Cooperative Deep Reinforcement Learning for Dynamic Pollution Plume Monitoring using a Drone Fleet

Mohamed Sami Assenine, Walid Bechkit, Ichrak Mokhtari, Hervé Rivano, Karima Benatchba

▶ **To cite this version:**

Mohamed Sami Assenine, Walid Bechkit, Ichrak Mokhtari, Hervé Rivano, Karima Benatchba. Cooperative Deep Reinforcement Learning for Dynamic Pollution Plume Monitoring using a Drone Fleet. IEEE Internet of Things Journal, 2023, pp.1-14. 10.1109/JIOT.2023.3328242 . hal-04316013

HAL Id: hal-04316013
https://inria.hal.science/hal-04316013

Submitted on 30 Nov 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Cooperative Deep Reinforcement Learning for Dynamic Pollution Plume Monitoring using a Drone Fleet

Mohamed Sami Assenine[1,2,a], Walid Bechkit[1,a], Ichrak Mokhtari[1,a], Hervé Rivano[1,a], and Karima Benatchba[2,b]

[1]Univ Lyon, INSA Lyon, Inria, CITI, EA3720, 69621 Villeurbanne, France

[2]École nationale Supérieure d'Informatique, Laboratoire de Méthodes de Conception des Systèmes, Algiers, Algeria

[a]{mohamed-sami.assenine,walid.bechkit,ichrak.mokhtari,herve.rivano}@insa-lyon.fr, [b]k_benatchba@esi.dz

***Abstract*—Monitoring pollution plumes is a key issue, given the harmful effects they cause. The dynamic of these plumes, which may be important due to meteorological conditions, makes their study difficult. Real-time monitoring in order to obtain an accurate mapping of the pollution dispersion is helpful and valuable to mitigate risks. In this work, we consider a fleet of cooperative drones carrying pollution sensors and operating in order to assess a pollution plume. The latter is assumed to follow a Gaussian Process (GP) with varying parameters. For this use case, we propose an efficient approach to characterize spatially and temporarily the plume while optimizing the path planning of drones. In our approach, drones are guided by a Deep Reinforcement Learning (DRL) model called Categorical Deep Q-Network (Categorical DQN) to maximize the plume coverage while considering budget constraints. Specifically, we develop a scalable Independent Q-Learning (IQL) scheme that shares team rewards based on each drone's deployment relevance and therefore ensures cooperation. We evaluate the performance of the plume parameter estimation as well as the maps generated by the GP regression. By testing our framework on several plume scenarios, we show that it offers good results in terms of both estimation quality and run-time efficiency.**

***Index Terms*—Drone fleet, Pollution plume monitoring, Reinforcement Learning, Independent Q-Learning, Gaussian Process, Multi-agent Informative Path Planning.**

I. Introduction

Within the smart cities paradigm, urban monitoring, particularly for the assessment of toxic, explosive, or irradiating pollutants, plays a central role in ensuring urban safety and well-being. This task involves the particular use case of timely pollution plume mapping following accidents or deliberate releases, which potentially may have a drastic effect on both human health and environment. In these situations, conventional static sensors and dispersion models fall short due to the unknown pollution sources and release rates, the unpredictable pollution dynamics, the inflexibility of static measurements solutions, and the strong need for quick responses [1]. Addressing this issue requires leveraging real-time data collection, advanced sensor networks, machine learning algorithms, and interconnected urban systems to help improve the speed and efficiency of response in accidental pollutant release situations [2].

Mobile sensing is emerging as a powerful means of efficiently gathering crucial environmental data, particularly in emergency situations where human intervention can be dangerous. This effectiveness is largely attributed to the recent democratization of mobile platforms, such as land rovers and drones, coupled with rapid advances in small and low-cost sensors [3], [4]. Furthermore, leveraging the capabilities of the Internet of Things (IoT) on these mobile devices allows establishing the fundamental infrastructure needed to address our challenges. By outfitting these mobile platforms with wireless communication capabilities, one can enable the creation of an IoT network that communicates with ground-based stations [5]. Equipped with adequate environmental sensors, this IoT network provides real-time data and remote monitoring capabilities, facilitating swift and secure responses [5]. Nevertheless, a common limitation of these sensor-equipped mobile robots is their dependence on batteries, which intrinsically limits their range of movement. To tackle this challenge in vast areas, deploying multiple collaborating robots is necessary for rapid characterization of the studied phenomenon [4], [6].

In this paper, we develop a Multi-Agent Reinforcement Learning (MARL) [7] scheme to optimize the mobility of a fleet of robots while characterizing an air pollution plume as well as possible. Without loss of generality, we consider in the rest of this paper a fleet of drones. The objective of the fleet is to collaborate in order to plan the most informative trajectories over a target area while respecting the budgetary constraints of each drone. We address two fundamental issues. The first issue is to design and evaluate an effective model for the instantaneous 2D mapping of a dynamic phenomenon, while combining sensor measurements [3] with an online spatio-temporal learning model [8], [9]. The second underlying issue concerns the optimal trajectory planning for a fleet of drones, in a large area, in order to collect the most informative measurements on the phenomenon and consequently improve the maps. This problem is called Multi-agent Informative Path Planning (MIPP) [6], [10].

In the following, we propose to model the pollution plume as a Gaussian Process (GP) [11]. The latter have been widely adopted to characterize the distribution of spatial data (ex. seawater salinity [8], [9], WiFi signal strength [3], [6], etc.).

With appropriate hyperparameters and an adequate kernel, the GP can be used to model processes of varying degrees of regularity and estimate concentrations at any location [8]. In our case, finding the relationship between positions on a map and pollutant concentrations is challenging due to the fact that these variables interact in a nonlinear fashion with each other. The underlying mechanism is complicated and difficult to be adequately captured by traditional regression analysis. In light of that, our objective is to develop a statistical procedure, which captures a pollution plume by using the least measurement effort. In this work, the pollution plume assumes the form of a GP, which is highly flexible and able to capture practically any continuous functional relationships. GP is chosen over other powerful nonlinear models because of its statistical inference capability [11], which allows for uncertainty quantification and provides spatio-temporal predictions.

Based on GP, Mutual Information (MI) [12] is proposed as a criterion to measure the informativeness of a path when data is collected by a mobile agent along it. While making use of MI theory to define rewards, we propose a trajectory planning algorithm based on Deep Reinforcement Learning (DRL) [13] for the guidance of several drones in the pollution area. These paths allow improving and extending the knowledge about the plume. Concretely, we use a Categorical Deep Q-Network (DQN-C51) [14] as a DRL framework for each drone agent. We opt for Independent Q-Learning (IQL) [15] where each learning agent (drone) has its own neural structure. However, team rewards are shared between them. To allow each agent to learn and update its neural network, we use a reward sharing mechanism according to each agent's contribution to the global team reward. For dynamic environments, the accuracy of the GP mapping degrades over time because it does not incorporate the temporal variation of the phenomenon. To solve this problem, we re-estimate the hyperparameters repeatedly at appropriate times. This mechanism for triggering the re-estimate depends on the intensity of the temporal variations of the phenomena to follow.

To summarize, here are the main contributions of our work:

- We propose a timely pollution plume characterization solution allowing both (i) a mapping based on GP regression and the idea of continuous online learning using drone data (ii) an adaptive estimation of the underlying GP parameters based on a weighted forgetting mechanism.
- We design an IQL-based MARL model using Categorical DQN for drone trajectory planning while proposing a hybridization of the latter with a random approach to improve performance by increasing exploration in certain novel situations.
- We conduct extensive simulations to validate our overall framework while considering both stable and time-varying parameters. The results show that our solution enables rapid convergence, scaling up, and adaptation to changes in the studied phenomenon's parameters.

The rest of this article is organized as follows. Section II presents previous work related to our theme. The problem of dynamic air pollution monitoring by a fleet of drones driven by a reinforcement learning algorithm is then formulated in section III. Further in section IV, we present the basic notions on reinforced learning, GPs and entropy theory for the benefit of path planning as well as a dispersion model of a plume rise. Section V describes the strategy put out to solve the considered problem, including information on its various components and the advanced assumptions. Section VI elaborates the validation methodology and the experimental setup. In the section VII, we present the evaluation results by showing the effectiveness of our framework while section VIII concludes this article and outlines future work.

II. Related Work

Most of the existing work defines trajectory planning for surface inspection as an offline coverage planning problem that seeks to identify a path that can fully cover the surface [16]. In fact, trajectory planning is a multi-objective optimization problem with multiple constraints where we seek to find a sequence of translation and rotation from a start position in order to reach an end point or try to cover an area of interest as much as possible while avoiding obstacles in the environment and respecting the imposed constraints [17].

A. General path planning approaches

According to the survey [17], we can categorize approaches solving conventional path planning problems as classical algorithms and more recent heuristic algorithms. The first class includes graph exploration through breadth-first traversal, depth-first traversal, Dijkstra's algorithms and A*. In the same class, we can also find the Potential Field technique [18] and the tree-building target search algorithm called Rapidly-exploring Random Trees (RRT) [19]. The second class of approaches contains most recent algorithms, such as nature-inspired algorithms (like Genetic Algorithms, Particle Swarm Optimization...) and Fuzzy Logic techniques [20] combined generally with machine learning.

B. RL based path planning approaches

Recently, Reinforcement Learning (RL) methods have appeared in this field thanks to their simplicity of implementation and their satisfactory results [4]. The problem of finding the optimal path to reach a target position has been widely studied in the literature and several solutions have been proposed, such as PRIMAL and PRIMALc [21]. In the context of our work, we do not detail these methods because they are not directly applicable in our case where the objective is to achieve a good spatio-temporal estimate of a physical phenomenon, not to find a target location in 3D space. Aydin et al. [22] have proposed a technique based on the integration of Deep Q-Network (DQN) into particles of a Particle Swarm Optimization (PSO) algorithm. The purpose of this method is to train the drones which have a circular coverage of communication not to move too far from each other and not to overlap to ensure optimal coverage. A problem arises when two particles are moving at the same time because they cannot predict what the other particles will do, leading to proximity issues. In situations

with dynamic phenomena, it is judicious to move the drones continuously and at the same time to follow the phenomenon in real time. This constrains the use of this approach in our use case moreover for our mapping no coverage radius exists.

C. GP regression based path planning approaches

In this paragraph, we discuss solutions that come close to our use case. In [9], authors presented a trajectory planning method for autonomous underwater vehicles in order to maximize information on sea salinity. Assuming GP modeling and use of Mutual Information (MI) theory, a Recursive Greedy algorithm was utilized to generate near-optimal paths while guaranteeing that the vehicle stays out of high traffic zones for predetermined time intervals. The Recursive Greedy algorithm used is faster than exhaustive search of course, but it is still time-consuming for medium to large-sized areas. Their approach involves a rough discretization of possible waypoints for a glider, but computational time still limits the scalability. This disadvantages this method moreover that it relies on a simple heuristic algorithm, which may not perform well with complicated problems. In [6], the authors studied the problem of planning efficient paths for several mobile robots in order to collect, thanks to a RL framework, spatial data on the strength of the Wi-Fi signal inside a room. The challenge was to adapt single-agent RL models to a system with several cooperative agents. For this, the authors proposed two methods of sharing rewards to allow agents to learn independently from each other while collaborating. The spatial phenomenon followed was modeled by a GP that produced signal maps that were remarkably accurate, showing promising results. Given the static nature of the phenomenon being seen, offline spatial mapping has been successful in producing satisfactory results. However, in cases involving dynamic phenomena, such as a pollution plume, it is essential to use an online data collection. In a recent work, Zhu et al. [16] presented an informative online path planning approach for actively gathering information on three-dimensional surfaces using a single drone. They assume that the phenomenon follows manifold Gaussian Processes (mGPs) with geodetic kernel functions to map surface information fields and plan informative paths in an active way. The challenge in this work was to combine their proposed solution with a global path planner to try to escape local minima. The findings of this study show that their online informative path planning method outperforms traditional approaches such as 3D coverage planning and random exploration in terms of information-theoretic metrics. Using an online data collection approach as in the last article, we will map the pollutant concentrations on a given surface considering a GP and a spatial kernel function. Then, we use Multi-agent Deep RL to construct informative paths while the authors of [16] used a single-agent Sequential Greedy Waypoint Search heuristic to maximize time-averaged information gain.

III. Problem Statement

In this work, we consider drones that are deployed over an area of interest in order to characterize and monitor a pollution plume in real time. For this, we consider the case of a fleet of drones equipped with adequate environmental sensors, Global Positioning System (GPS), processing capabilities and a wireless communication system. In order to ensure a better characterization of the phenomenon in a timely manner while minimizing drone mobility, our approach should continuously combine the generation of maps of the monitored phenomenon, made possible by spatio-temporal characterization model, with optimized trajectory planning algorithms.

The challenge behind the design of such a pollution plume monitoring system consists of two closely related sub-issues. The first concerns the mapping of the phenomenon to understand its evolution over time and the second consists of the proper use of the knowledge acquired on the phenomenon to better plan the trajectories allowing the generation of more precise maps of the aforementioned phenomenon.

We assume that the spatial domain, where we study the plume, is discretized into a finite number of V measurement locations. For each subset $\mathcal{A} \subseteq V$, we note $f(\mathcal{A})$ the quality of measurement, i.e. the informative character of the observation of the plume at the $\mathcal{A}$ locations. We also associate to each location $v \in V$, a cost $C_s(v) > 0$ quantifying the cost of sensing a measure on this location. When a drone travels between two locations, u and v, it incurs a moving cost $C_m(u, v) > 0$.

A drone traverses a path $\mathcal{P}$ from v_1 to v_l means that $\mathcal{P}$ is a sequence of l locations starting with node v_1 and arriving at v_l. The total cost $C(\mathcal{P})$ of traversing the path $\mathcal{P} = (v_1, v_2, \cdots, v_l)$ is the sum of the sensing costs at each location and the costs of moving between each two consecutive ones, i.e. $C(\mathcal{P}) = \sum_{i=1}^{l} C_s(v_i) + \sum_{i=2}^{l} C_m(v_{i-1}, v_i)$.

For a collection of N paths $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_N\}$, one for each drone, $f(\mathcal{P}) = f(\mathcal{P}_1 \cup \cdots \cup \mathcal{P}_N)$ quantifies the quality of the measurements collected on all N paths. The solution to our problem is to find a collection $\mathcal{P}^*$ of N paths, with known starting locations, which are the most informative, i.e., the value of $f(\mathcal{P}^*)$ is maximized. This while considering that the path taken by the i-th drone $\mathcal{P}_i$ has a bounded cost $C(\mathcal{P}_i) \leq \mathbf{b}^i$ where $\mathbf{b}^i$ is the maximum budget of the i-th drone.

More formally, we define our problem as a Multi-agent Informative Path Planning (MIPP) which we can represent by a quadruple $\langle G, \mathbf{v}_s, \mathbf{b}, f_D(\mathcal{P}) \rangle$ such that:

- $G = (V, E)$: is the graph representing the area to be monitored. V is the set of vertices representing the points of interest and E is the set of edges between these points.
- $\mathbf{v}_s$: is the vector of the initial locations of each of the N drones.
- $\mathbf{b}$: is the vector of the initial budgets of each of the N drones. It also stores their remaining budgets during an episode of flying.
- $f_D(\mathcal{P})$: is the informativeness function, which measures the usefulness and the new knowledge learned about the plume by measuring the pollution concentrations on the points of interest of the set of paths $\mathcal{P}$. We propose and discuss in section IV-C the definition of our informativeness function. D refers to previously collected measurements and are initially used to estimate the hyperparameters of the GP model.

Mathematically, the objective of an MIPP is written as shown by the following equation:

$$\begin{aligned} &\mathcal{P}^* = \underset{\mathcal{P}}{\operatorname{argmax}}\ f_D(\mathcal{P}),\ s.t.\ C(\mathcal{P}_i) \leq \mathbf{b}^i, \\ &\mathcal{P}_i[0] \in \mathbf{v}_s^i,\ \forall\, i \in \{1, \cdots, N\} \end{aligned} \tag{1}$$

Here $\mathcal{P}_i[0]$ designates the starting node of the drone i.

IV. Background

As we will detail in the following, we consider a pollution plume following a Gaussian Process (GP) and we will take advantage of Reinforcement Learning (RL) to achieve our objectives. In what follows, we present a background on the basic notions used in our work.

A. Reinforcement Learning and Q-Learning

RL is a type of machine learning where an agent must learn on its own to perform a task by taking actions while interacting with an uncertain and dynamic environment [13]. The basic principle is not to direct the agent towards which action to choose, but rather to leave it alone to discover which actions yield the most rewards, in the short and long term, by trying them [13].

Generally, RL problems are modeled by Markovian Decision Processes (MDP) for decision making in discrete, stochastic and sequential environments. Each time the agent observes a state, it performs an action, which results in rewards to be maximized. A MDP problem is represented by a tuple (S, A, R, p), where S is the underlying state space, A is the set of actions, $R : S \times A \times S \rightarrow \mathbb{R}$ is the immediate reward function which associates to each transition tuples (current state s, action taken a, next state s') a real number evaluating the relevance of the chosen action, and $p(s'|s, a)$ is the transition function which gives the probability of passing to the state s'. The agent's objective, at time t, is to maximize the accumulated rewards G_t also called "Discounted Return".

$$G_t = \sum_{k=0}^{T-t} \gamma^k \cdot r_{t+k} \tag{2}$$

$T \in \mathbb{N} \cup \{\infty\}$ is the last time step corresponding to the last action, r_t is the reward for the action taken at time t and γ is a reduction factor (discount factor) which ensures the convergence of the return G_t. Maximizing the return amounts to finding the most close to optimal policy, π^*, which achieves the maximum expected return of all states,

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ Q^\pi(s, a)\,, \tag{3}$$

where Q^π is called value function and $Q^\pi(s, a)$ is called Q-value of state-action pair (s, a) under the policy π. Formally $Q^\pi(s, a) = \mathbb{E}[G_t|S_t = s, A_t = a]$ and it allows us to quantify how good it is to choose the action a in a state s [23].

RL algorithms are often classified into Policy-based and Value-based approaches [24]. One of the most used algorithms is Q-Learning [25] which is part of the Value-based classification. This algorithm is based on a tabular and iterative approach, which seeks to best estimate the Q-values according to the following formula with a learning rate α.

$$Q_{t+1} \longleftarrow (1 - \alpha) \cdot Q_t(s, a) + \alpha[r_t + \gamma \cdot \max_{a'} Q_t(s', a')] \tag{4}$$

The number of Q-values increases exponentially with respect to the size of the action space and therefore classical Q-Learning is not very adequate for our problem. A nonlinear representation that maps both state and action to a value can be used to solve this problem. One of the most recent solutions is the use of deep neural networks to approximate value functions [23], [26]. This method is known as Deep Q-Network (DQN).

B. Deep Q-Network and the categorical variant

The authors in [27] have proposed a solution for a simple and efficient implementation of DQN. This latter is based on Q networks (neural networks) allowing to approach the value function. Usually the most used action picking policy is called ϵ-greedy. In fact, following this policy, the learning agent chooses a random action with a probability ϵ otherwise the action corresponding to the largest Q-value returned by the Q Network is chosen. This policy allows finding a trade-off between exploration and exploitation of the action space. Note that the value of ϵ can be fixed or variable throughout the training progress. In the same article [27], the authors proposed a novel approach to train DQNs. Instead of feeding the neural network with successive transition tuples and immediately training it on them, these transitions are stored in a huge buffer called Experience Replay Memory (ERM) or Replay Buffer (RB) capable of storing hundreds of thousands of transitions. The agent can now randomly sample transitions from the ERM to form mini-batches and train the DQN with them. Training such a Q network amounts to minimizing a cost function called TD Loss. It is calculated by averaging over a mini-batch sampled from the ERM as follows:

$$Loss = \mathbb{E}\left[\left(\underbrace{r_t + \gamma \cdot \max_{a'} Q_\theta(s_{t+1}, a')}_{\text{Target}} - \underbrace{Q_\theta(s_t, a_t)}_{\text{Prediction}}\right)^2\right] \tag{5}$$

where (s_t, a_t, s_{t+1}, r_t) is a transition tuple belonging to the mini-batch used for training, γ is the reduction factor and θ is the neural network parameters. In fact, the cost function as defined in 5 is very unstable. So instead of calculating it with the current θ parameters, the authors suggest calculating it with an older version of the neural network called target network with the θ^- parameters. The target network is updated approximately every tens of iterations with the learned parameters θ so the cost function becomes more stationary. This method is called Double Deep Q-Network (DDQN).

In this paper, we exploit the categorical variant of DQN called DQN-C51 [14]. C51 is a Q-Learning algorithm which can be used on any environment with a discrete action space. The main difference between C51 and classical DQN is that instead of just predicting the Q-value of each state-action pair, C51 predicts a histogram model for the probability distribution of each Q-value. By learning the distribution, C51 is able

to remain more stable during training, which improves the final performance [28]. An important parameter to consider in categorical algorithms is the number of atoms. This represents the number of support points in the probability distribution estimates. As one can see, from the name C51, the default atom number is 51 which gives a good granularity of Q-values support points [14], [28].

C. Gaussian Process and entropy-based path planning

Informative path planning is a special case of general path planning problems where the reward of a path is defined by the informativeness of the data collected along it. In information theory, informativeness can be measured by Mutual Information (MI) [12]. We will introduce, in the following, the computation of the informativeness of a path $\mathcal{P}$, denoted $f(\mathcal{P})$, based on GPs and MI. More details concerning GPs, entropy and MI can be found in [3], [6], [11] and [29].

Given a graph where the set of vertices is denoted by V and for each node $v \in V$ of coordinates $\mathbf{x}_v$, the corresponding data (e.g. pollutant concentrations) is denoted by $\mathbf{y}_v$. Assuming the data to be collected can be modeled by a GP, this means that the data $\mathbf{y}_V$ at all locations V follow a multivariate joint Gaussian distribution, whether note:

$$\mathbf{y}_V \sim \mathcal{N}\left(\begin{bmatrix} m(\mathbf{x}_1) \\ \vdots \\ m(\mathbf{x}_n) \end{bmatrix}, \begin{bmatrix} k(\mathbf{x}_1,\mathbf{x}_1) & \cdots & k(\mathbf{x}_1,\mathbf{x}_n) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_n,\mathbf{x}_1) & \cdots & k(\mathbf{x}_n,\mathbf{x}_n) \end{bmatrix}\right) \tag{6}$$

where $m(\cdot)$ is the mean function, $k(\cdot,\cdot)$ is the kernel and $n = |V|$ is the total number of vertices. For simplicity, we denote the multivariate Gaussian distribution by $\mathcal{N}(m(X_V), \sum_V)$, where X_V is a $n \times 2$ matrix of the 2D coordinates of V and $\sum_V$ is the covariance matrix $n \times n$ as defined by the kernel function [6]. Among the most used kernels in the literature, There is the exponential covariance kernel (called RBF) where:

$$k(\mathbf{x}_p, \mathbf{x}_q) = \sigma_f^2 \cdot exp\left(-\frac{||\mathbf{x}_p - \mathbf{x}_q||^2}{2l^2}\right), \tag{7}$$

such that σ_f^2 is the common variance of $\mathbf{y}_V$ and l is the length scale. The variance controls the maximum covariance between variables, and the length scale plays the role of an attenuation coefficient of the correlation function according to the Euclidean distance between the points p and q.

The continuous differential entropy of $\mathbf{y}_V$ is given by:

$$H(\mathbf{y}_V) = \frac{1}{2}\ln|\sum\nolimits_V| + \frac{n}{2}(1 + \ln(2\pi)) \tag{8}$$

Given $\mathcal{P}$ the set of paths taken by each of the drones, the latter are supposed to collect data on each of the points along their paths. We denote all sample locations by X_S and the corresponding measurements by $\mathbf{y}_S$. The posterior distribution of $\mathbf{y}_V$ given $\mathbf{y}_S$ is denoted $\mathcal{N}(\mu', \sum')$ where [30]:

$$\begin{aligned}\mu' = {} & m(X_V) + K(X_V, X_S) \cdot (K(X_S, X_S) + \sigma_n^2 I)^{-1} \\ & \cdot (\mathbf{y}_S - m(X_S))\end{aligned} \tag{9}$$

$$\begin{aligned}\sum{}' = {} & K(X_V, X_V) - K(X_V, X_S)[K(X_S, X_S) + \sigma_n^2 \cdot I]^{-1} \\ & \cdot K(X_S, X_V)\end{aligned} \tag{10}$$

Here σ_n represents the underlying Gaussian noise variance and $K(A, B)$ is the kernel matrix generated by $k(\cdot,\cdot)$ with pairwise entries in sets A and B. Then, the conditional differential entropy of $\mathbf{y}_V$ given $\mathbf{y}_S$ is then given by:

$$H(\mathbf{y}_V|\mathbf{y}_S) = \frac{1}{2}\ln|\sum\nolimits'| + \frac{n}{2}(1 + \ln(2\pi)) \tag{11}$$

The informativeness of an MI-based $\mathcal{P}$ path set can be calculated as follows.

$$\begin{aligned} f(\mathcal{P}) &= MI(\mathbf{y}_V; \mathbf{y}_S) \\ &= H(\mathbf{y}_V) - H(\mathbf{y}_V|\mathbf{y}_S) \end{aligned} \tag{12}$$

The intuition behind MI is that it rewards more locations that most significantly reduce uncertainty over locations where no action has been taken. It is interesting to notice that the entropy depends only on the covariance matrix. Therefore, the informativeness can be calculated offline analytically without traversing the paths and taking the measurements. However, in order to estimate the hyperparameters σ_f, l and σ_n of the RBF kernel (formulas 7, 9, 10), a small amount of pilot data is required $D = (X_D, \mathbf{y}_D)$. Given the set D, informativeness can be calculated as the following:

$$\begin{aligned} f_D(\mathcal{P}) &= MI(\mathbf{y}_V; \mathbf{y}_S \cup \mathbf{y}_D) \\ &= H(\mathbf{y}_V) - H(\mathbf{y}_V|\mathbf{y}_S \cup \mathbf{y}_D) \end{aligned} \tag{13}$$

D. Gaussian dispersion model for pollution plume

The theoretical study of the atmospheric dispersion of pollutants is mainly based on the theory of fluid mechanics [31]. In this work, we focus on dispersion models of steady-state plume rise. In order to simulate realistic average pollution concentrations and without loss of generality, we consider the basic Gaussian plume dispersion model [1]. Assuming that the wind direction is along the x axis and that the pollutant is released at point $(0, 0, h_s)$ in free space, this model allows computing the pollutant concentration at a point (x, y, z) using the following formula [32], [33]:

$$C(x,y,z) = \frac{Q}{2\pi u \sigma_y \sigma_z} e^{\left(\frac{-y^2}{2\sigma_y^2}\right)} \left[e^{\left(\frac{-(z-H_e)^2}{2\sigma_z^2}\right)} + e^{\left(\frac{-(z+H_e)^2}{2\sigma_z^2}\right)}\right] \tag{14}$$

With:

- C : Concentration of pollutants (g/m^3).
- Q : Emission rate at source (g/s).
- u : Wind speed (m/s).
- σ_y : Horizontal dispersion coefficient ($\sigma_y = a_y \cdot |x|^{b_y}$).
- σ_z : Vertical dispersion coefficient ($\sigma_z = a_z \cdot |x|^{b_z}$).
- H_e : Effective pollutant release height (m).

The effective height of the pollutant release H_e is equal to the sum of the pollution source height h_s and the elevation of the plume Δh. Briggs formula [32], [33] is commonly used for calculating Δh. In the case where the temperature of the pollutant T_s is higher than that of the ambient air T_a, Briggs' formula giving the value of Δh is as follows:

$$\begin{cases} \Delta h = \dfrac{1.6\, F^{1/3} \cdot x^{2/3}}{u} \\ F = \dfrac{g}{\pi} \cdot V_f \cdot \dfrac{T_s - T_a}{T_s} \end{cases} \tag{15}$$

With:

- F : Buoyancy (m^4/s^3).
- x : Distance from the source in direction of the wind (m).
- g : Gravity constant $(\approx 9.8\, m/s^2)$.
- V_f : Volume flow (m^3/s).

V. Methodology

Before presenting the details of our solution, we first present the modeling of the target area on which the pollution plume evolves and secondly, the set of assumptions that we have made to facilitate the development of our framework.

We assume that the geographical area to be monitored is discretized into a regular graph (grid or lattice) as shown by the example of Figure 1. The numbered vertices represent all the points of interest where the measurements and the estimates of the pollution concentrations are made. The drones move from one vertex to an adjacent one taking measurements on the vertices, while hovering.

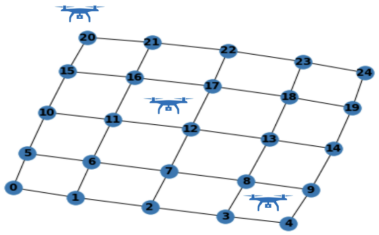

Fig. 1. Regular grid of size 5×5 representing a target area

Although we focus our work on air quality monitoring, our framework can be adapted for the assessment of any physical phenomenon. An important assumption is to assume that this phenomenon follows a Gaussian Process (GP). These processes have proven effective in modeling several phenomena such as water temperature [34], humidity [35] and pollution [36], [37].

We consider a fleet of connected drones which must be deployed as soon as the alert is received and collaborate in order to monitor the polluting plume in real time. This article focuses on the design and development of two bricks: mapping of pollutant concentrations and anticipated trajectory planning. Our framework works in an iterative way such that an iteration begins by taking environmental measurements using the various sensors embedded in the drones. From current measurements and possibly those of the past, a GP regression is able to generate a map modeling the state of the monitored phenomenon as well as the variance corresponding to the concentration estimates. Based on these maps as well as the current positions of the drones, we propose to use a Reinforcement Learning (RL) model to calculate the appropriate new positions for each drone and plan the optimal trajectories for them. The measurement procedure is repeated when the drones have moved to the following positions, and so on until their budget is exhausted. A drone's budget indicates its maximum travel distance. As shown in Figure 2, our system consists essentially of four blocks working together to correctly and quickly characterize a pollution plume by finding the pollutant concentrations in each point of the grid $\mu = (\mu_1, \cdots, \mu_n)$, the variance (σ_f^2) and the length scale (l).

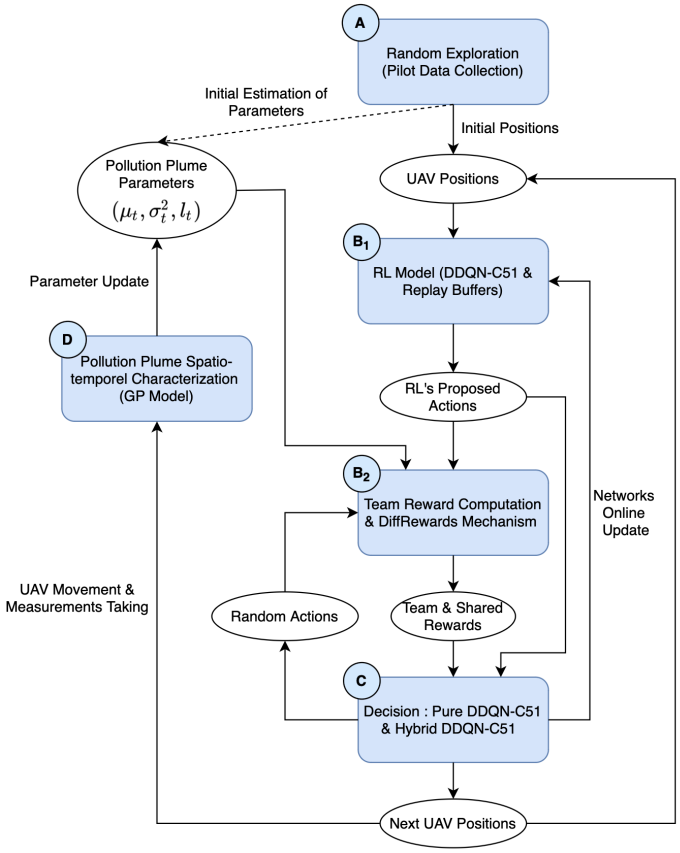

Fig. 2. Overall diagram of our solution

At the beginning, the drones perform a random exploratory flight of a few steps over the target area (Block A). This will allow collecting some measurements which are used to generate an initial estimation of pollution plume parameters. To better adjust these parameters characterizing the plume, we need to guide the drones in their movements in order to intelligently collect the most informative information about the plume and consequently update optimally the parameter estimates. For this, we propose an RL model based on Independent Q-Learning (IQL) with a reward sharing mechanism (Block B). The team reward is calculated using the Mutual Information (MI). We note that the reward of the RL model is based on the pollution plume parameters and this to determine the team incentives, which reflect the rates of uncertainty reduction in the plume's pollution concentrations. To keep the right compromise between exploration and exploitation, we propose a hybridization between a random choice of actions and the actions returned by the RL model (Block C). For this, we compare the immediate rewards of the two action choice strategies, and we take the actions corresponding to the best reward. Finally, the GP model (Block D) is updated by the new measurements collected by the drones in order to take into consideration, in addition to the spatial aspect, the potential temporal evolution of the plume parameters. In the following subsections, we will detail each of the four previous blocks.

A. Random exploration for pilot data collection

Before starting RL learning for path planning, we need to have some knowledge of the pollutant concentrations at some locations on the grid. This knowledge is called Pilot Data and it is collected following a random exploration of the drones during a few stages in the area of interest. Using this pilot data, we generate an initial parameter estimate and an approximate map of the pollution plume.

B. RL for an optimal path planning

For the learning scheme, our model is based on an IQL [38]. This form of learning allows each agent to learn and choose its actions independently of the others. This approach is not as independent as it appears. In fact, each of the agents takes into consideration the position of the other agents before deciding on the action to be taken. In addition, the environment gives a common reward to the whole team. This reward is the reduction of the uncertainty on the plume after the displacement of each agent towards its new position.

Our IQL scheme contains as many RL models as there are agents. Each agent model is similar to the model presented in Figure 3 where the inputs are the coordinates. These last feed a fully connected neural network of type DDQN-C51 which is detailed in IV-B. The last layer outputs the distributional diagrams of possible actions. For the choice of the best action, i.e. the next jump node, we use the policy ϵ-greedy which allows choosing with an ϵ probability a random action otherwise the best action is taken.

Formally, the agent state and the set of possible actions are defined as follows:

- State of an agent (drone): The state of an agent corresponds to the set of 2D coordinates of each of the N agents of the framework.
- Set of actions: The actions available at time t for agent i are the neighboring vertices of its current location. Given the differences between agents in terms of budgets, some may finish sooner than others. To handle this, we add a dummy action to the action space of an agent who has exhausted its budget. When all agents finish, a training episode ends. Given the current position of the i-th agent $v_t^i \in V$, the valid actions are given by the following formula:

$$A_t^i(v_t^i) = \{v \in \text{Adj}(v_t^i) \,|\, C_m(v_t^i, v) + C_s(v) \leq \mathbf{b}_t^i\}, \quad (16)$$

  where "Adj" represents adjacent vertices. C_m, C_s and $\mathbf{b}_t$ are respectively the moving cost, the sensing cost and the remaining budgets at time t, as defined in section III.

When the actions set of all agents is executed, the environment sends a team reward signal. This team reward r_t at time t is equal to the difference between the informative function (refer to the subsection IV-C) of the set of paths at time t ($\mathcal{P}_t$) and this set at the previous time $t-1$ ($\mathcal{P}_{t-1}$) as shown by the formula 17. This reward can be expressed, also, as the decrease in uncertainty of the measurements collected at time $t-1$ after collecting new data at time t, see formula 18.

$$r_t = f_D(\mathcal{P}_t) - f_D(\mathcal{P}_{t-1}) \quad (17)$$

$$r_t = H(\mathbf{y}_V | \mathbf{y}_{\mathcal{P}_{t-1}} \cup \mathbf{y}_D) - H(\mathbf{y}_V | \mathbf{y}_{\mathcal{P}_t} \cup \mathbf{y}_D) \quad (18)$$

This reward is distributed among all agents according to a mechanism of reward sharing also called credit assignment [39]. Among the most widespread sharing mechanisms [40], [41], we opt for sharing in relation to the marginal contribution of each agent in the team reward, namely the DiffRewards [6]. Let η be the function defined by: $S \times A \times R \longrightarrow \mathbb{R}^N$ associating to each (s_t, u_t, r_t) a vector of partial rewards $[r_t^1, \cdots, r_t^N]$. In particular, the reward of an agent i at time t can be calculated by the following formula:

$$r_t^i = H(\mathbf{y}_V | \mathbf{y}_{\mathcal{P}_t^{-i}} \cup D) - H(\mathbf{y}_V | \mathbf{y}_{\mathcal{P}_t} \cup D) \quad (19)$$

where $\mathcal{P}^{-i}$ designates the set of all the paths of the agents except that of the agent i and this in order to be able to evaluate its contribution. Since the sum of these partial rewards does not necessarily equal the total reward r_t ($\sum_{i=1}^N r_t^i \neq r_t$), then we must normalize them as shown the (20).

$$r_t^i \leftarrow r_t^i \cdot \frac{r_t}{\sum_{j=1}^N r_t^j} \quad (20)$$

The transition tuples containing the states, partial rewards, and actions taken by each agent will be saved in their experience buffers $\mathcal{M} = \{\mathcal{M}^i \,|\, i \in \{1, \cdots, N\}\}$. A random sample of these tuples is used after in the computation of the TD Loss function (see formula 5) which allows adjusting the weights of the different neural networks. Algorithm 1 presents the proposed IQL scheme based on the DDQN-C51 model and DiffRewards mechanism for sharing rewards.

C. Hybrid DDQN-C51

In addition to the DDQN-C51 model, we also propose a different iteration of the latter called Hybrid DDQN-C51. As we will show in VII, this version allows performance improvements especially in some new situations not seen before by the DDQN-C51 model where random movement can give better gains. The principle of the Hybrid DDQN-C51 approach is to simulate the execution of the following two cases:

- random actions,
- actions proposed by the already trained DDQN-C51.

Then we compute the team reward corresponding to each of the two prior cases and we select the set of actions that yield the best reward.

We note that this hybrid approach is used during the inference while deploying the drones. However, in a DDQN-C51, the consideration of random actions is only conducted during training to discover new applicable actions. During inference, this model only exploits and therefore executes actions returned by the neural network.

D. Spatio-temporal characterization of a pollution plume

As presented before, our objective is to estimate the pollution plume parameters. In what follows, we present our approach to estimate these parameters taking into account their potential temporal evolution.

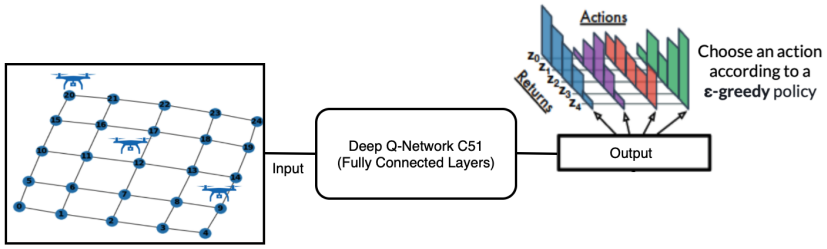

Fig. 3. Global single-agent Deep RL scheme

Algorithm 1: Multi-agent IQL scheme based on DDQN-C51 model and DiffRewards

Data: $\langle G, \mathbf{v}_s, \mathbf{b}, f_D(\cdot)\rangle$, N : number of agents, H : entropy function, K : target network update period.
Result: Neural Network parameters θ.

initialization of replay buffers $\mathcal{M} = \{\mathcal{M}^i \,|\, i \in \{1, \cdots, N\}\}$
initialize $\theta = \{\theta^i \,|\, i \in \{1, \cdots, N\}\}$ randomly and set $\theta^- = \theta$
initialization of the maximum size of an episode T
for *episode* $e = 1, 2, ...$ **do**
  get initial global state s_0
  for *step* $t = 1, 2, ..., T$ **do**
    for *agent* $i = 1, 2, ..., N$ **do**
      get actual position of i-th agent v_t^i
      with a probability ϵ, choose a random action $a_t^i \in A_t^i(v_t^i)$ (formula 16)
      else choose $a_t^i = \text{argmax}_a \; Q^{\theta^{i^-}}(s_t, a)$
    execute action $\mathbf{a}_t = (a_t^1, a_t^2, ..., a_t^N)$
    calculate, based on H, the team reward r_t (formula 18)
    calculate, based on r_t, the reward of each agent $[r_t^1, ..., r_t^N]$ (formulas 19 and 20)
    store each transition tuple $(s_t, a_t^i, r_t^i, s_{t+1})$ in $\mathcal{M}^i$
    for each agent i sample a batch of $(s_j, a_j^i, r_j^i, s_{j+1})$ from $\mathcal{M}^i$ and update θ^i by minimizing the TD Loss
  update $\theta^- = \theta$ with some period K
return θ

*1) **GP regression for spatial mapping**:* GP regression with an exponential spatial kernel is used in our general framework to generate pollution maps. Furthermore, we need to feed this GP regression with some pollutant concentrations and the parameters it needs. Using this information and based on the maximum likelihood estimation combined with k-fold cross-validation, it performs a spatial regression over the entire target area [11].

*2) **Consideration of temporal dynamics**:* For phenomena with changing characterizing parameters, the accuracy of the GP regression degrades over time because it does not integrate the temporal variation of the phenomenon. To solve this problem, we use the mechanism of re-estimating hyperparameters repeatedly at appropriate times [8]. The mechanism for triggering the re-estimate depends on two factors. The first factor stems from computational complexity. In fact, any re-estimate is a regression and its calculation is costly. The second factor concerns the dynamics of spatio-temporal variations. Indeed, the repetitive re-estimates of the hyperparameters of the kernel function must reflect the intensity of the temporal variation of the phenomenon.

*3) **Variance and length scale estimator**:* In order to estimate these two parameters, we propose an iterative method while including a partial forgetting mechanism with weighting so that the values remain up to date and follow the variations of the plume.

In our framework, the drones continuously move from one step to the next, hovering above each node to assess the measurements of pollutants there. Classically, to calculate the variance of the measurements at a given point, we save, each time, the new measurement taken on this point and recalculate the mean and variance based on this history. To improve the performance in terms of space memory and computation time, we propose to calculate the latter as follows. Suppose that at time t, a drone is on the point $v \in V$ and takes a $(m+1)$th measurement $\mathbf{y}_v$, the new average $\mathbb{E}_t(v)$ as well as the new variance $\mathbb{V}_t(v)$ in this point are given as the following:

$$\begin{cases} \mathbb{E}_t(v) = \dfrac{m \cdot \mathbb{E}_{t-1}(v) + \mathbf{y}_v}{m+1} \\ \\ \mathbb{V}_t(v) = \dfrac{m \cdot (\mathbb{V}_{t-1}(v) + \mathbb{E}_{t-1}^2(v)) + \mathbf{y}_v^2}{m+1} - \mathbb{E}_t^2(v) \end{cases} \tag{21}$$

In order to adapt the variance to changes in the dynamics of a phenomenon, we propose a partial and weighted mechanism for forgetting old measurements. For this, it is necessary to give more priority and importance to recent measures compared to old ones. We then introduce a forgetting factor γ_f often close to 1 to the formulas 21 in order to implement this mechanism, see formulas 22.

$$\begin{cases} \mathbb{E}_t(v) = \gamma_f \cdot \mathbb{E}_{t-1}(v) + (1-\gamma_f) \cdot \mathbf{y}_v \\ \\ \mathbb{V}_t(v) = \gamma_f \cdot (\mathbb{V}_{t-1}(v) + \mathbb{E}_{t-1}^2(v)) + (1-\gamma_f) \cdot \mathbf{y}_v^2 - \mathbb{E}_t^2(v) \end{cases} \tag{22}$$

To further simplify the calculations, we center the measures with respect to the estimate of their means. With this centered

values, we directly update the variance as shown in the following:

$$\sigma^2_{f_t} = \mathbb{E}\left[\gamma_f \cdot \sigma^2_{f_{t-1}} + (1-\gamma_f)(\mathbf{y}_v - \mathbb{E}_t(v))^2 \,|\, v \in \mathcal{V}_t\right] \quad (23)$$

where $\mathcal{V}_t$ is the set of points visited at time t by the N agents.

In order to estimate the covariance at time t between the two points u and v where we have just collected the measurements $\mathbf{y}_u$ and $\mathbf{y}_v$ respectively, we propose to follow the same idea of weighted forgetting as for the variance, which results in the following:

$$\mathbb{C}_t(u,v) = \mathbb{C}_{t-1}(u,v) + \mathbb{E}_{t-1}(u)\cdot\mathbb{E}_{t-1}(v) + (1-\gamma_f)\cdot\mathbf{y}_u\mathbf{y}_v - \mathbb{E}_t(u)\cdot\mathbb{E}_t(v) \quad (24)$$

Then, to deduce the length scale, we use the inverse function of the covariance function (formula 7), which gives the following formula:

$$l_t = \mathbb{E}\left[||u,v|| \cdot \sqrt{-\frac{1}{2}\cdot \ln^{-1}\left(\frac{\mathbb{C}_t(u,v)}{\sigma^2_{f_t}}\right)}\right],\ \forall\ (u,v) \in V^2 \quad (25)$$

VI. Validation Methodology and Experimental Setup

We recall that the phenomenon studied is modeled by a GP. In order to validate our solution, we first set up a simple simulator of pollution plume dispersion. As a proof of concept and without loss of generality, we opted for the use of the basic Gaussian dispersion plume model, defined above in the section IV-D. In the following subsections, we will further detail the plume simulation and experimental setup.

A. Pollution plume simulation

For the simulation of a real dispersion of pollutants over an area, we consider a geographical area in rectangular form. This area is discretized into a regular rectangular grid of size 21×9. For the generation of a plume realization, we used the pollutant dispersion model that we present in IV-D and which is based on the Gaussian dispersion formula of fluid mechanics. In the table I, we summarize all the parameters used by the formula 14 in order to create our simulation of a plume with three identical pollutant sources. Figure 4 provides the heat map of the pollution plume simulated.

To generate, from the plume presented in Figure 4, several other realizations over time, we consider a random function based on a multivariate Gaussian law taking the real covariance matrix of the studied plume that we search to estimate.

B. Experimental setup

We implement the proposed framework on an Apple M1 Max chip with 64GB of memory using Python as a main programming language as well as the TensorFlow (TF) platform and the following principal libraries:

General parameters		
Grid size of the target area	21×9	
Number of agents (N)	$\{1, 2, 3, 5, 8, 10\}$	
Distance between two adjacent nodes	$50\,m$	
Forgetting factor (γ_f)	0.95	
Pollution plume simulation parameters		
Pollutant source height (h_s)	$25\,m$	
Emission rate at source (Q)	$1.59\,g/s$	
Wind speed (u)	$5\,m/s$	
Ambient temperature (T_a)	$288.15\,K$	
Pollutant temperature (T_s)	$303.15\,K$	
Volume flow (V_f)	$6.03 \times 10^{-10}\,m^3/s$	
Horizontal dispersion coefficients	$a_y = 0.34, b_y = 0.82$	
Vertical dispersion coefficients	$a_z = 0.275, b_z = 0.69$	
RL parameters		
Size of an episode (d)	60 steps	
Number of training steps	$15000 - 40000$ depending on the number of agents	
Learning rate (α)	1×10^{-4}	
Reduction / Discount factor (γ)	0.99	
DDQN target network update period	5 steps	
Gradient descent optimization method	RMSProp	
Neural structure of the hidden layers ($j \times i-$unit layer)	$N \leq 3$:	3×64-unit layer
	$N = 5$:	4×64-unit layer
	$N > 5$:	5×64-unit layer

TABLE I
Summary of some general and other parameters used for the simulation of a pollution plume and RL

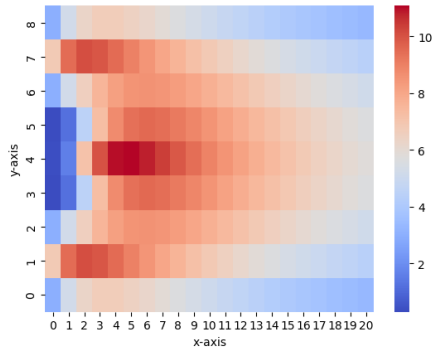

Fig. 4. Heat map of pollutant concentrations (in mg/m^3) of the plume over the target area presented as a 21×9 rectangular grid.

- **GPy**[1] **(v1.10):** a GP framework implementing the basic GP spatial regression as well as several other variants. It allows to compute the predictions and their uncertainties.
- **TF-Agents**[2] **(v0.11):** a TF library dedicated to RL offering a wide variety of modular components.

[1]https://github.com/SheffieldML/GPy.
[2]https://www.tensorflow.org/agents?hl=en.

Method	Episode inference run time in seconds					
	1 agent	2 agents	3 agents	5 agents	8 agents	10 agents
Random	11.18	12.04	14.15	21.75	24.87	33.20
DDQN-C51	11.82	12.94	16.11	22.55	26.86	34.86
Hybrid DDQN-C51	14.70	16.93	20.22	33.33	39.40	55.13
TSH	17.30	32.50	114.85	3481.62 ($\approx$ 0.97h)	12h $\ll$	-

TABLE II
COMPARISON OF EPISODE INFERENCE RUN TIMES OF THE METHODS USED WITH A DIFFERENT NUMBER OF AGENTS

VII. PERFORMANCE EVALUATION

In this section, we analyze the overall performance of the two main bricks that make up our solution, namely: the RL model and the parameter estimation. First of all, we start with the RL model and the two variants that we propose. Indeed, we compare these models with two baseline approaches to solve the MIPP problem using the uncertainty reduction metric. Secondly, we consider the task of monitoring a pollution plume over the area as represented by Figure 4. The results show that our proposed methods of gathering useful measurements can produce very accurate estimates of the plume's parameters. Finally, we present a series of tests that highlight the way our approach responds to changes in the phenomenon and adapts to the timely-varying nature of the plumes.

A. Evaluation of RL models

The first tests we conducted aim to validate the effectiveness of our RL models (DDQN-C51 and Hybrid DDQN-C51) compared to a model that makes the drones move randomly (Random) and a near-optimal heuristic model that we call Tree Search Heuristic (TSH). TSH is inspired by Monte Carlo Tree Search algorithm [42], [43]. The idea behind the TSH is to find a solution to the MIPP problem that is close to the optimal by exploring the tree of possible solutions as much as possible. Searching for the exact solution, one must navigate a tree with d depth levels (corresponding to the size of an episode) where the total number of leafs is exactly equal to $a^{d \times N}$ where N is the number of agents and a is the number of possible actions of each agent. This makes an extremely high number of possible solutions. This is why we opt for a pruning-based heuristic, selecting at each step some of the best combinations of actions maximizing the reward function.

Before the validation, we train our models using several initial configurations with different numbers of drones mobilized. The environment's features and the setup used in our many tests are outlined in the table I. We test all the proposed methods over 30 episodes by averaging the results. These latter are summarized in Figures 5 and 6.

From Figure 5, we notice that the DDQN-C51 method is more efficient, on average, by 40.42% than the random method and can even in some cases exceed the 50% threshold. The Hybrid DDQN-C51 method increases the performance by 7.22%. This improved RL method manages to approach, on average, 85.44% of the total reward found by the TSH method in the different situations. TSH needs an exponentially growing time to explore the tree (see table II), even with a greedy method that takes only the best combination at each step. It is not relevant for real deployment. On the other hand, the RL approach finds a good compromise between performance and inference time. In Figure 6, we show a comparison between a DDQN-C51 and a Hybrid DDQN-C51 with initial setups containing more agents than Figure 5.

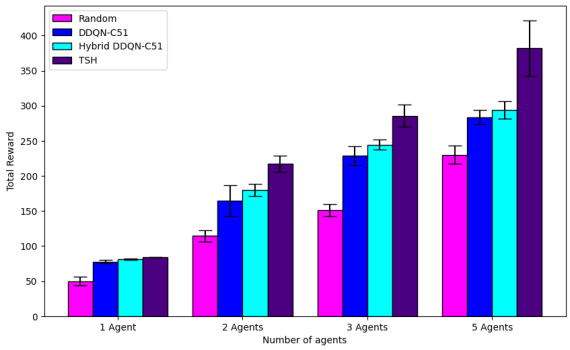

Fig. 5. Comparison between the Random, DDQN-C51, Hybrid DDQN-C51 and TSH methods with several configurations of the number of agents.

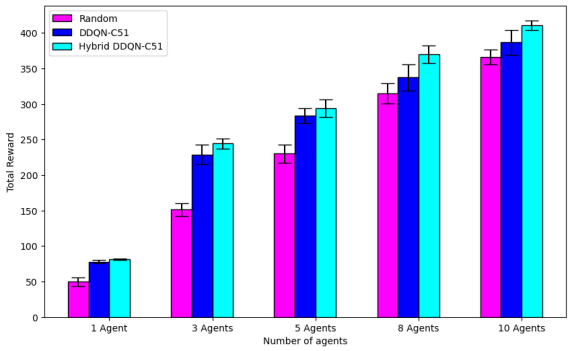

Fig. 6. Comparison between the DDQN-C51 and the Hybrid DDQN-C51 with several configurations of the number of agents mobilized.

To see in more detail the performance of our RL models, we consider the case of a system with 5 agents to perform the rest of the tests. The results represented by the Figures 7, 9, 11 and 12 show both the means and the 95% confidence intervals. Our framework initially approximates the values of the plume parameters. These initial values appear as starting values at times 0 in Figures 9, 11, 12 and 14.

In Figure 7, we have plotted the curves of decreasing uncertainty on the plume during a drone flight episode. We notice that our two RL models (Figures 7-(2) and 7-(3)) are doing quite well better than random (Figure 7-(1)) and close to

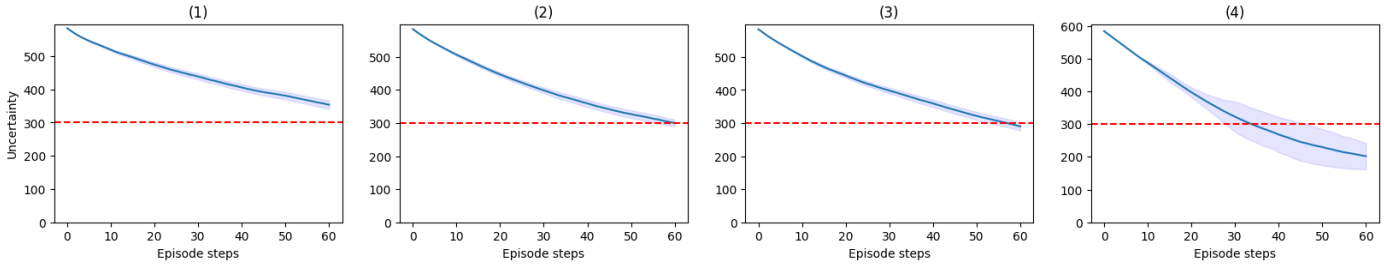

Fig. 7. Curves and 95% confidence intervals of the decrease in uncertainties during a complete drone flight episode. (1): case of a random solution, (2): case of a solution based on a DDQN-C51 method, (3): case of a solution based on a Hybrid DDQN-C51 method, (4): case of a solution based on the TSH heuristic.

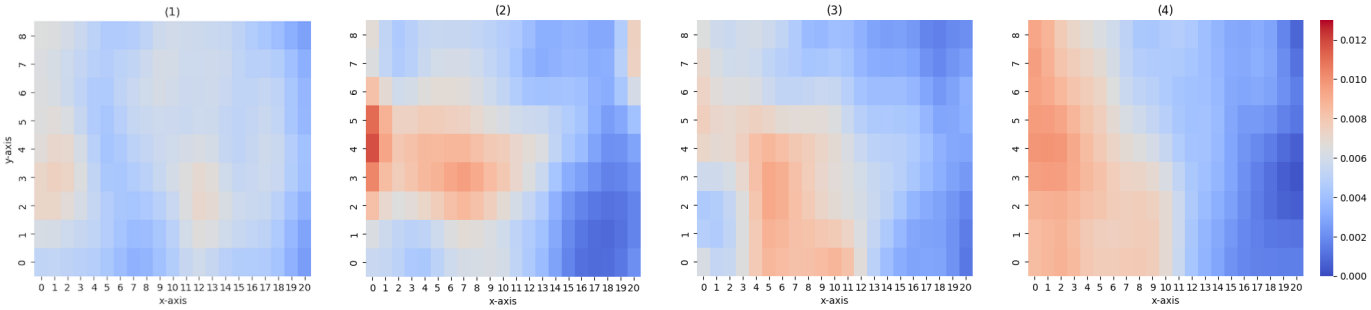

Fig. 8. Heat maps of drone locations after a full flight episode. (1): case of a random solution, (2): case of a solution based on a DDQN-C51 method, (3): case of a solution based on a Hybrid DDQN-C51 method, (4): case of a solution based on the TSH heuristic.

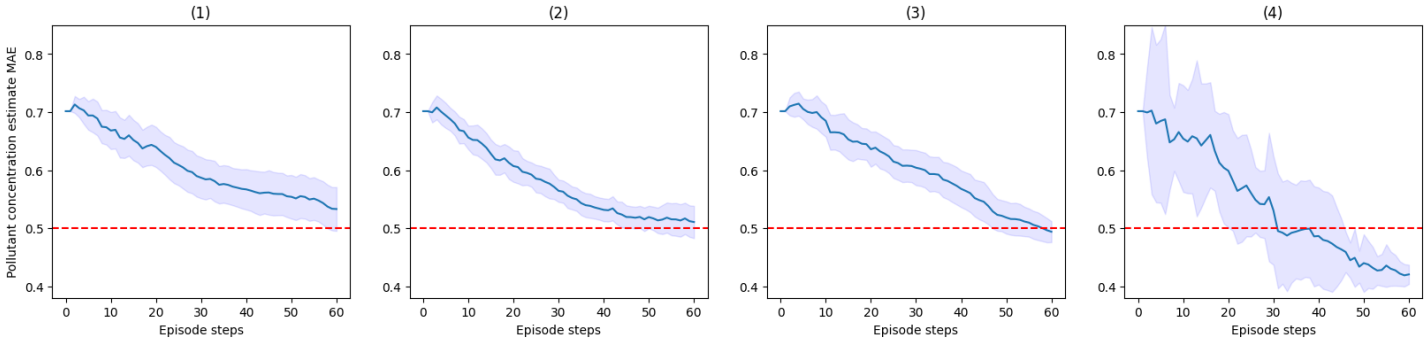

Fig. 9. The curves and the 95% confidence intervals of the Mean Average Error (MAE) of the concentrations of pollutants during a complete episode of drone flight. (1): case of a random solution, (2): case of a solution based on a DDQN-C51 method, (3): case of a solution based on a Hybrid DDQN-C51 method, (4): case of a solution based on the TSH heuristic.

TSH (Figure 7-(4)). We also note the fact that the confidence intervals of the results returned by our models are small. Therefore, our solution is stable and converges regardless of where the drones start on the map.

Figure 8 illustrates the different locations of the drones after a complete flight episode. To better interpret these movement heat maps, we determined the correlation between a node's number of visits and its corresponding pollutant concentrations. In the case of the random solution (Figure 8-(1)) the correlation is -0.009. It is very weak which is logical because we expect an equiprobable distribution on the whole map. For the RL solutions (Figures 8-(2) and 8-(3)), the correlation is a little higher (0.27) but it still remains low. The area coverage reaches averages of 80% of the entire area to be monitored. On the other hand, the results of the TSH solution are almost similar to those of the RL, in terms of correlation (0.23) and movement heat map (Figure 8-(4)) with an average area coverage of 90%.

B. *Computation efficiency*

In addition to total rewards, an important evaluation metric is the computation time. In this paragraph, we give the time complexities of the different path planning inference methods that we have used. For RL-based schemes, we focus on the computational time of path inference, since policies are trained only once beforehand.

- Random path inference time complexity:

$$O(d \cdot N) \tag{26}$$

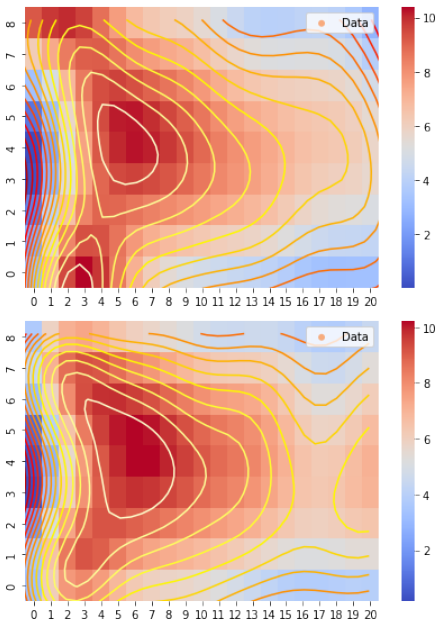

Fig. 10. Two maps of estimated pollutant concentrations (in mg/m^3) of the plume over the target area at the end of two independent drone flight episodes occurring chronologically from top to bottom.

- DDQN-C51 path inference time complexity:

$$O(d \cdot N \cdot (|\theta| + a)) \tag{27}$$

- Hybrid DDQN-C51 path inference time complexity:

$$O(d \cdot N \cdot (|\theta| + a)) + 2 \cdot d \cdot T_r \tag{28}$$

- TSH path inference time complexity:

$$w^d \cdot a^N \cdot T_r + O\left(w^d \cdot a^N \cdot N \cdot \log(a)\right) \tag{29}$$

Where T_r correspond to the time complexity of the reward computation algorithm, w is the exploration width of the search tree in TSH and $|\theta|$ is the number of neural network weights while a is the number of possible actions at each step.

C. Evaluation of pollution plume parameter estimation

With regard to the estimates of the pollutant concentrations made by the GP regression, we have plotted in Figure 9 the mean absolute errors of the latter with respect to the ground truth during a drone flight episode. Our two RL models (Figures 9-(2) and 9-(3)) perform better than the random model (Figure 7-(1)). In addition, we note that their confidence intervals are narrower compared to random and TSH approaches (Figure 9-(4)). TSH reduces the error by 0.07 compared to RL which may not be worth considering its run time (see table II).

The GP regression generates improved pollution maps each time the drones take new measurements. Figure 10 depicts three maps that were taken at the end of three independent episodes. These clearly show that the GP regression manages to capture the general shape of the plume (Figure 4), its direction, its amplitude and the three sources of pollution that generate it.

In order to validate the correct estimation of the variance and the length scale, we plot the evolution of the estimated errors of these parameters over several independent episodes of drone flight. The results are summarized in Figures 11 and 12. These show that our estimation model makes it possible to approach the true values of the variance and the length scale rapidly whatever the method of resolution used. Indeed, the average variance estimation error is between 2% and 3.5%, while that of the length scale is between 2.5% and 10%. In addition, we clearly see an advantage for the solution with the Hybrid DDQN-C51 method which manifests itself in greater stability and much better estimates.

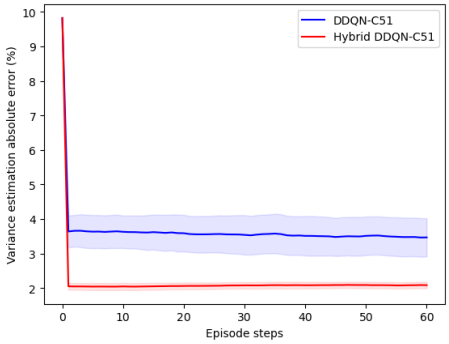

Fig. 11. Evolution of the variance estimate absolute error and its 95% confidence intervals over an entire drone flight episode.

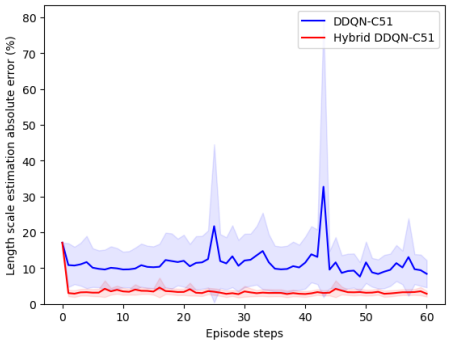

Fig. 12. Evolution of the length scale estimate absolute error and its 95% confidence intervals over an entire drone flight episode.

D. Behavior of our system in the case of a parameter change

In these paragraphs, we will analyze the response of our model to various changes in the input parameters that need to be estimated after seeing how it generally behaves in a scenario with stationary parameters. For this, we considered a scenario where we reduce the emission rate of pollution sources ($1\,g/s$ instead of $1.59\,g/s$ previously) as well as apply a wind inclination of $15°$ relative to the abscissa axis. In addition to this change, we have also increased the variance by 0.2 which means it goes from 0.5 to 0.7. The plume produced is shown on the map to the left of the Figure 13.

Figure 14 illustrates the way our approach behaves when estimating the parameters of a new plume. Note that all the

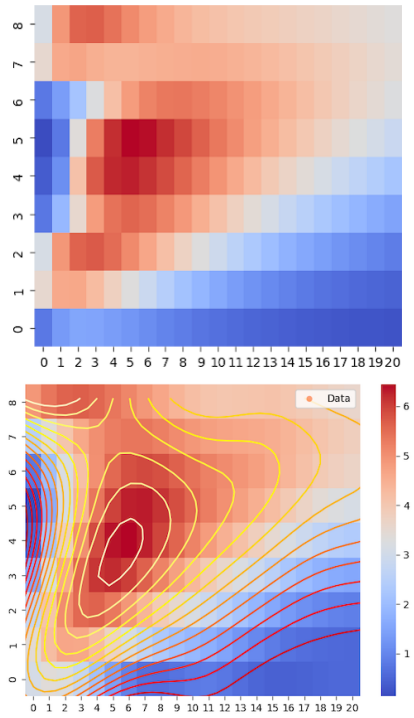

Fig. 13. Maps of pollutant concentrations (in mg/m^3) of the plume over the target area. The map above: unknown ground truth. The map below: estimated concentrations at the end of a drone flight episode.

changes made to the plume occurred at the step represented by the vertical dotted red line. The old and new real values of the variance and the average of the pollutant concentrations are represented by the gray and black dashed horizontal lines. At the beginning of the test, we see that the estimates of the various parameters converge very well towards their real values. The estimation model reacts quite abruptly when the scenario changes as shown by the increasing and decreasing peaks at step 100. However, this last quickly begins to find its pace of convergence. Looking at the curves, we see that convergence is found after approximately 100 steps, which means that it converges after less than two episodes. We also notice that there is a slight advantage for the hybrid method in terms of stability and rapidity of convergence especially for the estimation of the length scale where it is more stable and clearly faster. These results prove that our framework can adapt to changes and plume dynamics. Indeed, it learns to modify its behavior to switch to the steady state and thus correctly estimate the new values of the plume parameters. Our solution still manages to generate a fairly correct map of the pollution plume that matches the real map well by finding the new wind direction and the new pollutant concentrations (Figure 13).

VIII. Conclusion and Future Work

In this work, we study the MIPP problem for an air pollution characterization assuming that the latter follows a GP. We propose a framework based on MARL to guide a collaborative fleet of drones to carry out our task. In order to define an adequate characterization of the studied pollution plume, the

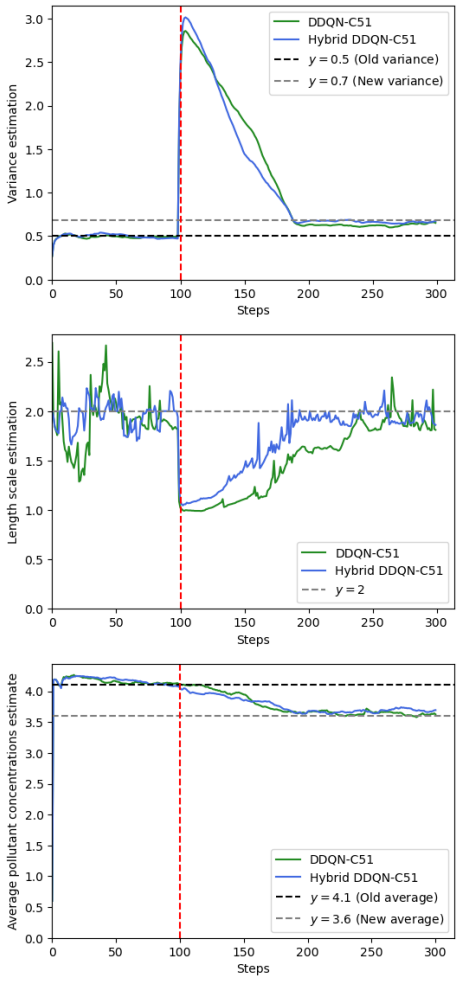

Fig. 14. Behavior of our framework following a change in variance, pollution plume emission rate and wind direction.

RL models of each agent try to learn conjointly the optimal positions. These positions allow gathering the most informative measurements about the pollutants, considering a reward proportional to the decrease in terms of uncertainty using MI. The new measurements ensure the improvement of the GP regression outputs. To validate the concepts proposed, we conduct a set of experiments using a Gaussian dispersion model to simulate realistic scenarios of pollution plume. Results show that our framework achieves good performances and manages to make estimates very close to the ground truth. From the maps generated, we can see the general appearance of the plume and differentiate the sources of pollution. Moreover, our model converges quickly and adapts to cases of change in the parameters of a plume. This is thanks to the weighted forgetting mechanism which makes it possible to estimate accurate updated variance and length scale.

In the future, we seek to further improve the proposed spatio-temporal characterization model to make it more flexible and accurate. For this, we intend to explore the data assimilation track while assuming the GP phenomenon. This allows defining a new reward function, which should ideally incorporate the connectivity factor between drones in flight.

ACKNOWLEDGMENT

This work was supported by the French National Research Agency (ANR) under Project ANR-21-CE25-0003 (DRON-MAP) and it was also partially supported by the French Embassy in Algeria through the project FSPI N°2021-016 LISEN 2020-02.

REFERENCES

[1] A. Boubrima, W. Bechkit, and H. Rivano, "Optimal wsn deployment models for air pollution monitoring," *IEEE Transactions on Wireless Communications*, vol. 16, no. 5, pp. 2723–2735, 2017.

[2] N. Cheng, S. Wu, X. Wang, Z. Yin, C. Li, W. Chen, and F. Chen, "Ai for uav-assisted iot applications: A comprehensive review," *IEEE Internet of Things Journal*, 2023.

[3] Y. Wei and R. Zheng, "Informative path planning for mobile sensing with reinforcement learning," in *IEEE INFOCOM 2020-IEEE Conference on Computer Communications*. IEEE, 2020, pp. 864–873.

[4] G. Duflo, G. Danoy, E.-G. Talbi, and P. Bouvry, "A q-learning based hyper-heuristic for generating efficient uav swarming behaviours," in *Intelligent Information and Database Systems: 13th Asian Conference, ACIIDS 2021, April 7–10, 2021, Proceedings 13*. Springer, 2021, pp. 768–781.

[5] F. Qi, X. Zhu, G. Mang, M. Kadoch, and W. Li, "Uav network and iot in the sky for future smart cities," *IEEE Network*, vol. 33, no. 2, pp. 96–101, 2019.

[6] Y. Wei and R. Zheng, "Multi-robot path planning for mobile sensing through deep reinforcement learning," in *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 2021, pp. 1–10.

[7] L. Buşoniu, R. Babuška, and B. De Schutter, *Multi-agent Reinforcement Learning: An Overview*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–221. [Online]. Available: https://doi.org/10.1007/978-3-642-14435-6_7

[8] K.-C. Ma, L. Liu, and G. S. Sukhatme, "Informative planning and online learning with sparse gaussian processes," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 4292–4298.

[9] J. Binney, A. Krause, and G. S. Sukhatme, "Informative path planning for an autonomous underwater vehicle," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 4791–4796.

[10] A. Singh, A. Krause, C. Guestrin, and W. J. Kaiser, "Efficient informative sensing using multiple robots," *Journal of Artificial Intelligence Research*, vol. 34, pp. 707–755, 2009.

[11] C. E. Rasmussen, *Gaussian Processes in Machine Learning*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 63–71. [Online]. Available: https://doi.org/10.1007/978-3-540-28650-9_4

[12] C. Guestrin, A. Krause, and A. P. Singh, "Near-optimal sensor placements in gaussian processes," in *Proceedings of the 22nd international conference on Machine learning*, 2005, pp. 265–272.

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

[14] M. Hessel, J. Modayil, H. van Hasselt, T. Schaul, G. Ostrovski, W. Dabney, D. Horgan, B. Piot, M. Azar, and D. Silver, "Rainbow: Combining improvements in deep reinforcement learning," 2017. [Online]. Available: https://arxiv.org/abs/1710.02298

[15] B. H. Abed-Alguni, D. J. Paul, S. K. Chalup, and F. A. Henskens, "A comparison study of cooperative q-learning algorithms for independent learners," *International Journal of Artificial Intelligence*, vol. 14, no. 1, pp. 71–93, 2016.

[16] H. Zhu, J. J. Chung, N. R. Lawrance, R. Siegwart, and J. Alonso-Mora, "Online informative path planning for active information gathering of a 3d surface," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1488–1494.

[17] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robotics and Autonomous Systems*, vol. 86, pp. 13–28, 2016.

[18] Y. K. Hwang, N. Ahuja *et al.*, "A potential field approach to path planning." *IEEE transactions on robotics and automation*, vol. 8, no. 1, pp. 23–32, 1992.

[19] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation.*, vol. 2. IEEE, 2000, pp. 995–1001.

[20] M. Wang and Liu, "Fuzzy logic based robot path planning in unknown environment," in *2005 International Conference on Machine Learning and Cybernetics*, vol. 2, 2005, pp. 813–818 Vol. 2.

[21] G. Sartoretti, J. Kerr, Y. Shi, G. Wagner, T. S. Kumar, S. Koenig, and H. Choset, "Primal: Pathfinding via reinforcement and imitation multi-agent learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2378–2385, 2019.

[22] M. E. Aydin and R. Fellows, "Building collaboration in multi-agent systems using reinforcement learning," in *International Conference on Computational Collective Intelligence*. Springer, 2018, pp. 201–212.

[23] Y. Chen, X. Lin, T. Khan, and M. Mozaffari, "A deep learning approach to efficient drone mobility support," in *Proceedings of the 2nd ACM MobiCom Workshop on Drone Assisted Wireless Communications for 5G and Beyond*, 2020, pp. 67–72.

[24] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, "Deep reinforcement learning: A brief survey," *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017.

[25] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[26] O. Delalleau and Y. Bengio, "Shallow vs. deep sum-product networks," *Advances in neural information processing systems (NeurIPS)*, vol. 24, 2011.

[27] 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.

[28] G. TensorFlow, "Dqn c51/rainbow," 2022. [Online]. Available: https://www.tensorflow.org/agents/tutorials/9_c51_tutorial

[29] Y. Wei, C. Frincu, and R. Zheng, "Informative path planning for location fingerprint collection," *IEEE Transactions on Network Science and Engineering*, vol. 7, no. 3, pp. 1633–1644, 2019.

[30] R. E. Blahut, "25 - information theory and coding," in *Reference Data for Engineers (Ninth Edition)*, ninth edition ed., W. M. Middleton and M. E. Van Valkenburg, Eds. Woburn: Newnes, 2002, pp. 25–1–25–31. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780750672917500273

[31] A. Daly and P. Zannetti, "Air pollution modeling–an overview," *Ambient air pollution*, pp. 15–28, 2007.

[32] J. Weil and R. Brower, "An updated gaussian plume model for tall stacks," *Journal of the Air Pollution Control Association*, vol. 34, no. 8, pp. 818–827, 1984.

[33] J. M. Stockie, "The mathematics of atmospheric dispersion modeling," *Siam Review*, vol. 53, no. 2, pp. 349–372, 2011.

[34] R. Grbić, D. Kurtagić, and D. Slišković, "Stream water temperature prediction based on gaussian process regression," *Expert systems with applications*, vol. 40, no. 18, pp. 7407–7414, 2013.

[35] S. Shabani, S. Samadianfard, M. T. Sattari, S. Shamshirband, A. Mosavi, T. Kmet, and A. R. Várkonyi-Kóczy, "Modeling daily pan evaporation in humid climates using gaussian process regression," *arXiv preprint arXiv:1908.04267*, 2019.

[36] O. Hamelijnck, T. Damoulas, K. Wang, and M. Girolami, "Multi-resolution multi-task gaussian processes," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[37] H. Liu, C. Yang, M. Huang, D. Wang, and C. Yoo, "Modeling of subway indoor air quality using gaussian process regression," *Journal of hazardous materials*, vol. 359, pp. 266–273, 2018.

[38] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[39] S. D. McDougle, M. J. Boggess, M. J. Crossley, D. Parvin, R. B. Ivry, and J. A. Taylor, "Credit assignment in movement-dependent reinforcement learning," *Proceedings of the National Academy of Sciences*, vol. 113, no. 24, pp. 6797–6802, 2016.

[40] C. Claus and C. Boutilier, "The dynamics of reinforcement learning in cooperative multiagent systems," *AAAI/IAAI*, vol. 1998, no. 746-752, p. 2, 1998.

[41] T. Balch *et al.*, "Learning roles: Behavioral diversity in robot teams," in *AAAI Workshop on Multiagent Learning*, 1997.

[42] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of monte carlo tree search methods," *IEEE Transactions on Computational Intelligence and AI in games*, vol. 4, no. 1, pp. 1–43, 2012.

[43] C. Paxton, V. Raman, G. D. Hager, and M. Kobilarov, "Combining neural networks and tree search for task and motion planning in challenging environments," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 6059–6066.