the queue length of vehicles along each lane.

The proposed approach extends the traditional MADRL approach with two main mechanisms. Firstly, each agent has an enhanced communication-learning mechanism with three main blocks to overcome the shortcomings of partial observations among distributed agents: (a) the information extract block extracts relevant information from high-dimensional raw data, and uses local states and Q-values of the distributed agents in the last time step rather than that in the current time step to increase robustness against delay; (b) the information exchange block has a centralized agent that learns the significance of the information of each distributed agent, and sends the information to the rest of the distributed agents; and (c) the policy learning block predicts the Q-values of the actions of each distributed agent based on the observations and information gathered by the information extract and exchange blocks. In short, this mechanism enables each distributed agent to use relevant information based on its significance. Secondly, an enhanced communication among agents. The hierarchical approach provides three main advantages: (a) provides the abstraction of complex problems; specifically, agent i learns from relevant information rather than high-dimensional raw data; (b) enables agent i to learn over temporal spans using knowledge from the last time step; and (c) provides the segregation of agents into smaller groups based on their capabilities (i.e., roles and positions), which naturally helps to decompose complex problems into smaller ones. The proposed scheme has been shown to reduce the congestion level (P.6).

6.3. SADRL with Actors and Critics Applied to Centralized Entities in a Hierarchical Multi-Agent Environment

DRL can be implemented using actor and critic networks, in which an agent i has: (a) a *critic network* to calculate the temporal difference (TD)-error, which represents the quality of agent i's current action, based on the states and actions of all distributed agents; and (b) a *actor network* to update its Q-values or policy using temporal difference, rather than the immediate reward. The actor–critic approach has been shown to expedite learning.

In Figure 4, the centralized entity i gathers the $(s_t^i, a_t^i, r_{t+1}^i(s_{t+1}^i), s_{t+1}^i)$ experience from distributed agents $N = I$, stores it in a replay memory D^i, learns, and distributes learned knowledge (i.e., the weights of the actors $\boldsymbol{\mu}$) to distributed agents. At each time step, the centralized entity i selects a random set of experiences from the replay memory D and updates four networks as follows:

- *Actor network* $\boldsymbol{\mu} = (\mu_1(s_1|\theta_1^\mu), \ldots, \mu_I(s_I|\theta_I^\mu))$ is parameterized by weight $\boldsymbol{\theta^\mu} = (\theta_1^\mu, \ldots, \theta_I^\mu)$. It updates the weight θ_i^μ of agent i using sampled policy gradient according to $\nabla_{\theta_i^\mu} J_i = \mathbb{E}_{\mathbf{s},\mathbf{a}\sim D}[\nabla_{a_i} Q_i(\mathbf{s},\mathbf{a}|\theta_i^Q)|_{a_i=\mu_i(s_i)}]\nabla_{\theta_i^\mu}\mu_i(s_i|\theta_i^\mu)$ where $J_i = \mathbb{E}[R_i]$ and $\mathbb{E}$ represents environment.
- *Target actor network* $\boldsymbol{\mu'} = (\mu_1'(s_1|\theta_1^{\mu'}), \ldots, \mu_I'(s_I|\theta_I^{\mu'}))$ is a copy of the actor network that updates the weight $\theta_i^{\mu'} \leftarrow \tau\theta_i^\mu + (1-\tau)\theta_i^{\mu'}$ slowly (e.g., with a small $\tau = 0.01$ value) to ensure convergence.
- *Critic network* $\mathbf{Q} = (Q_1(\mathbf{s},\mathbf{a}|\theta_1^Q), \ldots, Q_I(\mathbf{s},\mathbf{a}|\theta_I^Q))$ is parameterized by weight $\boldsymbol{\theta^Q} = (\theta_1^Q, \ldots, \theta_I^Q)$. It updates the weight θ_i^Q of agent i by minimizing its loss $Loss(\theta_i^Q) = \mathbb{E}_{\mathbf{s},\mathbf{a},r_i,\mathbf{s}'}[(Q_i(\mathbf{s},\mathbf{a}|\theta_i^Q) - y_i)^2]$ where $y_i = r_i + \gamma Q_i'(\mathbf{s}', \boldsymbol{\mu'}(\mathbf{s}'|\boldsymbol{\theta^{\mu'}})|\theta_i^{Q'})$.
- *Target critic network* $\mathbf{Q'} = (Q_1'(\mathbf{s},\mathbf{a}|\theta_1^{Q'}), \ldots, Q_I'(\mathbf{s},\mathbf{a}|\theta_I^{Q'}))$ is a copy of the critic network that updates the weight $\theta_i^{Q'} \leftarrow \tau\theta_i^Q + (1-\tau)\theta_i^{Q'}$ slowly.

Algorithm 5 shows the SADRL with actors and critics approach embedded in the centralized entity. The critic uses the $(s_t^i, a_t^i, r_{t+1}^i(s_{t+1}^i), s_{t+1}^i)$ experience from agent i to calculate the temporal difference $\delta_t^i(s_t^i, a_t^i)$ of the state–action pair of agent i. Subsequently, the temporal difference $\delta_t^i(s_t^i, a_t^i)$ is used by the actor to update the Q-value $Q_{t+1}^i(s_t^i, a_t^i)$.

Algorithm 5: Actor-Critic RL algorithm.

```
1: Procedure
2:     /*Critic*/
3:     Observe current state $s_t^i$
4:     Select action $a_t^{i,*}$
5:     Receive immediate reward $r_{t+1}^i(s_{t+1}^i)$
6:     Calculate temporal difference $\delta_t^i(s_t^i, a_t^i)$
7:     /*Actor*/
8:     Update Q-value $Q_{t+1}^i(s_t^i, a_t^i)$
9: End Procedure
```

SADRL with actors and critics can be applied to centralized entities (X.3.1) to receive experiences from distributed entities, and coordinate the operations of both centralized and distributed entities. This approach addresses the shortcomings of separate centralized and distributed entities. For increasing scalability (O.5), the centralized agent: (a) gathers data from some distributed agents, rather than all distributed agents, to reduce the signaling overhead (O.4) [35]; and (b) uses less data for learning to reduce the computational load [35]. For increasing stability (O.1), the centralized agent considers the distributed agents to take actions in a sequential manner [35]. For reducing the signaling overhead (O.4), the distributed agents do not exchange local information among themselves [35]. For reducing complexity (O.3) and increasing learning speed (O.2), the centralized agent performs the complex learning task, rather than the distributed agents [35].

Li's Hybrid SADRL Approach with Actors and Critics

In [35], Li et al. embedded SADRL with actors and critics in the centralized agent, which is the base station shown in Figure 4. The agent maximizes individual rewards in a competitive (X.2.1) manner. The *centralized agent* gathers experiences (including the channel information, such as channel quality and interference levels) from a predefined number of neighboring distributed agents, learns, and distributes learned knowledge (i.e., the weights of the actors) to distributed agents, which are the transmitter and receiver node pairs (or links) (X.3.1). Based on the knowledge given by the centralized agent, the distributed agents select their resources (i.e., the transmission opportunities characterized by the operating channel, the time of transmission, and the transmission power) for D2D communication so that they can communicate with each other directly without passing through the base station while meeting the SINR constraint (A.1). Node pairs use different operating channels to reduce interference, and to increase the reliability and capacity of both cellular communication (i.e., between the node pair and the cellular base station) and D2D communication if they are physically close with each other. The node pairs can reuse the same operating channels to improve spectral efficiency if they are far apart from other node pairs. The proposed approach addresses two challenges: (a) ultra-densification (C.1) caused by a large number of node pairs; and (b) high dynamicity (C.2) due to the dynamic condition of the operating channels.

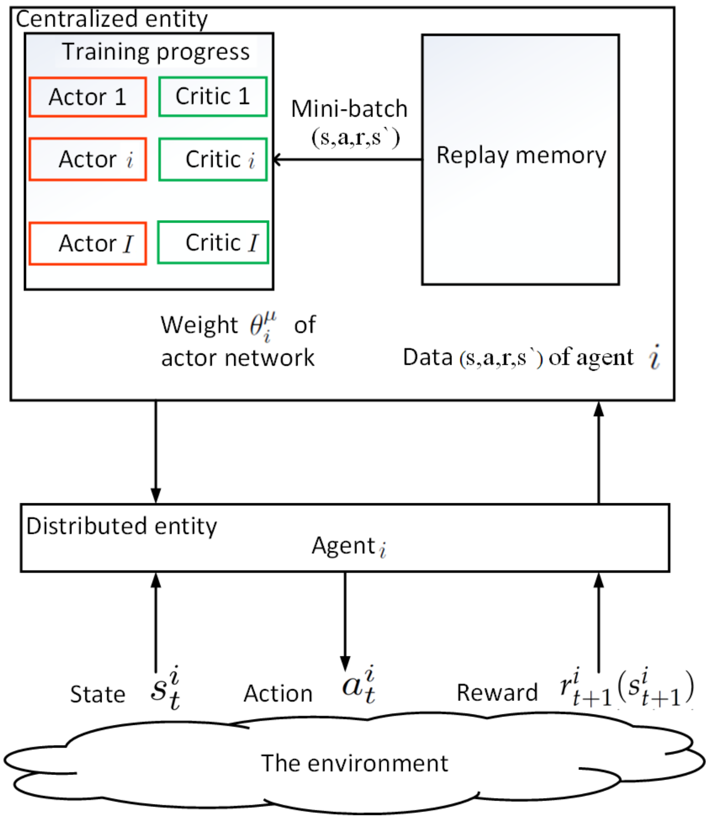

Figure 4. The centralized entity gathers experiences from agents $1 \ldots i \ldots I$, stores them in its replay memory, and performs training. Shown in the figure is one of the distributed agents, namely, agent i, that performs action selection.

Consider a predefined set of all distributed agents N maintained by the centralized agent, and a predefined set of neighboring distributed agents $I \in N$. Each D2D node pair (or link) i represents state s_t^i with a four-tuple information, including: (a) the channel information of the link i; (b) the channel information of the cellular link; (c) the interference level of the D2D link n at the previous time instant $t-1$; and (d) the transmission opportunity of the D2D link n at the previous time instant $t-1$. The action a_t^i represents a transmission opportunity at time t. The reward $r_{t+1}^i(s_{t+1}^i) = \log(1+\xi_t^i)$ represents the link rate of the D2D link i, where ξ_t^i represents the SINR of the receiver of the D2D link i based on the Shannon capacity. The reward is applicable when the SINR of cellular nodes is higher than a predefined threshold; otherwise, the reward is a negative constant value.

Each distributed agent receives the weights of the target actor $\mu_n'(s_n|\theta_n^{\mu'})$ from the centralized entity, observes its state s_t^n, selects action a_n^t independently, and receives reward $r_{t+1}^i(s_{t+1}^i)$. Subsequently, the agent i sends the (s_t^n, a_t^n, $r_{t+1}^n(s_{t+1}^n)$, s_{t+1}^n) experience to the centralized entity for subsequent learning. The SADRL with actor and critic networks approach has been shown to increase accumulated reward (P.2) and throughput (P.3), and reduce outage probability (P.1).

A similar approach has been applied in the literature [52]. In [52], Ishan et al., use this approach so that multiple distributed agents (i.e., node pairs) with different channel conditions can select their resources (i.e., transmission opportunities) for D2D communication (A.1), which helps them to achieve fairness (P.5) among the distributed agents and co-exist in a dynamic environment.

6.4. SADRL with Actors and Critics Applied to Distributed Entities in a Hierarchical Multi-Agent Environment

This section presents the use of SADRL to distributed agents so that they can learn independently without the help of the centralized entity (X.3.2).

6.4.1. Chen's SADRL Approach with Actors and Critics

In [77], Chen et al. embedded SADRL with actors and critics in distributed agents, which are the users, to select their resources (i.e., channels with transmission opportunities) (A.1) based on their own observations without knowing the channel states of other agents and exchanging information with other agents, while reducing the signaling overhead (O.4). The agents maximize their individual rewards by selecting the best possible channels in a competitive (X.2.1) manner. The proposed approach addresses the challenge of ultra-densification (C.1) caused by a large number of node pairs, and high dynamicity (C.2) due to the dynamic condition of the operating channels.

Each user i employs a separate set of actor and critic networks. The user i represents state s_t^i with the availability of its own channels. The action a_t^i represents the selected channel. The reward $r_{t+1}^i(s_{t+1}^i)$ is based on the SINR of the selected channel and whether a collision has occurred or not. The SADRL with actor and critic networks approach has been shown to increase throughput (P.3).

A similar approach has been applied in the literature [70]. In [70], Xu et al. embedded this approach in distributed agents (i.e., users) to select their resources (i.e., operating channels) (A.1) based on the the successful and unsuccessful (e.g., collision) transmissions, while addressing the challenges of ultra-densification (C.1) caused by a large number of users and high dynamicity (C.2), in order to increase throughput (P.3). Distributed agents store important (or prioritized) observations, which have less noise in the channel as estimated by a channel estimator, in the replay memory to increase the learning speed (O.2). Long Short Term Memory (LSTM) is used to predict the next state based on historical information (i.e., historical states and actions).

6.4.2. Zou's SADRL Approach with Actors and Critics

In [78], Zou et al. embedded this approach in distributed agents (i.e., D2D links) that select their resources (i.e., channels with transmission opportunities) (A.1) based on channel quality and interference levels in order to meet the co-channel interference constraint. The proposed approach addresses the challenges of ultra-densification (C.1) caused by a large number of UEs, and high dynamicity (C.2) due to the dynamic condition of the operating channels.

Each D2D link i represents state s_t^i with historical knowledge, particularly selected actions in the past. The action a_t^i represents the selected time frames for transmission. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the number of sub-time frames that can be used by the agent i, and the penalty when a collision occurs. Each D2D link selects action independently without knowing the traffic load condition of the network, and so it is partially observable in nature. The SADRL with actor and critic networks approach has been shown to increase throughput (P.3), while ensuring a fair coexistence with various networks in unlicensed channels.

6.5. MADRL with Actors and Critics Applied to Distributed Entities in a Multi-Agent Environment

This section presents the use of MADRL among distributed agents so that they exchange knowledge (i.e., action, state, and rewards), learn, and perform joint actions to increase scalability (O.5).

6.5.1. Chu's MADRL Approach with Actors and Critics

In [79], Chu's et al. embedded MADRL in distributed agents, which are the traffic lights (A.2). The MADRL approach enables a distributed agent to exchange information (i.e., policy) with neighboring agents. A multi-agent environment is collaborative (X.2.2) in nature. Using the proposed approach, distributed agents: (a) share knowledge with each other (X.1.1) to increase stability (O.1); and (b) scale down information from far-away distributed agents to increase learning speed (O.2) and scalability (O.5). The proposed approach addresses the challenges of high dynamicity (C.2) due to the dynamic condition of the road traffic pattern, and high dimension (C.3) with a large number of state–action pairs characterizing the large number of vehicles at an intersection.

At an intersection, the traffic light i represents state s_t^i with a two-tuple information, including: (a) the cumulative delay of the first vehicle in each incoming lane; and (b) the queue length of vehicles along each lane. The action a_t^i represents a selected traffic phase. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the queue length of vehicles in each lane. Each agent i manages the congestion level by exchanging knowledge (i.e., the last known policy π_{t-1}^j) with neighboring agent $j \in J$, and use actors and critics to find the optimal policy. The latest sampled policy π_{t-1}^j of neighboring agent j and the current state s_t^i are included as the inputs of agent i's DNN. The proposed scheme has been shown to reduce the congestion level (P.6).

A similar approach has been applied in the literature [87,88]. In [87], Yu et al., enable distributed agents with MADRL to coexist and share knowledge with each other in a competitive (X.2.1) and dynamic environment in order to ensure stability (O.1). MADRL addresses the challenges of high dynamicity (C.2) and ultra-densification (C.1) in wireless networks. The distributed agents sense spectrum, and use actors and critics to predict the unoccupied frequency bands by licensed users. The proposed approach has been shown to increase QoS—specifically throughput (P.3). In [88], the actor and critic networks of distributed agents predict idle frequency bands without knowledge exchange.

6.5.2. Hurmat's MADRL Approach with Actors and Critics

In [80], Hurmat et al. embedded MADRL with actors and critics in both centralized agent, which is the data center, and distributed agents, which are VNFs. Both centralized and distributed agents select actions to maximize their individual rewards, and hence resources, particularly transmission opportunities, are allocated in a competitive (X.2.1) manner. The centralized agent selects Internet of thing (IoT) devices to be served, and subsequently the distributed agents allocate resources to the IoT devices (A.1). The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the network.

The data center i updates the system state and performs action a_t^i, which is selecting the IoT devices to be served. Each VNF n represents state s_t^n with a three-tuple information: (a) the capacity of IoT devices; (b) the number of service requests; and (c) the number of unserved service requests in the previous time instant $t-1$. The action a_t^n represents a transmission opportunity and it is part of the joint action a_t decided by the centralized data center i and the distributed VNF n. The reward $r_{t+1}^n(s_{t+1}^n)$ of a distributed VNF n represents throughput. The proposed scheme has been shown to increase the overall throughput (P.3).

6.5.3. Li's MADRL Approach with Actors and Critics

In [81], Li et al. embedded MADRL with actors and critics in distributed agents, which are the control units at road intersections in vehicular networks (A.2). The MADRL approach enables distributed agents to share information among themselves (X.1.2), estimate their policies, and use the estimation to select their own actions. Distributed agents select actions that maximize individual rewards in a collaborative manner (i.e., (X.2.2)). The MADRL approach has two main features: (a) enabling collaboration with neighboring distributed agents to provide a partial observation of the operating environment; and (b) addressing the curse of dimensionality due to the complexity of the joint actions of multiple neighboring agents. The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the road traffic pattern.

The control unit i represent state s_t^i with a two-tuple information, including: (a) the current traffic phase; and (b) the queue length of vehicles in each lane. The action a_t^i represents a control signal that extends the current traffic phase or switch to another traffic phase. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the reduction of the average delay of vehicles at all incoming lanes associated with the control units.

The proposed approach extends the traditional MADRL approach with enhanced knowledge sharing mechanisms. Firstly, each agent stores and updates its own observation at a centralized knowledge container. Secondly, each agent accesses the centralized knowledge container to obtain knowledge prior to action selection. During action selection, agents take into account their own observations and centralized knowledge. Since each agent interprets the collective knowledge differently, the centralized knowledge container can be reconstructed based on historical knowledge of all agents. In the proposed model, a single shared critic network learns using observations in the centralized knowledge container during training, and distributed actors make decisions during execution. The proposed approach has been shown to reduce the congestion level (P.6).

6.6. Wu's MADRL Approach Using Target Updated with Neighboring Agents' States and Actions Information

In [36], Wu et al. embedded MADRL in distributed agents, which are the traffic lights (A.2). The MADRL approach enables a distributed agent to exchange state and action information with neighboring agents (X.1.1), estimate their policies, and use the estimation to select its own actions. This enables a distributed agent to perform learning that mimic the centralized learning approach, and select action based on local states. A multi-agent environment is collaborative (X.2.2) in nature. The MADRL approach has two main features: (a) enabling collaboration with neighboring distributed agents to provide a partial observation of the operating environment; and (b) addressing the curse of dimensionality due to the complexity of the joint actions of multiple neighboring agents. Using the proposed approach, distributed agents: (a) share knowledge with each other to increase the learning speed (O.2) and stability (O.1); and (b) learn in a distributed manner to increase scalability (O.5). The proposed approach addresses the challenges of high dynamicity (C.2) due to the dynamic condition of the road traffic pattern, and high dimension (C.3) with a large number of state–action pairs characterizing the large number vehicles at an intersection.

At an intersection, the traffic light i represents state s_t^i with a five-tuple information, including: (a) the current traffic phase; (b) the queue length of vehicles in each lane; (c) the location of the vehicles at each lane; (d) the velocity of the vehicles; and (e) the number of pedestrians waiting to cross the intersection. The action a_t^i represents a selected traffic phase. The reward $r_{t+1}^i(s_{t+1}^i)$ is a weighted value based on: (a) the sum of the queue lengths of vehicles in all its lanes; (b) the total waiting time of vehicles in all its lanes; (c) the system delay; (d) the total number of vehicles crossing the intersection at time t; (e) the changes of the blinking condition, which is an intermediate signal (e.g., a flashing red signal when the signal changes from red to green) in the current traffic phase; and (f) the total waiting time of pedestrians waiting to cross the intersection.

Figure 5 shows the proposed MADRL architecture. Both main and target networks are based on LSTM to use historical states and actions during training and action selection. The target network of an agent i receives state and action information from neighboring agents, and use this information to estimate its own Q-value $Q_t^{i,j}(s_t^{i,j}, a_t^{i,j}; \theta^{i,-})$, which is used to generate the target y_j^i, subsequently used to minimize the local loss function $L^i(\theta^i)$ and update the main network. In other words, each agent i uses global states and actions in its target network to estimate neighboring agents' policies. During action selection, each agent i selects actions based on local states and its own policy using the main network in an independent manner. Specifically, the proposed approach undergoes centralized learning and distributed execution. The proposed scheme has been shown to reduce the congestion level (P.6).

A similar approach has been applied in [82]. In [82], Qingyong et al. embedded this approach in distributed agents (i.e., base stations) that allocate resources (i.e., transmission opportunities) (A.1) to mobile users based on the SINR level and the capacity of the base station (i.e., the number of mobile users served by the base station), while addressing the challenge of high dynamicity (C.2) due to the dynamic channel conditions, in order to increase energy efficiency (P.4). Distributed agents share experience (i.e., states and actions) with neighboring distributed agents (X.1.1) to ensure stability (O.1) and increase the learning speed (O.2).

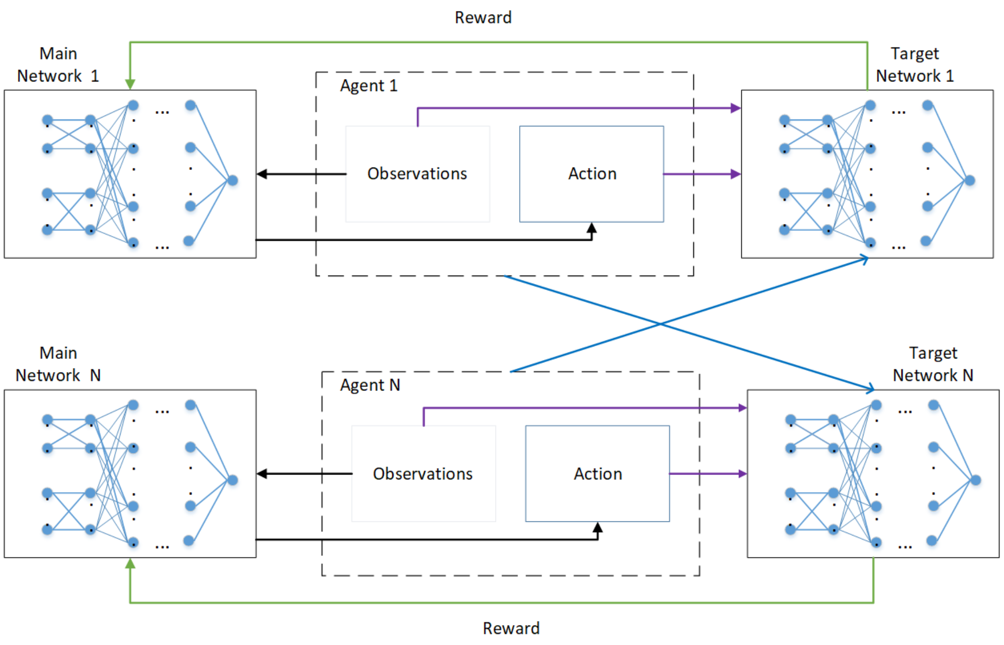

Figure 5. Architecture of the coordinating MADRL model.

6.7. Ge's MADRL Approach Using Target Updated with Neighboring Agents' Q-Values

Similarly to [36], in [33], Ge et al. embedded MADRL in distributed agents, which are the traffic lights (A.2), to exchange optimal Q-values (or knowledge) with neighboring agents (X.1.1). Figure 6 shows the abstract model of the proposed MADRL approach. The proposed approach enables agents select action collaboratively by taking into account the optimal Q-value from neighboring agents J (X.1.1), which is subsequently used to calculate the loss function of the Q-network of agent i as follows:

$$L_j^i(\theta^i) = \frac{1}{m}\sum_{t=1}^{m}\{r_t^i - \gamma[\max_{a\in A} Q_t^i(s_{t+1}^i, a; \theta^{i,-}) + \sum_{j\in N} \eta^{i,j} Q_{t-1}^j(s_{t-1}^j, a_{t-1}^j; \theta^j)] - Q_t^i(s_t^i, a_t^i; \theta^i)\}^2 \tag{7}$$

where m is the batch size, and $\max_{a\in A} Q_t^i(s_{t+1}^i, a; \theta^{i,-})$ is the optimal target Q-value for all actions under the state s_{t+1}^i. Using this loss function enables agent i to select actions based on its own and neighboring agents' Q-values.

At an intersection, the traffic light i represents state s_t^i with a two-tuple information, including the position and speed of the vehicles in each lane entering the intersection. The action a_t^i represents a selected traffic phase. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the changes of the average queue length of vehicles at the intersection. The proposed scheme has been shown to reduce the congestion level (P.6).

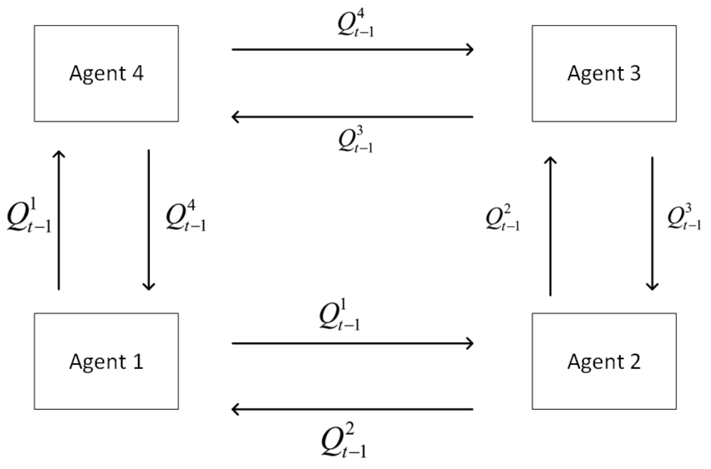

Figure 6. Architecture of cooperative MADQN.

6.8. Tian's MADRL Approach with Bootstrapping in a Multi-Agent Environment

Similarly to [33,36,76,79], in [83], Tian et al. embedded MADRL in distributed agents, which are the traffic lights (A.2), to enhance their exploration strategy. A multi-agent environment is collaborative (X.2.2) in nature. The two key features are that: (a) distributed agents use bootstrapping, whereby the value function $v_\pi^i(s_t^i)$ is sampled per episode to increase learning speed (O.2); and (b) distributed agents share knowledge (i.e., rewards) with neighboring agents (X.1.1) to ensure stability (O.1). The proposed approach addresses the challenge of high dynamicity (C.2) due to the dynamic condition of the road traffic pattern.

The proposed approach improves exploration by identifying uncertainties in Q-value estimation using bootstrapping. Bootstrapping enables an agent i to perform multiple actions and learn at the end of an episode rather than to learn right after taking an action. Figure 7 shows the bootstrapped DQN architecture that consists of $k > 10$ heads sharing a convolutional network. Bootstrapped DQN uses a masking distribution M over possible values to generate a mask m_t^k, which is a binary vector of length k used to decide whether or not the Q-value of the k^{th} head and an experience should be updated during training at time step t. Each head k, which is a separate DQN trained based on different data slices, bootstraps Q-value functions for different actions in parallel. Each head k uses a DQN, whereby the Q-value function of the main network $Q_t^{i,k}(s_t^i, a_t^i; \theta^i)$ is trained against its own target network $Q_t^{i,k}(s_t^i, a_t^i; \theta^{i,-})$. This allows the agent i, which is a traffic light, to learn the posterior distribution over different Q-value functions $Q_t^{i,k}(s_t^i, a_t^i; \theta^i)$ to minimize the loss function without replay memory data shared among the heads. During action selection, in each episode, the agent i observes state s_t^i, samples a single optimal Q-value $Q_t^{i,k}(s_t^i, a_t^i; \theta^i)$ per episode of interactions, takes the optimal actions a_t^i according to the sampled Q-value function throughout the episode, receives reward $r^i_{t+1}(s^i_{t+1})$, and then uses the m^k_t mask to select whether or not to store the $(s^i_t, a^i_t, r^i_{t+1}(s^i_{t+1}), m^k_t)$ experience in a replay buffer B^k (i.e., a distinct memory buffer for the k-head). The experience $(s^i_t, a^i_t, r^i_{t+1}(s^i_{t+1}), m^k_t)$ is used to train the k-th head if it is activated with $m^k_t[k-1] = 1$.

At an intersection, the traffic light i represents state s^i_t with the waiting time of all vehicles in all incoming lanes. The action a^i_t represents a selected traffic phase. The reward $r^i_{t+1}(s^i_{t+1})$ represents the number of vehicles waiting at lanes with red signals. The proposed scheme has been shown to reduce the congestion level (P.6).

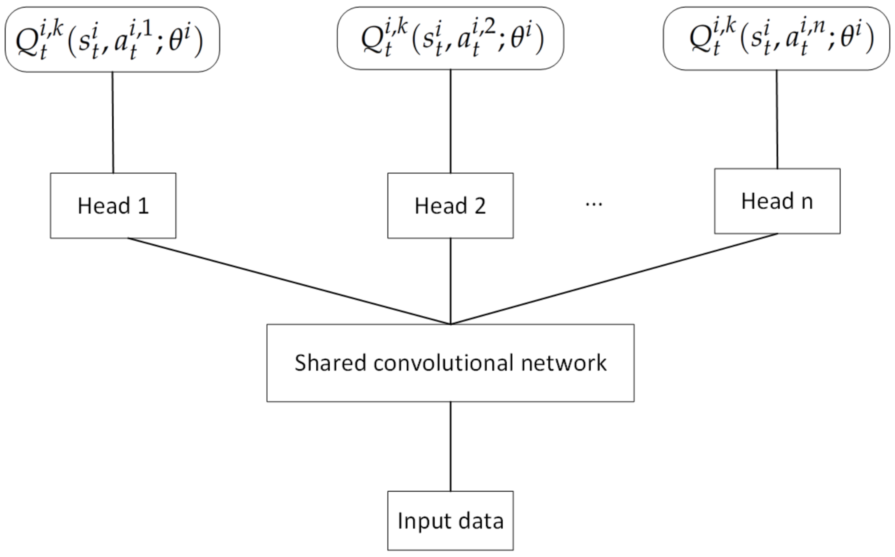
$Q^{i,k}_t(s^i_t, a^{i,1}_t; \theta^i)$
$Q^{i,k}_t(s^i_t, a^{i,2}_t; \theta^i)$
$Q^{i,k}_t(s^i_t, a^{i,n}_t; \theta^i)$
Head 1
Head 2
...
Head n
Shared convolutional network
Input data

Figure 7. Architecture of the bootstrapped MADQN approach.

6.9. Baldazo's MADRL Approach with Shared Reward

In [61], Baldazo et al. embedded MADRL in distributed agents, which are drones, to monitor floods (A.4) and plan relief works. The MADRL approach enables distributed agents to exchange states with other distributed agents (X.1.2). A multi-agent environment is collaborative in nature (X.2.2). Using the proposed approach, distributed agents: (a) share knowledge (i.e., states) with each other to increase the learning speed (O.2); and (b) share discounted reward, which is the mean of the independent rewards to ensure stability (O.1). The proposed approach addresses the challenges of high dimension (C.3) with a large number of state–action pairs related to the agents and the environment (i.e., the terrain).

Drones take images of surrounding areas and use DRL to detect whether or not a flood has occured. Each drone i is a distributed agent that selects a position in the terrain and adjusts its angle to take images using camera. Each agent i represents state s^i_t with four-tuple information, including: (a) the surface water depth; (b) the position of the drone; (c) the heading angle of the drone; and (d) the angle of the drone. The action a^i_t represents the selected angle. Each agent i receives an independent reward $r^i_{t+1}(s^i_{t+1})$ representing the sum of: (a) the distance from the flood; (b) the dry terrains nearby; (c) the use of high angles; and (d) the distance to other agents. All agents share the same reward consisting of the mean of the independent rewards. The proposed approach enables each agent i to collect knowledge (i.e., state information of its own and all other agents (X.1.2)) and use MADRL to optimize the strategy for trajectory planning by finding the optimal policy. The proposed scheme has been shown to increase accumulated reward (P.2).

6.10. Lei's MADRL with Action Discovery Strategy

In [55], Lei et al. embedded MADRL in distributed agents which are controllers in power grids (A.1). The MADRL approach enables distributed agents to explore, evaluate, and store new actions, and store the explored new actions as experiences for training purpose. This enables distributed agents to update their actions set by replacing inappropriate actions with more appropriate ones from time to time, and learn under random and stochastic disturbances. The controllers maximize the global reward in a collaborative (X.2.2) manner. Using this approach, the newly explored actions, which are more appropriate, reduces the overall need for exploration, and so it increase learning speed (O.2). The proposed approach addresses the challenge of high dynamicity (C.2) with the dynamic conditions of the operating environment, whereby the equilibrium (or the steady state) changes from one to another based on power supply and demand.

The agent seeks an optimal action set by exploring new actions using: (a) the cost function $c(s_t^i, s_{t+1}^i)$ that measures the transition cost from states s_t^i to s_{t+1}^i in which the action a_t^i represents the carrier of the state transition; and (b) the potential function $\varphi(s)$ evaluates the state brought about by the newly explored action for ensuring its suitability in improving the stability of the multi-agent environment (O.1)—in other words, whether the newly discovered action is worth exploring in the future. The agent i chooses an action a_t^i with a bounded cost function $c(s_t^i, s_{t+1}^i) \leq \infty$ and ignores other actions. Subsequently, the potential function $\varphi(s)$ is used to determine whether or not the newly explored action is worth exploring in the future.

Each controller i represents state s_t^i with area control error, which is the difference between scheduled and actual energy level dispatched in a control area of the power grid. The action a_t^i represents a selected power adjustment signal. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the stable operation of the overall controlled power grid system. The proposed scheme has been shown to increase energy efficiency (P.4).

6.11. Yu's Fully Distributed MADRL with Hierarchical Characteristics

In [84], Yu et al. embedded MADRL in distributed agents, which are drones, to collect data from sensor nodes within fixed time periods based on the residual energy of drones (or unmanned aerial vehicles, UAVs) (A.1). The proposed approach extends the traditional MADRL approach using options in an extended MDP (S, A, O, R, P). Due to the complexity of the action space of the drones, each drone selects an option (or a task), which represents a sequence of actions based on the option's policy for completing a task. This allows agents to learn at the general level (or the task level) rather than the specific level (or the action level). A multi-agent environment is competitive in nature (X.2.1) due to the goal of maximizing the individual rewards. Using the proposed approach, distributed agents: (a) decompose the complex problem into several sub-problems using hierarchical MADRL; (b) use options (i.e., a generalization of actions) to incorporate sequences of actions in order to ensure stability (O.1); and (c) have prior knowledge about the environment, which reduces complexity (O.3). The proposed approach addresses the challenge of ultra-densification (C.1) caused by a large number of sensor nodes in the multi-agent environment. The proposed model uses a fully distributed MADRL approach with options to obtain an optimal trajectory to sensor nodes. Sensor nodes are clustered, and each UAV agent communicates with neighboring agents via a UAV-to-UAV communication mechanism.

As an example, the "opening a door" option consists of: (a) controlling (or twitch) the body muscles; (b) standing up; and (c) moving to the door. As another example, the "going to school" option consists of: (a) controlling the body muscles; (b) getting dressed; and (c) riding the bus. As for the "collecting data within a fixed time period" option, it consists of: (a) flying to sensor nodes and collecting data; (b) flying to a charging station and getting charged for a fixed time period; and (c) flying to the charging station when the mission is completed. Since more than a single action can be taken in an option, the state may change several times between two decision epochs. Specifically, each option in the option space $o_t^i \in O$ has three components: (a) G represents the condition that initiates a task; (b) π represents a deterministic policy for an option o_t^i; and (c) B represents the condition that terminates the task.

At time instant t, an agent i observes state $s_t^i \in G$ and selects an option o_t^i based on the option learning policy μ. Then, the agent performs a sequence of actions in the selected option $a_t^i \in o_t^i$, and receives reward $r_{t+1}^i(s_{t+1}^i)$ when the selected option terminates at B. Each drone i represents state s_t^i with three-tuple information: (a) the amount of data collected by sensor nodes; (b) its position in the terrain; and (c) its residual energy. The action a_t^i is either flying to a certain sensor node, getting charged, or collecting data. The reward $r_{t+1}^i(s_{t+1}^i)$ represents the maximum time that an agent i needs to finish its task (or for an option to terminate). Each agent i is punished when it runs out of battery. The proposed scheme has been shown to increase energy efficiency (P.4).

6.12. Elhadji's MADRL with Concurrent Learning

In [85], Elhadji et al. embedded MADRL in distributed agents, which are computer players in a pong game (A.6). The agents have global goals (i.e., winning as a team) and independent decision-making capabilities influenced by one another's decisions. Each agent has its own DQN, receives a common reward, and attempts to improve the team score via concurrent learning to ensure stability (O.1). Concurrent learning is used to obtain a coordinated policy among distributed agents to address the challenge of policy adaption in a shared environment, in which distributed agents must adapt to others' current policies. The MADRL approach is collaborative in nature (X.2.2). The proposed approach addresses the challenge of high dynamicity (C.2) due to the players' high mobility.

There are two distributed agents in the environment. Each player i represents state s_t^i that contains the screen pixels. The action a_t^i represents the agent's movement (i.e., up, down, or stay at the same place). The reward $r_{t+1}^i(s_{t+1}^i) = 1$ is awarded when the agent wins; otherwise, $r_{t+1}^i(s_{t+1}^i) = -1$. Figure 8 shows two agents with joint states, rewards, and actions that perform concurrent learning. The agents must learn to divide their area of responsibility in the pong game for a collaborative play. One of the agents may select a non-optimal action and loose the ball. Thus, both agents jointly learn to cooperate. The proposed scheme has been shown to increase accumulated rewards (P.2).

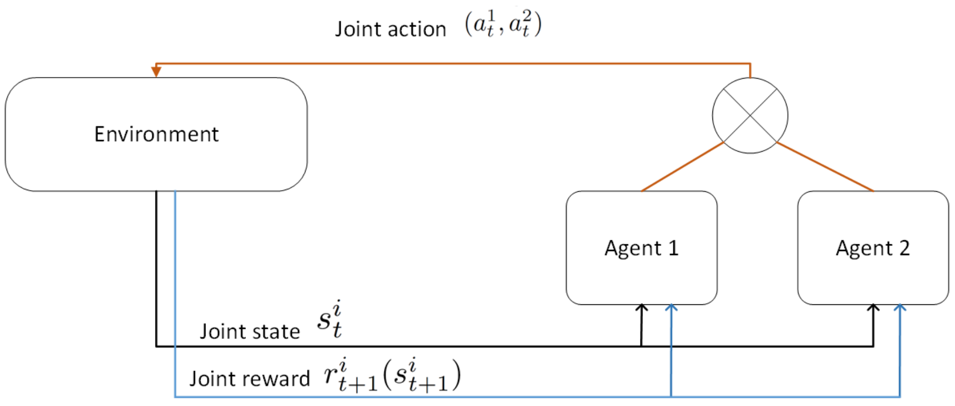
Joint action (a_t^1, a_t^2)
Environment
Agent 1
Agent 2
Joint state s_t^i
Joint reward $r_{t+1}^i(s_{t+1}^i)$

Figure 8. Architecture of MADRL concurrent learning.

A similar application has been applied in [86]. In [86], Zhang et al. embedded MADRL in distributed agents, which are controllers that control the energy consumption of electric generators to distribute energy (A.1) based on the voltage levels of all buses, while addressing the challenge of high dynamicity (C.2) (e.g., time variation of distributed and generated loads) in power grid systems in order to increase energy efficiency (P.4). Distributed agents share knowledge with all agents (X.1.2) to ensure stability (O.1) and increase the learning speed (O.2). The approach performs concurrent learning with centralized training and decentralized execution in a shared environment [7].

MADRL models and algorithms presented in this section can be applied to real-world multi-agent scenarios. Table 4 presents a comparative analysis of the MADRL and SADRL attributes, including applications, of the state-of-the-art approaches presented in Section 6.

For example, In Table 4, the MADRL approach [84] ensures stability (O.1) while reducing complexity (O.3) by enhancing the traditional MADRL approach using options, which generalize the complex set of agents actions in order to address the challenge of ultra-densification (C.1) caused by a large number of agents in the multi-agent environment. The proposed approach is applied to increase energy efficiency (P.4) in resource sharing and allocation (A.1), which have a competitive multi-agent environment (X.2.1). Hence, MADRL and SADRL approaches enhance the traditional MADRL and solve various real-world problems.

7. Open Issues and Future Directions

This section presents open issues that can be pursued in this research area.

7.1. Enhancing Communication between Agents in Large Scale MADRL

Communication plays a critical role in achieving coordination and improving tasks that require synchronization. In large-scale MADRL, the amount of information exchanged between agents is large. Some MADRL algorithms require agents to share actions and observations during training, which performs well in both collaborative (X.2.2) and competitive (X.2.1) environments [89]. However, agents do not generally develop a form of communication based on past experiences, which hinders scalability (O.5) [90]. The frequent information exchange between agents can increase the complexity (O.3) of the MADRL approach. Communication between agents faces the challenges of ultra-densification (C.1) due to the large number of agents in the environment, high dynamicity (C.2) due to the rapid changes in the operating environment, and high dimensionality (C.3) due to the large number of state and action spaces characterizing the operating environment of the large number of agents. Communication can be reduced by enhancing the learning mechanism [76] that normally stores knowledge exchanged with neighboring agents in a memory device [91]. However, a memory device has a limited capacity which is suitable for a small number of agents only. Communication in large-scale MADRL can be enhanced by: (a) Limiting communication between interacting agents. For example, agents know which agents to send their information to, and when to scale down information received from some agents (e.g., far away agents). (b) Enabling agents to remember at least the current and past experience gained from interactions with their operating environment [91]. Enhancing communication enables agents to ensure a higher QoS (P.3) in synchronization and coordination tasks.

7.2. Ensuring a Balanced Trade-off between Complexity and Network-Wide Performance in MADRL

In general, the MADRL applications in the literature consider that agents have similar (or homogeneous) capabilities in terms of resources (e.g., computational abilities) while contributing to the same decision-making process. However, in real-world applications, agents may be heterogeneous with different decision-making capabilities, although agents may have similar goals. As an example, in [62], MADRL is embedded in vehicles with different computational resources, and they are cooperative with a common goal of offloading computationally intensive tasks to base stations (A.5). Agents may also have different goals, such as finding the shortest path in network routing applications (A.3), and they are competitive with each other. In this case, agents with higher computational capabilities are able to find the shortest path faster than those with lower computational capabilities, causing unfairness and lower overall network-wide performance. Moreover, deriving optimal solutions comes at the cost of computational complexity (O.3) [9]. Hence, distributed approaches ensure a lower computational complexity (O.3), and they can achieve a lower network-wide performance due to the heterogeneous capabilities of distributed agents, particularly in competitive (X.2.1) multi-agent environments. While centralized approaches can achieve a fair network-wide performance, it has a high computational complexity (O.3). One possible solution is to use a hybrid approach that consists of centralized and distributed approaches. Hybrid approaches can help to achieve globally and locally optimal network performances, such as achieving a balanced complexity (O.3), while ensuring energy efficiency (P.4).

7.3. Ensuring Convergence in Cooperative MADRL

The changes of an agent's policy during learning, and the agent being part of the operating environment in the presence of other agents, results in a non-stationary environment. In addition, the delay incurred in information exchange between agents in MADRL, which is important to learning, can reduce the convergence rate, causing the agents to take a longer time to select an optimal joint action. While centralized learning approaches [92] improve convergence in cooperative MADRL, they face the challenge of ultra-densification (C.1) due to the large number of cooperating (X.2.2) agents. Parameter sharing [93], which enables agents to share parameters (e.g., weights) for cooperative learning, and so learning may take place in a single network and then the knowledge is used by all agents. Hence, parameter sharing enables multiple agents to share a single learning algorithm; specifically, each agent can train the learning algorithm for a faster convergence and increase the total accumulated reward (P.2) in a cooperative MADRL. Parameter sharing has two main limitations, namely, it: (a) can only be applied to cooperative environments [93]; and (b) can only be applied to a set of homogeneous agents (i.e., the policy of one agent can be traded with other agents without affecting the behavior or the outcome) [92]. To overcome these limitations, information about an agent can be included in the policy (i.e., the policy of an agent can include the agents that it sees) to distinguish each agent for learning an optimal policy. Existing parameter sharing approaches do not enable agents to communicate with each other, instead they share their respective learning networks, and so investigation can be made to enable communication among agents for faster convergence.

7.4. Enhancing Training Performance in MADRL Using Delayed Rewards

In some MADRL environments, agents select actions and receive delayed rewards, whereby the reward of an action is received after it has been taken for some time. The low frequency of rewards can hinder the training performance of MADRL to meet the additional needs of agents in competitive (X.2.1) and collaborative (X.2.2) environments. Moreover, training in MADRL with delayed binary rewards [94] becomes significantly difficult when agents receive binary rewards 1 or 0 only without intermediate rewards in between because: (a) delayed binary rewards are free from traditional reward shaping and are prone to developers' bias and domain knowledge; and (b) agents in MADRL environment with delayed binary rewards may visit state transitions that never produce any rewards, which makes learning more difficult. MADRL with delayed rewards faces the challenge of high dimension (C.3) due to the large state–action spaces. Training performance can be enhanced by: (a) enabling agents to receive rewards at each training step, including dense reward function that produces reward values for majority of transitions, enabling agents to receive rewards in almost every time step, particularly at the early stage of learning [95], for achieving optimal accumulated reward; (b) tailor-made reward functions by experts to assign rewards to behaviors that lead to optimal goal with faster learning speed (O.2); and (c) using credit assignment (or reward shaping [94]) that assigns credits to an action that produces reward to identify the particular action that triggers the reward [92]. Overall, properly designed reward functions ensure a higher convergence speed (O.2) and accumulated reward (P.2).

7.5. Enhancing Learning in Large Scale MADRL

Learning in MADRL is more complex than that in SADRL [28]. Agents in MADRL learn to coordinate with each other, and they share experience (e.g., state, action, reward, and Q-values) with each other to improve their individual or joint actions as time goes by. The presence of multiple learning and interacting agents results in the moving target problem [96], which can cause instability in learning due to a partial observation of the operating environment. Learning in MADRL faces the challenge of ultra-densification (C.1) with a large number of interacting agents required to explore a joint action space. By enabling agents to inform each other's behavior, they can coordinate among themselves using decentralized approaches to reduce complexity (O.3). The decentralized approaches train agents to predict other agents' behavior, while giving only their own local observations [97], which fosters the predictability of agents and promotes coordination. Promoting coordination can also be done by enabling agents to recognize different situations and select the corresponding sub-behaviors, which is also called the sub-policy selection [97,98]. Improving coordination, particularly collaborative tasks, can increase accumulated rewards (P.2) and QoS (P.3).

Enhancing Learning in large scale MADRL can be achieved by investigating six main open issues. Firstly, a light-weight MADRL algorithm can be designed to reduce the high computational power requirement while addressing real-world complex problems. Secondly, an efficient framework with low computational complexity can be designed for MADRL agents. Thirdly, MADRL has been used to solve small-scale problems, and it can be extended to large-scale complex problems with heterogeneous agents. Fourthly, model-based MADRL enables agents to learn optimal policies with a higher learning speed (O.2) based on a model of the operating environment. While it has been applied in the single-agent environment, it can be extended to MADQN in a multi-agent environment. Fifthly, learning in a competitive environment (X.2.1) requires multiple agents to exchange knowledge with neighboring agents to maximize their individual accumulated rewards, although agents can learn from historical actions rather than exchanging knowledge [6]. However, such a partial observation requires agents to remember the current state (or action) and past states (or actions), which requires high computational efficiency. Sixthly, both centralized and distributed agents can collaborate with each other to achieve a balanced trade-off between local and global network performances.

8. Conclusions

In this paper, MADRL models, algorithms, and applications were presented. Examples of the state-of-the-art applications are resource allocation and sharing, traffic signal controllers in vehicular networks, network routing, flood monitoring, and service migration and games. MADRL aims to achieve five main objectives (i.e., higher stability, higher learning speed, lower complexity, lower signaling overhead, and higher scalability), and addresses three main challenges (i.e., ultra-densification, high dynamicity, and high dimension). The MADRL models are attributed to three main characteristics: (a) collaborating entities (i.e., agents either collaborate with neighboring agents or all agents); (b) the relationship between agents (i.e., competitive or collaborative); and (c) the involvement of a centralized entity (i.e., whether the complex task is performed in a centralized or distributed manner). The MADRL models have shown to provide six main performance measures: less outages (or blocking probability), higher accumulated reward, higher quality of service (QoS), higher energy efficiency, higher fairness, and a lower congestion level. Towards the end, this article provided open issues and future directions expected to contribute to MADRL research, and to motivate researchers to investigate this research area.

Author Contributions: Conceptualization, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; methodology, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; investigation, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; resources, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; data curation, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; writing—original draft preparation, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; writing—review and editing, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; visualization, A.M.I., K.-L.A.Y., Y.-W.C. and C.W.; project administration, K.-L.A.Y.; funding acquisition, K.-L.A.Y. and Y.-W.C. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Malaysian Ministry of Education under Fundamental Research Grant Scheme through the "A Novel Clustering Algorithm based on Reinforcement Learning for the Optimization of Global and Local Network Performances in Mobile Networks" project under grant FRGS/1/2019/ICT03/SYUC/01/1. This research was also supported by Publication Fund under Research Creativity and Management Office, Universiti Sains Malaysia.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing not applicable.

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

References

1. Arulkumaran, K.; Deisenroth, M.P.; Brundage, M.; Bharath, A.A. A brief survey of deep reinforcement learning. *arXiv* **2017**, arXiv:1708.05866.
2. Arel, I.; Liu, C.; Urbanik, T.; Kohls, A.G. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intell. Transp. Syst.* **2010**, *4*, 128–135. [CrossRef]
3. Su, S.; Tham, C.K. SensorGrid for real-time traffic management. In Proceedings of the 2007 3rd International Conference on Intelligent Sensors, Sensor Networks and Information, Melbourne, VIC, Australia, 3–6 December 2007; pp. 443–448.
4. Lasheng, Y.; Marin, A.; Fei, H.; Jian, L. Studies on hierarchical reinforcement learning in multi-agent environment. In Proceedings of the 2008 IEEE International Conference on Networking, Sensing and Control, Sanya, China, 6–8 April 2008; pp. 1714–1720.
5. Jang, B.; Kim, M.; Harerimana, G.; Kim, J.W. Q-learning algorithms: A comprehensive classification and applications. *IEEE Access* **2019**, *7*, 133653–133667. [CrossRef]
6. You, X.; Li, X.; Xu, Y.; Feng, H.; Zhao, J.; Yan, H. Toward packet routing with fully-distributed multi-agent deep reinforcement learning. *arXiv* **2019**, arXiv:1905.03494.
7. Nguyen, T.T.; Nguyen, N.D.; Nahavandi, S. Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications. *IEEE Trans. Cybern.* **2020**, *50*, 3826–3839. [CrossRef] [PubMed]
8. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. Is multiagent deep reinforcement learning the answer or the question? A brief survey. *Learning* **2018**, *21*, 22.
9. OroojlooyJadid, A.; Hajinezhad, D. A review of cooperative multi-agent deep reinforcement learning. *arXiv* **2019**, arXiv:1908.03963.
10. Da Silva, F.L.; Warnell, G.; Costa, A.H.R.; Stone, P. Agents teaching agents: A survey on inter-agent transfer learning. *Auton. Agents Multi-Agent Syst.* **2020**, 34, 1–17. [CrossRef]
11. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. *Handb. Reinf. Learn. Control* **2021**, *325*, 321–384.
12. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2021**, 1–49. [CrossRef]
13. Sutton, R.S.; Barto, A.G. *Introduction to Reinforcement Learning*; MIT Press: Cambridge, MA, USA, 1998; Volume 135.
14. Puterman, M.L. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*; John Wiley & Sons: Hoboken, NJ, USA, 2014.
15. Bennett, C.C.; Hauser, K. Artificial intelligence framework for simulating clinical decision-making: A Markov decision process approach. *Artif. Intell. Med.* **2013**, *57*, 9–19. [CrossRef] [PubMed]
16. Watkins, C.J.C.H. *Learning from Delayed Rewards*; King's College: Cambridge, UK, 1989; pp. 1–229. Available online: http://www.cs.rhul.ac.uk/~chrisw/thesis.html (accessed on 30 September 2021).
17. Bowling, M.; Burch, N.; Johanson, M.; Tammelin, O. Heads-up limit hold'em poker is solved. *Science* **2015**, *347*, 145–149. [CrossRef] [PubMed]
18. Tsitsiklis, J.N. Asynchronous stochastic approximation and Q-learning. *Mach. Learn.* **1994**, 16, 185–202. [CrossRef]
19. Even-Dar, E.; Mansour, Y.; Bartlett, P. Learning Rates for Q-learning. *J. Mach. Learn. Res.* **2003**, *5*, 1–25.
20. Tsitsiklis, J.N.; Van Roy, B. An analysis of temporal-difference learning with function approximation. *IEEE Trans. Autom. Control* **1997**, *42*, 674–690. [CrossRef]
21. Hu, J.; Wellman, M.P. Multiagent reinforcement learning: Theoretical framework and an algorithm. *ICML* **1998**, *98*, 242–250.
22. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In Proceedings of the Machine Learning Proceedings, New Brunswick, NJ, USA, 10–13 July 1994; pp. 157–163.
23. Bernstein, D.S.; Givan, R.; Immerman, N.; Zilberstein, S. The complexity of decentralized control of Markov decision processes. *Math. Oper. Res.* **2002**, *27*, 819–840. [CrossRef]
24. Hansen, E.A.; Bernstein, D.S.; Zilberstein, S. Dynamic programming for partially observable stochastic games. *AAAI* **2004**, *4*, 709–715.
25. Laurent, G.J.; Matignon, L.; Fort-Piat, L. The world of independent learners is not Markovian. *Int. J. Knowl.-Based Intell. Eng. Syst.* **2011**, *15*, 55–64. [CrossRef]
26. Guestrin, C.; Lagoudakis, M.; Parr, R. Coordinated reinforcement learning. *ICML* **2002**, *2*, 227–234.
27. Busoniu, L.; De Schutter, B.; Babuska, R. Multiagent Reinforcement Learning with Adaptive State Focus. In *BNAIC* Proceedings of the Seventeenth Belgium-Netherlands Conference on Artificial Intelligence, Brussels, Belgium, 17–18 October 2005; pp. 35–42.
28. Busoniu, L.; Babuska, R.; De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Trans. Syst. Man, Cybern. Part C* **2008**, *38*, 156–172. [CrossRef]

29. Stettner, Ł. Zero-sum Markov games with stopping and impulsive strategies. *Appl. Math. Optim.* **1982**, *9*, 1–24. [CrossRef]
30. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjel, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
31. Chen, X.; Li, B.; Proietti, R.; Zhu, Z.; Yoo, S.B. Multi-agent deep reinforcement learning in cognitive inter-domain networking with multi-broker orchestration. In Proceedings of the Optical Fiber Communication Conference, San Diego, CA, USA, 3–7 March 2019; Optical Society of America: Washington, DC, USA, 2019.
32. Rasheed, F.; Yau, K.L.A.; Noor, R.M.; Wu, C.; Low, Y.C. Deep Reinforcement Learning for Traffic Signal Control: A Review. *IEEE Access* **2020**, *8*, 208016–208044. [CrossRef]
33. Ge, H.; Song, Y.; Wu, C.; Ren, J.; Tan, G. Cooperative deep Q-learning with Q-value transfer for multi-intersection signal control. *IEEE Access* **2019**, *7*, 40797–40809. [CrossRef]
34. Tan, M. Multi-agent reinforcement learning: Independent vs. cooperative agents. In Proceedings of the Tenth International Conference on Machine Learning, Amherst, MA, USA, 27–29 July 1993; pp. 330–337.
35. Li, Z.; Guo, C. Multi-Agent Deep Reinforcement Learning based Spectrum Allocation for D2D Underlay Communications. *IEEE Trans. Veh. Technol.* **2019**, *69*, 1828–1840. [CrossRef]
36. Wu, T.; Zhou, P.; Liu, K.; Yuan, Y.; Wang, X.; Huang, H.; Wu, D.O. Multi-Agent Deep Reinforcement Learning for Urban Traffic Light Control in Vehicular Networks. *IEEE Trans. Veh. Technol.* **2020**, *69*, 8243–8256. [CrossRef]
37. Fan, J.; Wang, Z.; Xie, Y.; Yang, Z. A theoretical analysis of deep Q-learning. In Proceedings of the Learning for Dynamics and Control, Berkeley, CA, USA, 10–11 June 2020; pp. 486–489.
38. Sorokin, I.; Seleznev, A.; Pavlov, M.; Fedorov, A.; Ignateva, A. Deep attention recurrent Q-network. *arXiv* **2015**, arXiv:1512.01693.
39. Bowling, M. Convergence problems of general-sum multiagent reinforcement learning. *ICML* **2000**, 89–94.
40. Littman, M.L. Friend-or-foe Q-learning in general-sum games. *ICML* **2001**, *1*, 322–328.
41. Tesauro, G. Extending Q-learning to general adaptive multi-agent systems. *Adv. Neural Inf. Process. Syst.* **2003**, *16*, 871–878.
42. Kapetanakis, S.; Kudenko, D. Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems. In *Adaptive Agents and Multi-Agent Systems II*; Springer: Berlin/Heidelberg, Germany, 2004; pp. 119–131.
43. Lauer, M.; Riedmiller, M. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, USA, 29 June-2 July 2000.
44. Melo, F.S.; Meyn, S.P.; Ribeiro, M.I. An analysis of reinforcement learning with function approximation. In Proceedings of the 25th International Conference on Machine Learning, Helsinki, Finland, 5–9 July 2008; pp. 664–671.
45. Ernst, D.; Geurts, P.; Wehenkel, L. Tree-based batch mode reinforcement learning. *J. Mach. Learn. Res.* **2005**, *6*, 503–556.
46. Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; Moritz, P. Trust region policy optimization. In Proceedings of the International Conference on Machine Learning, Lille, France, 7–9 July 2015; pp. 1889–1897.
47. Jaderberg, M.; Mnih, V.; Czarnecki, W.M.; Schaul, T.; Leibo, J.Z.; Silver, D.; Kavukcuoglu, K. Reinforcement learning with unsupervised auxiliary tasks. *arXiv* **2016**, arXiv:1611.05397.
48. Van Hasselt, H.; Guez, A.; Silver, D. Deep reinforcement learning with double q-learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 March 2016; Volume 30.
49. Wang, Z.; Schaul, T.; Hessel, M.; Hasselt, H.; Lanctot, M.; Freitas, N. Dueling network architectures for deep reinforcement learning. In International Conference on Machine Learning, New York, NY, USA 20–22 June 2016; pp. 1995–2003.
50. 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, New York, NY, USA, 20–22 June 2016; pp. 1928–1937.
51. Liang, L.; Ye, H.; Li, G.Y. Spectrum sharing in vehicular networks based on multi-agent reinforcement learning. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2282–2292. [CrossRef]
52. Budhiraja, I.; Kumar, N.; Tyagi, S. Deep Reinforcement Learning Based Proportional Fair Scheduling Control Scheme for Underlay D2D Communication. *IEEE Internet Things J.* **2020**, *8*, 3143–3156. [CrossRef]
53. Yang, H.; Alphones, A.; Zhong, W.D.; Chen, C.; Xie, X. Learning-based energy-efficient resource management by heterogeneous RF/VLC for ultra-reliable low-latency industrial IoT networks. *IEEE Trans. Ind. Inform.* **2019**, *16*, 5565–5576. [CrossRef]
54. Zhao, N.; Liang, Y.C.; Niyato, D.; Pei, Y.; Wu, M.; Jiang, Y. Deep reinforcement learning for user association and resource allocation in heterogeneous cellular networks. *IEEE Trans. Wirel. Commun.* **2019**, *18*, 5141–5152. [CrossRef]
55. Xi, L.; Yu, L.; Xu, Y.; Wang, S.; Chen, X. A novel multi-agent DDQN-AD method-based distributed strategy for automatic generation control of integrated energy systems. *IEEE Trans. Sustain. Energy* **2019**, *11*, 2417–2426. [CrossRef]
56. Tai, C.S.; Hong, J.H.; Fu, L.C. A Real-time Demand-side Management System Considering User Behavior Using Deep Q-Learning in Home Area Network. In Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC), Bari, Italy, 6–9 October 2019; pp. 4050–4055.
57. Zhao, L.; Liu, Y.; Al-Dubai, A.; Zomaya, A.Y.; Min, G.; Hawbani, A. A novel generation adversarial network-based vehicle trajectory prediction method for intelligent v ehicular networks. *IEEE Internet Things J.* **2020**, *8*, 2066–2077. [CrossRef]
58. Aljeri, N.; Boukerche, A. Mobility Management in 5G-enabled Vehicular Networks: Models, Protocols, and Classification. *ACM Comput. Surv. (CSUR)* **2020**, *53*, 1–35. [CrossRef]
59. Bui, D.T.; Ngo, P.T.T.; Pham, T.D.; Jaafari, A.; Minh, N.Q.; Hoa, P.V.; Samui, P. A novel hybrid approach based on a swarm intelligence optimized extreme learning machine for flash flood susceptibility mapping. *Catena* **2019**, *179*, 184–196. [CrossRef]

60. Yang, S.N.; Chang, L.C. Regional Inundation Forecasting Using Machine Learning Techniques with the Internet of Things. *Water* **2020**, *12*, 1578. [CrossRef]
61. Baldazo, D.; Parras, J.; Zazo, S. Decentralized Multi-Agent deep reinforcement learning in swarms of drones for flood monitoring. In Proceedings of the 2019 27th European Signal Processing Conference (EUSIPCO), A Coruna, Spain, 2–6 September 2019; pp. 1–5.
62. Yuan, Q.; Li, J.; Zhou, H.; Lin, T.; Luo, G.; Shen, X. A Joint Service Migration and Mobility Optimization Approach for Vehicular Edge Computing. *IEEE Trans. Veh. Technol.* **2020**, *69*, 9041–9052. [CrossRef]
63. Li, S.; Li, B.; Zhao, W. Joint Optimization of Caching and Computation in Multi-Server NOMA-MEC System via Reinforcement Learning. *IEEE Access* **2020**, *8*, 112762–112771. [CrossRef]
64. Kerk, S.G.; Hassan, N.U.; Yuen, C. Smart Distribution Boards (Smart DB), Non-Intrusive Load Monitoring (NILM) for Load Device Appliance Signature Identification and Smart Sockets for Grid Demand Management. *Sensors* **2020**, *20*, 2900. [CrossRef] [PubMed]
65. Khan, M.F.; Yau, K.L.A.; Noor, R.M.; Imran, M.A. Survey and taxonomy of clustering algorithms in 5G. *J. Netw. Comput. Appl.* **2020**, *154*, 102539. [CrossRef]
66. Rasheed, F.; Yau, K.L.A.; Low, Y.C. Deep reinforcement learning for traffic signal control under disturbances: A case study on Sunway city, Malaysia. *Future Gener. Comput. Syst.* **2020**, *109*, 431–445. [CrossRef]
67. Yau, K.L.A.; Qadir, J.; Khoo, H.L.; Ling, M.H.; Komisarczuk, P. A survey on reinforcement learning models and algorithms for traffic signal control. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–38. [CrossRef]
68. Luo, Z.; Chen, Q.; Yu, G. Multi-Agent Reinforcement Learning Based Unlicensed Resource Sharing for LTE-U Networks. In Proceedings of the 2018 IEEE International Conference on Communication Systems (ICCS), Chengdu, China, 19–21 December 2018; pp. 427–432.
69. Jiang, N.; Deng, Y.; Nallanathan, A.; Chambers, J.A. Reinforcement learning for real-time optimization in NB-IoT networks. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1424–1440. [CrossRef]
70. Xu, Y.; Yu, J.; Buehrer, R.M. The application of deep reinforcement learning to distributed spectrum access in dynamic heterogeneous environments with partial observations. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 4494–4506. [CrossRef]
71. Jain, A.; Powers, A.; Johnson, H.J. Robust Automatic Multiple Landmark Detection. In Proceedings of the 2020 IEEE 17th International Symposium on Biomedical Imaging (ISBI), Iowa City, Iowa, 3–7 April 2020; pp. 1178–1182.
72. Vilà, I.; Pérez-Romero, J.; Sallent, O.; Umbert, A. A Novel Approach for Dynamic Capacity Sharing in Multi-tenant Scenarios. In Proceedings of the 2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications, London, UK, 31 August–3 September 2020; pp. 1–6.
73. Chen, R.; Lu, H.; Lu, Y.; Liu, J. MSDF: A Deep Reinforcement Learning Framework for Service Function Chain Migration. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Korea, 25–28 May 2020; pp. 1–6.
74. Luis, S.Y.; Reina, D.G.; Marín, S.L.T. A Multiagent Deep Reinforcement Learning Approach for Path Planning in Autonomous Surface Vehicles: The Ypacaraí Lake Patrolling Case. *IEEE Access* **2021**, *9*, 17084–17099. [CrossRef]
75. Chen, S.; Dong, J.; Ha, P.; Li, Y.; Labi, S. Graph neural network and reinforcement learning for multi-agent cooperative control of connected autonomous vehicles. *Comput.-Aided Civ. Infrastruct. Eng.* **2021**, *36*, 838–857. [CrossRef]
76. Xie, D.; Wang, Z.; Chen, C.; Dong, D. IEDQN: Information Exchange DQN with a Centralized Coordinator for Traffic Signal Control. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
77. Zhong, C.; Lu, Z.; Gursoy, M.C.; Velipasalar, S. A Deep Actor-Critic Reinforcement Learning Framework for Dynamic Multichannel Access. *IEEE Trans. Cogn. Commun. Netw.* **2019**, *5*, 1125–1139. [CrossRef]
78. Zou, Z.; Yin, R.; Chen, X.; Wu, C. Deep Reinforcement Learning for D2D transmission in unlicensed bands. In Proceedings of the 2019 IEEE/CIC International Conference on Communications Workshops in China (ICCC Workshops), Changchun, China, 11–13 August 2019; pp. 42–47.
79. 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]
80. Shah, H.A.; Zhao, L. Multi-Agent Deep Reinforcement Learning Based Virtual Resource Allocation Through Network Function Virtualization in Internet of Things. *IEEE Internet Things J.* **2020**, *8*, 3410–3421. [CrossRef]
81. Li, Z.; Yu, H.; Zhang, G.; Dong, S.; Xu, C.Z. Network-wide traffic signal control optimization using a multi-agent deep reinforcement learning. *Transp. Res. Part C Emerg. Technol.* **2021**, 125, 103059. [CrossRef]
82. Su, Q.; Li, B.; Wang, C.; Qin, C.; Wang, W. A Power Allocation Scheme Based on Deep Reinforcement Learning in HetNets. In Proceedings of the 2020 International Conference on Computing, Networking and Communications (ICNC), Big Island, HI, USA, 17–20 February 2020; pp. 245–250.
83. Tan, T.; Chu, T.; Wang, J. Multi-Agent Bootstrapped Deep Q-Network for Large-Scale Traffic Signal Control. In Proceedings of the 2020 IEEE Conference on Control Technology and Applications (CCTA), Montreal, QC, Canada, 24–26 August 2020; pp. 358–365.
84. Zhang, Y.; Mou, Z.; Gao, F.; Xing, L.; Jiang, J.; Han, Z. Hierarchical Deep Reinforcement Learning for Backscattering Data Collection with Multiple UAVs. *IEEE Internet Things J.* **2020**, *8*, 3786–3800. [CrossRef]

85. Diallo, E.A.O.; Sugiyama, A.; Sugawara, T. Learning to coordinate with deep reinforcement learning in doubles pong game. In Proceedings of the 2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA), Cancun, Mexico, 18–21 December 2017; pp. 14–19.
86. Zhang, Y.; Wang, X.; Wang, J.; Zhang, Y. Deep Reinforcement Learning Based Volt-VAR Optimization in Smart Distribution Systems. *arXiv* **2020**, arXiv:2003.03681.
87. Zhang, Y.; Cai, P.; Pan, C.; Zhang, S. Multi-agent deep reinforcement learning-based cooperative spectrum sensing with upper confidence bound exploration. *IEEE Access* **2019**, *7*, 118898–118906. [CrossRef]
88. Kassab, R.; Destounis, A.; Tsilimantos, D.; Debbah, M. Multi-Agent Deep Stochastic Policy Gradient for Event Based Dynamic Spectrum Access. *arXiv* **2020**, arXiv:2004.02656.
89. Tuyls, K.; Weiss, G. Multiagent learning: Basics, challenges, and prospects. *Ai Mag.* **2012**, *33*, 41. [CrossRef]
90. Jiang, J.; Lu, Z. Learning attentional communication for multi-agent cooperation. *arXiv* **2018**, arXiv:1805.07733.
91. Pesce, E.; Montana, G. Improving coordination in small-scale multi-agent deep reinforcement learning through memory-driven communication. *Mach. Learn.* **2020**, *109*, 1727–1747. [CrossRef]
92. Gupta, J.K.; Egorov, M.; Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In Proceedings of the International Conference on Autonomous Agents and Multiagent Systems, São Paulo, Brazil, 8–12 May 2017; Springer: Cham, Switzerland, 2017; pp. 66–83.
93. Terry, J.K.; Grammel, N.; Hari, A.; Santos, L.; Black, B.; Manocha, D. Parameter sharing is surprisingly useful for multi-agent deep reinforcement learning. *arXiv* **2020**, arXiv:2005.13625.
94. Seo, M.; Vecchietti, L.F.; Lee, S.; Har, D. Rewards prediction-based credit assignment for reinforcement learning with sparse binary rewards. *IEEE Access* **2019**, *7*, 118776–118791. [CrossRef]
95. Gaina, R.D.; Lucas, S.M.; Pérez-Liébana, D. Tackling sparse rewards in real-time games with statistical forward planning methods. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 1691–1698.
96. Shoham, Y.; Powers, R.; Grenager, T. If multi-agent learning is the answer, what is the question? *Artif. Intell.* **2007**, *171*, 365–377. [CrossRef]
97. Roy, J.; Barde, P.; Harvey, F.G.; Nowrouzezahrai, D.; Pal, C. Promoting Coordination through Policy Regularization in Multi-Agent Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1908.02269.
98. De Hauwere, Y.M.; Vrancx, P.; Nowé, A. Learning multi-agent state space representations. In Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems, Toronto, ON, Canada, 10–14 May 2010; Volume 1, pp. 715–722.