Proceedings of the 2021 AIAA/IEEE 40th Digital Avionics Systems Conference (DASC), 3-7 October, San Antonio, TX, USA
DOI: 10.1109/DASC52595.2021.9594300

Adaptive UAV Swarm Mission Planning by Temporal Difference Learning

Shreevanth Krishnaa Gopalakrishnan
School of Aerospace, Transport and Manufacturing
Cranfield University
Milton Keynes, United Kingdom
S.Gopalakrishnan@cranfield.ac.uk

Saba Al-Rubaye
School of Aerospace, Transport and Manufacturing
Cranfield University
Milton Keynes, United Kingdom
S.Alrubaye@cranfield.ac.uk

Gokhan Inalhan
School of Aerospace, Transport and Manufacturing
Cranfield University
Milton Keynes, United Kingdom
Inalhan@cranfield.ac.uk

Abstract— **The prevalence of Unmanned Aerial Vehicles in precision agriculture has been growing rapidly. This paper tackles the UAV global mission planning problem by first incorporating a greater capacity for human-machine teaming in the design of a flexibly autonomous, near-fully-distributed Mission Management System for UAV swarms. Subsequently, to maximize the efficiency with which missions are carried out, the two problems of global mission planning: task assignment/routing and path planning, were solved together, for small problem sizes, by an integrated solution. This consists of a geometric clustering algorithm which prioritizes the minimization of overall mission time, and an off-policy, model-free Temporal Difference Learning global agent capable of learning about an initially unknown mission environment through simulations. The latter component makes the solution adaptive to missions with different requirements.**

Keywords—Reinforcement Learning, Temporal Difference Learning, UAV, Global Mission Planning, Precision Agriculture.

I. INTRODUCTION

Unmanned Aerial Vehicles (UAVs), or drones, are aerial robots that can support and transport cameras, communication equipment, sensors, and other payloads specific to the mission requirements. UAVs can be classified into three main types based on their profile and propulsion method as fixed-wing, single-rotor, multi-rotor, and hybrid fixed-wing Vertical Take-off and Landing (VTOL) [1]. Although originally, they were primarily employed for 'dull, dirty, or dangerous' [2] missions, due to their increasing networking capabilities, 'intelligence' and payload diversity, their benefits have been realized for broader applications in the civil domain such as photography, construction, logistics, remote surveillance, precision agriculture etc.

To cope with the expected world population of 9 billion by 2050, there is a need for a 70% increase in agricultural yield and products. UAVs and other Information and Communication Technologies (ICT) such as broadband connectivity, Internet of Things (IoT), sensors etc., are currently being leveraged to help the agricultural sector manage crops and resources more efficiently [3]. This falls into the domain of Precision Agriculture (PA) which is the use case selected for this paper.

PA's objectives are as follows: (i) to increase the yield of crops; (ii) to improve the quality of products; (iii) to make more efficient use of agrichemical products; (iv) to save energy; and (v) to protect the physical environment against pollution [3].

UAVs are particularly useful since they can grant the farmers insights into spatially and temporally varying data regarding the crop's health and nutrient requirements, soil composition, topography etc., thereby allowing the creation of more productive farming practices [3].

In this paper, first, a flexibly autonomous Unmanned Aerial System (UAS) swarm architecture for mission management is proposed. It is designed to provide it with a capability to adapt to the intricacy and reliability requirements of the mission. This is achieved by extracting, integrating, and repurposing state-of-the-art architectural elements from the literature. Subsequently, to address the prevalent challenges of limited endurance and drone operator fatigue [4], a swarm global mission planner (GMP) is designed and developed using algorithms from the field of Reinforcement Learning (RL). The overall aim of this work is to increase the efficiency of UAVs for short missions.

The contributions of this paper are three-fold: 1) highlighting the importance and relevance of the field of precision agriculture; 2) designing a flexibly autonomous UAV mission management architecture; and 3) developing an adaptive, deep RL-based global mission planning algorithm architected using traditional Object-Oriented Programming principles, combined with a simple geometric clustering algorithm to increase system decentralization.

II. TECHNICAL BACKGROUND

A. *Unmanned Aerial Systems*

UASs comprise the entire ecosystem that enables a UAV to function. Broadly, they consist of three principal sub-systems as demonstrated by Fig. 1 [5]:

- Aerial platform: airframe, navigation system (flight controller, autopilot, localization (GPS/GNSS), inertial system), power and propulsion system (actuators), and payload(s). The flight controller is the central processing unit of the drone.
- Ground Control Station (GCS): human-machine interface platform e.g., portable remote control, full-fledged command station etc., human supervisor(s), and other relevant software for high-level mission planning, control, and data exchange in real time.
- Communication system: transmission and reception equipment selected according to GCS-swarm and intra-swarm operating distance, and environmental conditions.

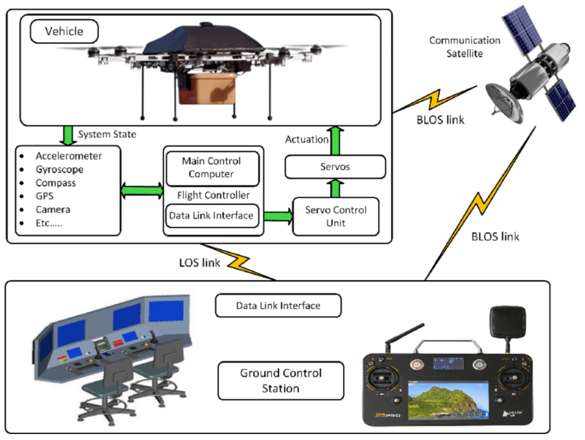

Fig. 1. High-level UAS architecture [5]

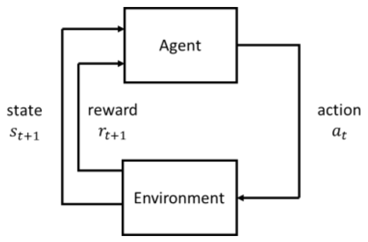

Fig. 2. RL Agent seeks to maximize cumulative reward obtained

B. *Intelligent Agents and Reinforcement Learning*

Autonomous UAVs, by definition, can operate according to a predefined mission plan without human intervention [6]. The level of autonomy of UAVs is typically divided into Classes 'A' to 'E', with 'E' representing fully-remote-controlled, and 'A' representing fully-autonomous operation [7]. Autonomy is capable of easing operator workload, reducing error rate associated with attention-intensive control tasks, enabling faster response to developing operational conditions due to eliminated datalink time loss, optimizing battery usage, and improving landing accuracy.

In the domain of Artificial Intelligence (AI), each UAV can be abstracted to being an intelligent agent which is a goal-oriented, self-governing entity that perceives its environment through sensors and influences it using actuators. There are five main types of agents: simple reflex, model-based, goal-based, utility-based, and learning [8]. In swarm-based systems, agents also incorporate a social layer for communication with peer agents.

The two architectures of interest in this paper are utility-based and learning agents. Utility-based agents choose actions that maximize their utility function/internal performance measure. In addition to considering the importance of mission goals, they assign 'value' to states in the world. This allows them to reach the goal optimally [8].

Reinforcement Learning, a branch of AI, deals with the 'learning' problem for intelligent agents. RL agents search for more effective ways to understand and navigate around initially unknown environments by learning from the impact of their interactions [9]. The fundamental schematic of an RL agent working is shown in Fig. 2.

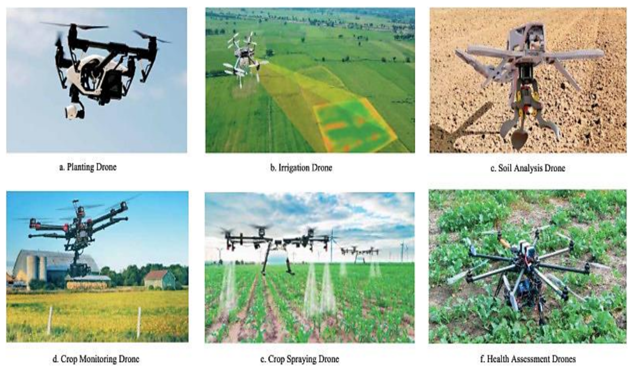

Fig. 3. Types of agricultural drones [10]

TABLE I. MAIN PRECISION AGRICULTURE PRACTICES [3]

Process	**Description**
Production mapping	Crop efficacy calculation and identification of production determining factors.
Soil mapping	Determination of the variability of the soil's chemical composition.
Electrical conductivity mapping	Identification of homogeneous soil management zones.
Remote sensing (RS)	Remote information capture by measuring the crops' electromagnetic radiation etc.
Variable Rate Application (VRA)	Distribution of agrochemical products (pest control, fertilizer) in different doses, including water for irrigation.

C. *Precision Agriculture*

With regards to types of UAVs, multirotor (quadrotor) UAVs, as seen in Fig. 3, are typically selected for precision agriculture due to their ability to hover, take-off and land without considerable space (runway) or launch equipment (catapult etc.). However, when endurance is more important than maneuverability, fixed-wing UAVs are optimal. Some examples of common PA tasks are explained in TABLE I.

Variable Rate Application (VRA) of agrochemical products such as fertilizers is the main interest of this paper. VRA is typically carried out in a map-based or sensor-based method. In the former case, the UAV relies on GPS and a product prescription map for dosage. Alternatively, specialized UAVs with dedicated sensors can measure the characteristics of each area in real-time [11]. UAV swarms are typically preferred due to the possibility of dividing the farmland into areal blocks and tackling them independently. These UAVs would consist of an agrochemical product dispenser system, which, for fluids, comprises a tank, pipeline, pump, and nozzle [1].

D. *UAV Mission Planning Taxonomy*

The UAV mission planning problem can be sub-divided into vehicle routing and trajectory optimization [12].

The routing problem deals with the assignment of UAVs to achieve a set of predetermined tasks while optimizing cost, time, distance, or energy. Additional constraints may be in place to account for the payload weight, environmental effects (obstacles, wind etc.), battery life, and demand (node visits) [13]. There are several variants of the routing problem including the Capacitated Vehicle Routing Problem (CVRP), the Travelling Salesman Problem (TSP) and the UAV Task Assignment Problem (UAVTAP). However, this category of problems is Non-deterministic Polynomial time-hard (NP-hard), implying that scalability is an issue [14].

The other half of mission planning is to do with trajectory optimization (TO) and path planning (PP). TO problems involve determining the trajectory of a system (control inputs for desired maneuvers) while minimizing a scalar performance index (flight time, fuel consumption etc.), and satisfying a set of boundary conditions and constraints regarding system kinematics and dynamics [12].

Path planning is a geometric problem which deals with finding a collision-free path from a pre-determined start position to a goal position, for an agent in Euclidean space. This problem has also been shown to be NP-hard if the vehicle velocity is unbounded and rotation is not considered. It can be solved in a discrete or continuous space. Discrete formulations (graph space) rely on exact or heuristic solvers and usually output polygonal paths, thereby warranting separate curvature inclusion [12].

III. System Design

A. *Related Work on UAS Mission Management Systems*

As the name suggests, a UAS mission management system (MMS) is responsible for all tasks beginning from the receipt of mission objectives from the GCS, to the completion of the mission by individual UAVs in swarm.

Nieuwenhuisen and Behnke [15] designed a layered mission planning and navigation system targeted towards multi-rotor micro air vehicles (MAVs) which are required to operate in partially observable environments. In this system, the GCS is tasked with high-level, low frequency mission planning and global path planning. Subsequently, an allocentric map is passed onto the MAVs for local path refinements and obstacle avoidance at higher frequencies.

Rudnick and Schulte [16] proposed an agent-based architecture which allows a range of human independence/ control levels over the UAV. In this scenario, the planning is carried out by a Hierarchical Task Network (HTN). Tasks allowed higher levels of independence are placed higher in the tree. Importantly, the operator is offered guided access to the HTN to prevent jeopardisation of the mission intent.

The design of a multi-drone UAS MMS also entails the specification of the control architecture. Compared to a centralized system, it has been shown that distributed control results in higher reliability, and requires lower computational resources and communication [17], [18].

However, of greatest interest in this paper is the work by Sanchez-lopez et al. on ‘AEROSTACK’ [19]. This is full-fledged mission management architecture that offers the benefits of high mission autonomy, versatility and swarm operation. It is offered as an open-source package which contains a system architecture and a multi-purpose software framework. It consists of five layers:

- Social layer: for intra-UAS communication.
- Reflective layer: for supervision of other layers, to assess if the system is progresssing towards its goals, and for troubleshooting the current position of the agent.
- Deliberative layer: for generation of global solutions for mission planning and path planning.
- Executive layer: for generation of instructions to the reactive layer based on the inputs from the Deliberative layer.
- Reactive layer: for low-level control through sensor (percept) – actuator (action) loops.

B. *Core System Requirements Specification*

Distilling the key priorities from the relevant literature, functionally, the MMS should enable the operator to communicate the target locations to be visited on a farm through a user interface. This information must then be passed onto a Global Mission Planner (GMP) which is part of the GCS software suite. The GMP’s role involves the following: 1) computing the optimal UAV swarm size through task assignment, 2) determining task sequence for each UAV for VRA of fertilizers without colliding with obstacles, 3) passing this information to a dedicated swarm coordinator UAV through the wireless datalink equipment.

The GMP will be designed to learn (plan) the initial, global mission objectives before the mission commences. However, the MMS must provide the operator the ability to intervene with a mission when necessary or abort it. E.g., in case the operator wishes to interrupt the agricultural UAV VRA to carry out manual soil mapping or health monitoring.

At the operational level, the architecture must facilitate cooperative drone working in a swarm and not be dependent on significant levels of communication between drones for completion of the mission, i.e., a near-fully-distributed architecture. The MMS should implement a hierarchical structure of mission planning, wherein the global task sequence is refined by the UAVs locally and adaptively in accordance with dynamic environmental conditions. Finally, the MMS must enable the storage of up-to-date knowledge about the position of obstacles and No-Fly-Zones.

C. *Proposed Architecture*

It is envisaged that the GMP abstract agent will have hybrid characteristics: utility-based and learning. Fig. 4 and Fig. 5 demonstrate the proposed flexibly autonomous UAS architecture. It possesses the following features:

- Hierarchical mission planning workflow, with the GCS at the global level, the swarm coordinator at the interim, and the local UAVs at the bottom.

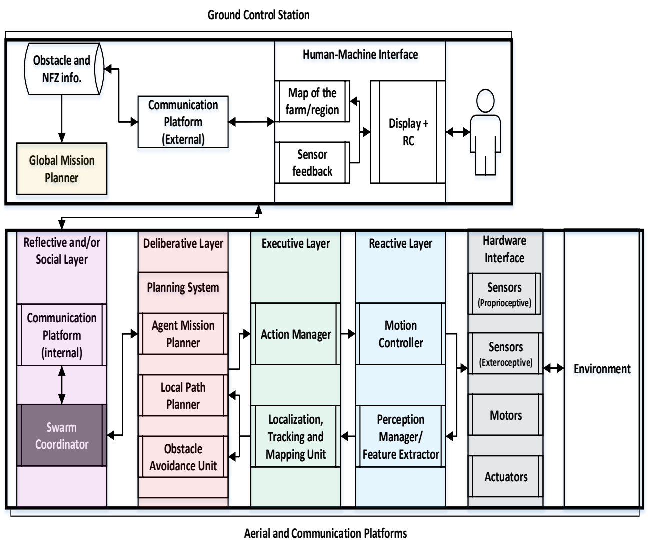

Fig. 4. Reconfigurable UAS Mission Management System architecture

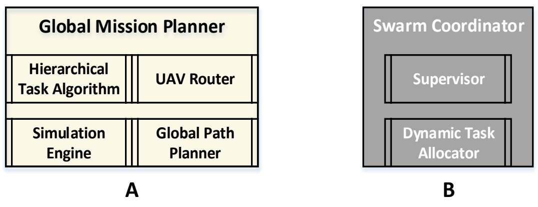

Fig. 5. Architecture subcomponents: A) Global Mission Planner with added flexibility; B) Swarm coordinator

- AEROSTACK' backbone which can manage drone swarming and full autonomy inherently.
- Flexible autonomy incorporated through software changes within the GMP. This leverages the strengths of human-machine teaming.

Having been constrained to a set of tasks following the hierarchy of mission planning, the local UAVs are able to operate in a near-fully-distributed manner. The in-built HTN and Simulation engine interface, as per [16], shown in Fig. 5, enable the operator to modify any of the original objectives without impeding the overall intent of the mission. It has been highlighted in the literature that this combination of routing and trajectory optimization is essential for increasing the efficiency of real-world UAV operations [12].

There is a single, assigned swarm coordinator which is responsible for receiving (from the GMP) and relaying the respective task sequences to each member of the swarm, supervising their progress during the mission, and forwarding the information back to the GCS for displaying to the operator. The flow of information is shown in Fig. 6.

Finally, it is proposed that the intra-swarm communication is conducted according to a Flying Ad-hoc Network (FANET) layout [20], as shown in Fig. 7.

Designing and developing the GMP's routing and path planning algorithms for the VRA application will be the focus of the rest of this paper.

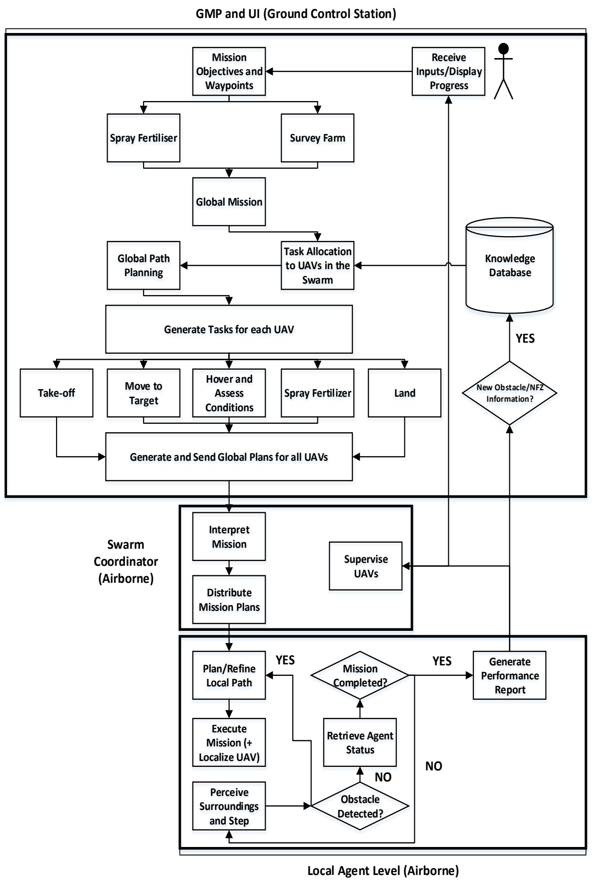
GMP and UI (Ground Control Station)
Mission Objectives and Waypoints
Receive Inputs/Display Progress
Spray Fertiliser
Survey Farm
Global Mission
Global Path Planning
Task Allocation to UAVs in the Swarm
Knowledge Database
Generate Tasks for each UAV
YES
Take-off
Move to Target
Hover and Assess Conditions
Spray Fertilizer
Land
New Obstacle/NFZ Information?
Generate and Send Global Plans for all UAVs
Swarm Coordinator (Airborne)
Interpret Mission
Distribute Mission Plans
Supervise UAVs
Plan/Refine Local Path
YES
Mission Completed?
YES
Generate Performance Report
Execute Mission (+ Localize UAV)
Retrieve Agent Status
NO
NO
Perceive Surroundings and Step
Obstacle Detected?
Local Agent Level (Airborne)

Fig. 6. Interaction between GCS, coordinator, and local agents

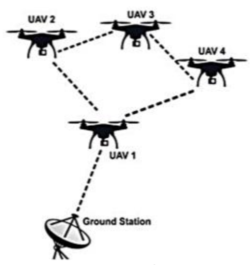

Fig. 7. Flying Ad-hoc Network configuration proposed [20]

IV. Problem Formulation

Prior to designing the Global Path Planner, the farm environment (mission area) within which the UAS will operate, and the sample problem(s) to be solved, will be formulated and described.

A. *Environment Model and Mission Description*

The operating environment of the UAVs is modelled as a square grid W of size M x $M \in \mathbb{N}$, where M is 500 meters, M x MM x $M \in$ **N**, and is comprised of square cells c of length 1 meter. A 10-task scenario, which was one of the problems considered, is shown in Fig. 8. Let C_o denote the list of static obstacles, such that $C_o = \{O_1, O_2, O_3, \dots, O_{N_o}\}$, and let C_Z denote the list of static obstacles and No-Fly-Zones (NFZs), such that $C_Z = (\{Z_1, Z_2, Z_3, \dots, Z_{N_Z}\})$.

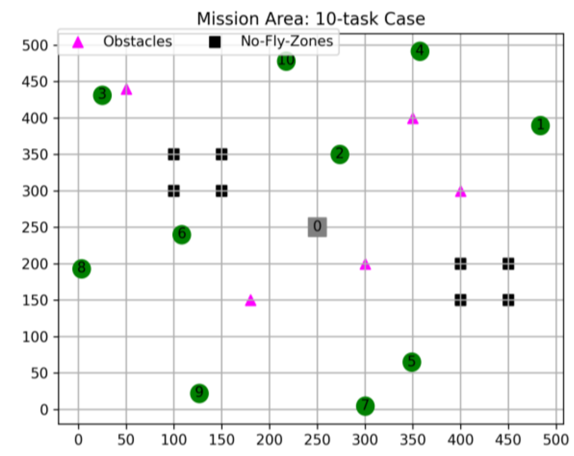

Fig. 8. 10-target mission scenario with a central depot

TABLE II. MISSION PLANNING PROBLEM ASSUMPTIONS

Category	**Characteristics**	
	Type	**Description of Assumptions**
UAVs	Quantity and traits	Multiple, homogeneous UAVs; fleet size variable; vehicles not capacitated.
	Modelling properties	No vehicle flight dynamics (geometric problem only) or equations of motion considered; Dubin's model not assumed.
Waypoints	Quantity	Multiple, unordered.
	Constraints	Single visit allowance.
Environment	Conditions	Obstacles and NFZs present; no wind.
	Dimensionality	2D plane.
Launching	Quantity of depots	Single, central depot; no inter-depots.
Time	Fixed/Variable	Time and velocity are fixed (not to be optimized).

Thus, the only traversable spaces are the free spaces which are denoted by C_F, where $C_F = (W \cap (C_o \cup C_Z))^C$. The complete set of target states C_T, where $C_T = \{T_1, T_2, T_3, \dots, T_{N_T}\} \subseteq C_F$, is finite in length ($N_T$).

The mission is formulated and constrained as per a standard TSP/VRP as follows. Operating from a central depot D, a swarm of agricultural UAVs are tasked with visiting a subset of the targets each for fertilizer spraying in a near-optimal path (time and distance). Once a target is visited, it is added to a list $V \subseteq C_T$. The motion of each UAV is from one state (T_i) to another (T_j) such that $i, j < N_T$, $i \neq j$, and $T_j \notin V$, i.e., revisits are not allowed.

Finally, as stated earlier, during the mission, the UAVs must avoid collisions with obstacles and other UAVs, and must not enter the pre-defined No-Fly Zones (NFZs). Three mission environments were created: 5, 10 and 20 tasks. TABLE II. presents the key assumptions made.

B. *Agent-Based Problem Formulation*

The fundamental building block of RL is a Markov Decision Process (MDP) [9], and it can be defined by the tuple: (S, A, R, P) with state space S, action space A, reward function R, and probabilistic transition function P mapping from $S \times A \times S \rightarrow \mathbb{R}$. For finite horizon MDPs, the time index t is used as subscript.

The agent's observation at each time step in the simulation can be represented by $s_t = (P_t, V_t)$, which consists of two elements:

- $P_t \in \mathbb{B}^{N_T}$ contains the current position of the agent in one-hot-encoded format, where $\mathbb{B}$ represents the Boolean domain $\{0, 1\}$.
- $V_t \in \mathbb{B}^{N_T}$ contains the set of visited positions of the agent at time t. Once a task is completed, its position in V_t is marked as TRUE.

$$S = \underbrace{\mathbb{B}^{N_T}}_{Curr.\ Pos.} \times \underbrace{\mathbb{B}^{N_T}}_{Hist.\ Pos} \tag{1}$$

Since the positions of obstacles are not represented within the state space, this problem can be categorized as a Partially Observable MDP (POMDP). Additionally, since the agent needs to remember the set of tasks it has already completed and the rewards are 'non-stationary', it is essential to pass V_t to the agent. This makes the problem Markovian, thereby preventing the need for 'memory'.

Since the agent jumps from task-to-task, neglecting the interim states, the action space can be defined as:

$$A = T_1, T_2, T_3, \dots, T_{N_T} = C_T \tag{2}$$

In model-free RL, there is no need to know the probabilistic transition function, and hence it has been preferred in the literature in most cases of UAV path planning where the environment's mathematical model is unknown as a prior [21]. The final element of the MDP is the reward function R. It was designed as follows:

- r_{pen} which represents the penalty (negative) given to the agent upon attempting to do a task it is not allowed to. This would account for crashes into obstacles and NFZs.
- r_{bon} which represents the bonus (positive) given when the agent completes a task appropriately.

Hence, the narrowed down problem scope can be formally divided into two parts: 1) allocate tasks to a suitable number of UAVs for the mission. Each UAV (uav_i) would therefore need to perform $C_{T,i} \subseteq C_T \ \forall i \in 1, 2, \dots, N_{uav}$, 2) plan a path for each UAV such that it visits each task only once, does not collide with obstacles, NFZs or other UAVs, and returns to the depot.

V. ALGORITHM DEVELOPMENT

A. *Related Work on Task Allocation Algorithms*

With regards to the objective of task allocation, Li et al [22] developed a Variable Neighborhood Descent-enhanced Particle Swarm Optimization algorithm for the flight path optimization of multiple agricultural UAVs. Their conclusion was that for point-to-point missions, it is more efficient to minimize mission time by minimizing the length of the longest UAV path, as opposed to total distance.

In a similar line of work, Ann et al [23] devised an area allocation algorithm for the coverage path planning problem of multiple UAVs based on the clustering (K-means) and the graph method. Their algorithm enabled the division of a mission zone into multiple collision-free sub-areas consisting of obstacles, each covered by a different UAV.

These two works suggest that geometric clustering approaches like K-means could be sufficient for task allocation in a UAV MMS.

B. Background of Reinforcement Learning Algorithms

RL algorithms are particularly suitable for sequential decision-making problems like path planning. Generically, RL agents search for an optimal behavior policy π^*, that consistently maximizes their reward when interacting with the environment [9]. Temporal Difference (TD) Learning refers to the category of algorithms in which the agents update their estimates of the value function V and action-value function Q, as per the Bellman equations, using only previously learned estimates and the observed reward R_{t+1} via bootstrapping [9].

Q-learning is a model-free, off-policy, TD Learning technique. Its premise is the iterative improvement of the Q function to guide and evaluate the process of learning π^*. As shown in Fig. 2 earlier, the agent does this by sequentially observing its current state $s_t \in S$, performing an action $a_t \in A$ according to its current policy $\pi(s_t, a_t): S \times A \to \mathbb{R}$, receiving a reward $r(s_t, a_t) \in R$, and repeating the process at $t+1$ [9]. The Q-update formula for finding the optimal policy is given by the following two equations.

$$\delta_t^{Q} = R_{t+1} + \gamma \max_{\forall a \in A} Q(s_{t+1}, :) - Q(s_t, a_t) \quad (3)$$

$$Update: Q(s_{t+1}, a_{t+1}) \leftarrow Q(s_t, a_t) + \alpha \delta_t^{Q} \quad (4)$$

The selection of the maximum Q-value at time $t+1$ is attributed to the greedy strategy used for finding the optimal policy. α refers to the learning rate of the algorithm.

Q-learning suffers from issues regarding lack of generalization and scalability when implemented using a two-dimensional ($S \times A$) lookup table. One well-known solution to this is a Deep Q Network (DQN) which utilizes a multi-layered neural network Q-function approximator [24]. However, for this *paper*, since DQNs have been proven to overestimate V and Q due to the $\max$ operator, a Double DQN (DDQN) algorithm [25] was narrowed down on.

To improve computational stability, Mnih et al [26] suggested the simultaneous use of two separate networks: a policy network (with parameters $\boldsymbol{\theta}$) and a target network ($\boldsymbol{\theta'}$); and experienced replay.

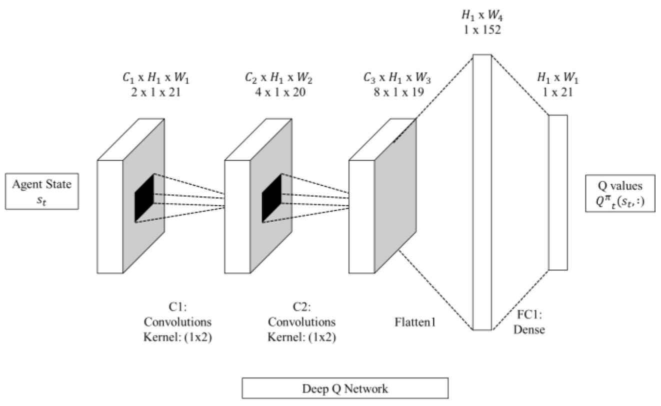

Fig. 9. Neural network architecture of a DQN

The NN takes the agent's current state s as input, and outputs the vector $Q(s, :; \boldsymbol{\theta})$. The target network has the same structure as the original (online/policy) network, however, $\boldsymbol{\theta'}$ are updated/copied only every τ steps from $\boldsymbol{\theta}$. On the other hand, experience replay is a uniform random sampling technique that was proposed to reduce correlations in the training data which is otherwise sequential. The agent's experience is typically stored in a finite replay buffer D in a quadruple format: (s, a, r, s') [25]. The DDQN TD target is shown below in:

$$A = T_1, T_2, T_3, \dots, T_{N_T} = C_T \quad (5)$$

The novelty introduced by DDQN is that the TD target is determined by gathering the Q values associated with the action selected by the policy network [25].

A recent work by Theile et al [27] addressed the coverage path planning problem using a DDQN. Their network architecture interprets 3-channel map-like input through convolutional layers to generate the observation.

More recently, Xie et al [28] formulate UAV path planning as a POMDP. They use recurrent neurons to handle the partial observability by extracting crucial information from historical state-action pairs, and convolutional neurons to capture spatial feature information from the observation prior to determining the Q values of a state. This is referred to as a DRQN algorithm.

Therefore, the literature suggests that the DDQN algorithm is highly relevant for the path planning problem, and that the network can be combined with convolutional or recurrent neurons to provide it with secondary capabilities.

C. Double Deep Q-Learning Model and Simulation Setup

The DQN neural network architecture shown in Fig. 9 below was adopted for the policy network and the target network, and hence, the DDQN-based agent was created.

Two 2D convolutional layers with a flat kernel of size (1x2) are used to extract the spatial features from the agent's state s_t tensor. The resulting tensor of shape (8x1x19), for, say a 20-task problem, is flattened prior to being passed to a densely connected layer for reshaping into the action space A's dimensions (1 x 21). The resulting vector represents the predicted Q values corresponding to each state-action pair at time t.

Action selection at each step is carried out based on the predicted Q values from the policy network according to an ϵ -greedy algorithm as shown below. P refers to the probability of a case occurring.

$$a_t = \begin{cases} \underset{a}{\text{argmax}}\ Q(S_{t+1}, :; \boldsymbol{\theta}_t), & P = 1 - \epsilon \\ \text{randin}(\text{range}(0, N_T)), & P = \epsilon \end{cases} \quad (6)$$

The ϵ-greedy algorithm is one of the ways to solve the exploration-exploitation dilemma during agent training [9].

During each training step, as per the standard neural network training procedure, the loss is computed for each batch sampled from D, and the network parameters ($\boldsymbol{\theta}$) are trained through backpropagation by an optimizer. For this paper, the Huber Loss function was selected since it combines the beneficial properties of both, Mean Absolute Error (MAE) and MSE). It is represented by:

$$L_{Huber} = \begin{cases} 0.5(x_n - y_n)^2, \ if \ |x_n - y_n| < \delta \\ \delta(|x_n - y_n| - 0.5\delta), \ \text{otherwise} \end{cases} \quad (7)$$

Adam [29] was selected as the optimizer due to its robustness and limited requirement for hyperparameter optimization resulting from as a result of its adaptive learning rate selection.

The simulation environment was designed in Python. The PyTorch library [30] was used for designing the neural networks and the training infrastructure.

The elbow point method was selected for use in tandem with K-means clustering to determine the optimal number of UAVs (N_{uav}) needed. This method seeks to trade-off the minimization of the Within Cluster Sum of Squares (WCSS) with the number of clusters. Clustering was repeated iteratively to ensure that each task was reachable within each cluster, and the UAV had a safe path to the depot at the end of the mission.

With respect to task assignment, to prevent collisions, some of the edges were labelled as 'blocked paths' and associated with a negative reward. They were determined by comparing the closest point of approach between each path and the obstacles and NFZs with a threshold of 10 meters.

In Object-Oriented Programming terms, the Agent and Environment were created as two separate classes. The interaction between the agent and the environment objects is demonstrated in Fig. 10.

The training was set to run for several hundred or a thousand simulation episodes, and each episode was run for up to 150 timesteps or until all the agents completed their designated tasks and reached the goal state (depot).

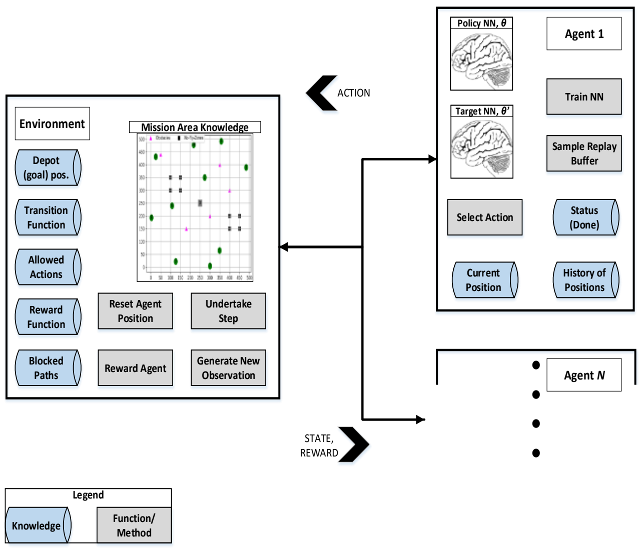

Fig. 10. Object-Oriented Programming-inspired structure of the simulation environment

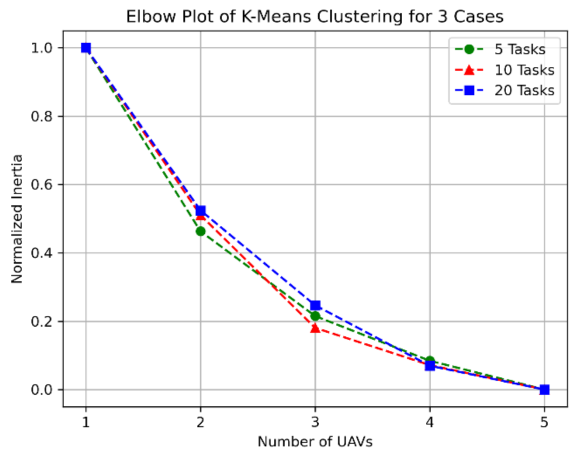

Fig. 11. Elbow point method shows s smooth curve with a relatively ambiguous elbow point

VI. Simulation Results and Analysis

A. *Task Allocation*

The implementation of the elbow point method for the 3 different missions revealed that, for the 5-task and 10-task missions, the optimal UAV swarm size is 3. However, for the 20-task mission, the gradient was found to continue to decline relatively steeply beyond 3, and hence, the optimal swarm size was determined to be 4. Refer to Fig. 11.

However, upon further analysis with regards to the positions of the obstacles, the algorithm revealed that, to provide accessibility to all tasks in the 5-task mission, a maximum of 2 UAVs/clusters would be possible.

B. *Reinforcement Learning Algorithms' Validation*

To validate the RL algorithms, the problem was simplified to a single agent scenario, and the blocked paths were not considered to simplify the set-up of optimization problem (multi-vehicle VRP).

TABLE III. RL ALGORITHM PARAMETERS FOR VALIDATION

Parameter	Value	Justification
Episodes (E)	500 (5, 10) or 1000 (20)	Enables path exploration
Timesteps (TS)	100 (5, 10) or 150 (20)	Encourages path completion
Optimizer LR (α)	0.001	With *Adam*, it is adaptive
Discount Factor (γ)	0.99	Prioritises future rewards
Epsilon range (ε_{init} - ε_{min})	0.90 – 0.1	Encourages path exploration
Epsilon decay (λ)	0.995 (5, 10) - 0.998 (20)	Slow exponential decay
Decay threshold	2x or 3x ideal steps	RBES: More leeway for 20-target scenario with 3x.
Target model update frequency (τ)	10 timesteps	Stabilizes NN optimization
Training batch size	512 timesteps	Trade-off between optimizer step accuracy and compute
Replay buffer($\mathcal{D}$) length	10000 timesteps	At least 100 episodes' data
Min. Reward (r_{pen})	0	Only positive reinforcement
Max. Reward (r_{bon})	+100	Same as above
Distance reward scaler (z)	10	Encourages nearest neighbour-seeking

Branch-and-Bound (B&B) refers to a family of algorithms which are used to produce exact solutions to NP-hard problems like path planning [31]. Here, since the number of tasks was small, the B&B method was used. Both the candidate RL algorithms: single DQN, and DDQN were set up according to the architecture shown in Fig. 9 using the parameters as per TABLE III.

To maximize exploration, the initial ϵ was set to a high value of 0.9. Subsequently, this was decayed exponentially, at a rate of 0.999, each time the number of timesteps to complete the trajectory was below a threshold. Finally, for the last 10% of episodes, it was manually forced to 0.1 to maximize consolidation of the gathered knowledge.

The 'Step' function, shown in Fig. 10, is responsible for reflecting the action taken by the agent. Working with the 'Reward Agent' function, it assigns a negative reward ($r_{t+1_i} = r_{pen}$, for Agent i) if the agent visits the depot prematurely, or attempts to repeat a task or travel via a blocked path. In case these conditions are not violated, it assigns a positive reward to each agent as follows: 1) compute distance ($dist_{t_i}$) from s_{t_i} to a_{t_i}, 2) identify shortest (non-zero) path possible:$dmin$, 3) calculate distance reward $r_{\text{dist}} = r_{bon} \times z \times dmin/dist_{t_i}$, 4) count previously completed tasks: vis_{t_i}, 5) observe length of agent $i'si$'s subspace: $space_i$, 6) calculate visited cells reward $r_{vis} = r_{bon} \times vis_i/space_i$, and finally, 7) return $r_{t+1_i} = r_{\text{dist}} + r_{vis}$. Hence, it is apparent that r_{dist} encourages the agent to find its nearest neighbor from any position. z was used as a scaling factor to trade-off with reinforcing the agent to complete new tasks.

The results in Fig. 12 and TABLE IV. demonstrate that the RL algorithms perform better with smaller mission sizes. Additionally, it is evident that the algorithms are comparably accurate to the exact B&B method, and hence, the simulation setup methodology is validated. Overall, it can also be observed that the DDQN algorithm performs better than the 'vanilla' DQN.

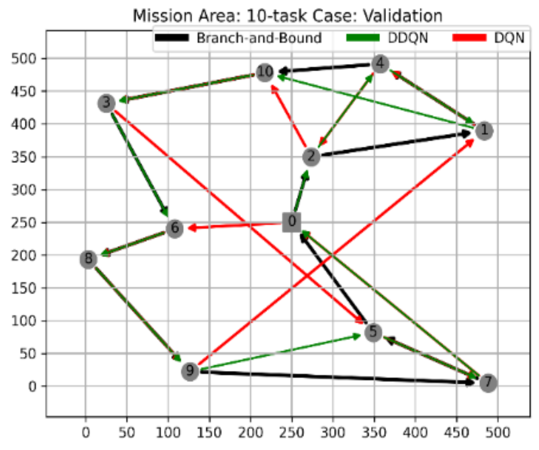

Fig. 12. RL algorithms' validation for the 5-task mission

TABLE IV. RL AGENT VALIDATION RESULTS: DISTANCES

Algorithms	Number of targets		
	5	10	20
B&B	1274m	2068m	2128m
DDQN	1274m	2174m	2975m
DQN	1746m	2623m	3270m

TABLE V. UPDATED PARAMETERS FOR MISSION PLANNING

Parameter	Value
Episodes (E)	1000
Timesteps (TS)	100
Epsilon decay (λ)	0.995
Decay threshold	3x (5) or 4x (10) ideal steps

C. Metrics for Mission Planning Evaluation

Moving forward, two of the three mission scenarios have been selected for comparison: 5 and 10 tasks. Two metrics are used to compare the performance of the candidate algorithms:

- *Average reward* ($r_{\text{av}_{\text{ep}}}$): This is calculated as the sum of all agents' rewards, averaged by the number of timesteps to complete the episode t_{comp}.

$$r_{\text{av}_{\text{ep}}} = 1/t_{\text{complete}_{\text{ep}}} . \sum_{t=1}^{t_{\text{complete}_{\text{ep}}}} \sum_{i=1}^{N_{\text{uav}}} r_{t_i} \qquad (8)$$

- *Number of ideal trajectories* (E_{ideal}): The 'ideal' number of timesteps for any UAV ($C_{T,i} \subseteq C_T$ $\forall i \in 1,2,..,N_{\text{uav}}$) is the sum of the cluster size and one (to return to the depot).

D. Results and Evaluation

This section provides the results obtained from the full simulations (four scenarios) of the vanilla DQN and the DDQN global path planning algorithms. TABLE V. presents the amendments made to the algorithms' parameters to account for the increased problem complexity resulting from the inclusion of obstacles and NFZs in the mission area. TABLE VI. presents the results in terms of the average episode reward and the number of ideal trajectories.

TABLE VI. RL ALGORITHMS PATH PLANNING RESULTS

Algorithms	Metrics	Number of targets	
		5	10
DDQN	E_{ideal}	113	51
	Maximum $r_{\text{av}_{\text{ep}}}$	1278	2205
DQN	E_{ideal}	76	37
	Maximum $r_{\text{av}_{\text{ep}}}$	1265	2114

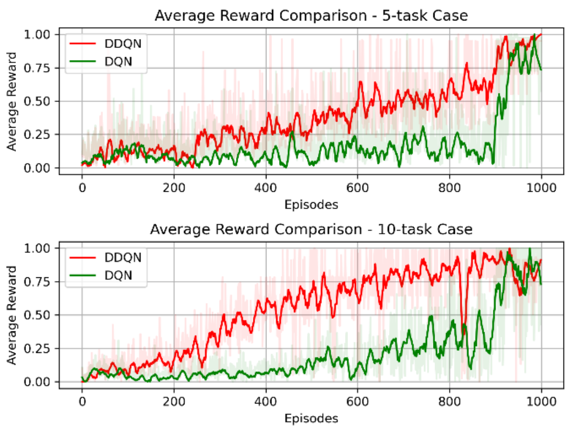

Fig. 13. Average Reward growth. A 2nd order Savitsky-Golay filter with window size of 21 was used for smoothing.

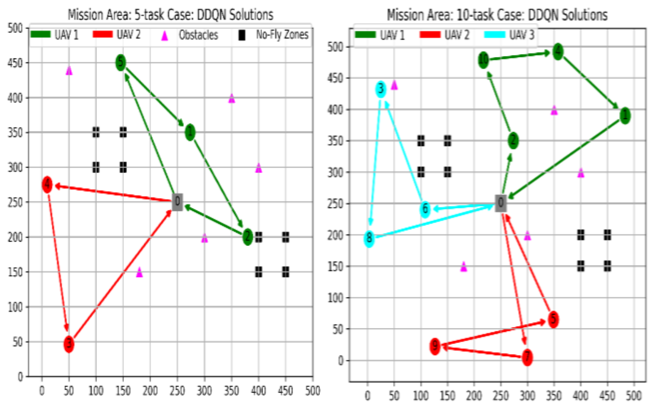

Fig. 14. DDQN GMP agent predictions: a) 5-tasks b) 10-tasks

Referring to Fig. 13 which shows the improvement in the average reward per episode over the training period, it can be observed that the DQN algorithm performed relatively poorly compared to DDQN for both cases.

Fig. 14 presents the planned trajectories for each of the UAVs by the DDQN-based GMP. During training, there were scenarios when the agent did not converge to the optimal trajectory. This can be attributed to the stochasticity in the model training because of the ε-greedy action selection, the limited training episodes, the impact of the reward-based epsilon decay strategy, and the dominance of the nearest neighbor-seeking tendency (z).

Therefore, it can be inferred that the DDQN-based GMP is able to explore the state space more efficiently (larger value of E_{ideal}) and converge faster to a solution in both scenarios. This can be attributed to its greater accuracy with respect to estimating the Q values of the environment, as explained previously.

E. Final Discussion and Scope for Future Work

Hence, RL models offer significant flexibility in addressing missions of different types, each with different requirements and constraints, without the need for a mathematical model of the environment as a prior. Tunable properties of the simulation environment such as the reward function and the strategy of ε decay were found to contribute greatest to this adaptiveness. Alternatively, this can be perceived as a limitation, since, in this work, no single combination of hyperparameters was found to be suitable for all mission sizes and constraints. The search for such a best set of hyperparameters would require a computationally expensive grid search, which is common in the field of Machine Learning (ML). This is one of the areas of recommended future work.

In this paper, keeping in mind the application of the UAS for small-scale VRA on farmlands, the neural network models were made relatively shallow, thereby allowing it to be implemented on inexpensive GPUs. For these small-scale scenarios, the combination of geometric clustering for task allocation and DDQN for path planning proved to work effectively in a near-fully-distributed manner.

It is noteworthy that the overall UAV MMS was designed to be suitable for larger scale mission scenarios as well. However, while scaling, the impact of NP-hardness of path planning on both the task assignment and path planning tasks must be borne in mind and supported by greater computational resources, larger NNs, and a more efficient clustering algorithm.

To improve the quality of the paths provided to the local agents by the swarm coordinator, it is mainly recommended that the path planning problem be migrated to a Gridworld-based environment which considers only static obstacles [9]. It is envisaged that, while this would require greater development time, and could slow down training and prediction time at the GCS end, it would greatly reduce the computational requirements of the distributed local agents.

VII. CONCLUSIONS

In this paper, first, to leverage the strengths of human-machine teaming, a flexibly autonomous UAV Mission Management System was architected.

Second, based on the design and development work towards building a novel Global Mission Planner (GMP) it was observed that model-free, off-policy algorithms such as Double Deep Q-Learning enable the GMP to be adaptive to different mission types since it does not need a model of the environment as a prior and its operation can be finetuned through hyperparameters.

Overall, the prospect of using Reinforcement Learning as a Global Mission Planner for UAV swarms carrying out missions in rapidly developing avenues such as precision agriculture is promising and warrants further research.

VIII. References

[1] M. F. F. Rahman, S. Fan, Y. Zhang, and L. Chen, "A comparative study on application of unmanned aerial vehicle systems in agriculture," *Agric.*, vol. 11, no. 1, pp. 1–26, 2021, doi: 10.3390/agriculture11010022.

[2] B. Liu, "Recent Advancements in Autonomous Robots and Their Technical Analysis," *Math. Probl. Eng.*, vol. 2021, 2021, doi: 10.1155/2021/6634773.

[3] P. Radoglou-Grammatikis, P. Sarigiannidis, T. Lagkas, and I. Moscholios, "A compilation of UAV applications for precision agriculture," *Comput. Networks*, vol. 172, no. February, p. 107148, 2020, doi: 10.1016/j.comnet.2020.107148.

[4] R. W. Wohleber *et al.*, "The impact of automation reliability and operator fatigue on performance and reliance," *Proc. Hum. Factors Ergon. Soc.*, pp. 211–215, 2016, doi: 10.1177/1541931213601047.

[5] R. Altawy and A. M. Youssef, "Security, privacy, and safety aspects of civilian drones: A survey," *ACM Trans. Cyber-Physical Syst.*, vol. 1, no. 2, 2017, doi: 10.1145/3001836.

[6] A. Atyabi, S. MahmoudZadeh, and S. Nefti-Meziani, "Current Advancements on Autonomous Mission Planning and Management Systems: an AUV and UAV perspective," *arXiv*, 2020.

[7] K. Nonami, "Present state and future prospect of autonomous control technology for industrial drones," *IEEJ Trans. Electr. Electron. Eng.*, vol. 15, no. 1, pp. 6–11, 2020, doi: 10.1002/tee.23041.

[8] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[9] R. S. Sutton and A. G. Barto, "An introduction to reinforcement learning," *Decis. Theory Model. Appl. Artif. Intell. Concepts Solut.*, pp. 63–80, 2011, doi: 10.4018/978-1-60960-165-2.ch004.

[10] T. Talaviya, D. Shah, N. Patel, H. Yagnik, and M. Shah, "Implementation of artificial intelligence in agriculture for optimisation of irrigation and application of pesticides and herbicides," *Artif. Intell. Agric.*, vol. 4, pp. 58–73, 2020, doi: 10.1016/j.aiia.2020.04.002.

[11] J. Campos, M. Gallart, J. Llop, P. Ortega, R. Salcedo, and E. Gil, "On-Farm Evaluation of Prescription Map-Based Variable Rate Application of Pesticides in Vineyards," *Agronomy*, vol. 10, no. 1, p. 102, 2020, doi: 10.3390/agronomy10010102.

[12] W. P. Coutinho, M. Battarra, and J. Fliege, "The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review," *Comput. Ind. Eng.*, vol. 120, no. April, pp. 116–128, 2018, doi: 10.1016/j.cie.2018.04.037.

[13] D. Rojas Viloria, E. L. Solano-Charris, A. Muñoz-Villamizar, and J. R. Montoya-Torres, "Unmanned aerial vehicles/drones in vehicle routing problems: a literature review," *Int. Trans. Oper. Res.*, vol. 28, no. 4, pp. 1626–1657, 2021, doi: 10.1111/itor.12783.

[14] B. Alidaee, H. Gao, and H. Wang, "A note on task assignment of several problems," *Comput. Ind. Eng.*, vol. 59, no. 4, pp. 1015–1018, 2010, doi: 10.1016/j.cie.2010.07.010.

[15] M. Nieuwenhuisen and S. Behnke, "Layered mission and path planning for MAV navigation with partial environment knowledge," *Adv. Intell. Syst. Comput.*, vol. 302, no. July, pp. 307–319, 2016.

[16] G. Rudnick and A. Schulte, "Scalable autonomy concept for reconnaissance UAVs on the basis of an HTN agent architecture," *2016 Int. Conf. Unmanned Aircr. Syst. ICUAS 2016*, pp. 40–46, 2016, doi: 10.1109/ICUAS.2016.7502534.

[17] Z. ZHEN, P. ZHU, Y. XUE, and Y. JI, "Distributed intelligent self-organized mission planning of multi-UAV for dynamic targets cooperative search-attack," *Chinese J. Aeronaut.*, vol. 32, no. 12, pp. 2706–2716, 2019, doi: 10.1016/j.cja.2019.05.012.

[18] W. Yao, N. Qi, N. Wan, and Y. Liu, "An iterative strategy for task assignment and path planning of distributed multiple unmanned aerial vehicles," *Aerosp. Sci. Technol.*, vol. 86, pp. 455–464, 2019, doi: 10.1016/j.ast.2019.01.061.

[19] J. L. Sanchez-lopez, H. Bavle, and C. Sampedro, "AEROSTACK : An Architecture and Open-Source Software Framework for Aerial Robotics," 2016.

[20] M. A. Khan, I. M. Qureshi, and F. Khanzada, "A hybrid communication scheme for efficient and low-cost deployment of future flying AD-HOC network (Fanet)," *Drones*, vol. 3, no. 1, pp. 1–20, 2019, doi: 10.3390/drones3010016.

[21] A. T. Azar *et al.*, "Drone deep reinforcement learning: A review," *Electron.*, vol. 10, no. 9, pp. 1–30, 2021, doi: 10.3390/electronics10090999.

[22] X. Li, Y. Zhao, J. Zhang, and Y. Dong, "A Hybrid PSO algorithm based Flight Path Optimization for Multiple Agricultural UAVs," 2016, doi: 10.1109/ICTAI.2016.107.

[23] S. Ann *et al.*, "Area Allocation Algorithm for Multiple UAVs Area Coverage Based on Clustering and Graph Method," *IFAC-PapersOnLine*, vol. 48, no. 9, pp. 204–209, 2015, doi: 10.1016/j.ifacol.2015.08.084.

[24] V. Mnih *et al.*, "Playing Atari with Deep Reinforcement Learning," 2013, [Online]. Available: http://arxiv.org/abs/1312.5602.

[25] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," *30th AAAI Conf. Artif. Intell. AAAI 2016*, pp. 2094–2100, 2016.

[26] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, Feb. 2015, [Online]. Available: http://dx.doi.org/10.1038/nature14236.

[27] M. Theile, H. Bayerlein, R. Nai, D. Gesbert, and M. Caccamo, "UAV coverage path planning under varying power constraints using deep reinforcement learning," *arXiv*, pp. 1444–1449, 2020, doi: 10.1109/IROS45743.2020.9340934.

[28] R. Xie, Z. Meng, L. Wang, H. Li, K. Wang, and Z. Wu, "Unmanned Aerial Vehicle Path Planning Algorithm Based on Deep Reinforcement Learning in Large-Scale and Dynamic Environments," *IEEE Access*, vol. 9, pp. 24884–24900, 2021, doi: 10.1109/ACCESS.2021.3057485.

[29] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization." 2014, [Online]. Available: http://arxiv.org/abs/1412.6980.

[30] A. Paszke *et al.*, "PyTorch: An Imperative Style, High-Performance Deep Learning Library," in *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d\textquotesingle Alché-Buc, E. Fox, and R. Garnett, Eds. Curran Associates, Inc., 2019, pp. 8024–8035.

[31] D. R. Morrison, S. H. Jacobson, J. J. Sauppe, and E. C. Sewell, "Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning," *Discret. Optim.*, vol. 19, pp. 79–102, 2016, doi: 10.1016/j.disopt.2016.01.005.