drones

MDPI

*Systematic Review*

# Scalable and Cooperative Deep Reinforcement Learning Approaches for Multi-UAV Systems: A Systematic Review

**Francesco Frattolillo** *,† **, Damiano Brunori** *,† **and Luca Iocchi**

Department of Computer, Control and Management Engineering (DIAG), Sapienza University of Rome, 00185 Rome, Italy

* Correspondence: frattolillo@diag.uniroma1.it (F.F); brunori@diag.uniroma1.it (D.B.)

† These authors contributed equally to this work.

**Abstract:** In recent years, the use of multiple unmanned aerial vehicles (UAVs) in various applications has progressively increased thanks to advancements in multi-agent system technology, which enables the accomplishment of complex tasks that require cooperative and coordinated abilities. In this article, multi-UAV applications are grouped into five classes based on their primary task: coverage,adversarial search and game, computational offloading, communication, and target-driven navigation. By employing a systematic review approach, we select the most significant works that use deep reinforcement learning (DRL) techniques for cooperative and scalable multi-UAV systems and discuss their features using extensive and constructive critical reasoning. Finally, we present the most likely and promising research directions by highlighting the limitations of the currently held assumptions and the constraints when dealing with collaborative DRL-based multi-UAV systems. The suggested areas of research can enhance the transfer of knowledge from simulations to real-world environments and can increase the responsiveness and safety of UAV systems.

**Keywords:** unmanned aerial vehicles; multi-UAV; deep reinforcement learning; multi-agent cooperation



**Citation:** Frattolillo, F.; Brunori, D.; Iocchi, L. Scalable and Cooperative Deep Reinforcement Learning Approaches for Multi-UAV Systems: A Systematic Review. *Drones* **2023**, *7*, 236. https://doi.org/10.3390/drones7040236

Academic Editor: Marija Popović and Inkyu Sa

Received: 28 February 2023
Revised: 13 March 2023
Accepted: 14 March 2023
Published: 28 March 2023



## 1. Introduction

Unmanned aerial vehicles (UAVs) have become excellent candidates for addressing various problems due to their high mobility in three-dimensional space, easy deployment, and relatively low production costs. UAVs are utilized in a wide range of applications, from critical humanitarian purposes such as firefighting support [1] and search and rescue (SAR) assistance [2] to purposes aimed at improving the quality of life (QoL) such as parcel delivery (which can decrease delivery costs, see, e.g., [3]), and becoming integrated into industries [4] to monitor the industrial Internet of Things (IoTs) for Industry 4.0. Among the applications of UAVs in communication systems, one example is the use of UAVs to collect data from Internet of Things (IoTs) devices using deep learning (DL) techniques to minimize the Age of Information (AoI) of the data [5]. Another application is to use UAVs to capture views to maximize the fidelity of video reconstruction for remote users [6]. Several applications have been recently developed for scenarios involving surveillance, medical (i.e., blood delivery), agricultural, and high-risk human operations. Many future UAV applications will be based on their cooperative and coordinated interaction, i.e., a fundamental feature of any efficient, robust, and reliable application deployment. Thus, in this article, we decided to examine multi-UAV systems specifically.

The development of deep reinforcement learning (DRL) and its recent successes have made it an excellent candidate for solving complex problems such as those involving multi-UAV applications. The increase in the computational power of modern machines, along with recent advances in deep learning, has led to remarkable results using DRL, such as playing at a superhuman level.In ATARI 2600 games [7], beating the world champion in the *Go* strategy board game [8], defeating the world champion in the complex multiplayer

game DOTA2 [9], and solving real-life control tasks such as solving a Rubik's cube using a robotic hand under external disturbances [10]. A general overview of the research trends associated with the two main keywords, i.e., *reinforcement learning* and *UAV*, is presented in Figure 1. There has been an exponential increase in the number of related publications since 2017.

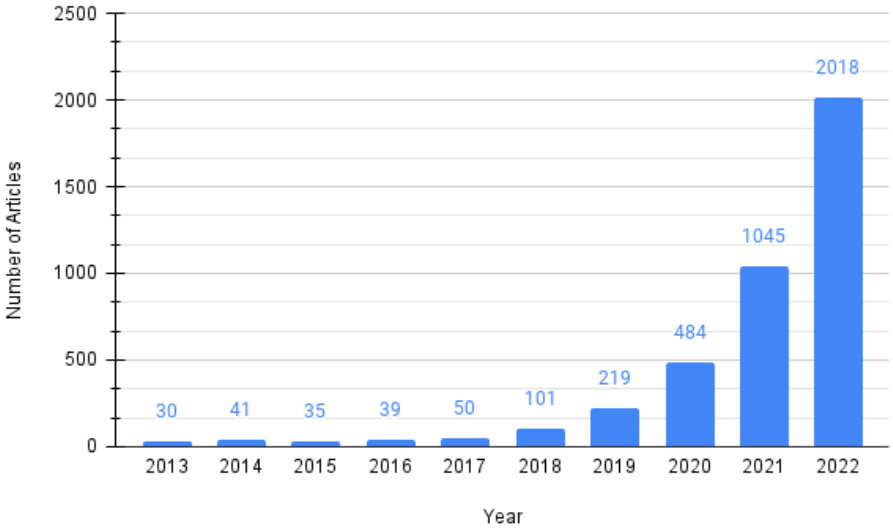



**Figure 1.** Number of works per year retrieved using the keywords "UAV" and "reinforcement learning".

We then decided to investigate a particular subclass of works, i.e., those involving deep reinforcement learning techniques applied to scalable and collaborative multi-UAV systems. To the best of our knowledge, this article is the first to present an extensive and comprehensive comparative analysis of cooperative DRL-based multi-UAV systems. Our research can guide future studies in this field, which is anticipated to attract increasing attention in the coming years.

Several reviews and surveys of UAVs have already been published but with different perspectives than this article. A more general survey of all the machine learning (ML) techniques applied to UAV systems was provided in the study by Bithas et al. [11], which investigated UAV-based communication applications. A similar topic was covered by Aissa and Letaifa [12], where ML techniques were studied to address some of the constraints of UAV wireless communications. Other works either focus on a specific UAV application or do not focus on DRL techniques. More general AI applications (including RL) for UAV path planning problems were investigated in [13], and specific UAV scenarios, e.g., Internet of Things (IoT) assistance [14], were examined by describing the main approaches used (not only (D)RL-based). It is worth mentioning the study of Azar et al. [15], which focused on investigating DRL techniques related to three main UAV macro-problems, namely path planning, navigation, and control, without specifically addressing collaborative multi-UAV systems and their applications. Their study provided a detailed explanation of some DRL algorithms, which can be useful for better understanding the DRL approaches mentioned in Section 4 of this article.

In this review, we highlight the most widely explored research directions in DRL for multi-UAV systems and present other research directions that are either neglected or not fully explored. We also emphasize the most significant features associated with the distribution of real multi-UAV applications. Our review is not aimed at providing implementation details or detailed explanations about the multi-agent deep reinforcement learning (MADRL) algorithms but rather is aimed at critically analyzing, through a comparative study, all the common and dissimilar aspects of the various DRL techniques used when addressing multi-UAV problems. Nevertheless, we will provide some basic RL

(mainly MARL) concepts as necessary and sufficient conditions to facilitate comprehension and readability.

The rest of this article is structured as follows. In Section 2, we introduce the essential background concepts about single and multi-agent reinforcement learning by highlighting some solutions for cooperation and coordination. Section 3 describes the methodology used to select the works related to DRL-based multi-UAV applications. In Section 4, we classify multi-UAV systems into different classes based on their applications and present a comparison of the most relevant features in each class. In Section 5, a comprehensive discussion about the selected works is presented, whereas Section 6 contains a summary of this review, as well as the conclusions and considerations.

## 2. Background

Some basic and main notions will be provided to understand better how a cooperative multi-UAV system can be investigated when applying deep reinforcement learning techniques. Part of the following definitions is adapted from the book by Sutton and Barto [16] (refer to it for additional insights).

### *2.1. Single-Agent Markov Decision Process*

A Markov decision process (MDP) is a mathematical framework modeling sequential decision-making problems. In MDPs, the learner (i.e., the decision maker) is called an *agent*, and everything that interacts with it is called the *environment*: a continuous interaction between the agent and the environment takes place during the learning process. An MDP can be formally defined by the tuple $\langle \mathbb{S}, \mathbb{A}, \mathcal{T}, \mathcal{R} \rangle$ :

- $\mathbb{S}$ is a set of states $\langle s_0, \ldots, s_n \rangle$, with $s_i \in \mathbb{S}$;
- $\mathbb{A}$ is the a set of actions $\langle a_0, \ldots, a_m \rangle$ with $a_i \in \mathbb{A}$ possible in each state $s$;
- $\mathcal{T} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to [0, 1]$ is the probability transition matrix representing the probability of switching from the state $s$ at time $t$ to the state $s'$ at time $t + 1$ by picking the action $a$;
- $\mathcal{R} : \mathbb{S} \times \mathbb{A} \times \mathbb{S} \to \mathbb{R}$ is the reward function that returns the reward obtained by transitioning from state $s$ to $s'$ by picking action $a$.

where $\mathcal{P}$ and $\mathcal{R}$ represent the environmental dynamics and the long-term rewards, respectively; they also define the world model, i.e., the model of the considered environment. The full state space can always be observed in a single-agent problem (i.e., full observability of the MDP).

### *2.2. Multi-Agent Markov Decision Process*

A multi-agent Markov decision process is referred to as a Markov game [17], i.e., a generalization of the standard MDP (used in SARL problems) to the multi-agent scenario. A Markov game is defined by a tuple $\langle \mathcal{P}, \mathbb{S}, \mathbf{\mathbb{A}}, \mathcal{T}, \mathcal{R} \rangle$:

- $\mathcal{P}$ is the number of $n \in \mathcal{N}$ players (agents);
- $\mathbb{S}$ is the set of environmental states shared by all the agents $\langle s_0, ..., s_k \rangle$ with $s_i \in \mathbb{S}$;
- $\mathbf{\mathbb{A}}$ is the set of joint actions $\langle A_1 \times A_2 \cdots \times A_n \rangle$ with $A_i \in \mathbf{\mathbb{A}}$, where $A_i$ is the set of actions of agent $i$;
- $\mathcal{T} : \mathbb{S} \times \mathbf{\mathbb{A}} \times \mathbb{S} \to [0, 1]$ is the probability transition matrix representing the probability of switching from the state $s$ to $s'$ by picking the joint action $\boldsymbol{a}$;
- $\mathcal{R}$ is the set of rewards $\langle \mathcal{R}_1, \mathcal{R}_2, \ldots, \mathcal{R}_N \rangle$, where $\mathcal{R}_i : \mathbb{S} \times \mathbf{\mathbb{A}} \times \mathbb{S} \to \mathbb{R}$ is the reward function related to the agent $i$, which returns the reward obtained by transitioning from the state $s$ to the state $s'$ by taking the action $\boldsymbol{a}$.

In this case, the MDP can be classified into three main groups: (i) fully observable MDP (FOMDP), where the agent can access the full state space; (ii) partially observable MDP (POMDP), in which the agent only knows local state information; (iii) mixed observability MDP (MOMDP), allowing the agent to obtain both the global and local states.

### 2.3. Single-Agent Reinforcement Learning (SARL)

Reinforcement learning (RL) is a machine learning technique where autonomous learners (agents) are supposed to take the best action based on a specific desired goal and by interacting with an environment. In a single-agent reinforcement learning (SARL) system, only one agent acts within the environment by modifying it through its actions. This problem is usually formalized through the single-MDP framework. The learner receives a reward according to the action selected, and its main general goal is choosing actions in such a way as to maximize a cumulative reward over time. One of the main challenges is setting a good trade-off between the *exploration* phase, which pushes the learner to choose different actions in order to observe new possible paths in the operative environment, and the exploitation phase, which pushes the learner instead to continue to select actions that have been already proven successful. When using RL, the goal of the agent is to maximize a numerical reward signal over time:

$$G_t = R_{t+1} + R_{t+2} + R_{t+3} + ... + R_T$$

where $T$ is the final time step. In order to prevent the previous function from blowing up in the long term and thus focusing on the effect of the short-term actions, the previous formula can be rewritten as follows:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} + ... = \sum_{k=0}^{T} \gamma^k R_{t+k+1}$$

where $0 \leq \gamma \leq 1$ is called the *discounted rate*.

An RL agent uses the previous cumulative reward to improve a policy, a state-value function, or a state-action value function. A *policy* $\pi$ represents a mapping function linking a state to an action distribution, and $\pi(a|s)$ defines the probability that the agent will pick the action $a$ in the state $s$.

The *value function* or *state-value function* estimates how good it is for an agent to be in a given state. A value function of a state $s$ under the policy $\pi$ is defined as follows:

$$V_\pi(s) = \mathbb{E}_\pi[G_t|\mathcal{S}_t = s]$$

where $\mathbb{E}_\pi$ is the expected value obtained starting from state $s$ and successively choosing actions using the policy $\pi$

The *Q-function* or *action-value function* instead estimates how good it is to choose an action $a$ in a given state $s$. The action-value function of a state $s$ and an action $a$ under the policy $\pi$ is defined as follows:

$$Q_\pi(s, a) = \mathbb{E}_\pi[G_t|\mathcal{S}_t = s, \mathcal{A}_t = a]$$

where $\mathbb{E}_\pi$ is the expected value obtained starting by selecting the action $a$ in the given state $s$ and then following the policy $\pi$. Reinforcement learning algorithms may be classified in different ways using different criteria, one of which depends on the function type learned by the agent:

- *Policy-based methods.* The agent learns an explicit representation of a policy $\pi$ during the learning process;
- *Value-based methods.* The agent learns a value function to derive an implicit policy $\pi$. The selected actions are the ones that maximize the value function;
- *Actor–critic methods.* The agent learns both a policy function and a value function. These methods are mainly related to deep reinforcement learning and can be considered a mix of the previous ones.

We can further distinguish among the other two main general RL approaches, namely *model-based* and *model-free* solutions. In the former, the model of the environment is already given and partially updated over time during the learning process, while in the latter,

the environment model is unknown (the agent interacts with it through a trial-and-error procedure by trying to learn its model from scratch).

Finally, we report here, below, the main MDP assumptions associated with most SARL problems:

- *Past State Independence.* Future states depend only on the current state;
- *Full Observability.* The MDP is often considered to be a FOMDP;
- *Stationary Environment.* $\mathcal{P}$ and $\mathcal{R}$ are constant over time.

For what concerns *deep reinforcement learning* (DRL) techniques, we can simply and generically describe them as methods combining the usage of neural networks (NNs) as function approximators with RL frameworks in order to allow the agent to learn more generalizable and scalable behavior.

### 2.4. Multi-Agent Reinforcement Learning (MARL)

A multi-agent reinforcement learning (MARL) problem can be modeled through a Markov game framework, where multiple agents act simultaneously by shaping the environment accordingly. In MARL problems, as for the SARL case, the agents try to improve a policy according to a specific desired task. However, the policy is obviously associated with a system of agents. Sometimes, it can also happen that each agent improves only its policy selfishly without taking into account the goal of the whole system, either because it is required for a very specific and limited case or because it is competing against the other agents involved in the considered scenario. Based on the type of training and execution paradigm used, which can be centralized or decentralized, three main schemes for multi-agent reinforcement learning can be identified [18]:

- *Centralized training with centralized execution* (CTCE) provides a single-cooperative system policy but relies on stable communication among all the agents. The need for a centralized unit makes this approach unmanageable when the number of agents becomes large. Standard SARL algorithms may be adapted to work in this setting;
- *Decentralized training with decentralized execution* (DTDE) allows independent agents to perform their policies without neither communication nor cooperation. This paradigm is sometimes referred to as independent learning, and SARL algorithms (just as in the previous case) can be used for any individual agent;
- *Centralized training with decentralized execution* (CTDE), where the agents can access global info during the training time but not at the execution time. The system cooperation is ensured by the centralized training, while the distributed agents execution can be performed without the need for communication, hence providing a more adaptive behavior with respect to the non-stationarity of the environment (see Figure 2 for a schematic visualization);

A fourth *mixed* paradigm can also be considered together with the previous ones. It can be represented by a combination of all those approaches, mixing the already mentioned paradigms and/or combining them with other additional techniques explained in Section 2.5.

Moreover, based on the structure of the reward function, the MARL problem can be defined as follows:

- Fully Cooperative. Each agent receives the same reward at each time step $\mathcal{R}_t^1 = \mathcal{R}_t^2 = ... = \mathcal{R}_t^n$
- Fully Competitive (or zero-sum). The sum of the rewards of all the agents is zero $\mathcal{R}_t^1 + \mathcal{R}_t^2 + ... + \mathcal{R}_t^n = 0$
- Partially Cooperative. A part of the reward is shared among agents, but they could have additional individual rewards, i.e., $\mathcal{R}_{Tot}^i = \mathcal{R}_{sh} + \mathcal{R}_{ind}^i$, where $\mathcal{R}_{sh}$ is the shared part of the reward and $\mathcal{R}_{ind}^i$ represents the individual reward associated with agent $i$;
- Mixed. No specific constraint on the structure of the reward function.

Finally, we want to highlight the most relevant MARL challenges:

- Partial Observability. Action selection takes place under incomplete assumptions;
- Scalability. The state and action spaces increase exponentially with the number of agents;
- Non-stationarity. All the agents learn and change their policies at the same time;
- Credit Assignment. Every agent needs to understand its contribution to the joint system reward.

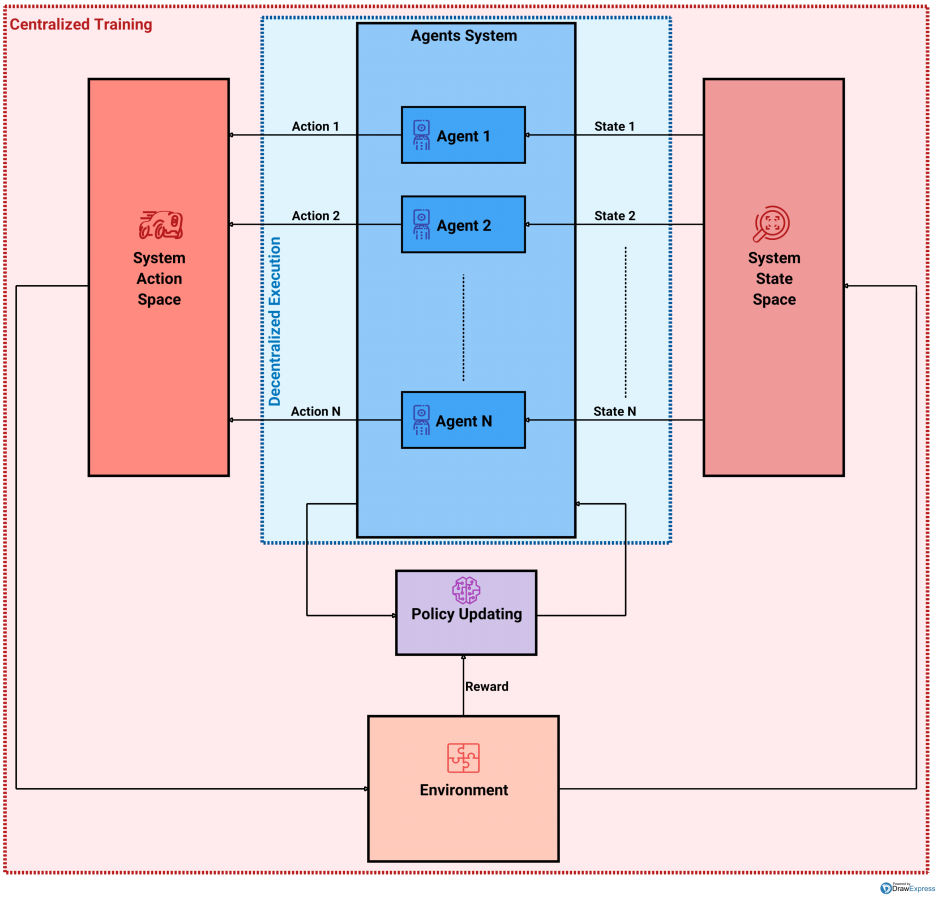



**Figure 2.** Centralized training decentralized execution (CTDE) paradigm scheme (this image was generated through *Draw Express Lite* [19]).

### *2.5. Additional Techniques*

In this article, the reader may come across specific techniques not necessarily based on (D)RL, such as path planning; mean field; attention mechanism; and, more generally, deep learning strategies. Here, we provide a brief definition of these techniques and additional references that may be useful to clarify their mechanism.

Path planning can be applied globally or locally. The associated techniques allow the agents to find a path from a starting location to a target one by avoiding obstacles (if any). The best possible path can be found based on some specific desired goal, such as energy-saving maximization, traveling time minimization, or even exploration tasks. Some of the most known path planning algorithms are represented by algorithms such as $A^*$, $D^*$, *RRT*, $RRT^*$, and *Dijkstra* (see the survey by Karur et al. [20] for an in-depth explanation).

Mean field techniques are based on the mean-field game theory. The idea behind them is to decompose a general and complex problem containing many interacting units in a simpler problem where only local interactions between a small number of units are considered. In RL applications, units are usually referred to as agents.

The attention mechanism [21] is used in deep learning techniques to keep the most relevant features of specific input sequences by discharging the less significant ones: this is performed through a weighted combination of all the input vectors.

Some other techniques can be considered hybrid as they use other deep learning approaches on top of (D)RL methods and/or a combination of the latter.

## 3. Articles Selection Methodology

Here, we describe the methodology used to filter all the studies characterized by the features of our interest.

Initially, we investigated significant conferences (e.g., International Conference on Autonomous Agents and Multiagent Systems AAMAS, the European Conference on Machine Learning ECML, The International Conference on Machine Learning ICML, and Neural Information Processing Systems NeurIPS). However, since conferences do not allow for an indexed and detailed search, we moved to a keyword search on some of the most relevant scientific databases (e.g., IEEE Xplore and ScienceDirect). The review process started in early 2022, and we initially focused our search on collecting articles from 2013 onwards by combining, through boolean operators, the following keywords: *UAV*, *multi-UAV*, *reinforcement learning*, *drone*, *multi-agent*. Different websites offer different advanced search utilities, such as keyword searching in the abstract, in the full text, in metadata, etc. We noticed that searching the keywords in the full text of the document usually led to misleading results, as the topic of some of the papers found was not related to multi-UAV RL-based systems: this happened even though one or more keywords still appeared in the sections associated with the introduction, the reference, or the related works. Searching instead for the presence of the keywords only in the abstract section yielded more significant results.

With all these things considered, we resolved to search only for the keywords *multi-UAV* and *reinforcement learning* in the abstract section of the papers collection related to the year 2022 (the last inspection was on 18 December 2022). Indeed, similar but different keywords such as *Drone* and *UAV* often returned articles on single-agent applications, thus not what we were looking for. Due to the extremely high number of works found in the literature during the paper-collection phase, we decided to apply the following additional "filters" in order to select the papers in which we were interested specifically:

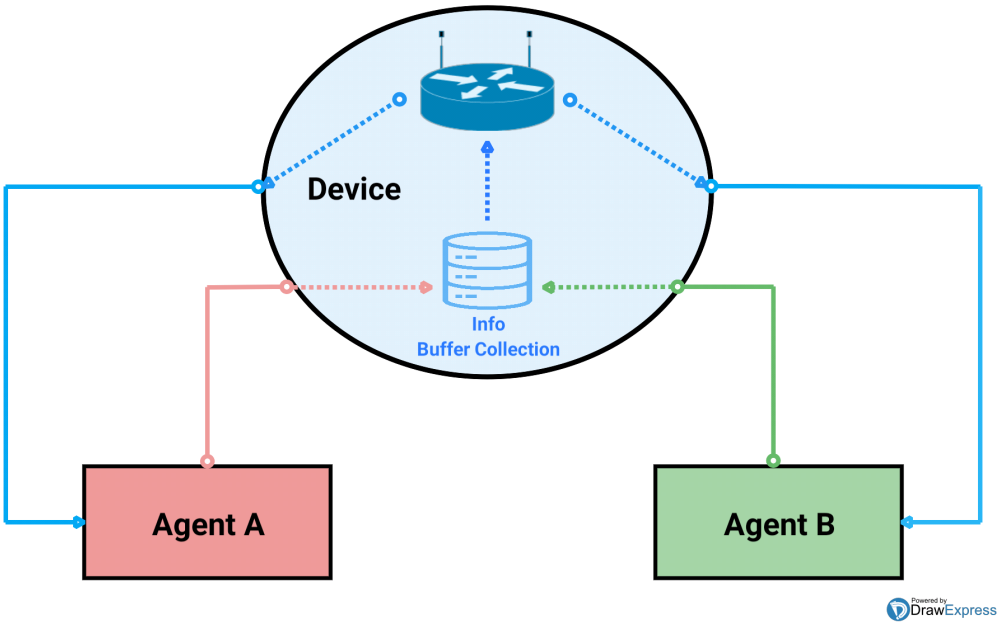
- **Scalable and cooperative approaches.** The multi-agent reinforcement learning solution should use either a centralized training decentralized execution (CTDE) scheme (see Section 2.4 for more details) or even a *fully decentralized* (*FD*) approach. However, in the latter case, it should be augmented through *communication* and/or a *shared reward* (*SR*) among the agents. When the reward is shared, communication cannot even be present. When the agents are sharing a reward (even if partially), they can learn to collaborate without communication and solve the non-stationarity issue. All the works using either only a centralized training/centralized execution or a decentralized training/decentralized execution paradigm will not be considered: we want our agents to be able to act individually and to cooperate at the same time without being necessarily constrained to the global (and maybe not always accessible) state system information during the execution phase unless a specific communication method is provided. We will also not take into account any works in which the training phase is performed sequentially by keeping fixed the policy of all the other UAVs involved during the learning step and, thus, actually not solving the non-stationary online problem (see Section 2 for more details). Collaboration can be considered as such only if all the UAVs can obtain either the same reward or different rewards but share some common observations through direct (Figure 3a) or indirect (Figure 3b) communication. The observation communicated can result either from the concatenation of all the agents' individual observations (i.e., *global communication*, referred to as *GC*) or from a local and/or partial info exchange (i.e., *local communication*, referred as *LC*): if the shared observations are local, they should not necessarily be partial. Info exchange can be about any feature (e.g., sensor detections, gradients, and q-values). When the

training phase is centralized, then cooperation is intrinsically guaranteed. Even if the communication module is not based on a real transmission protocol, this is still a valid feature for UAVs' cooperation. Our criteria reject all the cases where there is no clear info about how the global observations are supposed to be locally available on the agent side at execution time. Even though some works study multi-UAV systems resulting in a final collaborative task, they were not selected if the agents involved are not aware of each other and, thus, every time that the cooperation could be implicitly derived only from the interaction between the environment objects and the agents. For example, the latter case could happen in some specific application scenarios where different users can be served only by one agent at a time; these specific interactions implicitly lead to cooperation among agents, which will then avoid all the already served users. These first filtering choices are meant to be focused on scalable solutions (i.e., without a central control unit at execution time) and cooperative approaches through the usage of shared resources or communications. Indeed, it is well known that completely centralized approaches suffer from the curse of dimensionality;

- **Deep RL.** We only consider reinforcement learning approaches that use function approximators based on deep neural networks. This choice is guided by the fact that we want to focus on generalizable solutions: in real environments, some unpredictable events which are not experienced during the learning phase can occur, and RL techniques cannot handle them as smoothly as DRL approaches can;
- **Number of UAVs.** The number of UAVs effectively used in the experiments should be at least greater than two (comparison analyses including a number of drones ranging from 2 to $N$, with $N > 2$, have still been taken into account). For the sake of clarity, we highlight that we filtered an article out even though it was focused on a system (which could be heterogeneous) made up of a number of agents equal to or greater than three (but with a number of UAVs involved in the considered system not greater than two). In addition, we did not select any papers where the agents are not represented by the UAVs even if UAVs are somehow involved (indeed, sometimes, it could happen that the agents are represented by a central controller such as a server or a base station);
- **Clear description of the framework.** The MDP and the framework structure must be explicitly defined. If not, their main features (i.e., state space, action space, or reward function) must be clearly or easily derivable from the context or figures.


**(a)** Agent to Agent (A2A).

**Figure 3.** *Cont.*




**(b)** Agent to Device (A2D).

**Figure 3.** A2A (**a**) and A2D (**b**) communications: (**a**) represent a direct communication between agents, while in (**b**), communication takes place through a device, which can be a server, a central communication node (supervisor), or any device that is supposed to collect info from each agent and then broadcast the global state to all the agents. Only two agents have been represented here for easy representation, but there can be more than two. These images were generated through *Draw Express Lite* [19].

## 4. DRL-Based Multi-UAV Applications

In this section, we show the main features of the selected works, which have been divided into five main classes, assigned according to each multi-UAV application considered:

- Coverage;
- Adversarial Search and Game;
- Computational Offloading;
- Communication;
- Target-Driven Navigation.

Before moving on to the multi-UAV classes analysis, we want to highlight some particular aspects that can help better understand what follows. First of all, the terms *UAVs*, *drones*, and *agents* are used in an interchangeable way and with the same meaning.

Every class feature comparison table (e.g., Table 1) shows different features associated with each application class. The following features will be considered in every multi-UAV class (some useful abbreviations that could be used for these features are reported in the abbreviation list in Section 6):

- *Algorithm* indicates the algorithm used by the authors;
- *Paradigm* refers to the learning paradigm used:
  - Centralized training with decentralized execution (CTDE);
  - Fully decentralized (FD) with either local communication (FDLC), global communication (FDGC), or shared reward (FDSR). A combination of some or all the previous ones is also possible (e.g., FDLCSR or FDGCSR);
- *No. of UAVs* points out the number of UAVs considered in the experiments;
- *Collision avoidance* is specified by a flag value (either a check mark or a cross) indicating whether collision avoidance is taken into account;
- *Action space* is associated with a categorical value, i.e., it can be specified by *discrete* (D), *continuous* (C), or *hybrid* (H) in order to indicate the value type of the considered action

space. The type of action space is not indicated whenever the authors do not clearly define it or not smoothly inferrable (a cross will be present in this particular case);
- *State space*: as for the action space, but here, it is referred to as the state space;
- *3D* is a flag value indicating if the UAVs can move in a three-dimensional space or not (i.e., only 2D);
- *UAV dynamic model* shows whether a UAV dynamic model is used (a check mark will be present) or not (a cross will be reported even if the UAV dynamic model is either not specified or not clearly inferable). The dynamic model must include low-level control inputs such as forces and/or torques and, thus, take into account also the mass of the UAVs. All the works using a simulator provided with realistic UAVs models, e.g., *AirSim* [22] and *Gazebo* [23], will be considered as solutions that include the UAV dynamic model by default. This feature does not apply to any studies using the dynamic model of a material point, even when including the forces and/or torques applied on the UAVs' center of mass;
- *Propulsion energy consumption* is specified by a flag value (either a check mark or a cross), indicating whether the energy consumption due to the flight phase is considered or not.

Other task-dependent features vary from class to class. They will be indicated by * in the associated class table (their meaning will be explained inside each dedicated class section).

It is worth mentioning that the number of UAVs (indicated by *No. of UAVs*) shown inside the comparative tables follows the rules described below:

- *i-j* points out that a number of UAVs between *i* and *j* (it could also include all the integer numbers among them) is used for the considered case;
- *i,j,k,...* indicates instead that a number of UAVs respectively and exactly equal to *i,j,k,...* is tested for the specific considered case.

Finally, we considered as a 2D case every scenario in which the UAVs were assumed to fly at a fixed and constant height. Indeed, even though the UAVs' height is specified, if it is fixed, the movement along the Z-axis is not actually considered.

### 4.1. Coverage

In this section, we consider all those applications involving a fair distribution of UAVs with respect to a series of requirements.

The coverage task is usually associated with a fairness index that requires all points of interest (PoI) to be treated fairly without prioritizing some over others. The models of the environments in the *coverage* category share a common structure. More in detail, a number $\mathcal{N} = \{n = 1, 2, \ldots, N\}$ of UAVs with a limited sensing range and a set $\mathcal{K} = \{k = 1, 2, \ldots, K\}$ of PoIs to be covered is considered (usually $K > N$); additional constraints may include obstacle avoidance and limited battery. One of the main functions used to bring geographical fairness is *Jain's Fairness* index:

$$[f_t = \frac{(\sum_{k=1}^{K} c_t(k))^2}{K \sum_{k=1}^{K} c_t(k)^2}] \quad (1)$$

where $c_t(k)$ is a generic function that evaluates the coverage score of a specific PoI $k$ at time step $t$.

The target applications for this class are usually related to mobile crowd sensing (MCS) [24,25] and mobile-assisted edge computing (MEC) [26]. In MCS, a group of UAVs equipped with sensors must collect data in contexts such as smart cities, emergency rescues, agricultural productions, solar panel inspections, and many others. In MEC applications, a team of UAVs support ground user equipment in order to offload part of their tasks. Such a scenario can be modeled with different levels of complexity by considering or not additional constraints such as collision avoidance, energy consumption, dynamic PoI, or unconstrained UAVs' movement on a 3D plane.

Table 1 contains different features associated with the *coverage* class. In particular, the *PoI Model* represents works in which the points of interest are not static and predefined at the beginning, but their dynamics are explicitly modeled.

**Table 1.** Coverage class features.The mark ∗ indicates class-dependent features and is thus considered only for this class.

| Ref. | Algorithm | Par. | No. of UAVs | Col. Av. | Prop. Eg. | Action Space | State Space | 3D | UAV Dyn. | PoI Model * |
|---|---|---|---|---|---|---|---|---|---|---|
| [24] | EDICS | CTDE | 2–5 | ✓ | ✓ | C | H | X | X | X |
| [27] | DDPG | FDGCSR | 3–8 | X | ✓ | C | C | X | X | X |
| [26] | MADDPG | CTDE | 3,4 | ✓ | ✓ | C | H | X | X | X |
| [28] | PPO | FDSR | 3,4 | X | ✓ | D | C | X | X | ✓ |
| [29] | HGAT | FDLC | 10,20,30,40 | X | ✓ | D | C | X | X | X |
| [30] | SBG-AC | FDSR | 3–9 | ✓ | ✓ | D | C | ✓ | X | X |
| [25] | DRL-eFresh | CTDE | up to 50 | ✓ | ✓ | C | H | X | X | X |
| [31] | MFTRPO | FDLC | 5–10 | X | ✓ | C | H | X | X | X |
| [32] | SDQN | CTDE | 10 | X | X | D | D | X | X | X |

In this category, the work of Nemer et al. [30] is the only one to explicitly consider the 3D movement of UAVs in the DRL algorithm. The authors proposed a distributed solution to address the challenge of fair coverage and energy consumption for UAVs to provide communication to ground users. In their scenario, UAVs are free to move in a 3D space, but the ground region is divided into cells, and the center of each cell is a PoI. Each UAV is required to cover a number of PoIs for a certain time. They modeled both the channels for communication and energy consumption. The problem is formulated as a state-based potential game, and it is combined with the DDPG algorithm; the authors refer to this solution as State-Based Game with Actor–Critic (SBG-AC). This approach was able to compete and sometimes outperform both the centralized DRL-EC3 [33] and distributed Distributed-DRL [27] under different metrics such as the average coverage score, the fairness index, and the normalized average energy consumption. Instead, Bai et al. [28] are the only ones considering dynamic PoIs. Here, the goal is to use a team of UAVs to provide fair communication quality of service (QoS) to all the users, whose movement is defined through a random walk model. Although the collision avoidance task is not taken into account, obstacles are present and may obstruct communication. They used the PPO algorithm, training a single network with the experience of all the UAVs, and then, they distributed the network to all the UAVs during the testing phase. Thus, the authors define their method as CTDE, but we argue that this approach can be defined as a decentralized approach with parameter sharing and cooperative reward. In addition to the geographical fairness, in Wang et al. [26], the authors also take into consideration the fairness of load balance at each UAV. More in detail, multi-UAV-assisted mobile edge computing is here considered, where UAVs support ground user equipment (UEs) by allowing them to offload part of their tasks. The UEs (and not the UAVs) decide to offload or to compute a task locally based on a minimum energy consumption principle: this is why we did not include this work in the computational offloading class. The goal is to maximize the geographical fairness among the UEs, the fairness of UE-load at each UAV, and to minimize the energy consumption for all the UEs. Similarly to the previous case, a CTDE solution for the mobile crowd sensing (MCS) problem is provided by Liu et al. [24]. Here, both UAVs and driverless cars are used as mobile terminals (MTs) to collect data. They are constrained by their carrying capacity and sensing range: the main objective is to provide an efficient route for the MTs given a sensing area. The authors proposed an energy-efficient distributed mobile crowd sensing (Edics) algorithm based on MADDPG with an N-step return and a prioritized experience replay. They compared Edics with four baseline algorithms (including the classical MADDPG approach) and used metrics such as energy efficiency, geographical fairness, data collection, and energy consumption ratios. They were able to outperform the baseline in almost every scenario.

Similarly, an MCS scenario is also considered by Dai et al. [25], but differently from the previous case, the authors also add a constraint on data freshness by setting a duration dead-

line for the validity of the sensed information. In particular, the UAVs need to maximize the amount of data collected and the geographical fairness and to minimize energy consumption by avoiding obstacles and guaranteeing the freshness of data. The starting point for their proposed solution is the proximal policy optimization algorithm (PPO), called DRL-eFresh: a shared global PPO model is trained while multiple workers receive a copy of the global PPO model in order to interact independently with the environment. A different approach for the coverage problem based on game theory is given by Chen et al. [31]. The main goal here is to provide fair coverage to a set of ground users (or PoIs) while minimizing the UAVs' energy consumption. To solve this problem, the authors formalized it as a mean field game (MFG) and used the Hamilton–Jacobi–Bellman/Fokker–Planck–Kolmogorov (HJB/FBP) equation to find the Nash-mean field equilibrium (Nash-MFE). Nevertheless, this approach presents some problems as it is required to solve stochastic partial differential equations (SPDEs), and the MFE is obtained through two coupled SPDEs whose formulation is prone to human error. Since the authors noticed that the solution to the HJB/FPK equation could also be obtained by using neural networks and reinforcement learning, they proposed a solution based on a combination of MFG and trust region policy optimization (TRPO), named as Mean-Field TRPO (MFTRPO). An agent collects the state of the environment using the action from MFTRPO and sends information to its neighbors.

Liu et al. [27] consider UAVs as mobile base stations (BSs) to provide long-term communication coverage for ground mobile users. The authors use an actor–critic approach with a centralized critic. The target networks are not just copied from the main networks but are slowly updated. The UAVs receive a common reward based on average coverage fairness, which helps them to work cooperatively. With this approach, the authors obtained a better performance than their previous state-of-the-art approach $DRL - EC^3$ [33], which was based on a fully centralized solution. While the previously described solutions were built for ad hoc use cases, Chen et al. [29] provide a generic solution by applying it to two different multi-UAVs operations, i.e., UAV recon and predator-prey scenarios. Below, we will discuss only the design choices related to the UAV recon scenario. Here, the states of the environment are represented as graphs, and two UAVs can communicate only if they are inside a specific communication range. In particular, an agent builds two graphs: an observation graph containing all the entities inside the observation range of the agent and a communication graph containing all the agents inside the communication range of the considered agent. An adjacency matrix is then built for both of these graphs. An encoder layer gives the network structure for both entities and agents inside a group, followed by two hierarchical graph attention (HGAT) modules, one for the entities inside the observation range and one for the entities inside the communication range. The output of the two HGATs is then concatenated and passed on to a recurrent unit. They used an actor–critic framework and conducted experiments with up to 40 UAVs, showing the scalability of their approach.

A more complex scenario is presented by Mou et al. [32], where the goal is to cover a 3D irregular terrain surface through a two-level UAV hierarchy made up of high-level leaders UAVs (LUAVs) and low-level follower UAVs (FUAVs). Each sub-swarm of UAVs uses a star network communication topology where the leader acts as the communication center and can communicate with all the FUAVs. In contrast, the FUAVs can only communicate with their LUAVs. LUAVs have their leader communication network (LCN), where two LUAVs can establish a communication link to communicate with each other if their signal power exceeds a certain threshold. The reinforcement learning agents are represented by the LUAVs, which are supposed to select the next position in the discretized environment. Although the problem is described as a 3D coverage scenario, the reinforcement learning algorithm is used to learn a policy to select one of four possible directions: therefore, we considered it a 2D model in Table 1. The reward comprises a shared part assessing the overall coverage rate of the entire terrain and connectivity conditions of the LCN and a private part punishing the LUAV for revisiting the positions covered previously. The DRL algorithm proposed here, called Swarm Deep Q-learning (SDQN), is based on the

value decomposition method and, thus, structured through a CTDE paradigm. Beyond the specific reinforcement learning algorithm used, as we have already seen in previous papers, some other techniques were adopted, such as parameter sharing [28], prioritized experience replay [24,26], N-step return [24], and the usage of target networks [24,26,27,30].

Eventually, it is worth noticing that different architectures are considered in addition to the standard fully connected layers. In particular, convolutional neural networks (CNNs) [24,25] are used to extract valuable information when data are represented as matrices, while recurrent networks are integrated to take historical data into account [25,29]. Hierarchical graph attention layers (HGAT) are also applied in order to combine both graph neural networks and the attention mechanism [29].

### *4.2. Adversarial Search and Game*

This class includes all the scenarios where two opposing UAV teams compete against each other. Even though from an outer perspective there is a winning and a losing UAV team and hence a *zero-sum* rewards structure, internal cooperation among the agents of the same team is required to defeat the opponents. Relevant features belonging to this class are referred to as *attack model* and *learning enemies* in Table 2. When two teams compete against each other, it could happen that only one team (the "good" team) is learning a specific behavior, while the other team (referred to as the "enemy" team) is not learning any policy. Thus, *learning enemies* indicates whether the opponent team is somehow learning and improving a policy over time either by using another algorithm or even by using the same algorithm as that used by the "good" team. The *attack model* is related instead to the explicit design of a model to decree whether an attack has been successful.

**Table 2.** Adversarial search and game scenario class features. The mark * indicates class-dependent features and is thus considered only for this class: *Att. Mod.* and *Learn. Enem.* stand, respectively, for *attack model* and *learning enemies*.

| Ref. | Algorithm | Par. | No. of UAVs | Act. Space | St. Space | 3D | UAV Dyn. | Col. Av. | Prop. Eg | Att. Mod.* | Learn. Enem.* |
|---|---|---|---|---|---|---|---|---|---|---|---|
| [34] | MADDPG | CTDE | 4 | D | D | ✓ | X | X | X | ✓ | X |
| [35] | MADDPG | CTDE | 2,3 | D | X | ✓ | X | X | X | ✓ | ✓ |
| [36] | WMFQ+ WMFAC | FDLC | 25,50,100 | D | D | X | X | X | X | ✓ | ✓ |
| [29] | HGAT | FDLC | 10,20,30,4 | D | D | X | X | X | X | X | ✓ |
| [37] | MADDPG | CTDE | 3 | C | C | X | X | X | X | ✓ | ✓ |

Wang et al. [36] tackled the problem of UAV swarm confrontation. Using decentralized approaches based only on local observation leads to the problem of non-stationarity since multiple agents are learning simultaneously. In this work, the *situation information* of a UAV is usually determined by itself and its neighboring UAVs. In particular, the authors applied the *mean field theory* to model the communication between UAVs and considered a virtual UAV as an average of its neighboring agents' value functions to model the interaction between the allies (the number of neighbor agents is predefined). Additionally, different weight coefficients were assigned to different neighboring UAVs based on an attention mechanism. This approach allows for handling the dynamism in the number of the UAVs due to the fact that some of them are removed from the scenario during the match: an enemy UAV has a certain probability of being destroyed if it is inside the attacking range of the opponent UAV, which corresponds to a circular sector. Two algorithms using mechanism attention are proposed here, i.e., the weighted Mean Field Q-learning (WMFQ) and the Weighted Mean Field Actor–Critic (WMFAC) approaches. Experiments with a 50 vs. 50 UAV swarm confrontation were performed, and the proposed solution was compared with other algorithms such as independent double Q-learning, independent actor–critic, MFQ, and MFAC. Similarly to the previous case, Li et al. [34] defined a cone-shaped attack area in front of the UAV (associated with a certain probability of being hit) based on the distance and the relative position between the UAV and its target. The proposed solution is based on a MADDPG algorithm with a multi-actor-attention-critic structure. The main

idea is to learn a centralized critic using an attention mechanism to let the agents learn to whom to pay attention. This approach is referred to as Multi-Agent Transformed-based Actor–Critic (MATAC), and it is based on Bidirectional Encoder Representations from Transformers (BERT) [38]. A gated recurrent unit in the actor network is used in order to use historical information in the decision-making process. Additionally, a dual-experience replay mechanism is used to ease the problem of sparse reward. The authors designed a 4 vs. 4 adversarial environment and compared their solution with multiple state-of-the-art algorithms such as MAAC, COMA, and MADDPG. A different approach based on an incomplete information dynamic game was proposed by Ren et al. [35]. The dynamic game was considered incomplete since the UAVs have missing information, such as the opponent's strategy. In the scenario considered here, the flight trajectories of the UAVs are decomposed into seven elementary maneuvers: the goal is to cooperatively decide which maneuvers best fit to achieve victory against the opponent team. The utility function of the dynamic game, which is given by the pay-off of certain mixed strategies, is also used as the reward for the underlying reinforcement learning algorithm. The maneuver decision is based on a situation assessment model, computed through a *situation function* and based on spatial geometric relationships between UAVs. The target enemy of a UAV is the one with the largest value related to the situation function. An attack is considered successful if the enemy is within the attack range of the UAV for a certain amount of time. A dynamic Bayesian network was designed to infer the opponent's tactical decision. Just like in the work by Wang et al. [36], the swarm confrontation problem is also tackled by Zhang et al. [37]. Each UAV is associated with an attacking distance and two different angles: the front angle, which represents the UAV attacking zone, and the rear angle, which indicates the UAV unprotected zone. A UAV can defeat the opponent if the former is within the attacking angle and distance of the considered UAV, which must be inside the unprotected zone of the opponent. Initially, the authors used the standard MADDPG algorithm and, then, added two additional techniques. The first is the *scenario-transfer learning*, which uses experience from one scenario to another. In particular, a two-step transfer learning was used with one of the teams not learning while the other one learns and vice versa. The second additional approach is represented by the usage of a *self-play* technique, which allows the agents to improve by playing against themselves. The teams were trained one at a time. The proposed model was tested in a 3 vs. 3 scenario by showing that the suggested solution has a faster convergence with respect to the standard MADDPG algorithm without using either scenario transfer or self-play techniques. The last work in this section is by Chen et al. [29], whose recon scenario part has already been described in the coverage class. We will now cover the part of the paper associated with the predator–prey scenario, where there is a certain number of predator and prey UAVs, with the predator UAVs having a lower speed than the prey UAVs. The prey is considered caught if its distance from a predator is below a certain threshold, while it is considered safe if a certain amount of time without being caught has elapsed. The action space is the same as the one in the coverage scenario, while the reward depends on the time needed to catch all the prey UAVs. The predator and prey UAVs were trained using the previously described HGAT algorithm and other DRL techniques (e.g., DQN and MADDP).

### *4.3. Computation Offloading*

In modern applications, there is a growing need for remote devices to exchange data with a central node, e.g., a server, in such a way that it can process the information received from any smart device in a significantly reduced time thanks to its much higher computational power. This process is mostly known as *computation offloading*. In real-time applications that employ autonomous vehicles, a critical aspect is finding a proper trade-off between increasing the vehicles' onboard resources and offloading vehicles' tasks to a cloud computing server. The former could increase the vehicles' manufacturing costs and energy consumption. At the same time, the latter could sometimes be unfeasible due to some possible constraints associated with the wireless communication delay: hence, a compromise is

needed. The works described in this section can also handle some system constraints which can be associated with other classes (e.g., *coverage* and/or *communication*), but our intention here is to classify in *computational offloading* all papers where the offloading of some tasks is explicitly mentioned (this is the core goal even if other sub-tasks could be needed).

Table 3 shows some features strictly associated with the *computation offloading* class, which needs to be explained more in detail. The offloading energy (*OFLD Eg*) is associated with any possible energy consumption related to any offloading task, while *OFLD* refers to the implementation of a likely computational offloading model. *Trajs.* indicates instead if the agents' trajectories are either *learned* (L) or set by *default* (D); sometimes, it is assumed that UAVs are not moving, and this particular case is referred to as *fixed* (F).

**Table 3.** Computation offloading class features. The mark $*$ indicates class-dependent features and is thus considered only for this class. *Com. Eg* stands for *communication energy*, while *Trajs.* means *trajectories* and *OFLD* is the abbreviation used for *offloading*.

| Ref. | Algorithm | Par. | No. of UAVs | Col. Av. | Act. Space | State Space | 3D | UAV Dyn. | Prop. Eg | OFLD Eg * | * OFLD | * Trajs. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [39] | MATD3 | CTDE | 2,3 | ✓ | C | C | ✓ | X | X | ✓ | ✓ | L |
| [40] | DQN | FDLC | 1–10 | X | C | C | X | X | X | X | X | F |
| [41] | MASAC | CTDE | 3–6 | X | C | C | X | X | ✓ | ✓ | ✓ | D |
| [42] | MARL AC | FDGC | 50,100,150,200 | X | D | C | X | X | X | ✓ | ✓ | F |
| [43] | MADDPG | CTDE | 2,3,9 | ✓ | C | H | ✓ | X | ✓ | X | ✓ | L |
| [44] | MADDPG | CTDE | 1–6 | ✓ | C | H | ✓ | X | ✓ | X | ✓ | L |

Four studies use a CTDE paradigm [39,41,43,44], and only two works [40,41] seem to vary in an appreciable way from some existing algorithms; in the latter, the mobile edge computing (MEC) technology was investigated, and both the centralized and distributed paradigms were proposed. For what concerns the distributed approach used, the authors could obtain agents' cooperation by using Deep Q-Networks (DQNs) with a Q-value transfer mechanism: the *i-th* UAV can learn a collaborative behavior as its loss function receives the optimal Q-value of neighboring agents at the latest time step. The distributed method here results in being competitive against the centralized one, even if it is worth mentioning that the performance gap in favor of the centralized one increases as the number of UAVs increases. A particular approach was applied by Cheng et al. [41], using a MADDPG framework. However, each UAV was provided with a Soft Actor–Critic (SAC) network instead of the DDPG algorithm usually used in a MADDPG approach: this method, referred to as Multi-Agent Soft Actor–Critic (MASAC), outperforms other well-known algorithms (e.g., SAC, DDPG, and MADDPG) used as baselines in a scenario in which the agents jointly offload tasks and energy to fog computing nodes (FCNs) placed on the ground. It is also worth mentioning the usage of MARL and blockchain technology by Seid et al. [44], where the problem of secure computational offloading with energy harvesting in the context of IoT devices was faced. As UAVs have limited energy resources, blockchain technology was used to secure transactions between resource providers (RPs) and resource requesters (RRs) and to perform operations such as consensus or transaction management. A MADDPG-based algorithm was used, and the proposed solution was evaluated against multiple baselines through different metrics. In computational offloading applications, UAVs are sometimes assumed to not be moving, and the learned actions only include offloading and/or resource allocation strategies, just as the ones used by Liu et al. and Sacco et al. [40,42]. In the latter, the agents were considered as nodes of a graph, where each UAV can communicate with each other through a consensus updating method, and the number of UAVs involved was noticeably larger than the other works belonging to the same class. Additionally, neighbor agents share a local value function. Note also that propulsion and offloading energy consumption were never both taken into account in the same work except for the one proposed by Cheng et al. [41]. Finally, we want to highlight that since the work by Gao et al. (August 2021) [45] does not show significant changes with respect to Gao et al. (December 2021) [43] except for some features (e.g., DDPG instead

of MADDPG) and baseline algorithms used, we will consider only the study by Gao et al. (December 2021) [43] in the comparative table (Table 3).

### *4.4. Communication*

Many multi-UAV systems were studied and designed to ease and support communication services. This task usually examines constraints such as energy, computational, and coverage limitations. We include in this class all the works considering the enhancement of a multi-UAV communication-based system regardless of other possible constraints which could be secondary (even if contributing to the achievement of the final goal). This policy is mainly motivated by the fact that some works belonging to the *communication* class can have intersection features with other classes such as *computation offloading* and/or *coverage*. Thus, we consider here only papers for which the main goal (explicitly or implicitly declared) was trying to optimize the performance of a UAV-based communication system through, for example, optimal resource allocation, throughput maximization, Age of Information (AoI) minimization, optimal data harvesting, and/or trajectories optimization.

Table 4 shows the main features associated with the *communication* class. The most class-dependent features need additional clarification and are provided below. In particular, we highlight that the *communication energy* feature is associated with all possible communication (e.g., among UAVs, among users) energy consumption, while *U2U* and *U2D communication models* refer to the implementation of a likely communication model for UAV-to-UAV (i.e., direct communication among agents) and UAV-to-device communications (a device can be a represented by anything other than a UAV, e.g., a user or a BS). *Trajectories* are also considered since the *communication* class is mainly focused on network aspects. Indeed, sometimes, UAVs' trajectories are assumed to be either predefined, indicated as *default* (D), or *fixed* (F), namely motionless UAVs, rather than *learned* (L).

**Table 4.** Communication class features. The mark ∗ indicates class-dependent features and is thus considered only for this class. *Com. Eg* stands for *communication energy*, while *Trajs.* is the abbreviation used for *trajectories*. *U2U* and *U2D* show, respectively, whether a UAV-to-UAV and/or a UAV-to-device communication model is used.

| Ref. | Algorithm | Par. | No. of UAVs | Col. Av. | Act. Sp. | St. Sp. | 3D | UAV Dyn. | Prop. Eg | Com. Eg ∗ | ∗ U2U | ∗ U2D | ∗ Trajs. |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| [46] | MAUC | CTDE | 2–6 | ✓ | C | C | X | X | ✓ | X | X | ✓ | L |
| [47] | DHDRL | FDLC | 10,15 | X | H | X | X | X | X | X | ✓ | ✓ | D |
| [48] | MAHDRL | CTDE | 3–7 | X | H | H | X | X | X | ✓ | X | ✓ | D |
| [49] | DE-MADDPG | CTDE | 8,10,12 | X | C | C | ✓ | X | X | X | ✓ | ✓ | L |
| [50] | HDRL | CTDE | 2–20 | ✓ | D | H | X | X | ✓ | X | X | ✓ | L |
| [51] | cDQN | FDGC | 3 | X | D | D | ✓ | X | X | X | ✓ | ✓ | L |
| [52] | MADDPG | FDGC | 1,2,3 | ✓ | C | D | ✓ | X | X | ✓ | X | ✓ | L |
| [53] | DQN | FDGC | 10 | X | D | D | X | X | X | X | X | ✓ | L |
| [54] | CA2C | FDGCSR | 2–6 | X | H | H | X | X | X | X | X | ✓ | L |
| [55] | MADRL-SA | FDLCSR | 2–10 | X | H | C | X | X | X | X | X | ✓ | D |
| [56] | DDPG | FDGC | 6,8,10,12 | X | C | C | X | X | X | X | X | ✓ | L |
| [57] | DTPC | CTDE | 4 | ✓ | C | C | X | X | ✓ | ✓ | X | ✓ | L |
| [58] | MADRL-based | CTDE | 3 | ✓ | H | C | X | X | X | X | X | ✓ | L |
| [59] | UAFC | CTDE | 2,3 | X | C | C | X | X | ✓ | ✓ | X | ✓ | L |
| [60] | CAA-MADDPG | CTDE | 3 | X | C | H | ✓ | X | X | X | X | ✓ | L |

Both the CTDE and the decentralized paradigms with communication are quite equally used among the papers in this class. In some works [46,48,52,53,55,56,58], already existing algorithms are directly used, or no significant variations are applied to them: e.g., Qin et al. [46] apply a general MADRL-based distributed UAV-BS Control (MAUC) approach, but actually, it is mainly MADDPG-based as it also uses a centralized critic and decentralized actors (MAUC considers the communication fairness, while other used approaches seem not to take it into account). Even though Zhao et al. [52] use a MADDPG algorithm with experience replay, we can consider it mostly as a classical MADDPG algorithm: similar reasoning can be applied for the work by Wu et al. [53], where a DQN algorithm with experience replay is applied. Another analogous situation is the one faced by Emami et al. [55], where the algorithm used is called Multi-UAV Deep Reinforcement

Learning-based Scheduling Algorithm (MADRL-SA), but it is mainly associated with a DQN approach based on agents cooperation; in this latter case, a data gathering problem is faced, and different ground sensors are used to record and share info about other UAVs with the current UAV that is scheduling the sensor (local communication).

More appreciable modifications to well-known algorithms can be found instead in other studies [47–51,54,57,59,60] which are analyzed in more depth below. Xu et al. [47] proposed a decomposed heterogeneous DRL (DHDRL) approach to enhance the collaborative backhaul rate (BR) with limited information exchange in UAV networks: a backhaul network is part of a network that generally links a central network to other distributed networks of the same hierarchical network. The used backhaul framework decomposes the joint action space into three submodules (action spaces), which are represented by the transmission target selection (TS), the power control (PC), and the power allocation (PA). More in detail, the TS problem is faced through deep Q-learning (DQL) to decrease the BR (also reducing a useless relay), while the PC and PA problems are solved by a DDPG approach, which allows for matching the traffic requirements with suitable transmission power. Even if each UAV executes each of these three modules with a distributed behavior, it is worth mentioning that U2U communication is implemented, and thus, neighbor UAVs can communicate and cooperate accordingly after receiving a local observation from *one-hop* neighbor agents: using graph theory, we can define one-hop neighbors as the nodes (i.e., the agents) close to a specific path between a source and a destination node. Cheng et al. [48] referred to their technique as MAHDRL, showing a similar structure to the QMIX algorithm but using a hybrid action space and an actor–critic structure in order to handle it: the joint UAV–user association, channel allocation, and transmission power control (J-UACAPC) problem is faced here. The main idea behind using the MAHDRL approach is to identify the hybrid action pair (i.e., discrete-continuous action pair) generating the maximum action value. A decomposed MADDPG (DE-MADDPG) algorithm was developed by Zhu et al. [49], who studied a scenario involving UAV clusters by rebuilding a connected network where a virtual navigator was proposed to address the instability issue of the state space: the virtual UAV is associated with the barycenter of the cluster of UAVs and thus allows for the relative coordinates to be considered rather than absolute ones. De-MADDPG combines the MADDPG with a single-agent DDPG using reward decoupling, i.e., a global and a local reward. UAVs are represented here as graph nodes, whose edges indicate the communication links between two agents and depend on the distance between each agent. Zhou et al. [50] modeled their problem as a POMPD by using a hybrid DRL approach with a QoE-driven adaptive deployment strategy: multiple UAVs are deployed at specific positions in a target area to serve moving ground users (GUs) through a wireless connection. Each agent is provided with a value network (distributed control) defined as a deep recurrent Q-network (DRQN), while a hybrid network (which is a feedforward NN) is used to pass the output coming from each agent to a layer aimed at taking the final step of the decision-making process (centralized training). In order to speed up the hybrid DRL training phase, the initial UAV deployment is performed through a *genetic algorithm* (GA), i.e., an algorithm based on a heuristic search inspired by evolution theory (see the review by Katoch et al. [61] for more details); cooperation is achieved by ensuring connectivity and collision avoidance between each UAV. The same scenario with slightly different goals (e.g., coverage maximization instead of QoE maximization) was already dealt with by Ma et al. [62] and further extended by Wu et al. [63], where an attention mechanism is also used. Considering the last three cited papers, only one of them [50] will be inserted into the comparative Table 4 as it provides the most detailed and clear info about the learning structure used, while the other ones result in being simply either a less exhaustive work [62] or a simple extension with not relevant changes except for the introduction of the attention mechanism when using the HDRL technique [63]. The study by Zhang et al. [51] represents instead a particular case in which some coverage constraints are also taken into account, but they are not sufficient (as explained at the beginning of this section) to allow us to classify it into the *coverage* class. The authors thus proposed a decentralized paradigm (indeed, UAVs

act as independent agents) with PS and a constrained DQN (cDQN) algorithm in order to design UAVs trajectories that allow for satisfying some coverage constraints just to improve the capacity of a communication system where ground terminals (GTs) are supposed to be covered by the agents.

The primary control and non-payload communication (CNPC) [64] among UAVs is used as a communication model among agents, and thus, each UAV can access the local positions of all the other UAVs. In some cases, the main goal could be trying to minimize the Age of Information (AoI) using UAVs [54], where drones perform different sensing tasks while improving the AoI. In this case, an algorithm called Compound-action Actor–Critic (CA2C) is used: it is based on the DQN and DDPG algorithms to handle actions with discrete and continuous variables. A distributed sense-and-send protocol is used with an actor–critic structure defined for each UAV, and cooperation among agents is obtained through shared reward information collected by a BS and then sent to all the UAVs. The 3D case is tested only with two UAVs, and since we are considering only UAV systems made up of a number of UAVs equal to or greater than three, the third dimension will not be considered as enabled for this work. The work presented by Hu et al. (2020) [54] seems to be an extension of another work [65]. Indeed, three authors out of five are the same: the main difference between these two studies is basically given by the fact that in the former, a more extensive comparative analysis is performed by varying, for example, the agents' flight altitude for different use-cases. For this reason, we consider only the study by Hu et al. (2020) [54] in Table 4. The AoI problem was also investigated by Wu et al. [56], who applied a DDPG-based algorithm. The main difference with respect to the classical DDPG approach is that, here, the training sample is generated after each cycle (including all its samples) and not after each state transition: a cycle is made up of temporal steps depending on some specific triggering events. An experience replay is also used in order to generate the training set. A BS here allows each UAV to know the other agents' state. This paper is in an extension of another work by Wu et al. (2020) [66], and for this reason, the latter will not be taken into account in Table 4, as it does not add/modify any feature to the work by Wu et al. [56]. Similar reasoning applies to the study by Wu et al. (2019) [67], which is basically a less extensive version of the already mentioned manuscript by Wu et al. (2020) [53]. It is worth noting also that the same four authors' names appear in all the latest four cited works (except for two cases in which other two authors are also involved): these works are indeed strictly related as they deal with the same scenario features by either changing the optimization problem (and the used algorithm accordingly) or including some slightly different features and/or baselines algorithms. This is why we decided to insert in the comparative Table 4 only the more exhaustive articles in terms of features and comparative analysis. Some other works can still be mentioned as *communication* is the application class with the largest number of studies. For example, Chen et al. [57] used UAVs as base stations to provide ground user coverage. The agents perceived partial information about the positions of the M nearest ground users and the position of all the UAVs. Those authors proposed a decentralized trajectory and power control (DTPC) algorithm based on MADDPG, but instead of using a different critic for each agent, they used a single centralized critic. A comparative analysis with multiple baselines such as Dueling DQN, Double DQN, DDPG, and MADDPG was also performed. Heterogeneity in the tasks of the UAVs can be found in some specific applications [60], where a cooperative jamming problem was studied by letting UAV jammers help UAV transmitters defend against ground eavesdroppers (GEs). The authors proposed an improved version of MADDPG called Continuous Action Attention MADDPG (CAA-MADDPG). In particular, the attention mechanism was used to give proper weights to the position of GUs (transmitter UAV case) and GEs (jammer UAVs case). The study presented by Zhang et al. [60] is an extension of another less representative work [68], where MADDPG was applied to the same use-case scenario. For this reason, the latter will not be included in the comparison Table 4. Finally, Zhou et al. [59] investigated a UAV-assisted communication scenario by using a UAV-assisted fair communication (UAFC) algorithm: basically, a multi-agent twin delayed

deep deterministic policy gradient (MATD3) algorithm was applied with an actor–critic structure and a CTDE paradigm. In particular, the *gates functions* allow the agents to select the required information from a central memory according to their observations. In addition, the actor network of the UAVs involves a decomposing and coupling structure: it can decouple the input vector into three categories and expand and aggregate three parts of the information as a unique vector, which helps reduce the state dimension and generate a better policy. The authors also defined a metric to maximize fair system throughput and to guarantee user fairness simultaneously.

Other additional final considerations can eventually be taken into account. The number of agents used for communication scenarios is similar among all the works, and no one used more than 15 UAVs. U2U communication is handled only in three studies [47,49,51], while propulsion and communication energy consumption were both considered in the same problem only in one work in the *communication* class [57]; the 3D scenario is seldomly used except for few cases [49,51,52,60], while the agents' trajectories are mostly learned but not in three of the analyzed situations [47,48,55], where trajectories are set by default. The collision avoidance task is considered only in about one-third of the works belonging to the *communication* class [46,50,52,57,58], even if in approximately four-fifths of the works [46,49–54,56–60], the trajectories are learned by agents, and thus, they should try to avoid collisions among other UAVs and/or obstacles.

### *4.5. Target-Driven Navigation*

Multi-UAV systems are often used in scenarios involving static or dynamic targets to reach, spot, or track. Based on the works selected for this review, we decided to divide this class into two main sub-classes, i.e., *Target Tracking* and *Target Reaching*.

#### 4.5.1. Target Tracking

Many recent applications are focused on trying to make autonomous systems able to track a specific target for different reasons, which could be for entertainment, academic, or safety and surveillance purposes. The main feature of this subclass is that the targets are dynamic, as they need to be tracked.

In Table 5, all the features associated with this specific class are shown. In particular, the feature defined as *multi-target* indicates whether multiple targets are taken into account during the tracking task: this means that we will consider as multi-target problems both the ones associating only one target to each different agent and the ones in which every UAV could track more than one target.

**Table 5.** Target tracking class features: The mark $*$ indicates class-dependent features and is thus considered only for this class.

| Ref. | Algorithm | Par. | No. of UAVs | Col. Av. | Action Space | State Space | 3D | UAV Dyn. | Prop. Eg | Multi-Target * |
|---|---|---|---|---|---|---|---|---|---|---|
| [69] | DRQN-RDDPG | FDSR | 3 | ✓ | D | C | ✓ | ✓ | X | X |
| [70] | MAC$^3$ | CTDE | 3 | ✓ | C | C | X | X | X | X |
| [71] | MAAC-R | FDLCSR | 5,10,20,50, 100,200,1000 | X | D | C | X | X | X | ✓ |
| [72] | PPO | CTDE | 2,5,10,15,20 | X | D | H | X | X | X | ✓ |
| [73] | COM-MADDPG | CTDE | 3,4 | ✓ | C | X | X | X | X | X |
| [74] | Fast-MARDPG | CTDE | 3,4 | ✓ | C | C | X | X | X | ✓ |

All the works belonging to the *target tracking* class apply different algorithms mainly using a CTDE paradigm (except for two cases [69,71]). Yan et al. [72] applies an approach based on the existing PPO algorithm by examining a search and rescue (SAR) problem. SAR problems are actually defined as a combination of searching and tracking tasks, but as we classified these two tasks into separate categories, we put SAR inside the *target tracking* class since the tracking part in a SAR mission is less exploratory than the searching task (and thus generally more suitable to an RL approach).

Some other works belonging to this class try instead to apply more noticeable variations of already known algorithms. For instance, Goh et al. [69] adopted a recurrent approach in two use cases based, respectively, on DQN and DDPG in a multi-UAV actor–critic system to integrate information over time with a combined reward. A reciprocal reward multi-agent actor–critic (MAAC-R) algorithm was used by Zhou et al. [71], who applied an algorithm based on a multi-agent actor–critic by using the parameter sharing technique: here, each agent received an individual reward and the maximum *reciprocal* reward, i.e., the dot product of the reward vector of all neighbor agents and the dependency vector between the considered agent and its neighbors. This work is an extension of a minor one [75], where the same multi-target tracking (MTT) problem was addressed by the same authors (except for one of them not involved here) but applying a D3QN approach by modeling local communication through graphs; a lower number of UAVs was involved also with the usage of cartogram feature representation (FR) to integrate variable length information into a fixed-shape input size. Thus, only the work by Zou et al. [71] will be considered in Table 5. Jiang et al. [73] proposed a MADDPG-based algorithm with the usage of communication networks (COM-MADDPG): an actor–critic structure was used and, more in depth, a critic, an actor, and communication networks were used and associated with their corresponding target networks. The communication network, which allows for exchanging information locally among the agents, obviously facilitated the UAVs' cooperation in achieving the desired goal. A Fast-MARDPG approach was instead proposed by Wei et al. [74]. Since the standard MADDPG algorithm can only act on the current state, the authors improved the recurrent deterministic policy gradient (RDPG) algorithm by combining RDPG and MADDPG. Other authors [70] used a method referred to as MAC$^3$ with curriculum learning (CL) and joint state and action tracker (JSAT), where a long short-term memory (LSTM) was applied to estimate global observation and action over time: the considered scenario involves pursuers defined as cellular-connected UAVs and an evader indicated as an unauthorized UAV. CL is a technique associated with the training phase, where complex scenario constraints are progressively introduced in such a way that the agents can gradually learn from a more and more different and challenging environment.

None of the studies in Table 5 took into account the energy UAV flight consumption, and a remarkable number of UAVs ranging between 5 and 10,000 was considered only in one of them [71]. An effective three-dimensional scenario seems to be involved only in the use-case described by Goh et al. [69], where a particular task associated with this application class was defined: indeed, here, multiple UAVs were supposed to capture the same scene or target simultaneously while flying for *aerial cinematography*. It is worth mentioning also that, even if an adversarial scenario is supposed to take place in the problem analyzed by Wei et al. [74], we do not consider this work as belonging to the *adversarial search and game scenario* class as a real game between different teams is not actually considered, but only target tracking tasks are really dealt with.

#### 4.5.2. Target Reaching

In this specific subclass, the targets are known by the UAVs. They aim to generate a trajectory to reach a specific target while satisfying other constraints, such as collision avoidance.

The CTDE learning paradigm is used in four works [76–79] out of seven, while decentralized training with either local or global communication is adopted in the remaining three papers belonging to this class. Well-known DRL algorithms are mostly used in different studies, but some interesting variations are sometimes applied. A particular case of study is the one associated with the multi-UAV flocking problem [79] faced through a digital twin (DT) DRL-based framework (with an actor–critic structure) reproducing a scenario in which UAVs must reach a target location. The approach here applied is referred to by the authors as a Behavior-Coupling Deep Deterministic Policy Gradient (BCDDPG), and it involves four sub-actor networks. Three of the used sub-networks take as input three different types of state information and output three different types of actions (associated with the related input), while the fourth sub-actor takes as input all the three actions gener-

ated by the other three sub-actors and the joint state space information derived from their single state information: the fourth sub-actor is thus able to output a desired final action. This method can help in generating learned policies with higher quality. In addition, one of the three sub-networks fed with the decomposed input uses an RNN (more in detail, an LSTM) to extract better information from the history of the data passed as input. Since energy consumption here is only used as a performance comparison metric and not used in the reward function, we will not consider it in Table 6 (the energy consumption is indeed indirectly computed as it can be associated with the average travel distance of the agents). A similar multi-UAV problem was faced by Zhao et al. [77], who proposed multi-agent joint proximal policy optimization (MAJPPO) using a state-value function (as in PPO) instead of the Q-value function (as in DQN). In order to enhance the cooperation and stability of the training, they used a moving window average of the state-value function between different agents. More in detail, each agent computes its state-value function through its critic network, and then, a weighted average is used to compute a joint value function for each of them. The joint state-value function is used to optimize the actor network, which controls the movement of the agents. The UAV dynamic model is also taken into account. A DDPG-based approach was used instead by Wang et al. [80], where a two-stage reinforcement learning method was used mainly to face the problem of multi-UAV collision avoidance under uncertainties (it is modeled as a POMPD). The first stage uses a supervised training technique to train the agent to follow well-known collision avoidance strategies, while the second stage uses a policy gradient technique to improve the previously learned strategies. Here, a single policy with experiences from all the UAVs was trained and then distributed during testing. Although the authors considered this approach a CTDE scheme, we argue that this is a decentralized approach with parameter sharing since there is not a real centralized network that uses the global state of the environment as the input. Another interesting study [81] can be considered as a sort of target discovery problem. However, since it seems to be the only one almost entirely related to this class type, we decided to associate it with the *target searching* category. The authors considered a surveillance task with multiple targets to be discovered by a group of UAVs. The scenario is modeled as a network distributed POMDP (ND-POMDP), meaning that not all the agents interact simultaneously: a DDQN algorithm with a parameter-sharing technique was applied. In some cases [76,78,82], a planning approach, and, more in detail, a path planning method, was combined with the DRL technique. Indeed, Qie et al. [76] proposed a solution based on multi-agent deep deterministic policy gradient (MADDPG) to solve multi-UAV target assignment and path planning problems simultaneously: the authors proposed STAPP, i.e., a Simultaneous Target Assignment and Path Planning algorithm. In this problem, the UAV must cover a target while cooperating to minimize the total travel cost. The goal of the target assignment problem was to minimize the traveling distance of UAVs, while path planning should minimize collisions. A comparative analysis with respect to other algorithms was not performed, but many experiments were executed and the results were reported by following some metrics such as Collision Rate Between Agent and Agent (CRBAA), Collision Rate Between Agent and Threat area (CRBAT), and the task completion rate. Lin et al. [78] also combined path planning with a DRL approach: MADDPG was applied to avoid collisions between agents, while path planning helped avoid collisions between agents and static obstacles by providing the agents with info about forbidden areas. A global planner was used instead by Walker et al. [82]. A target-reaching problem was investigated, where a planner coordinated different UAVs to reach some target location in a decentralized way. First, the planner collected local observations from the agents and sent them macro-actions indicating the path to achieve the desired location. Then, UAVs followed the global map information received from the global planner by communicating with each other some local observations when needed: each agent was provided with a local controller based on the PPO algorithm to tune the velocity of the considered UAV properly.

**Table 6.** Target-reaching class features:the mark ∗ indicates class-dependent features and is thus considered only for this class.

| Ref. | Algorithm | Par. | No. of UAVs | Col. Av. | Action Space | State Space | 3D | UAV Dyn. | Prop. Eg | Multi-Target * |
|---|---|---|---|---|---|---|---|---|---|---|
| [76] | STAPP | CTDE | 4,5 | ✓ | X | X | ✓ | X | X | ✓ |
| [77] | MAJPPO | CTDE | 3 | ✓ | C | C | ✓ | ✓ | X | X |
| [80] | DDPG | FDLC | 10,200 | ✓ | X | X | ✓ | ✓ | X | X |
| [82] | PPO | FDGC | 1–4 | ✓ | C | C | ✓ | ✓ | X | ✓ |
| [78] | MADDPG | CTDE | 3 | ✓ | D | C | X | X | X | ✓ |
| [79] | BCDDPG | CTDE | 6,9,12 | ✓ | C | C | X | X | X | X |
| [81] | DDQN | FDGC | 2,3,4 | ✓ | D | D | X | X | X | ✓ |

We noticed that energy consumption is never considered in this specific class, but the collision avoidance task is always taken into account. The three-dimensional use-case is dealt with in approximately half of the works considered here [76,77,80,82] and a significantly larger number of UAVs (up to 200) were involved only in one of them [80].

## 5. Discussion

From Section 4, it is possible to infer relevant considerations and significant aspects related to the DRL-based multi-UAV systems.

### *5.1. Technical Considerations*

We can genuinely state that whenever it is possible to assume that the static objects of the environment are known, a DRL model-based approach must be taken. Indeed, a model-free approach could be reasonable only in cases where UAVs operate in a completely unknown environment, such as a cave or a catastrophic scenario, which were mostly considered in SAR missions. Nevertheless, only a few works seem to use model-based techniques (e.g., [76,78,82]). It should also be noted that none of the selected papers included the delay in the learning process. In a delayed MDP framework [83], we could have three different types of delay:

- *Reward delay.* The reward of the agents can be received only after a specific time interval [84,85] due, for example, to some unpredictable constraints of the considered multi-agent system;
- *Observation delay.* Some observations can be delayed (e.g., [86]), and this may occur in real multi-UAV systems either for sensing or communication reasons;
- *Action delay.* In multi-UAV applications, some actions could take some time before being accomplished, resulting in a system affected by delayed actions [87].

Considering one, some, or all of the delays mentioned above can ease the knowledge transferability from a simulated environment to a real one. This transfer can also be facilitated by the usage of realistic UAV dynamic models involving low-level input quantities (such as forces and torques) controlling the UAVs. Only a few works [69,77,80,82] use this likely UAVs' representation and only through simulators in *target-driven navigation* applications. Some other and more technical observations can be made for what concerns the MDP and its main features (e.g., state space, action space, and reward function), which need to be described clearly and concisely and possibly defined by using the AI field terminology in order to avoid any kind of ambiguity and misunderstanding (e.g., in the work by Patrizi et al. [88], the MDP features are not clearly described and neither a pseudo-code nor a flowchart of the algorithm used is provided). In addition, we want to remark on using some particular learning paradigms associated with a sequential training phase [89], where during the training of the agent $k$, all the learned policies of the other agents are kept fixed. Proceeding in this way, the problem of the non-stationarity of the environment will still be present at execution time, and thus, it should be avoided real multi-UAV application deployment. It is worth mentioning that authors tend to implement custom scenarios for their considered use case: agreeing on a universal framework (or on a limited group of frameworks) is needed to perform extensive and fair comparative analyses in the most modular and scalable way possible. Moreover, the various comparison tables in Section 4

show that a 3D environment is not very often considered in multi-UAV systems. This choice could be a valid one regardless of whether it is always possible to perform a UAV flight level assignment, but in extremely dynamic scenarios (e.g., emergency or catastrophic), it was not possible to allocate the UAVs in a specific-altitude space slot both at the right time and safely. Thus, in all these cases, UAVs need to learn a policy that can be performed throughout three-dimensional space. Another remarkable aspect is that authors mainly focus on DRL-based baselines when performing a comparative analysis with their proposed approach. Researchers should try instead to also use baseline algorithms that are not based on DRL in order to perform a fairer comparison analysis: in this way, they could highlight more markedly and clearly the possible effectiveness of a DRL-based approach against already existing deterministic and classical methods.

Finally, a straightforward overview of the DRL approaches used in multi-UAV systems is given by the scheme shown in Figure 4. Inside the *additional features* set shown in this Figure, *attention* means the application of an attention mechanism to any neural network involved in the usage of a DRL algorithm, while *hybrid* represents a technique either mixing different DRL approaches or including also deep learning methods. This scheme is quite straightforward, and hence, we can infer the following:

- DDPG and PPO are the most used policy-based algorithms;
- The most used value-based algorithm is DQN;
- MADDPG, which is off policy and mainly associated with a continuous action space, is instead the unique algorithm used in all the macro multi-UAV applications;
- The most used techniques combined with DRL methods are the *others*, namely all methods varying the reward and/or gradient sharing and processing, and more generally not included in all the other *additional techniques* specified and shown in Figure 4;

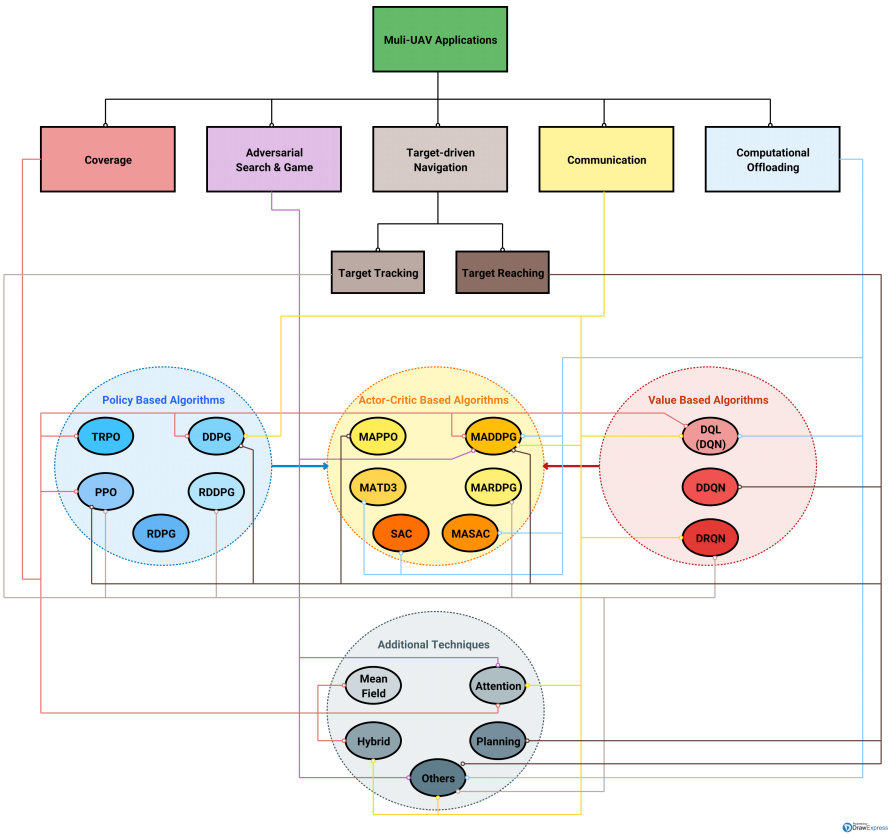
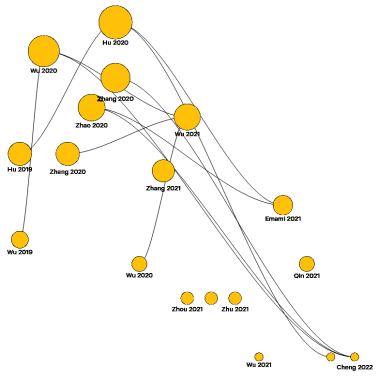



**Figure 4.** Multi-UAV applications and related DRL algorithms taxonomy. See Section 2.3 for more details on *policy-based*, *value-based* and *actor–critic* algorithms; for a better comprehension of the *additional techniques*, see instead Section 2.5.

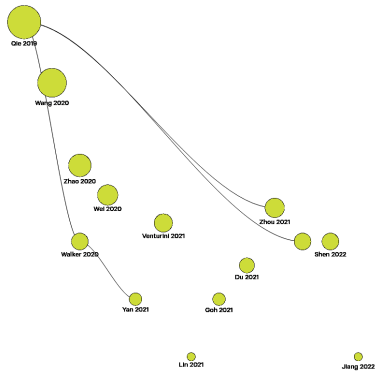
### 5.2. Driving Works and Links Analysis

Here, an overall and visual comparison among the selected papers is conducted in order to understand better whether and possibly how they are linked based either on the specific UAV application considered or on some other extra feature (Figures 5–7 are screenshots taken from interactive maps generated through *Litmaps* [90]).

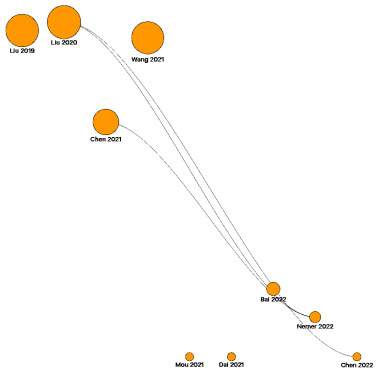
From Figure 5, we can notice that *communication* is the UAV application related to the most works and associations; *target-driven navigation* and *coverage* also present some connections among the papers belonging to the same class, but not as significant as the ones in *communication* application. Very few arches link instead the works associated with the *computational offloading* class, while the ones belonging to the *adversarial search and game* are not connected at all. A particular aspect to be noticed is that even though some of the selected studies have several citations (in absolute terms), they are not taken into account by the papers included in this review. Indeed, Figure 6 shows that some nodes associated with the most cited works (e.g., [24,40,80]) have few or no arches towards the other articles selected in our work: this aspect is quite unexpected as these studies are among the first ones (between all those here included) to apply DRL techniques to cooperative multi-UAV systems, and hence, they should have been analyzed at least by the authors dealing with the same application class. We can further note that the most citations among the papers present in this review are achieved by studies presented by Qie et al. [76] and Wang et al. [58] belonging to the *target reaching* and *coverage* classes, respectively, and thus not belonging to the first application class when it comes to the number of works and paper connections, i.e., the *communication*. Indeed, Figure 6 shows that these works are also mentioned in studies that do not deal with the same UAV application class.



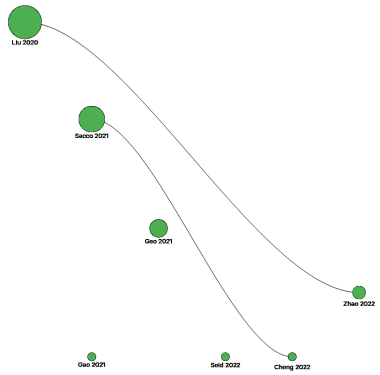

**(a)** Communication class.




**(b)** Target-driven navigation class.

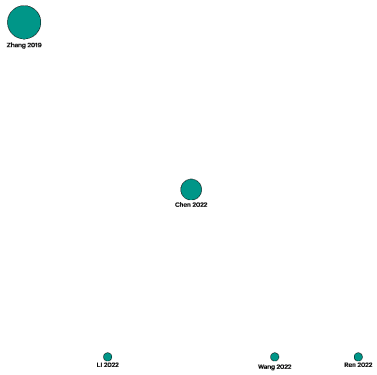



**(c)** Coverage class.




**(d)** Computational offloading class.

**Figure 5.** *Cont.*




**(e)** Adversarial search and game class.

**Figure 5.** Theanalyzed papers are shown here as nodes through five different maps indicating the UAV application classes: the bigger the node, the larger the absolute number of citations (regardless of its belonging class) of the paper associated with it. On each map, on the left side are shown the oldest papers, while on the right side are shown the most recent ones; on the bottom side are the less cited papers, while on the top side are the most cited ones. In order to avoid overlapping and to improve readability, the names of the articles are not always reported below the corresponding nodes. The maps are ordered from left to right and from top to bottom based on descending order of the number of *citation links* (i.e., the arches connecting the nodes) among the studies belonging to the class.

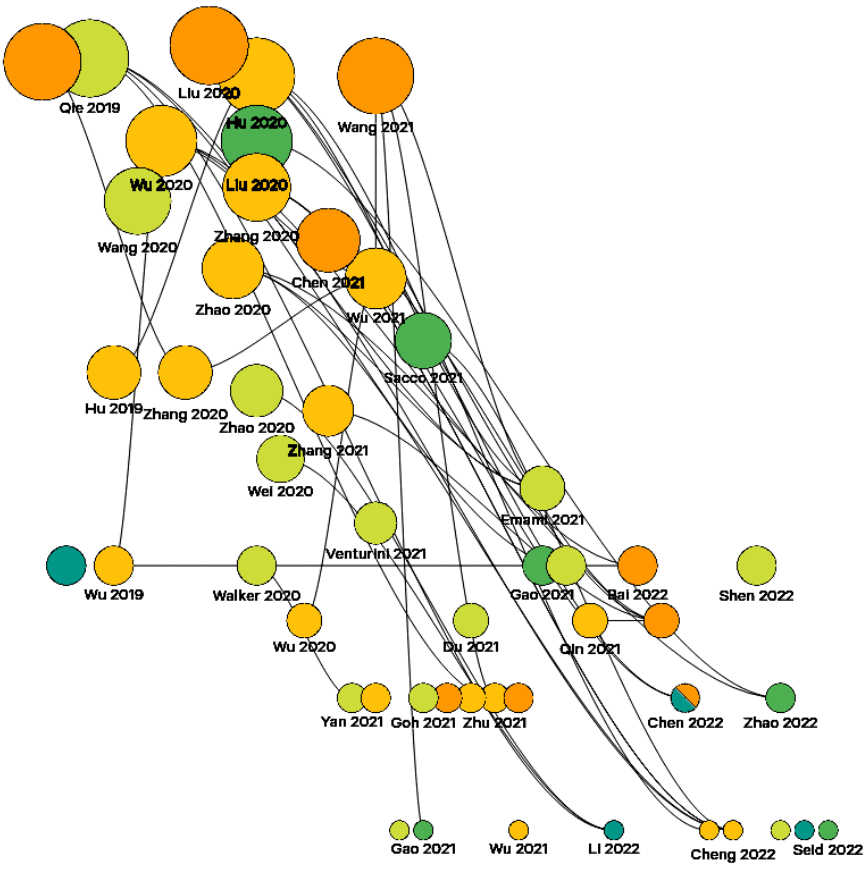



**Figure 6.** Overall paper comparison: the colors, sizes, and arrangement of the nodes associated with each paper follow the same scheme described in Figure 5.

A more general overview of the links associated with the papers included in this article is shown in Figure 7, where we can notice that some relevant works were not included in our review (i.e., the black circles placed inside the grey circular crown) as they do not comply with our filtering methodology described in Section 3. Some of them are associated with applications not focused on multi-UAV systems but dealing with well-known and relevant DRL algorithms such as DQN [7], DDPG [91], or MADDPG [92]. Other ones instead consider UAVs scenarios but without applying (D)RL-based solutions: Zhang et al. (2018) [93] provided a closed-form solution for the joint trajectory design and power control in a UAV relay network, while Zhang et al. (2019) [94] defined a cooperative sense-and-send protocol to improve the quality of service in UAV sensing tasks. Finally, there can also be found all the works related to UAV applications using DRL techniques but without satisfying some specific filters used in our selection methodology: these cases can be represented, for example, by some works in which a centralized learning approach is used [33], or a *tabular* reinforcement learning strategy is applied instead of a *deep* one [95].

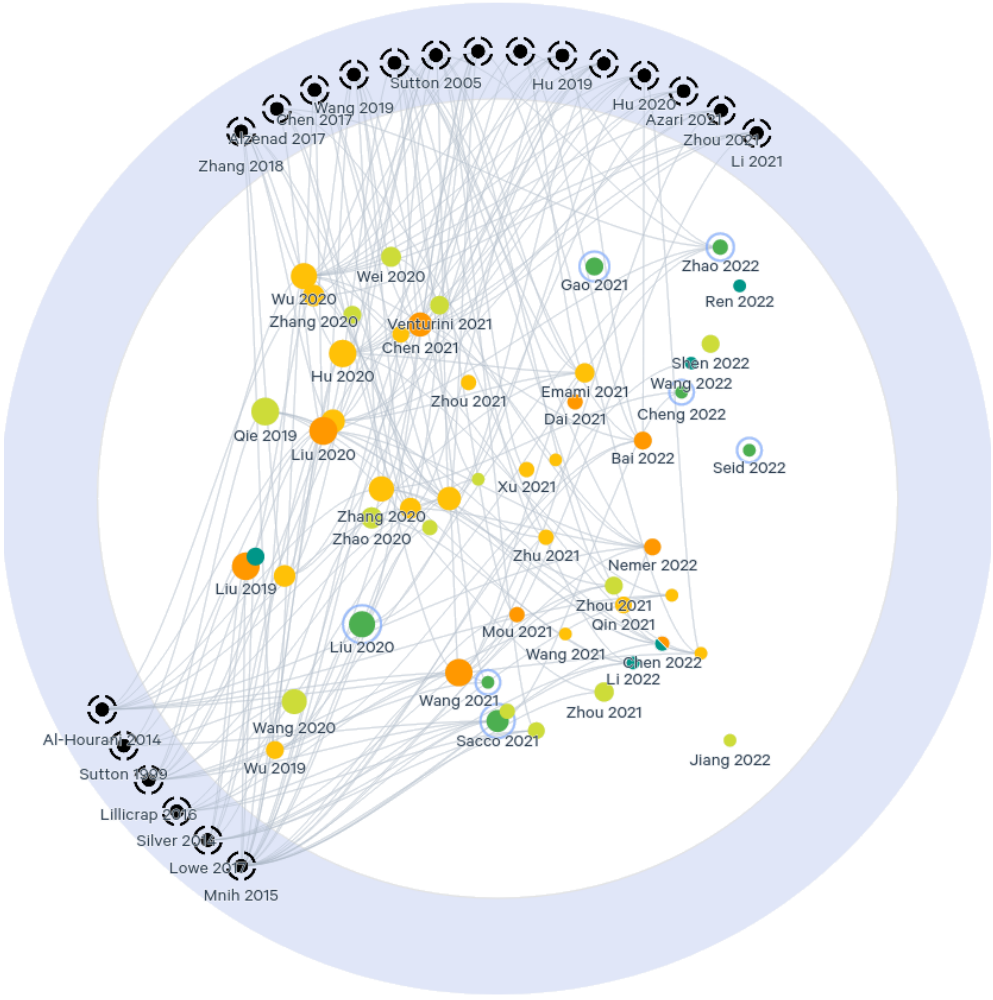



**Figure 7.** Overview of higher-level papers: the grey circular crown contains the papers not included in this review but which are most cited among those involved in it. The colors, sizes, and arrangement of the nodes associated with each paper follow the scheme described in Figures 5 and 6.

### *5.3. General Considerations on Current and Future Developments*

Section 4 allows us to conclude that there exists a preferred direction for what concerns the application scenarios involving multi-UAV DRL-based systems, and it is represented by the *communication* application class, immediately followed by other classes which are directly or indirectly associated with it, such as the *computational offloading* and *coverage*

categories. Additionally, *target-driven navigation* is mainly studied as it involves other subclasses related to target-reaching, target-spotting, and/or target-tracking tasks. The greatest concentration of the works on *communication* could be explained by the increasing need for a stable, reliable, and highly responsive interconnection and data exchange between devices, not only in urban but also in industrial scenarios. Special attention is deserved by all those multi-UAV applications involving monitoring and surveillance tasks that could result in a limitation of individual privacy and freedom whether or not they are properly designed: both the UAV usage and security aspects of data sensing and exchange (e.g., through the usage of a blockchain [44]) should be investigated more in detail. In this regard, we also highlight the psychological impact that multi-UAV DRL-based systems could have on people who, in their daily lives, are not familiar with either AI or drones. People should be made aware of these future scenarios in which drones will "fly over their heads" just as cars now run on the roads. Adequate awareness of people on this topic aimed at underlying the relevance of the usage of AI-based multi-UAV systems in crucial applications (e.g., SAR or medical deliveries) could significantly reduce possible demonstrations and/or intolerant attitudes against such future applications.

## 6. Conclusions

As multi-UAV applications and studies in DRL have progressively increased in the last few years, we decided to investigate and discuss the most used DRL techniques for the most emerging multi-UAV use cases in cooperative scenarios. We classified the main multi-UAV applications into five macro-categories: (i) *coverage*, (ii) *adversarial search and game*, (iii) *computational offloading*, (iv) *communication*, and (v) *target-driven navigation*. An extensive comparison of all the DRL strategies used for each class has been argued, and improvement considerations have been proposed. DRL is undoubtedly helpful in all cases in which a real-time task execution strategy modification based on dynamic feedback is needed (e.g., learning a new behavior through new and additional knowledge of the environment). For the particular topic covered in this article, system responsiveness is pivotal. Indeed, the eventual need to recompute a valid alternative strategy through default emergency schemes and classical deterministic methods could be heavy and slow in performance and, hence, risky for the system's safety and all the items involved in the considered environment. Thus, DRL results are a winning technique whenever it is needed to make a valid and feasible decision within a strictly limited time interval following a dynamic and/or unpredictable event. These requirements are crucial in multi-UAV application scenarios, and this review provides comprehensive and valuable material and suggestions to face and improve these systems using DRL techniques. In order to satisfy these systems' requirements, we suggest directing future works towards mainly solutions that are not fully centralized but cooperative and that explicitly consider the delay in the algorithm design: the former ensures the system scalability, safety, relatively low computational resources, and the independence from a single central control unit, while the latter eases the simulation-to-real knowledge transferability process.

**Author Contributions:** Conceptualization, F.F and D.B; methodology, F.F. and D.B.; investigation, F.F. and D.B.; visualization, F.F. and D.B.; writing—original draft preparation, F.F. and D.B.; writing—review and editing, F.F., D.B. and L.I.; funding acquisition, L.I. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partially funded by the BUBBLES Project under a grant agreement (No. 893206), by the ERC Advanced Grant WhiteMech (No. 834228), and by the PNRR MUR project PE0000013-FAIR.

**Conflicts of Interest:** The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| *Algorithms* | |
| CA2C | Compound-Action Actor–Critic |
| CAA-MADDPG | Continuous Action Attention Multi-Agent Deep Deterministic Policy Gradient |
| COM-MADDPG | Communication Multi-Agent Deep Deterministic Policy Gradient |
| D3QN | Dueling Double Deep Q-Network |
| DDPG | Deep Deterministic Policy Gradient |
| DHDRL | Decomposed Heterogeneous Deep Reinforcement Learning |
| DQL | Deep Q-Learning |
| DQN | Deep Q-Network |
| DRL | Deep Reinforcement Learning |
| DRQN | Deep Recurrent Q-Network |
| De-MADDPG | Decomposed MADDPG |
| FNN | Feedforward Neural Network |
| GA | Genetic Algorithm |
| HGAT | Hierarchical Graph Attention |
| MAAC-R | Reciprocal Reward Multi-Agent Actor–Critic |
| MADDPG | Multi-Agent Deep Deterministic Policy Gradient |
| MADRL-SA | Multi-UAV Deep Reinforcement Learning-Based Scheduling Algorithm |
| MAHDRL | Multi-Agent Hybrid Deep Reinforcement Learning |
| MAJPPO | Multi-Agent Joint Proximal Policy Optimization |
| MARL AC | Multi-Agent Reinforcement Learning Actor–Critic |
| MASAC | Multi-Agent Soft Actor–Critic |
| MATD3 | Multi-Agent Twin Delayed Deep Deterministic Policy Gradient |
| MAUC | Multi-Agent UAV Control |
| MFTRPO | Mean-Field Trust Region Policy Optimization |
| PPO | Proximal Policy Optimization |
| RDDPG | Recurrent Deep Deterministic Policy Gradient |
| RL | Reinforcement Learning |
| RRT | Rapidly Exploring Random Tree |
| SGB-AC | State-Based Game with Actor–Critic |
| STAPP | Simultaneous Target Assignment and Path Planning |
| TRPO | Trust Region Policy Optimization |
| WMFAC | Weighted Mean Field Actor–Critic |
| WMFQ | Weighted Mean Field Q-learning |
| cDQN | Constrained Deep Q-Network |
| *Paradigms* | |
| CTDE | Centralized Training with Decentralized Execution |
| FD | Fully Decentralized |
| GC | Global Communication |
| LC | Local Communication |
| PS | Parameter Sharing |
| SR | Shared Reward |
| *RL Terminology and Domain-Dependent Terms* | |
| Action Sp./Act Sp. | Action Space |
| Alg. | Algorithm |
| BS | Base Station |
| CNN | Convolutional Neural Network |
| Col. Av. | Collision Avoidance |
| Eg | Energy |
| FOMDP | Fully Observable Markovian Decision Process |
| HJB/FBP | Hamilton–Jacobi–Bellman/Fokker–Planck–Kolmogorov |
| LSTM | Long Short-Term Memory |
| MARL | Multi-Agent Reinforcement Learning |

| | |
|---|---|
| MCS | Mobile Crowd Sensing |
| MDP | Markovian Decision Process |
| MEC | Mobile Edge Computing |
| MFE | Mean Field Equilibrium |
| MFG | Mean Field Game |
| MOMDP | Mixed Observability Markovian Decision Process |
| MT | Mobile Terminal |
| POMDP | Partially Observable Markovian Decision Process |
| Par. | Paradigm |
| PoI | Points of Interest |
| Prop. Eg | Propulsion Energy |
| QoS | Quality of Service |
| RNN | Recurrent Neural Network |
| Ref. | Reference |
| SARL | Single-Agent Reinforcement Learning |
| SPDE | Stochastic Partial Differential Equations |
| State Sp./St. Sp. | State Space |
| U2D | UAV-to-Device |
| U2U | UAV-to-UAV |
| UAV | Unmanned Aerial Vehicle |
| UAV Dyn. | UAV Dynamic model |
| UE | User Equipment |

## References


1. Akhloufi, M.A.; Couturier, A.; Castro, N.A. Unmanned Aerial Vehicles for Wildland Fires: Sensing, Perception, Cooperation and Assistance. *Drones* **2021**, *5*, 15. [CrossRef]
2. Hayat, S.; Yanmaz, E.; Brown, T.X.; Bettstetter, C. Multi-objective UAV path planning for search and rescue. In Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA), Singapore, 29 May–3 June 2017; pp. 5569–5574. [CrossRef]
3. Aurambout Jean-Philippe, G.K.C.B. Last mile delivery by drones: an estimation of viable market potential and access to citizens across European cities. *Eur. Transp. Res. Rev.* **2019**, *11*, 30. [CrossRef]
4. Salhaoui, M.; Guerrero-González, A.; Arioua, M.; Ortiz, F.J.; El Oualkadi, A.; Torregrosa, C.L. Smart Industrial IoT Monitoring and Control System Based on UAV and Cloud Computing Applied to a Concrete Plant. *Sensors* **2019**, *19*, 3316. [CrossRef] [PubMed]
5. Zhou, C.; He, H.; Yang, P.; Lyu, F.; Wu, W.; Cheng, N.; Shen, X. Deep RL-based Trajectory Planning for AoI Minimization in UAV-assisted IoT. In Proceedings of the 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), Xi'an, China, 23–25 October 2019; pp. 1–6. [CrossRef]
6. Chakareski, J. UAV-IoT for Next Generation Virtual Reality. *IEEE Trans. Image Process.* **2019**, *28*, 5977–5990. [CrossRef]
7. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef]
8. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
9. Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
10. OpenAI.; Akkaya, I.; Andrychowicz, M.; Chociej, M.; Litwin, M.; McGrew, B.; Petron, A.; Paino, A.; Plappert, M.; Powell, G.; et al. Solving Rubik's Cube with a Robot Hand. *arXiv* **2019**, arXiv:1910.07113.
11. Bithas, P.S.; Michailidis, E.T.; Nomikos, N.; Vouyioukas, D.; Kanatas, A.G. A Survey on Machine-Learning Techniques for UAV-Based Communications. *Sensors* **2019**, *19*, 5170. [CrossRef]
12. Ben Aissa, S.; Ben Letaifa, A. UAV Communications with Machine Learning: Challenges, Applications and Open Issues. *Arab. J. Sci. Eng.* **2022**, 47, 1559–1579. [CrossRef]
13. Puente-Castro, A.; Rivero, D.; Pazos, A.; Fernandez-Blanco, E. A review of artificial intelligence applied to path planning in UAV swarms. *Neural Comput. Appl.* **2022**, 34, 153—170. [CrossRef]
14. Pakrooh, R.; Bohlooli, A. A Survey on Unmanned Aerial Vehicles-Assisted Internet of Things: A Service-Oriented Classification. *Wirel. Pers. Commun.* **2021**, *119*, 1541–1575. [CrossRef]
15. Azar, A.T.; Koubaa, A.; Ali Mohamed, N.; Ibrahim, H.A.; Ibrahim, Z.F.; Kazim, M.; Ammar, A.; Benjdira, B.; Khamis, A.M.; Hameed, I.A.; et al. Drone Deep Reinforcement Learning: A Review. *Electronics* **2021**, *10*, 999. [CrossRef]
16. Sutton, R.; Barto, A. Reinforcement Learning: An Introduction. *IEEE Trans. Neural Netw.* **1998**, *9*, 1054–1054. [CrossRef]
17. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. *Mach. Learn. Proc.* **1994**, 157–163. [CrossRef]
18. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2021**, *55*, 895–943. [CrossRef]

19. DrawExpress Lite [Gesture-recognition Diagram Application]. Available online: https://drawexpress.com/ (accessed on 27 February 2023).
20. Karur, K.; Sharma, N.; Dharmatti, C.; Siegel, J.E. A Survey of Path Planning Algorithms for Mobile Robots. *Vehicles* **2021**, *3*, 448–468. [CrossRef]
21. Niu, Z.; Zhong, G.; Yu, H. A review on the attention mechanism of deep learning. *Neurocomputing* **2021**, *452*, 48–62. [CrossRef]
22. Shah, S.; Dey, D.; Lovett, C.; Kapoor, A. AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles. *arXiv* **2017**, arXiv:1705.05065.
23. Koenig, N.; Howard, A. Design and use paradigms for Gazebo, an open-source multi-robot simulator. In Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566), Sendai, Japan, 28 September–2 October 2004; Volune 3, pp. 2149–2154. [CrossRef]
24. Liu, C.H.; Chen, Z.; Zhan, Y. Energy-Efficient Distributed Mobile Crowd Sensing: A Deep Learning Approach. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 1262–1276. [CrossRef]
25. Dai, Z.; Liu, C.H.; Han, R.; Wang, G.; Leung, K.; Tang, J. Delay-Sensitive Energy-Efficient UAV Crowdsensing by Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2021**, *1233*, 1–15. [CrossRef]
26. Wang, L.; Wang, K.; Pan, C.; Xu, W.; Aslam, N.; Hanzo, L. Multi-Agent Deep Reinforcement Learning-Based Trajectory Planning for Multi-UAV Assisted Mobile Edge Computing. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 73–84. [CrossRef]
27. Liu, C.H.; Ma, X.; Gao, X.; Tang, J. Distributed Energy-Efficient Multi-UAV Navigation for Long-Term Communication Coverage by Deep Reinforcement Learning. *IEEE Trans. Mob. Comput.* **2020**, *19*, 1274–1285. [CrossRef]
28. Bai, C.; Yan, P.; Yu, X.; Guo, J. Learning-based resilience guarantee for multi-UAV collaborative QoS management. *Pattern Recognit.* **2022**, *122*, 108166. [CrossRef]
29. Chen, Y.; Song, G.; Ye, Z.; Jiang, X. Scalable and Transferable Reinforcement Learning for Multi-Agent Mixed Cooperative-Competitive Environments Based on Hierarchical Graph Attention. *Entropy* **2022**, *24*, 563. [CrossRef] [PubMed]
30. Nemer, I.A.; Sheltami, T.R.; Belhaiza, S.; Mahmoud, A.S. Energy-Efficient UAV Movement Control for Fair Communication Coverage: A Deep Reinforcement Learning Approach. *Sensors* **2022**, *22*, 1919. [CrossRef] [PubMed]
31. Chen, D.; Qi, Q.; Zhuang, Z.; Wang, J.; Liao, J.; Han, Z. Mean Field Deep Reinforcement Learning for Fair and Efficient UAV Control. *IEEE Internet Things J.* **2021**, *8*, 813–828. [CrossRef]
32. Mou, Z.; Zhang, Y.; Gao, F.; Wang, H.; Zhang, T.; Han, Z. Three-Dimensional Area Coverage with UAV Swarm based on Deep Reinforcement Learning. *IEEE Int. Conf. Commun.* **2021**, *39*, 3160–3176. [CrossRef]
33. Liu, C.H.; Chen, Z.; Tang, J.; Xu, J.; Piao, C. Energy-efficient UAV control for effective and fair communication coverage: A deep reinforcement learning approach. *IEEE J. Sel. Areas Commun.* **2018**, *36*, 2059–2070. [CrossRef]
34. Li, S.; Jia, Y.; Yang, F.; Qin, Q.; Gao, H.; Zhou, Y. Collaborative Decision-Making Method for Multi-UAV Based on Multiagent Reinforcement Learning. *IEEE Access* **2022**, *10*, 91385–91396. [CrossRef]
35. Ren, Z.; Zhang, D.; Tang, S.; Xiong, W.; heng Yang, S. Cooperative maneuver decision making for multi-UAV air combat based on incomplete information dynamic game. *Def. Technol.* **2022** . [CrossRef]
36. Wang, B.; Li, S.; Gao, X.; Xie, T. Weighted mean field reinforcement learning for large-scale UAV swarm confrontation. *Appl. Intell.* **2022**, 1–16. [CrossRef]
37. Zhang, G.; Li, Y.; Xu, X.; Dai, H. Multiagent reinforcement learning for swarm confrontation environments. In Proceedings of the 12th International Conference, ICIRA 2019, Shenyang, China, 8–11 August 2019; Lecture Notes in Computer Science (including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). Springer: Berlin/Heidelberg, Germany, 2019; Volume 11742 LNAI, pp. 533–543. [CrossRef]
38. Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv*, **2018**, arXiv:1810.04805. [CrossRef]
39. Zhao, N.; Ye, Z.; Pei, Y.; Liang, Y.C.; Niyato, D. Multi-Agent Deep Reinforcement Learning for Task Offloading in UAV-Assisted Mobile Edge Computing. *IEEE Trans. Wirel. Commun.* **2022**, *21*, 6949–6960. [CrossRef]
40. Liu, Y.; Xie, S.; Zhang, Y. Cooperative Offloading and Resource Management for UAV-Enabled Mobile Edge Computing in Power IoT System. *IEEE Trans. Veh. Technol.* **2020**, *69*, 12229–12239. [CrossRef]
41. Cheng, Z.; Liwang, M.; Chen, N.; Huang, L.; Du, X.; Guizani, M. Deep reinforcement learning-based joint task and energy offloading in UAV-aided 6G intelligent edge networks. *Comput. Commun.* **2022**, *192*, 234–244. [CrossRef]
42. Sacco, A.; Esposito, F.; Marchetto, G.; Montuschi, P. Sustainable Task Offloading in UAV Networks via Multi-Agent Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 5003–5015. [CrossRef]
43. Gao, A.; Wang, Q.; Liang, W.; Ding, Z. Game Combined Multi-Agent Reinforcement Learning Approach for UAV Assisted Offloading. *IEEE Trans. Veh. Technol.* **2021**, *70*, 12888–12901. [CrossRef]
44. Seid, A.M.; Lu, J.; Abishu, H.N.; Ayall, T.A. Blockchain-Enabled Task Offloading With Energy Harvesting in Multi-UAV-Assisted IoT Networks: A Multi-Agent DRL Approach. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 3517–3532. [CrossRef]
45. Gao, A.; Wang, Q.; Chen, K.; Liang, W. Multi-UAV Assisted Offloading Optimization: A Game Combined Reinforcement Learning Approach. *IEEE Commun. Lett.* **2021**, *25*, 2629–2633. [CrossRef]
46. Qin, Z.; Liu, Z.; Han, G.; Lin, C.; Guo, L.; Xie, L. Distributed UAV-BSs Trajectory Optimization for User-Level Fair Communication Service With Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 12290–12301. [CrossRef]

47. Xu, W.; Lei, H.; Shang, J. Joint topology construction and power adjustment for UAV networks: A deep reinforcement learning based approach. *China Commun.* **2021**, *18*, 265–283. [CrossRef]
48. Cheng, Z.; Liwang, M.; Chen, N.; Huang, L.; Guizani, N.; Du, X. Learning-based user association and dynamic resource allocation in multi-connectivity enabled unmanned aerial vehicle networks. *Digit. Commun. Netw.* **2022** . [CrossRef]
49. Zhu, Z.; Xie, N.; Zong, K.; Chen, L. Building a Connected Communication Network for UAV Clusters Using DE-MADDPG. *Symmetry* **2021**, *13*, 1537. [CrossRef]
50. Zhou, Y.; Ma, X.; Hu, S.; Zhou, D.; Cheng, N.; Lu, N. QoE-Driven Adaptive Deployment Strategy of Multi-UAV Networks Based on Hybrid Deep Reinforcement Learning. *IEEE Internet Things J.* **2022**, *9*, 5868–5881. [CrossRef]
51. Zhang, W.; Wang, Q.; Liu, X.; Liu, Y.; Chen, Y. Three-Dimension Trajectory Design for Multi-UAV Wireless Network With Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2021**, *70*, 600–612. [CrossRef]
52. Zhao, N.; Liu, Z.; Cheng, Y. Multi-Agent Deep Reinforcement Learning for Trajectory Design and Power Allocation in Multi-UAV Networks. *IEEE Access* **2020**, *8*, 139670–139679. [CrossRef]
53. Wu, F.; Zhang, H.; Wu, J.; Song, L. Cellular UAV-to-Device Communications: Trajectory Design and Mode Selection by Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Commun.* **2020**, *68*, 4175–4189. [CrossRef]
54. Hu, J.; Zhang, H.; Song, L.; Schober, R.; Poor, H.V. Cooperative Internet of UAVs: Distributed Trajectory Design by Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Commun.* **2020**, *68*, 6807–6821. [CrossRef]
55. Emami, Y.; Wei, B.; Li, K.; Ni, W.; Tovar, E. Joint Communication Scheduling and Velocity Control in Multi-UAV-Assisted Sensor Networks: A Deep Reinforcement Learning Approach. *IEEE Trans. Veh. Technol.* **2021**, *70*, 10986–10998. [CrossRef]
56. Wu, F.; Zhang, H.; Wu, J.; Han, Z.; Poor, H.V.; Song, L. UAV-to-Device Underlay Communications: Age of Information Minimization by Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Commun.* **2021**, *69*, 4461–4475. [CrossRef]
57. Chen, B.; Liu, D.; Hanzo, L. Decentralized Trajectory and Power Control Based on Multi-Agent Deep Reinforcement Learning in UAV Networks. *IEEE Int. Conf. Commun.* **2022**, 3983–3988. [CrossRef]
58. Wang, W.; Lin, Y. Trajectory Design and Bandwidth Assignment for UAVs-enabled Communication Network with Multi - Agent Deep Reinforcement Learning. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Norman, OK, USA, 27–30 September 2021; pp. 1–6. [CrossRef]
59. Zhou, Y.; Jin, Z.; Shi, H.; Wang, Z.; Lu, N.; Liu, F. UAV-Assisted Fair Communication for Mobile Networks: A Multi-Agent Deep Reinforcement Learning Approach. *Remote Sens.* **2022**, *14*, 5662. [CrossRef]
60. Zhang, Y.; Mou, Z.; Gao, F.; Jiang, J.; Ding, R.; Han, Z. UAV-Enabled Secure Communications by Multi-Agent Deep Reinforcement Learning. *IEEE Trans. Veh. Technol.* **2020**, *69*, 11599–11611. [CrossRef]
61. Katoch, S.; Chauhan, S.S.; Kumar, V. A review on genetic algorithm: Past, present, and future. *Multimed. Tools Appl.* **2021**, *80*, 8091–8126. [CrossRef] [PubMed]
62. Ma, X.; Hu, S.; Zhou, D.; Zhou, Y.; Lu, N. Adaptive Deployment of UAV-Aided Networks Based on Hybrid Deep Reinforcement Learning. In Proceedings of the 2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall), Victoria, BC, Canada, 18 November–16 December 2020; pp. 1–6. [CrossRef]
63. Wu, J.; Cheng, X.; Ma, X.; Li, W.; Zhou, Y. A Time-Efficient and Attention-Aware Deployment Strategy for UAV Networks Driven by Deep Reinforcement Learning. In Proceedings of the 2021 IEEE 94th Vehicular Technology Conference (VTC2021-Fall), Norman, OK, USA, 27–30 September 2021; pp. 1–5. [CrossRef]
64. Zeng, Y.; Zhang, R.; Lim, T.J. Wireless communications with unmanned aerial vehicles: opportunities and challenges. *IEEE Commun. Mag.* **2016**, *54*, 36–42. [CrossRef]
65. Hu, J.; Zhang, H.; Bian, K.; Song, L.; Han, Z. Distributed trajectory design for cooperative internet of UAVs using deep reinforcement learning. In Proceedings of the 2019 IEEE Global Communications Conference, GLOBECOM 2019-Proceedings, Waikoloa, HI, USA, 9–13 December 2019. [CrossRef]
66. Wu, F.; Zhang, H.; Wu, J.; Song, L.; Han, Z.; Poor, H.V. AoI Minimization for UAV-to-Device Underlay Communication by Multi-agent Deep Reinforcement Learning. In Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
67. Wu, F.; Zhang, H.; Wu, J.; Song, L. Trajectory Design for Overlay UAV-to-Device Communications by Deep Reinforcement Learning. In Proceedings of the 2019 IEEE Global Communications Conference (GLOBECOM), Waikoloa, HI, USA, 9–13 December 2019; pp. 1–6. [CrossRef]
68. Zhang, Y.; Zhuang, Z.; Gao, F.; Wang, J.; Han, Z. Multi-Agent Deep Reinforcement Learning for Secure UAV Communications. In Proceedings of the 2020 IEEE Wireless Communications and Networking Conference (WCNC), Seoul, Republic of Korea, 25–28 May 2020; pp. 1–5. [CrossRef]
69. Goh, K.C.; Ng, R.B.; Wong, Y.K.; Ho, N.J.; Chua, M.C. Aerial filming with synchronized drones using reinforcement learning Multimedia Tools and Applications. *Multimed. Tools Appl.* **2021**, *80*, 18125–18150. [CrossRef]
70. Du, W.; Guo, T.; Chen, J.; Li, B.; Zhu, G.; Cao, X. Cooperative pursuit of unauthorized UAVs in urban airspace via Multi-agent reinforcement learning. *Transp. Res. Part Emerg. Technol.* **2021**, *128*, 103122. [CrossRef]
71. ZHOU, W.; LI, J.; LIU, Z.; SHEN, L. Improving multi-target cooperative tracking guidance for UAV swarms using multi-agent reinforcement learning. *Chin. J. Aeronaut.* **2022**, *35*, 100–112. [CrossRef]
72. Yan, P.; Jia, T.; Bai, C. Searching and Tracking an Unknown Number of Targets: A Learning-Based Method Enhanced with Maps Merging. *Sensors* **2021**, *21*, 1076. [CrossRef]

73. Jiang, L.; Wei, R.; Wang, D. UAVs rounding up inspired by communication multi-agent depth deterministic policy gradient. *Appl. Intell.* **2022** . [CrossRef]
74. Wei, X.; Yang, L.; Cao, G.; Lu, T.; Wang, B. Recurrent MADDPG for Object Detection and Assignment in Combat Tasks. *IEEE Access* **2020**, *8*, 163334–163343. [CrossRef]
75. Zhou, W.; Liu, Z.; Li, J.; Xu, X.; Shen, L. Multi-target tracking for unmanned aerial vehicle swarms using deep reinforcement learning. *Neurocomputing* **2021**, *466*, 285–297. [CrossRef]
76. Qie, H.; Shi, D.; Shen, T.; Xu, X.; Li, Y.; Wang, L. Joint Optimization of Multi-UAV Target Assignment and Path Planning Based on Multi-Agent Reinforcement Learning. *IEEE Access* **2019**, *7*, 146264–146272. [CrossRef]
77. Zhao, W.; Chu, H.; Miao, X.; Guo, L.; Shen, H.; Zhu, C.; Zhang, F.; Liang, D. Research on the multiagent joint proximal policy optimization algorithm controlling cooperative fixed-wing uav obstacle avoidance. *Sensors* **2020**, *20*, 4546. [CrossRef] [PubMed]
78. Lin, J.S.; Chiu, H.T.; Gau, R.H. Decentralized Planning-Assisted Deep Reinforcement Learning for Collision and Obstacle Avoidance in UAV Networks. In Proceedings of the 2021 IEEE 93rd Vehicular Technology Conference (VTC2021-Spring), Helsinki, Finland, 25–28 April 2021; pp. 1–7. [CrossRef]
79. Shen, G.; Lei, L.; Li, Z.; Cai, S.; Zhang, L.; Cao, P.; Liu, X. Deep Reinforcement Learning for Flocking Motion of Multi-UAV Systems: Learn From a Digital Twin. *IEEE Internet Things J.* **2022**, *9*, 11141–11153. [CrossRef]
80. Wang, D.; Fan, T.; Han, T.; Pan, J. A Two-Stage Reinforcement Learning Approach for Multi-UAV Collision Avoidance under Imperfect Sensing. *IEEE Robot. Autom. Lett.* **2020**, *5*, 3098–3105. [CrossRef]
81. Venturini, F.; Mason, F.; Pase, F.; Chiariotti, F.; Testolin, A.; Zanella, A.; Zorzi, M. Distributed Reinforcement Learning for Flexible and Efficient UAV Swarm Control. *IEEE Trans. Cogn. Commun. Netw.* **2021**, *7*, 955–969. [CrossRef]
82. Walker, O.; Vanegas, F.; Gonzalez, F. A Framework for Multi-Agent UAV Exploration and Target-Finding in GPS-Denied and Partially Observable Environments. *Sensors* **2020**, *20*, 4739. [CrossRef] [PubMed]
83. Katsikopoulos, K.; Engelbrecht, S. Markov decision processes with delays and asynchronous cost collection. *IEEE Trans. Autom. Control.* **2003**, *48*, 568–574. [CrossRef]
84. Arjona-Medina, J.A.; Gillhofer, M.; Widrich, M.; Unterthiner, T.; Hochreiter, S. RUDDER: Return Decomposition for Delayed Rewards. *Adv. Neural Inf. Process. Syst.* **2019**, *32* .
85. Kim, K. Multi-Agent Deep Q Network to Enhance the Reinforcement Learning for Delayed Reward System. *Appl. Sci.* **2022**, *12*, 3520. [CrossRef]
86. Agarwal, M.; Aggarwal, V. Blind Decision Making: Reinforcement Learning with Delayed Observations. *Proc. Int. Conf. Autom. Plan. Sched.* **2021**, *31*, 2–6. [CrossRef]
87. Chen, B.; Xu, M.; Li, L.; Zhao, D. Delay-aware model-based reinforcement learning for continuous control. *Neurocomputing* **2021**, *450*, 119–128. [CrossRef]
88. Patrizi, N.; Fragkos, G.; Tsiropoulou, E.E.; Papavassiliou, S. Contract-Theoretic Resource Control in Wireless Powered Communication Public Safety Systems. In Proceedings of the GLOBECOM 2020-2020 IEEE Global Communications Conference, Taipei, Taiwan, 7–11 December 2020; pp. 1–6. [CrossRef]
89. 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.* **2021**, *8*, 3786–3800. [CrossRef]
90. Litmaps [Computer Software]. 2023. Available online: https://www.litmaps.com/spotlight-articles/litmaps-2023-redesign (accessed on 27 February 2023).
91. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the 4th International Conference on Learning Representations, ICLR 2016-Conference Track Proceedings, San Juan, Puerto Rico, 2–4 May 2016.
92. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6380–6391.
93. Zhang, S.; Zhang, H.; He, Q.; Bian, K.; Song, L. Joint Trajectory and Power Optimization for UAV Relay Networks. *IEEE Commun. Lett.* **2018**, *22*, 161–164. [CrossRef]
94. Zhang, H.; Song, L.; Han, Z.; Poor, H.V. Cooperation Techniques for a Cellular Internet of Unmanned Aerial Vehicles. *IEEE Wirel. Commun.* **2019**, *26*, 167–173. [CrossRef]
95. Hu, J.; Zhang, H.; Song, L. Reinforcement Learning for Decentralized Trajectory Design in Cellular UAV Networks with Sense-and-Send Protocol. *IEEE Internet Things J.* **2019**, *6*, 6177–6189. [CrossRef]