IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Date of publication xxxx 00, 0000, date of current version xxxx 00, 0000.

Digital Object Identifier 10.1109/ACCESS.2017.DOI

Wildfire Front Monitoring with Multiple UAVs using Deep Q-Learning

ALBERTO VISERAS[1], MICHAEL MEISSNER[1,2], JUAN MARCHAL[1]

[1]Institute of Communications and Navigation of the German Aerospace Center (DLR), 82234, Oberpfaffenhofen, Germany (e-mail: alberto.viserasruiz@dlr.de, juan.marchalgomez@dlr.de)

[2]Department of Computer Science, Technische Universität München (TUM), 85748, Munich, Germany (e-mail: m.meissner@tum.de)

Corresponding author: Alberto Viseras (e-mail: alberto.viserasruiz@dlr.de).

"This work was supported by European H2020 project HEIMDALL under Grant 740689."

ABSTRACT Wildfires destroy thousands of hectares every summer all over the globe. To provide an effective response and to mitigate wildfires impact, firefighters require a real-time monitoring of the fire front. This paper proposes a cooperative reinforcement learning (RL) framework that allows a team of autonomous unmanned aerial vehicles (UAVs) to learn how to monitor a fire front. In the literature, independent Q-learners were proposed to solve a wildfire monitoring task with two UAVs. Here we propose a framework that can be easily extended to a larger number of UAVs. Our framework builds on two methods: multiple single trained Q-learning agents (MSTA) and value decomposition networks (VDN). MSTA trains a single UAV controller, which is then "copied" to each of the UAVs in the team. In contrast, VDN trains agents to learn how to cooperate. We benchmarked in simulations our two considered methods – MSTA and VDN – against two state-of-the-art approaches: independent Q-learners and a joint Q-learner. Simulation results show that our considered methods outperform state-of-the-art approaches in a wildfire front monitoring task with up to 9 fixed-wing and multi-copter UAVs.

INDEX TERMS Robot learning, Multi-robot systems, Unmanned aerial vehicles, Intelligent robots, Mobile robots, Robot control

I. INTRODUCTION

Massive wildfires are spreading across the globe in past years, while causing dramatic social, economical and environmental effects [1]. Most tragic social effects of wildfires include the loss of human lives, as well as injuries caused by heat and flames, smoke inhalation, vehicle crashes, debris and heart attacks. Economical effects of wildfires include the loss of houses, infrastructure, livestock, timber, etc. These losses are estimated to have reached billions of dollars for single large wildfires. Environmental effects include both a short term decrease in air quality due to smoke, and a change in global atmosphere composition, as large wildfires emit large amounts of carbon dioxide leading to a global temperature increase. Wildfires also affect biodiversity; in extreme cases leading to local extinctions.

In order to fight wildfires, it is vital to have accurate real-time information of the fire spread and movement of the fire front, fuel maps, and weather conditions. The availability of real-time information highly increases the success of the fire fighting effort and reduces the risk for the fire fighters [2].

Satellite images are currently one of the main sources to obtain real-time information of a wildfire. However, these are only received once the satellite passes over the area of interest. Moreover, satellite images lack from a high spatial resolution, which is not suitable for small fires [3]. Another alternative source of information are manned aircraft, which are costly and require a highly skilled pilot. One of the most promising solutions to gather real-time information during a wildfire is the use of autonomous unmanned aerial vehicles (UAVs) [3]. Autonomous UAVs have a flight controller that is able to accept external movement commands, in addition to the ones commanded by the remote controller. An example of external commands could be GNSS coordinates to which an UAV should fly to. External commands can be either sent from a base station using a communication link, or can be commanded directly from a microcomputer that is mounted on the drone and is connected physically to the flight controller. Autonomous UAVs require little human resources, as the system can be launched by a single operator. In practice, due to the current legislation in most countries, a safety pilot is however required to control the drone in case of emergency.

The use of multiple autonomous UAVs, instead of a

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

single one, has a clear advantage: it increases the system efficiency and robustness [2]. Moreover, by using multiple UAVs, smaller and cheaper aircraft could be used, which would reduce the total system cost and could increase the system adoption by organizations that are short on financial resources. Contrarily, multiple UAVs involve an additional challenge: the need of an algorithm that allows them to coordinate with each other. While infrared and visible-light image-based wildfire detection and drone hardware are near-operational, algorithms for cooperative control of multiple drones for wildfire monitoring are yet to be further investigated [4], [5], [6], [7], [8], [9].

In the literature, cooperative control is typically achieved by planning predefined trajectories, or by means of approaches based on decentralized control theory or receding horizon optimization. We refer the reader to [10] for more details. In [11] the authors present an approach that goes one step forward, and outperforms a receding horizon optimization method for monitoring a wildfire with two aircraft. The method proposed in [11] uses deep reinforcement learning (Deep-RL), and shows that learning-based approaches are a promising alternative for cooperative control tasks. In particular, the authors use independent Q-learners [12], [13], which is a simple and widely used approach in multi-robot learning. This has however the disadvantage that coordination among robots, arising in a cooperative multi-agent reinforcement learning (cMARL) setting, is ignored.

Our objective in this paper is to investigate approaches that can better deal with the challenges associated to cMARL in a wildfire monitoring task with multiple UAVs. Specifically, we propose the use of two methods that are based on deep Q-learning (DQN): multiple single-trained agents (MSTA) and value decomposition networks (VDN) [14]. MSTA is a simple learning strategy (even simpler than independent learning) that consists of first training single agents, and then testing with multiple agents. Although MSTA does not account for the multi-agent setting (like in independent learning), MSTA shows great benefits compared to it. Contrarily, VDN tackle problems arising in cMARL by using a special structure of the Q-function that is learned centrally for all agents.

We evaluate our proposed methods in simulations in a wildfire front monitoring task with up to 9 UAVs. To simulate the wildfire, we use an stochastic model that incorporates wind. Moreover we analyze two different UAV models: a multi-copter and a fixed-wing model. Results of the simulations demonstrate that our proposed methods outperform state-of-the-art approaches. In addition, we analyze the advantages and drawbacks of both MSTA and VDN algorithms in terms of the task complexity and of the scalability with the number of UAVs.

A. CONTRIBUTIONS AND PAPER OUTLINE

The main contribution of this paper is a novel formulation of a wildfire monitoring task as a cMARL problem. This formulation allows us to apply MSTA and VDN methods to the monitoring task. The use of MSTA and VDN brings us to the next paper contributions:

- Our proposed cMARL formulation outperforms state-of-the-art RL techniques both in learning speed and resulting score.
- We analyzed the effect of the UAV model on the algorithms's performance.
- We investigated the scalability of the algorithms for up to 9 UAVs. Previous work [11] only considered up to 4 UAVs. This demonstrates the scalability of our proposed methods.

The remainder of the paper is organized as follows. First, we review the related work in Section II. Then we state the problem formally in Section III. Next we introduce in Section IV the core concepts of RL and cMARL algorithms. This is followed in Section V by a detailed description of the algorithms we propose to solve a wildfire monitoring task. Then we test our algorithms in simulations in Section VI. We finalize with a summary and outlook of the paper in Sections VII and VIII.

II. RELATED WORK

The use of UAVs to gather information has been studied in a broad domain of applications. These include, but are not limited to, smart agriculture [15], search and rescue missions [16], post-disaster assessment tasks [17], mapping of hazardous material [18], or wildfire monitoring [2]. For a comprehensive survey on drone applications, we refer the reader to [19], [20].

In this paper, we focus on a wildfire monitoring application [2]. Infrared and visible light cameras, as well as the combination of both of them, have been used to detect wildfires, track the fire front or search for hot-spots after a fire [3]. In [21], the authors outlined several infrared-light-based techniques to track wildfires from UAVs and satellites.

Airborne wildfire monitoring and detection offers a higher spatial and temporal resolution compared to satellite imaging [3], [22]. In [23], the authors highlighted the operational aspects for wildfire monitoring with a remotely controlled UAV in large environments. Autonomous UAVs, in contrast to remotely controlled ones, reduce the personnel cost and operational requirements required to monitor a wildfire. In this respect, the use of autonomous UAVs, together with ground-based robots, was proposed in [24]. Multiple UAVs offer clear advantages in terms of efficiency with respect to a single-UAV system [3]. Algorithms to control a fleet of UAVs in a simulated wildfire monitoring task were proposed in [2], [7], [8], [25]–[27]. The authors in [28] proposed an online decentralized method to estimate the model of a wildfire. The proposed methods were tested using real hardware and data obtained from an stochastic fire simulator. In [29], the authors went one step further and proposed a cooperative algorithm, which they tested in field experiments.

Aforementioned works rely on control theory techniques to decide on the UAVs movement. In [11], the authors showed that Deep-RL-based techniques outperform control

theory techniques (in particular, a receding-horizon controller) for a wildfire monitoring task with 2 aircraft using an stochastic wildfire simulator. Specifically, DQN based independent learners were used in [11] to train the agents. Inspired by [11], we decided to further investigate the use of Deep RL for wildfire monitoring with multiple UAVs.

In [6], the authors investigated a Deep RL technique to find trees that are burning and apply a retardant to extinguish the fire. The work from [6] focuses on the detection and extinguishing part of the problem, and not on the monitoring aspect as we do in this paper.

There are multiple works in the literature that focus on the cooperative control of multiple agents for other applications that are not related to wildfire. For example, in [30] the authors used an equilibrium-based Q-learning variant to learn a coverage task. In [31], a decentralized Q-learning variant was proposed for a fully cooperative warehouse tool delivery problem. Deep-RL has been also used for information gathering tasks. In [32], the authors proposed DeepIG, which allows agents to learn how to gather information while avoiding inter-robot collisions. DeepIG was tested in experiments with 3 UAVs in a terrain mapping lab experiment.

The works in [11], [30]–[32] proposed algorithms to learn a multi-robot task and exemplified the challenges that arise in cMARL problems. In the literature, we can find a wide range of other algorithms to tackle cMARL problems. The most common approach is independent learning agents [12], [13]. This uses a single RL algorithm for each agent, while it ignores the rest of the agents in the environment. Independent learning has been combined with deep Q-learning [11], [33], and with policy gradient methods [32], [34].

One of the main problems that arises in independent learning is the non-stationarity of the environment [33], [35]. This is particularly relevant in DQN-based approaches [36], which rely on an experience replay memory (ERM) that might become obsolete for non-stationary environments [37].

Independent learning is, due to its simplicity, one of the standard benchmarks for cMARL algorithms. Another standard baseline is the use of a joint agent [35], which combines all agents into a central one. A joint agent does not scale with the number of agents, as the action space grows exponentially with the number of agents. One solution to tackle the scalability problem is central training of decentralized policies (CTDP) [38]. CTPD is a technique that combines independent and joint learners, and has become a standard paradigm in multi-agent learning. Examples of algorithms that use CTPD are COMA [39] and VDN [14], [40]. COMA uses a centralized critic at training time to train local policies. In particular, we use VDN [14], [40] as CTPD technique. VDN decompose the Q-function into local ones for each agent and use the decomposition only for central training. As benchmarks, we select independent learners and a joint agent, as these are the baselines that are typically considered in the literature.

In Table 1 we present an overview of the most relevant works in the context of this paper. This summary shows us that the only paper that addresses a wildfire monitoring task is [11]. Therefore, we decided to consider [11] as our benchmark.

III. PROBLEM STATEMENT

We aim to efficiently monitor a wildfire front with multiple autonomous UAVs. First, we introduce the wildfire and UAV model that will be used in the rest of the paper. Then we introduce the score, which is a metric that allows us to evaluate the performance of the monitoring task.

A. WILDFIRE MODEL

Our stochastic wildfire model builds on the one from [11]. We define a wildfire as a two-dimensional grid. Each grid cell (n, m) has three values associated to it: (i) an integer value $f_{n,m}$ that quantifies the amount of fuel remaining in the cell, (ii) a binary value $b_{n,m}$ that is 1 if (n, m) is burning and is 0 otherwise, and (ii) an ignition probability $\mathcal{P}_{n,m}$ that quantifies how likely cell (n, m) will start burning at current iteration.

Algorithm 1: Stochastic wildfire simulation

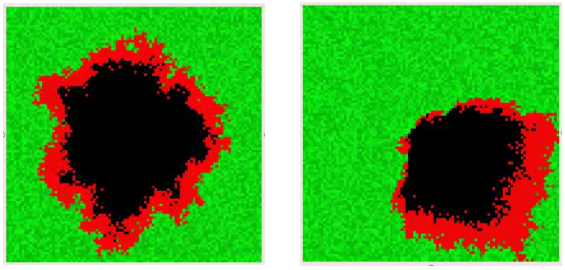
```
Result: Updated b, f and P data structures
Initialize b with one on the desired starting fire cells
  and zero on the rest;
Initialize f with the desired amount of fuel on each
  cell;
Initialize probability P with zero values for each cell;
while Simulation is not finished do
    for each cell (n,m) do
        if b_{n,m} == 1 then
            if f_{n,m} > 0 then
                f_{n,m} = f_{n,m} - 1 ;
            else
                b_{n,m} = 0;
        else
            for each neighbour cell (k,l) do
                update probability P_{n,m} with (1) ;
    Update each b_{n,m} based on P_{n,m};
```

Grid cell values are updated as follows: While cell (n, m) is burning, it consumes one unit of fuel each iteration, until there is no fuel left and it gets extinguished. If a cell has fuel and is not burning, it might ignite. The probability $\mathcal{P}_{n,m}$ of cell (n, m) igniting at any given time depends on all other cells, denoted by (k, l). Let us define $\mathbf{w}$ as a vector that determines wind speed and direction, $\mathbf{d}_{nmkl}$ the vector that connects cells (n, m) and (k, l), and $\mathcal{P}_{nmkl}$ denotes the probability that cell (k, l) ignites (n, m). The latter is separated in two terms: a term $p_{nmkl,0}$ that depends only on the distance between cells, and a term $p_{nmkl,w}$ that is related to wind. These results in the following definition for $\mathcal{P}_{n,m}$:

$$\mathcal{P}_{nm} = 1 - \prod_{k,l}^{\|\mathbf{d}_{nmkl}\| < d_{\max}} [1 - \mathcal{P}_{nmkl}], \qquad (1)$$

	[11]	[6]	[32]	This work
Used method	DQN independent learners	Novel extension of DQN	DeepIG	VDN and MSTA
UAV Type	Fixed-wing	Multi-copter	Multi-copter	Multi-copter and fixed-wing
Focused on wildfire monitoring	✓	∼	✗	✓
Results	Outperforms a receding-horizon controller	Outperforms the proposed heuristics	Outperforms entropy-driven exploration	Outperforms [11]

TABLE 1: Summary of the most relevant works in the context of this paper. Note that [6] focuses on forest fire detection and extinguishing, but not on wildfire monitoring. The only paper that addresses a wildfire monitoring task is [11], which is why we decided to consider [11] as our benchmark.



(a) Without wind. (b) With wind.

FIGURE 1: Visualization of two wildfires simulated with (1). On the left hand side we depict a wildfire that is not affected by wind, while on the right hand side we depict one that is affected by wind. Green cells correspond to cells that have fuel and are not burning, red cells are cells that are burning, and black cells are cells that run out of fuel.

with:

$$\mathcal{P}_{nmkl} = \max(0, \min(1, p_{nmkl,0} + p_{nmkl,\mathrm{w}})), \tag{2a}$$

$$p_{nmkl,0} = \kappa \frac{1}{\|\mathbf{d}_{nmkl}\|^2}, \tag{2b}$$

$$p_{nmkl,\mathrm{w}} = \kappa \frac{\mathbf{w} \circ \mathbf{d}_{nmkl}}{\|\mathbf{d}_{nmkl}\|^2}, \tag{2c}$$

where $\circ$ denotes a scalar product, and κ is a parameter to tune the probability of ignition of each cell. Also note that neighbors that are further apart than the maximum ignition distance $d_{\max}$ cannot ignite each other. This limits the computational demands of the model without losing generality for the generation of fire hotspots that are close to the fire front, which is a realistic assumption. We summarize the wildfire generation in Algorithm 1.

We depict in Figure 1 a visualization of two wildfires that were simulated with the afore-described model.

B. UAV MODEL

We consider two different UAV types: a multi-copter and a fixed-wing aircraft. We assume for both models that they are equipped with a flight controller that translates high-level actions into direct motor commands. The mounted sensor is pointing downwards, so no further conversions needs be done depending on the angle. The sensor detects burning cells within a range equal to d_{sight}. UAVs fly at different heights to avoid collisions.

A multi-copter is modeled as in [32]. Multi-copter's i position is denoted as (x_t^i, y_t^i). Possible actions are one grid cell displacement in north, east, west, or south direction. A fixed-wing is modeled as in [11]. This is a standard model that assumes that aircraft maintain a constant altitude and speed v. The aircraft position and heading are denoted as (x_t^i, y_t^i) and ψ_t^i, respectively. The aircraft is controlled by increasing or decreasing $5°$ the bank angle $\phi_t^i \in [-50°, 50°]$. This results in the following motion equations:

$$\begin{aligned} \dot{x_t^i} &= v \cos \psi_t^i, \\ \dot{y_t^i} &= v \sin \psi_t^i, \\ \dot{\psi_t^i} &= \frac{g}{v} \tan \phi_t^i, \end{aligned} \tag{3}$$

with g the standard gravity.

C. SCORE METRIC

The score metric quantifies the extent of the fire that is burning but is not being monitored by UAVs during a whole episode. It is used as a metric to compare performance between different methods. Let us define first the concept of belief: For each cell (n, m), we define a belief $\hat{b}_{n,m}$ that indicates whether (n, m) was measured as burning. Essentially, UAVs task is to reduce the error between the belief map $\hat{\mathbf{b}} = [\hat{b}_{1,1}, \hat{b}_{1,2}, ..., \hat{b}_{2,1}, ...]$, consisting of all belief cells, and the actual fire state $\mathbf{b} = [b_{1,1}, b_{1,2}, ..., b_{2,1}, ...]$. Let us also define the momentary scaled fire miss $\mathcal{M}_t$ for each time-step t, which counts how many cells are burning momentarily, but are not flagged as burning in the belief map. Fire miss $\mathcal{M}_t$ is given by the following expression:

$$\mathcal{M}_t = \frac{\sum_{n,m} \max(0, b_{n,m} - \hat{b}_{n,m})}{\sum_t \sum_{n,m} b_{n,m}}. \tag{4}$$

Note that $\mathcal{M}_t$ is scaled with respect to the sum of cells burning for a whole episode, which allows us to compare fires of different sizes.

The score metric $\mathcal{S}$ for a whole episode is the integral of $\mathcal{M}_t$ over time:

$$\mathcal{S} = \sum_t \mathcal{M}_t \in [0, 1], \tag{5}$$

where 0 means that each burning cell was detected immediately, and 1 means that UAVs missed the whole fire.

IV. COOPERATIVE MULTI-AGENT DEEP Q-LEARNING

In this section we introduce deep Q-learning, as well as the challenges that arise in a cooperative multi-agent setting. This is followed by a summary of the cMARL methods we chose to solve a wildfire monitoring task.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

A. DEEP Q-LEARNING

Reinforcement learning (RL) allows us to solve Markov decision processes (MDP) by trial and error [41]. It works as follows: an agent makes an observation $o(s)$ of environment's state s, and selects an action a according to a policy $\pi : o \mapsto a$. This action causes the environment to transition stochastically from s to state s'. Then the agent receives a reward r, which indicates whether pair (s, a) was a good (high reward) or a bad choice (low reward). Agent's objective in RL is to maximize the expected future reward.

In this paper, we use Q-learning as RL method. Q-learning learns the optimal state-action values $Q^*(s, a)$. These represent the expected future reward of selecting action a in state s when following an optimal policy afterwards. The Q values can be learned from transition tuples (s, a, r, s') by iteratively applying the so-called Bellman equation [41]. For real-world problems, storing all transition tuples is typically unfeasible from a memory perspective. DQN solves this by approximating the Q-values with a deep neural network [36]. This approximation can, however, lead to instability in training. To this end, two stabilization techniques are applied [36]: the use of two different neural networks to avoid oscillations. An experience replay memory (ERM) is implemented, that uses past transition tuples to decorrelate experiences. This avoids using two consecutive experiences that can be highly correlated and could lead to inefficient learning. The ERM stores a certain number of experiences, which are randomly sampled from the memory to train the network.

B. CHALLENGES IN cMARL

In a cMARL setting, we have N agents that interact with the environment, and receive a common team reward r [42]. Let us define a joint action $\mathbf{a} = [a^1, ..., a^N]$, joint observation $\mathbf{o} = [o^1, ..., o^N]$ and joint policy $\pi^J = [\pi^1, ..., \pi^N]$, with a^i, o^i, π^i the action, observation and policy of agent i with $i = 1, 2, ..., N$, respectively. Agents' goal is to find a joint optimal action that maximizes future team reward.

In the literature we identified fours challenges that are associated to cMARL problems. These are the following:

1) *The course of dimensionality*: it refers to the exponential growth of $\mathbf{a}$, $\mathbf{o}$, and policy $\pi : o \mapsto a$ as N increases [35].
2) *The coordination problem*: it arises when more than one optimal joint action exist for a given state. If agents select their individual actions from different optimal joint actions, the resulting joint action might not be optimal [12].
3) *The spurious reward problem*: it refers to the fact that, as agents receive a team reward, they cannot distinguish whether their individual actions contributed to the team reward. This causes the so-called "lazy agent" problem [14], where some agents might learn that the best action is "doing nothing".
4) *The non-stationary environment problem*: independent learners use a standard RL algorithm on every agent independently, which does not account for the existence of other agents explicitly. These other agents are therefore part of the environment for each agent in the standard RL framework. Since all agents learn simultaneously, the environment for each agent changes over the curse of training [33].

C. IMPLEMENTED cMARL ALGORITHMS

In this paper we investigated four algorithms that tackle and/or mitigate the aforementioned problems that arise in cMARL. They all have in common that they transform the cMARL problem into a single-RL one, which we solve using standard deep Q-learning. Note that the transformation of the cMARL problem into a single-RL one does not incur any loss of accuracy. Specifically, we investigate four methods that we term JOINT, INDI, MSTA and VDN. Next we summarize each of them:

- JOINT: it corresponds to a joint agent, for which a joint policy is trained centrally. Joint actions are also selected centrally, from which individual actions are assigned to corresponding agents. This agent solves challenges 2-4, while it especially suffers from challenge 1. JOINT algorithm is a typical baseline in cMARL.
- INDI: it refers to independent Q-learners. This means that each agent is trained with its own single-RL method, which has no explicit representation of the other agents in the environment. This solves challenge 1, while it ignores challenges 2-4. Nevertheless, INDI is the most used approach in the literature due to its simplicity and good performance. We would like to remark that our benchmark algorithm for wildfire monitoring [11] uses INDI algorithm.
- MSTA: it refers to multiple single-trained agents. MSTA uses a single agent in the environment for training, and multiple instances of that single-trained agent for testing. On the one hand, MSTA reduces the training complexity, as a single agent is trained. On the other hand, multi-agent coordination is not learned. Instead team performance is driven by the individual behavior of each agent. MSTA is particularly well suited for tasks that can also be solved by a single-agent. MSTA solves challenges 1,3,4 while it ignores challenge 2.
- VDN: it refers to value decomposition networks [14]. VDN is a DQN-based technique that implements CTDP. It defines the joint Q-function as the sum of individual Q functions for each agent:

$$Q^J(\mathbf{o}, \mathbf{a}) = \sum_{i=1}^{N} Q^i(o^i, a^i). \tag{6}$$

The intuition behind VDN is that each agent learns its own contribution to the team reward, which solves the spurious reward problem. Training is done centrally for Q^J, while testing is decentralized using the individual Q^i, for $i = 1, ..., N$. Challenge 1 is mitigated, especially

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

Problem	JOINT	INDI	MSTA	VDN
Course of dimensionality (1)	✗*	✓	✓	∼
Coordination (2)	✓	✗	✗	∼
Spurious reward (3)	✓	✗	✓	✓
Non-stationary environment (4)	✓	✗	✓	✓

TABLE 2: Summary of implemented algorithms according to their capabilities to solve the four most common problems that arise in cMARL. Symbols ✓, ✗, ∼ indicate that a problem is solved, unsolved or mitigated, respectively. Symbol ✗* indicates that a problem is not solved, and that the algorithm particularly suffers from it.

when parameter sharing is applied. Challenge 2 is also mitigated, while challenges 3,4 are solved with VDN.

Here we propose the use of MSTA and VDN for wildfire monitoring with multiple UAVs. We benchmark the two algorithms against JOINT and INDI, which are typical baselines in the literature. For each of the aforementioned algorithms, we summarize in Table 2 the cMARL problems that each of the algorithms solve or mitigate. In Table 3 we show an overview of the experience tuples, the Q-function evaluation, and the greedy action selection equation [36] implemented by each of the algorithms. Next we summarize our proposed algorithm.

V. WILDFIRE MONITORING WITH MULTIPLE UAVS

We propose a cMARL framework that permits multiple robots to learn how to gather information. In particular, without loss of generality, we focus here on a wildfire front monitoring task with UAVs. Our proposed framework is used by the four cMARL algorithms we consider in this work: INDI, JOINT, MSTA and VDN (see Section IV-C). First we provide an overview of our framework. Then we describe in detail the agent's observation, reward, and neural network architecture. These are based on [11], which permits a fair comparison between [11] (INDI) and our proposed cMARL methods (VDN and MSTA).

A. FRAMEWORK OVERVIEW

A block diagram that summarizes our proposed framework is depicted in Figure 2. This corresponds to the steps executed by a robot i with $i = 1, ..., N$. We assume an homogeneous team of robots, which implies that the framework is identical for all of them.

Robot i is the "brain" of our system, and it interacts with a "physical environment" and with "other robots". On the one hand, the "physical environment" refers to the wildfire and UAV hardware. They are described by models introduced in Section III-A and III-B, respectively. On the other hand, the "other robots" refers to the rest of robots in the team. That is, all robots $j = 1, .., N;\ j \neq i$. Interaction between robot i, and the "physical environment" and "other robots" takes place through three modules: a sensor, a communication and an actor module.

The "sensor module" samples the "physical environment" every time step. In particular, it samples the robot i state and the wildfire state. Robot i state is denoted as $\mathbf{m}_r^i$, with $\mathbf{m}_r^i = [x_t^i, y_t^i]$ for the multi-copter UAV, and $\mathbf{m}_r^i = [x_t^i, y_t^i, \phi_t^i, \psi_t^i]$ for the fixed-wing UAV. Wildfire state is denoted as $\mathbf{m}_w^i$. This is a vector that contains those grid cells that are burning, as measured by the UAV i thermal camera within a measurement range d_{sight}.

Measurement $\mathbf{m}^i = [\mathbf{m}_r^i, \mathbf{m}_w^i]$, for $i = 1, ..., N$, is shared among all robots through the "communication module". Measurement $\mathbf{m}^i$, together with measurements $\mathbf{m}^{\bar{i}} = \{\mathbf{m}^j\ \forall\ j = 1, ..., N \text{ and } j \neq i\}$, are then stored in the "information storage". The "information storage" keeps wildfire and robot's information separately in two separate submodules: map and data storage. The "map storage" saves most recent wildfire measurement for each grid cell, while the "data storage" saves current robots' state measurements. Note that the "map storage" information corresponds to belief map $\hat{\mathbf{b}}$.

The information stored in the "information storage" module is processed by the "observation renderer" (see Section V-B). The "observation renderer" maps the stored information into the format required by the "policy" module. Here the policy consists of a neural network that we train using one of the four cMARL methods we summarized in Section IV-C. Details on the reward used for training, as well as on the neural network architecture can be found in Section V-D and V-C, respectively.

The "policy" outputs a high-level action that is translated into low level action commands by the "actor module". Since we train and test our algorithms in simulations, the "actor module" essentially updates UAVs state according to the model described in Section III-B.

We propose a framework in which robot i policy can potentially communicate with policies of other robots in the team through a "inter-policy communication" module. This could be used to improve the cooperation between different agents by e.g. sharing features of the neural network [14].

Provided this framework overview, we describe next the agent's observation, reward and neural network architecture.

B. AGENT OBSERVATION

The agent observation has two components: an image and a vector observation. The image observation is depicted in Figure 3. Its number of pixels is equal to the number of the wildfire grid cells. The image observation has two layers. The first layer colors white the cells that are flagged as burning in belief map $\hat{\mathbf{b}}$. Otherwise, cells are colored black. The second layer is a coverage map, which is black for those cells in which a measurement was taken last time-step. Older measurements fade out to white, with white being a measurement that was taken more than 255 time steps ago. Both image layers are centered on each robots position for both drone models, and rotated into the direction of travel for the fixed-wing drone model.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

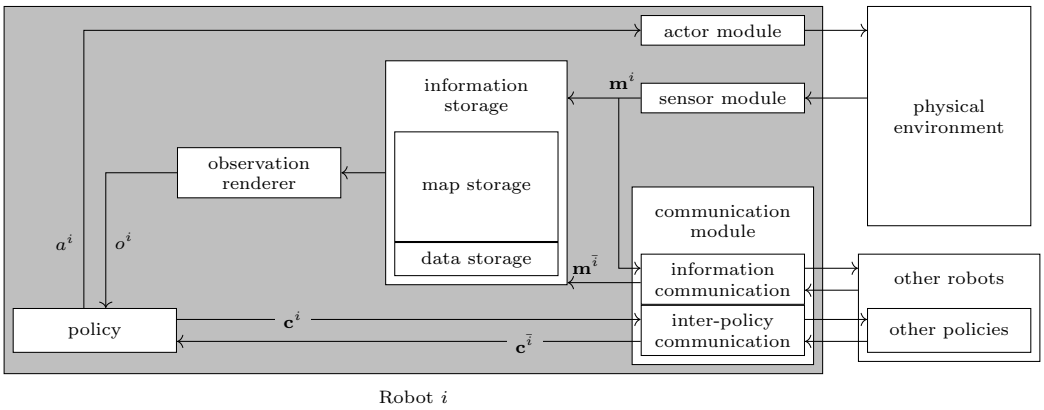
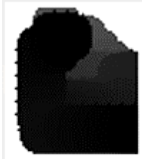


FIGURE 2: Overview of our framework for cMARL information gathering from the perspective of robot i. The grey shaded area corresponds to modules that belong to robot i, while the rest of the modules are external to the robot. Interaction between robot i and the external modules is carried out through the actor, sensor and communication modules.

Algorithm	Experience tuple	Q-function	Greedy action selection
MSTA	(o_t, a_t, r_t, o_{t+1})	$Q(s_t, a_t)$	$a = \text{argmax}_{\tilde{a}_t} Q(o_t, \tilde{a}_t)$
INDI	$(o^i_t, a^i_t, r_t, o^i_{t+1})$	$Q(s^i_t, a^i_t)$	$a^i = \text{argmax}_{\tilde{a}^i_t} Q(o^i_t, \tilde{a}^i_t)$
JOINT	$(\mathbf{o}_t, \mathbf{a}_t, r_t, \mathbf{o}_{t+1})$	$Q(\mathbf{o}_t, \mathbf{a}_t)$	$\mathbf{a} = \text{argmax}_{\tilde{\mathbf{a}}_t} Q(\mathbf{o}_t, \tilde{\mathbf{a}}_t)$
VDN	$(\mathbf{o}_t, \mathbf{a}_t, r_t, \mathbf{o}_{t+1})$	$\sum_{i=1}^N Q^i(o^i, a^i)$	$\mathbf{a} = [\text{argmax}_{\tilde{a}^i_t} Q(o^i_t, \tilde{a}^i_t)]_{i=1,2,...,N}$

TABLE 3: Summary of the experience tuples, Q-function and greedy action selection for the four methods considered in the paper: MSTA, INDI, JOINT and VDN.

(a) Belief map.



(b) Coverage map.

FIGURE 3: Image layers that constitute the agent image observation. (a) Belief map in which white cells are those that are flagged as burning in the belief map. Otherwise, cells are colored black. (b) Coverage map, which is black for those cells in which a measurement was taken last time-step. Older measurements fade out to white.

The vector observation holds information of UAVs states. This includes robot i state, as well as some relative quantities calculated between robot i and robot j states, for all $j = 1, ..., N$ and $j \neq i$. The vector information differs for the multi-copter and fixed-wing UAVs considered in this work. For a multi-copter the vector observation of robot i consists of:

- $[x^i_t, y^i_t]$: robot i position.
- $[\Delta x^{ij}_t, \Delta y^{ij}_t] = [x^j_t - x^i_t, y^j_t - y^i_t]$: relative position of robot j with respect to robot i.
- $d^{ij}_t = \sqrt{(\Delta x^{ij}_t)^2 + (\Delta y^{ij}_t)^2}$: distance between robot i and j.
- a^j_{t-1}: robot j last action.

for all $j = 1, ..., N$ and $j \neq i$.

For a fixed-wing the vector information of robot i consists of:

- ϕ^i_t: robot i bank angle.
- d^{ij}_t: distance between robot i and j.
- $\Delta\beta^{ij}_t = \arctan\left(\frac{\Delta y^{ij}_t}{\Delta x^{ij}_t}\right) - \psi^i_t$: bearing angle of drone j relative to robot i heading.
- $\Delta\psi^{ij}_t = \psi^j_t - \psi^i_t$: robot j heading relative to robot i heading.
- ϕ^j_t: drone j bank angle.

for all $j = 1, ..., N$ and $j \neq i$.

C. AGENT REWARD

Robots receive a reward of $+1$ for each cell that (i) is measured by any of the UAVs as burning, and (ii) is not marked as burning in belief map $\hat{\mathbf{b}}$. This definition of reward encourages UAVs to search for new burning cells, instead of monitoring cells that were already detected as burning. New burning cells typically correspond to the wildfire front, as the front corresponds to the areas towards which the wildfire propagates. It is possible that a burning cell is measured by more than one UAV. In this case, robots still receive a reward of $+1$; independently of the number of UAVs that measured that burning cell.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

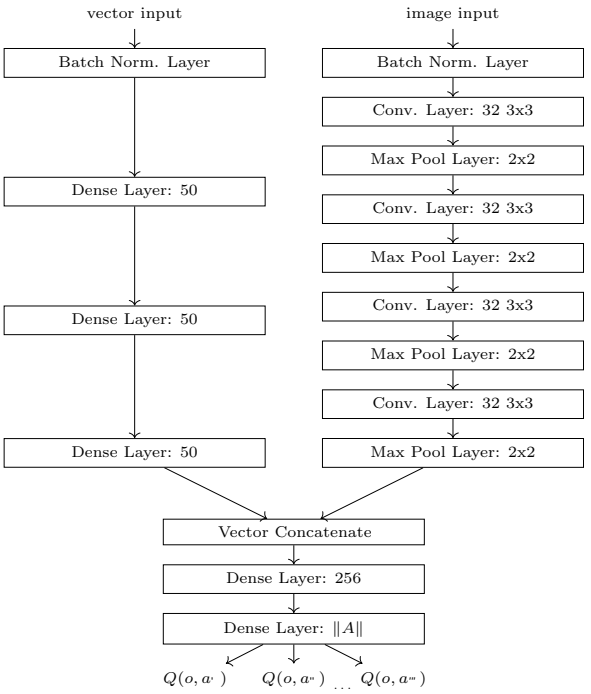


FIGURE 4: Neural network architecture.

Agents reward encourages UAVs to monitor the wildfire front. However, it does not favor, explicitly, monitoring it efficiently, which is our objective (see Section III). Note that, although not encouraged explicitly, efficiency is favored implicitly thanks to the RL discount factor [41]. This encourages agents to maximize the future expected reward as fast as possible, and, therefore, map the fire front efficiently.

D. NEURAL NETWORK ARCHITECTURE

The neural network architecture is shown in Figure 4. Vector and image observations (see Section V-B) are both fed through a batch normalization layer. On the one hand, the vector input is then processed by a dense network of three layers, with each layer containing 50 neurons. On the other hand, the image input is then processed by a convolutional network. This consists of four 3x3-convolutional layers with 32 features and stride of 1. Each layer is followed by a 2x2-max-pooling layer. The output of the dense network is concatenated with the flattened output of the convolutional network. The resulting vector is fed through a dense network, which consists of a dense layer with 256 neurons and a dense layer with one output for each possible UAV action. Note that the number of possible actions for the multi-copter and the fixed-wing model differ, as explained in Section III-B.

The output of the neural network is a Q-function value for each of the observation-action pairs. In order to calculate agents' action from the Q-values, we use a greedy action selection. This is different for each of the algorithms considered in the paper, and it is summarized in Table 3. Our framework allows us to use the same neural network architecture for the four methods considered in the paper (JOINT, INDI, MSTA and VDN), as well as for a varying number of agents and UAV models. This is achieved by simply modifying the input and output layers accordingly.

VI. SIMULATIONS AND DISCUSSION OF RESULTS

This section describes first the simulation setup that we use to validate our proposed methods. This is followed by a discussion of the results, which we organize in several subsections that focus on different aspects of our system. In particular, we analyze the following: learning progression, performance comparison against benchmarks, performance with respect to the UAV model, and scalability with the number of UAVs in the system.

A. SIMULATIONS SETUP

We consider a forest that measures $1000\text{m} \times 1000\text{m}$. The forest area is discretized in a 100×100 grid, which yields a grid cell size of $10\text{m} \times 10\text{m}$. In this forest we simulate a wildfire. For each of the simulation runs, wind and forest fuel are drawn from an uniform distribution. Wind speed is bounded by 0 and w_{max}, while wind direction takes a random cardinal direction. Fuel for each cell is bounded by f_{min} and f_{max}. We assume the initial fire as a 5×5 square located in the middle of the forest.

The fire is monitored either by a team of N multi-copter or fixed-wing UAVs, depending on the specific simulation. UAVs' position is noise-free. UAVs speed is constant and has a value of $20^{\text{m}}/_{\text{s}}$. To allow a finer control, UAVs can move five cells before the wildfire simulation gets updated ($n_{\text{fire update}}$). UAVs are equipped with a thermal camera pointing downwards that allows them to identify burning cells. We assume that the camera has a sight range $d_{sight} = 100\text{m}$, which approximately corresponds to an UAV using a thermal camera with a 53° x 38° field of view and a focal length of 8 mm while flying at a height of 100 m. UAVs starting position and orientation (for the fixed-wing model) is chosen randomly for each episode. Episode's length is $l = 320$ time steps. At the beginning of each episode, we let the fire propagate during 12 time steps before UAVs start the monitoring task.

We benchmark our two proposed cMARL methods (VDN and MSTA) against JOINT and INDI methods, which are both standard benchmarks to evaluate cMARL algorithms. Let us remark that INDI is the approached used in our reference paper [11]. For the four methods, we used our own implementation of double-Q-learning [43]. This uses an ERM that stores 800000 past transition tuples, as well as a target network. The target network is updated every $n_{\text{target update}}$ gradient steps. Exploration follows an ϵ-greedy strategy with ϵ being 1 for $e1$ frames, and then linearly annealed to reach a value of 0.1 at e_2 frames and 0 at e_3 frames. We used Adam [44] to optimize the neural network parameters. In Table 4 we summarize the parameters used for the simulations.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

Parameter	Description	Value
b	batch size	128
$n_{\text{target update}}$	target network update frequency	1000
e_1	exploration decay factor	80000
e_2	exploration decay factor	800000
e_3	exploration decay factor	3200000
γ	discount decay factor	0.9
α	Adam learning rate	0.0003
w, h	forest width and height	1000m
	forest grid cell dimensions	10m × 10m
$n_{\text{fire update}}$	fire update frequency	5
d_{max}	fire maximum ignition distance	2.5
κ	fire ignition factor	0.05
f_{max}	maximum initial fuel	20
f_{min}	minimum initial fuel	15
w_{max}	maximum wind speed	1
N	number of UAVs	$1, 2, ..., 9$
v	UAV speed	$20^{\text{m}}/_{\text{s}}$
d_{sight}	UAV sight range	$100m$
l	episode length	320

TABLE 4: Summary of simulation parameters.

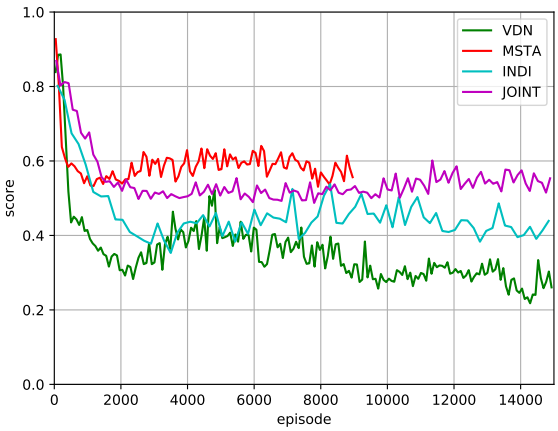


FIGURE 5: Score obtained during the training progress for the four considered cMARL methods using a fixed-wing UAV. Note that MSTA curve corresponds to a single-agent training, while the other methods were trained with three agents.

Agents training was implemented in Python using PyTorch and CUDA for GPU acceleration. We used an Intel I7 third generation CPU, a Nvidia GeForce GTX 1080 Ti GPU, and a 32GB DDR3 RAM. Trained agents were then tested in multiple simulations. Results correspond to the average score calculated over 100 test runs.

Next we evaluate in Section VI-B the learning progression, and compare in Section VI-C the four methods considered in this work. Then we evaluate in Section VI-D the performance of the methods with respect to the UAV model. Finally we analyze in Section VI-E the scalability of the methods as we increase the number of UAVs in the team.

B. LEARNING PROGRESSION

In this section we present the evolution of the score, calculated with equation (5), as the number of training episodes increases. In particular, we show training results for a one-agent and a three-agents system. The number of agents is selected to provide a fair comparison with previous work [11]. The one-agent training corresponds to MSTA, as a single-agent is trained for this method (see Section IV-C). The three-agents training is carried out for VDN, INDI and JOINT. The one-agent training converged after approx. 9000 episodes, while the three-agents one required 15000 episodes. In terms of training time, the one- and three-agents system converged after approx. 12h and 60h, respectively For this analysis, we considered the fixed-wing UAV model. The fixed-wing model has more complex dynamics, so it is harder for the network to learn it. The multi-copter dynamics are easier to learn as the network only has to learn high level commands that indicate the drone where to move next. In contrast, the fixed-wing model must learn how to control the bank angle, and the relationship between this angle and the next position. Therefore, the training complexity and time increases when using the fixed-wing model with respect to the multi-copter one.

Figure 5 depicts the learning progression of the four cMARL methods considered in this work. First of all, we would like to remark that our goal is to reach a low score. A low score implies that a little portion of the wildfire was missed by the UAVs (see Section III-C). MSTA is the fastest to converge, and it does it after approx. 1000 episodes. Training performance cannot be compared against the other methods, as only one agent is used during training, while three agents are used for the other methods. JOINT converges after 4000 episodes, followed by INDI and VDN, which require approx. 15000 episodes to converge. Although VDN requires longer to converge, it shows the best learning performance.

C. COMPARISON AGAINST BENCHMARKS

We benchmark our proposed methods (VDN and MSTA) against JOINT and INDI [11] in a wildfire monitoring tasks using pre-trained agents. Training was carried out as described in Section VI-B. Figure 6 depicts the fire miss over time, calculated with equation (4), for three fixed-wing UAVs. As in Section VI-B UAVs goal is to reach the lowest fire miss over time.

First thing we observe is that all curves present a local minima at around 50 time steps. This corresponds to the moment in which UAVs first reach and detect the fire, which significantly lowers the fire miss. From 50 steps till the end of the episode, the fire miss monotonously increases for all methods. This is essentially caused by the following: The fire size continuously increases with time, as new cells get ignited. Since agents thermal camera have a limited sight range, the number of new burning cells that are missed by the UAVs increases as the fire gets larger.

Provided a first explanation for the curves in Figure 6, we can now compare the performance of the different methods.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

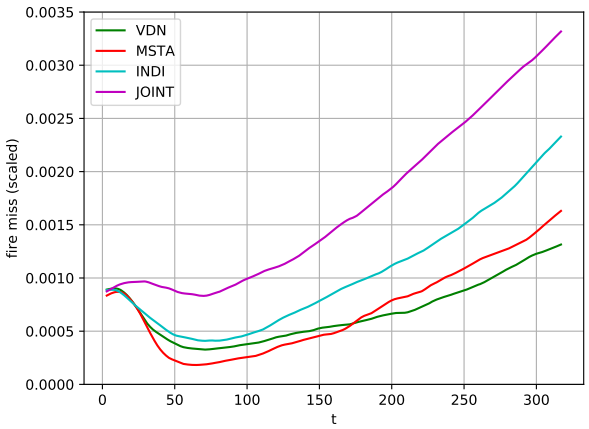


FIGURE 6: Fire miss over time for three fixed-wing UAVs. We benchmark our proposed methods (VDN and MSTA) against JOINT and INDI [11].

Score	All	Best	Intermediate values	Worst
VDN	**0.22**	**0.13**	**0.20**	0.43
MSTA	0.23	0.15	0.23	**0.32**
INDI	0.33	0.19	0.31	0.69
JOINT	0.54	0.40	0.51	0.87

(a) Fixed-wing.

Score	All	Best	Intermediate values	Worst
VDN	**0.27**	**0.16**	**0.25**	0.52
MSTA	0.27	0.20	0.27	**0.39**

(b) Multi-copter.

TABLE 5: Score obtained by the four considered methods for a system composed of three fixed-wing and multi-copter UAVs. "All" corresponds to the average score calculated over 100 tests runs. "Best" and "worst" are the average over the 10 best and worst runs, respectively. "Intermediate" corresponds to the average calculated over the remaining 80 test runs.

We can see that JOINT offers the worst performance. One would expect JOINT to outperform the rest of the methods. However, due to the bigger size of the joint action and observation spaces, compared to the other methods, the training only converged to a local minima (see Figure 5). INDI offers the second worst performance, and it is clearly outperformed by VDN and MSTA.

VDN and MSTA curves present a similar behaviour. We can nevertheless observe a small difference: MSTA performs better at the beginning of the episodes, but it gets surpassed by VDN as the episode progresses. MSTA agents learned to rapidly localize the fire, as a single-agent is sufficient to realize this task. On the contrary, VDN agents learned to cooperate to monitor the fire, but not to localize it efficiently. That is, as the training complexity of VDN is much higher than for MSTA, learning landed in a local minima. This caused that agents were not able to learn both behaviours – monitor and localize the fire – efficiently. This explains why MSTA outperforms VDN at the beginning of the episode. Once the fire grows during the episode, cooperation between UAVs acquires a higher relevance, as UAVs need to efficiently monitor the forest to find new burning cells. It is at this point when VDN outperforms MSTA, as VDN agents learned a cooperative policy.

In order to obtain a better understanding of the four considered methods, we also analyzed the best, intermediate and worst scores achieved by the methods. Results are summarized in Table 5a. Firstly, we can again conclude that VDN and MSTA share the best performance, and outperform both benchmarks (INDI and JOINT). Therefore, in the following subsections we focus our analyses on VDN and MSTA. Secondly, VDN outperforms MSTA on the best and intermediate scores, but not on the worst ones. The explanation for this behaviour is the same as the one expressed in previous paragraph. There exist complex fire behaviours that cannot be efficiently monitored neither by VDN nor by MSTA. However, MSTA might at least be able to localize the fire, while VDN might not. Just by localizing the fire, MSTA would obtains a higher score than VDN.

D. PERFORMANCE WITH RESPECT TO UAV MODEL

In this section we analyze how the performance of our proposed methods (VDN and MSTA) gets affected by the UAV model. Table 5 shows the results for a system of 3 fixed-wing (Table 5a) and multi-copter (Table 5b) UAVs. Let us remark that the fixed-wing model is more complex than the multi-copter model because the first has more complex dynamics. VDN and MSTA offer the best results for the most challenging scenario (Table 5a). Therefore, we further compare these two in a more simple scenario with a multi-copter model (Table 5b) to better understand the difference between the two algorithms.

We can observe that VDN and MSTA present a similar behaviour for the multi-copter UAVs (Table 5b), as well as for the fixed-wing UAVs (Table 5a), as we discussed in Section VI-C. However, performance for the multi-copter UAV is lower than for the fixed-wing UAV. This is because the multi-copter UAV cannot move diagonally and, therefore, moves slower than the fixed-wing UAV. Note that speed plays an important role; especially at the later stages of the episode, when the fire covers a large area. Also note that in a real world experiment speed also plays an important role, as the sensor accuracy will be influenced when flying over a certain speed. In such scenario, the UAV might need to reduce the speed when taking a picture depending on the used sensor.

We also observed that training converged faster when using a multi-copter UAV. The fixed-wing model is harder to learn because of its higher order of dynamics. When using a multi-copter UAV, actions directly translate into UAV movement commands. On the contrary, for the fixed-wing UAV, actions correspond to changes in the curvature, which

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

implies that UAV movement dynamics need to be learned. Nevertheless, VDN and MSTA were able to deal with the higher training complexity of the fixed-wing model. According to results from Table 5a, VDN and MSTA agents learned a policy that outperforms JOINT and INDI benchmarks.

E. SCALABILITY WITH NUMBER OF UAVS

We evaluate the performance of MSTA and VDN as we increase the number of UAVs in the system. In particular, we consider fixed-wing UAVs. We trained 3, 5, 7 and 9 fixed-wing agents using VDN, and a single agent using MSTA. Note that for a single agent, VDN and MSTA methods are identical.

Figures 7a and 7b depict the fire miss, calculated with equation (4), for VDN and MSTA methods, respectively. In addition, we calculate the score with (5) as we increase the number of agents from 1 to 9. This is depicted in Figure 7c, and it allows us to better analyze the performance with respect to the number of UAVs.

We can observe that the performance for VDN and MSTA significantly increases from one to three agents. For a larger number of agents, VDN performance stagnates and even decreases when 9 agents are considered. This is due to a combination of three factors: First, the problem complexity increases with a growing number of agents, as the coordination and spurious reward problem become more significant (see Section IV-B). In consequence, training might converge to a local minima. Second, since the neural network size is constant for a growing number of agents, it is possible that the network is not large enough to deal with the increasing complexity. Third, the linear approximation of the joint Q-function, in which VDN is based on (see Section IV-C), might be a inaccurate for our problem.

In contrast to VDN, MSTA performance steadily increases with a growing number of agents. This is plausible, since additional MSTA agents simply imply adding independent UAV's that were individually trained to fly towards the fire and circle it. Although MSTA agents do not explicitly cooperate, more new burning cells are discovered by a larger number of UAVs, which improves MSTA performance.

According to simulation results, we can conclude that VDN is the best alternative for a system that is composed of up to 3 fixed-wing UAVs. For a larger number of UAVs, MSTA offers the best performance. Nevertheless, we strongly believe that VDN has the potential to outperform MSTA for a wildfire monitoring task with multiple UAVs. In this respect, we outline in Section VII promising directions.

VII. CONCLUSION

In this paper, we developed a cooperative multi-agent reinforcement learning framework to learn a wildfire monitoring task with UAVs. In particular we proposed the use of multiple single-trained Q-learning agents (MSTA) and value decomposition networks (VDN) for this task. As baseline, we used independent Q-learners (INDI) and a joint Q-learner (JOINT). Both methods are standard baselines in the literature. Moreover, independent Q-learners were also used on a wildfire monitoring task [11].

We carried out simulations using a stochastic wildfire model and two types of UAVs: a fixed-wing and a multicopter UAV. First, we compared MSTA and VDN against the considered baselines using a system composed of 3 fixed-wing UAVs. Results showed that MSTA and VDN clearly outperform the two state-of-the-art baselines.

Simulations also allowed us to conclude that VDN performs better at the end of an episode, when the fire behaviour is more complex and inter-UAV coordination plays a big role. In contrast, MSTA shows a better performance at the beginning of an episode, when the main objective is to localize the fire and coordination is not that important. Moreover, MSTA shows a smaller variance in the overall score between different episodes, in comparison with VDN. This indicates that MSTA learned a robust behaviour, while VDN offers a less stable behaviour due to a higher learning complexity.

We also showed that MSTA scales better than VDN for a team of more than 3 UAVs. This could be due to the linear approximation used by VDN to approximate the joint Q-function.

VIII. FUTURE WORK

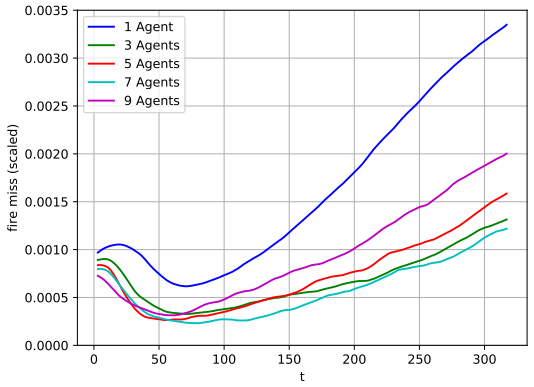
In order to solve the scalability problem of the VDN method, a possible solution could be to use a nonlinear learned approximation, as QMIX [40] does. Additionally, we strongly believe that the use of curriculum learning [45] could be of great advantage. We could design a curriculum that first trains a robust single-agent policy. In a latter stage, inter-agent cooperation could be learned with VDN for an increasing number of agents. Another promising approach to improve VDN performance is to exploit communication between agents' policies. Communication can be realized using predefined communication schemes [46], or letting agents learn how to communicate [47].

Communication is typically imperfect and is restricted. In this scenario, it might be necessary to compress the data. One possible approach is to use models that approximate the fire front based on the already taken measurements. The information of the individual models can be shared using parameter consensus techniques, reducing the amount of data that needs to be transmitted. In addition, models could interpolate between measurements, providing a better coverage of the fire.

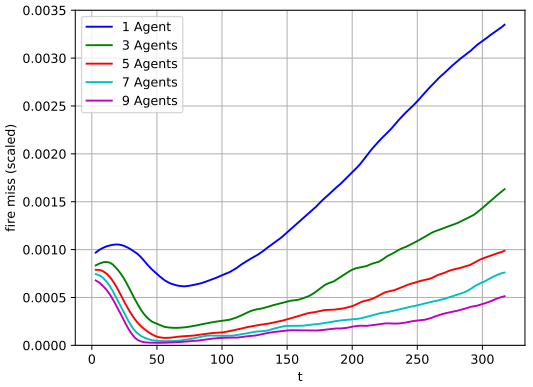
In this work, UAVs only obtain information from their surroundings. We believe that having a global view of the environment could help to improve the performance of the system. To this extent, we could incorporate extra channels with down-sampled information that cover a larger area but with lower resolution.

A natural next step is to perform experiments with real UAVs to observe how the uncertainties of the real system affects the network. In addition, performing such experiments during controlled burns will provide a better understanding of how the controller is able to extrapolate from simulations

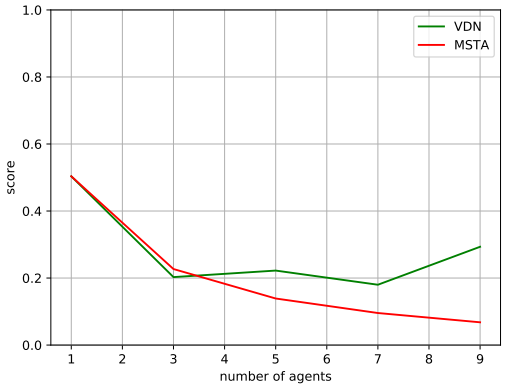
This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access



(a) VDN.



(b) MSTA.



(c) VDN vs. MSTA

FIGURE 7: VDN and MSTA scalability as we increase the number of fixed-wing UAVs in the system. On the top part, we depict the fire miss over an episode. On the bottom part, we show the algorithms' score for a growing number of UAVs.

to a real scenario. In this direction, first studies were carried out in [48].

REFERENCES

[1] A. M. Gill, S. L. Stephens, and G. J. Cary, "The worldwide "wildfire" problem," Ecological applications, vol. 23, no. 2, pp. 438–454, 2013.

[2] H. X. Pham, H. M. La, D. Feil-Seifer, and M. Deans, "A distributed control framework for a team of unmanned aerial vehicles for dynamic wildfire tracking," in 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2017, pp. 6648–6653.

[3] C. Yuan, Y. Zhang, and Z. Liu, "A survey on technologies for automatic forest fire monitoring, detection, and fighting using unmanned aerial vehicles and remote sensing techniques," Canadian journal of forest research, vol. 45, no. 7, pp. 783–792, 2015.

[4] F. A. Hossain, Y. Zhang, and C. Yuan, "A survey on forest fire monitoring using unmanned aerial vehicles," in 2019 3rd International Symposium on Autonomous Systems (ISAS), 2019, pp. 484–489.

[5] N. K. Ure, S. Omidshafiei, B. T. Lopez, A. Agha-Mohammadi, J. P. How, and J. Vian, "Online heterogeneous multiagent learning under limited communication with applications to forest fire management," in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 5181–5188.

[6] R. N. Haksar and M. Schwager, "Distributed deep reinforcement learning for fighting forest fires with a network of aerial robots," in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 1067–1074.

[7] Z. Lin and H. H. Liu, "Topology-based distributed optimization for multi-uav cooperative wildfire monitoring," Optimal Control Applications and Methods, vol. 39, no. 4, pp. 1530–1548, 2018. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/oca.2424

[8] F. Afghah, A. Razi, J. Chakareski, and J. Ashdown, "Wildfire monitoring in remote areas using autonomous unmanned aerial vehicles," in IEEE INFOCOM 2019 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2019, pp. 835–840.

[9] A. Sargolzaei, A. Abbaspour, and C. D. Crane, Control of Cooperative Unmanned Aerial Vehicles: Review of Applications, Challenges, and Algorithms. Cham: Springer International Publishing, 2020, pp. 229–255. [Online]. Available: https://doi.org/10.1007/978-3-030-34094-0_10

[10] S. Chung, A. A. Paranjape, P. Dames, S. Shen, and V. Kumar, "A survey on aerial swarm robotics," IEEE Transactions on Robotics, vol. 34, no. 4, pp. 837–855, 2018.

[11] K. D. Julian and M. J. Kochenderfer, "Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning," Journal of Guidance, Control, and Dynamics, pp. 1–11, 2019.

[12] C. Boutilier, "Planning, learning and coordination in multiagent decision processes," in Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge. Morgan Kaufmann Publishers Inc., 1996, pp. 195–210.

[13] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," PloS one, vol. 12, no. 4, p. e0172395, 2017.

[14] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls et al., "Value-decomposition networks for cooperative multi-agent learning," arXiv

preprint arXiv:1706.05296, 2017.

[15] L. Tang and G. Shao, "Drone remote sensing for forestry research and practices," Journal of Forestry Research, vol. 26, no. 4, pp. 791–797, 2015.

[16] J. Q. Cui, S. K. Phang, K. Z. Ang, F. Wang, X. Dong, Y. Ke, S. Lai, K. Li, X. Li, F. Lin et al., "Drones for cooperative search and rescue in post-disaster situation," in 2015 IEEE 7th International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE Conference on Robotics, Automation and Mechatronics (RAM). IEEE, 2015, pp. 167–174.

[17] F. Yamazaki, K. Kubo, R. Tanabe, and W. Liu, "Damage assessment and 3d modeling by uav flights after the 2016 kumamoto, japan earthquake," in 2017 IEEE International Geoscience and Remote Sensing Symposium (IGARSS). IEEE, 2017, pp. 3182–3185.

[18] P. G. Martin, S. Kwong, N. Smith, Y. Yamashiki, O. D. Payton, F. Russell-Pavier, J. S. Fardoulis, D. Richards, and T. B. Scott, "3d unmanned aerial vehicle radiation mapping for assessing contaminant distribution and mobility," International journal of applied earth observation and geoinformation, vol. 52, pp. 12–19, 2016.

[19] S. M. Adams and C. J. Friedland, "A survey of unmanned aerial vehicle (uav) usage for imagery collection in disaster research and management," in 9th International Workshop on Remote Sensing for Disaster Response, vol. 8, 2011.

[20] D. Giordan, Y. Hayakawa, F. Nex, F. Remondino, and P. Tarolli, "the use of remotely piloted aircraft systems (rpass) for natural hazards monitoring and management," Natural hazards and earth system sciences, vol. 18, no. 4, pp. 1079–1096, 2018.

[21] L. Hua and G. Shao, "The progress of operational forest fire monitoring with infrared remote sensing," Journal of forestry research, vol. 28, no. 2, pp. 215–229, 2017.

[22] R. S. Allison, J. M. Johnston, G. Craig, and S. Jennings, "Airborne optical and thermal remote sensing for wildfire detection and monitoring," Sensors, vol. 16, no. 8, p. 1310, 2016.

[23] T. J. Zajkowski, M. B. Dickinson, J. K. Hiers, W. Holley, B. W. Williams, A. Paxton, O. Martinez, and G. W. Walker, "Evaluation and use of remotely piloted aircraft systems for operations and research–rxcadre 2012," International Journal of Wildland Fire, vol. 25, no. 1, pp. 114–128, 2016.

[24] S. Lacroix, R. Alami, T. Lemaire, G. Hattenberger, and J. Gancet, "Decision making in multi-uavs systems: Architecture and algorithms," in Multiple heterogeneous unmanned aerial vehicles. Springer, 2007, pp. 15–48.

[25] D. W. Casbeer, R. W. Beard, T. W. McLain, S.-M. Li, and R. K. Mehra, "Forest fire monitoring with multiple small uavs," in Proceedings of the 2005, American Control Conference, 2005. IEEE, 2005, pp. 3530–3535.

[26] P. Sujit, D. Kingston, and R. Beard, "Cooperative forest fire monitoring using multiple uavs," in 2007 46th IEEE Conference on Decision and Control. IEEE, 2007, pp. 4875–4880.

[27] M. Kumar, K. Cohen, and B. HomChaudhuri, "Cooperative control of multiple uninhabited aerial vehicles for monitoring and fighting wildfires," Journal of Aerospace Computing, Information, and Communication, vol. 8, no. 1, pp. 1–16, 2011.

[28] N. K. Ure, S. Omidshafiei, B. T. Lopez, A. Agha-Mohammadi, J. P. How, and J. Vian, "Online heterogeneous multiagent learning under limited communication with applications to forest fire management," in 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, pp. 5181–5188.

[29] L. Merino, F. Caballero, J. R. Martínez-de Dios, J. Ferruz, and A. Ollero, "A cooperative perception system for multiple uavs: Application to automatic detection of forest fires," Journal of Field Robotics, vol. 23, no. 3-4, pp. 165–184, 2006.

[30] H. X. Pham, H. M. La, D. Feil-Seifer, and A. Nefian, "Cooperative and distributed reinforcement learning of drones for field coverage," arXiv preprint arXiv:1803.07250, 2018.

[31] Y. Xiao, J. Hoffman, T. Xia, and C. Amato, "Multi-robot deep reinforcement learning with macro-actions," arXiv preprint arXiv:1909.08776, 2019.

[32] A. Viseras and R. Garcia, "Deepig: Multi-robot information gathering with deep reinforcement learning," IEEE Robotics and Automation Letters, vol. 4, no. 3, pp. 3059–3066, 2019.

[33] J. Foerster, N. Nardelli, G. Farquhar, T. Afouras, P. H. Torr, P. Kohli, and S. Whiteson, "Stabilising experience replay for deep multi-agent reinforcement learning," in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 1146–1155.

[34] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch, "Emergent tool use from multi-agent autocurricula," arXiv preprint arXiv:1909.07528, 2019.

[35] L. Buşoniu, R. Babuška, and B. De Schutter, "Multi-agent reinforcement learning: An overview," in Innovations in multi-agent systems and applications-1. Springer, 2010, pp. 183–221.

[36] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," arXiv preprint arXiv:1312.5602, 2013.

[37] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in International Conference on Autonomous Agents and Multiagent Systems. Springer, 2017, pp. 66–83.

[38] Y. Li, "Deep reinforcement learning," arXiv preprint arXiv:1810.06339, 2018.

[39] J. N. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[40] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Qmix: monotonic value function factorisation for deep multi-agent reinforcement learning," arXiv preprint arXiv:1803.11485, 2018.

[41] R. S. Sutton and A. G. Barto, Reinforcement learning: An introduction. MIT press, 2018.

[42] F. A. Oliehoek, "Decentralized pomdps," in Reinforcement Learning. Springer, 2012, pp. 471–503.

[43] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in Thirtieth AAAI conference on artificial intelligence, 2016.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[45] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 41–48.

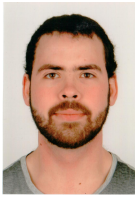
[46] J. R. Kok and N. Vlassis, "Collaborative multiagent reinforcement learning by payoff propagation," Journal of Machine Learning Research, vol. 7, no. Sep, pp. 1789–1828, 2006.

[47] J. Foerster, I. A. Assael, N. de Freitas, and S. Whiteson, "Learning to communicate with deep multi-agent reinforcement learning," in Advances in Neural Information Processing Systems, 2016, pp. 2137–2145.

[48] A. Viseras, J. Marchal, M. Schaab, J. Pages, and L. Estivill, "Wildfire monitoring and hotspots detection with aerial robots: Measurement campaign and first results," in 2019 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR). IEEE, 2019, pp. 102–103.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/ACCESS.2021.3055651, IEEE Access

ALBERTO VISERAS studied electrical engineering at the University of Malaga in Spain, and received his PhD in 2018 from the University Pablo de Olavide in Seville, Spain. During his PhD he carried out a research stay at the University of Sydney, where he investigated algorithms for the coordination of multiple aerial gliders. Alberto is since 2013 a researcher at the Institute of Communications and Navigation at the German Aerospace Center (DLR). His research interests focus on machine learning methods for coordination of multiple robots. In particular, he leads a research line on reinforcement learning methods for multi-robot exploration, with a special focus on Unmanned Aerial Vehicles (UAVs). Alberto is involved in national and European projects that investigate the use of UAVs for emergency management.



MICHAEL MEISSNER studied mechanical engineering at the Technical University of Munich, where he finished his Master's in 2019. His focus studying was control theory, as well as machine learning and embedded systems development. In his master's thesis at the Institute of Communications and Navigation at the German Aerospace Center (DLR), he addressed reinforcement learning for multi-UAV coordination in the context of wildfire monitoring.

JUAN MARCHAL studied Electronics Engineering at the University of Almeria in Spain, and finished his Master studies in 2017 at the University of Malaga in Spain. In 2016 he started as a Master student at the Institute of Communications and Navigation at German Aerospace Center (DLR), where he is currently working as a Research Associate. His research interests focus on multi-robot coordination using reinforcement learning and multi-robot area coverage using UAVs. Juan is involved in European projects that investigate the use of UAVs for emergency management.