Supporting UAVs Swarm Missions by Multi-Agent Reinforcement Learning

P. Fusco, L. Porcelli, F. Palmieri, and M. Ficco
Università degli Studi di Salerno
{pfusco, lporcelli, fpalmieri, mficco}@unisa.it

***Abstract*—The deployment of drone swarms is expected to increase significantly in the coming years, especially for monitoring and inspection scenarios, as well as hazardous situations, such as rescue missions in hostile or disaster-stricken areas. Smart Unmanned Aerial Vehicles (UAVs) are well-suited for these tasks due to their agility and maneuverability. However, their limited battery capacity presents substantial challenges, particularly during missions that demand complete coverage of extensive areas within a short time frame. This paper presents a Multi-Agent Reinforcement Learning-based approach for computing the coverage path of large regions by swarms of UAVs supported by mobile battery swapping stations. By implementing advanced path optimization algorithms, it is possible to enhance the efficiency and effectiveness of UAV operations. These algorithms can reduce travel distance, optimize battery swapping during the mission, minimize energy consumption, and ensure comprehensive area coverage. Incorporating real-time data and adaptive strategies can further refine path planning, enabling UAV swarms to adapt to environmental changes and mission requirements dynamically. This approach not only maximizes the operational lifespan of the UAVs, but also ensures the timely and accurate completion of critical tasks in challenging environments.**

***Index Terms*—Reinforcement Learning, Swarm of UAVs, Mobile Charging Stations, Energy Efficiency.**

I. Introduction

In recent years, technological advancements have led to a proliferation of Unmanned Aerial Vehicles (UAVs) across various civil and commercial applications, including logistic delivery, monitoring, inspection, etc. [1]–[4], [16]. UAV swarms have emerged as valuable tools in critical fields, notably the military and rescue sectors, for supporting coordinated operations within large and challenging scenarios. For instance, during disaster management and rescue efforts, small UAVs are particularly well-suited due to their exceptional maneuverability and rapid deployment. On the other hand, deploying multiple UAVs simultaneously in large and critical missions presents several challenges: (1) The swarm must efficiently cover the entire area in the shortest time, avoiding redundant path overlaps; (2) UAVs must avoid collisions with each other and with obstacles; (3) The path for each UAV must be coordinated to optimize energy consumption; and (4) Extended missions may require multiple battery replacements, leading to unnecessary energy and time waste from repeated trips to the stationary charging station.

The primary constraint on flight time for small UAVs is the limited capacity of the batteries that can be installed due to stringent space and weight restrictions. At present, the majority of small battery-powered UAVs on the market can fly for a maximum of one and a half hours [5]. This battery limitation becomes even more pronounced during missions requiring real-time processing, such as streaming and onboard imaging, or in harsh weather conditions.

Numerous strategies have been suggested to address the limitations of UAV batteries. In particular, since UAV batteries are easily interchangeable, some research has focused on integrating fixed stations along UAV flight paths to replace depleted batteries [6], [7]. On the other hand, the efficient joint coordination of UAVs and mobile charging stations remains an open research challenge, as well as effective solutions capable of recalculating their paths in real-time according to actual field conditions.

The Vehicle Routing Problem (VRP) involves efficiently dispatching a fleet of trucks with varying capacities to serve a group of geographically scattered customers. Over time, the classical VRP has evolved through numerous extensions to incorporate additional real-world factors and complexities. A recent taxonomic review of the VRP is detailed in [8]. The classical VRP is not enough for planning drone deliveries as it does not accommodate multiple trips to the depot, which can lead to an excess of drones. Furthermore, it overlooks the effects of battery and payload weight on energy consumption, resulting in costly or impractical routes, as highlighted in [9]. The study by [10] underscores the complexity of UAV routing and formalizes the UAV Routing and Trajectory Optimization Problem. In [11], a mobile ground vehicle (GV) restricted to a road network is proposed as a refueling station for UAVs. In [12], the authors examine a routing problem for UAVs monitor areas with varying accuracy requirements, determining the optimal height for each visit. Lastly, [13] focuses on rescue missions, introducing a VRP variant called the synchronized networks VRP with multiple UAVs and charging stations.

This study presents a *Multi-Agent Reinforcement Learning* (MARL)-based strategy for energy consumption optimization through the use of mobile ground stations (MBSs) for UAV battery swapping. It aims at minimizing the overall energy consumption and battery swapping while planning the path of MBSs and a swarm of UAVs tasked with complete coverage of a large area. An experimental campaign is presented, in which three different MARL-based algorithms, exploited to optimize in a coordinated manner the path planning of both the swarm of UAVs and MBSs, are compared.

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

II. Reinforcement Learning-Based Approach

Unlike other Machine Learning paradigms, such as supervised and unsupervised learning, Reinforcement Learning (RL) works in a trial-and-error fashion by interacting with its environment. Some key elements of RL are:

- *Agent*: It is a software program that interacts with the environment for learning and making intelligent decisions. In the RL context, an agent is a learner.
- *Environment*: It is the world where the agent lives and with which it interacts.
- *State and action*: A *state* is a position or a moment in the environment that the agent can be in. The state is usually denoted by s. The agent interacts with the environment and moves from one state to another by putting into effect an *action* denoted by a.
- *Reward*: Based on the performed action, the agent receives a reward. A reward is a numerical value that allows the agent to understand whether the action performed in a state is a valid choice. The reward is denoted by *r*.

In various applications, such as games and autonomous vehicle fleets, multiple agents often train concurrently while executing local policies independently ($i.e.$, without a central decision-maker). This scenario leads to *Multi-Agent Reinforcement Learning* (MARL), which presents a broader and more complex set of challenges compared to single-agent RL. MARL problems can be categorized into three groups based on the degree of collaboration and competition among agents:

- *Fully cooperative environments*: In this context, all agents in the environment work towards a common goal. Agents are credited equally for performance revealed by the environment, so there is no incentive for any of the agents to deviate from the common task.
- *Fully competitive environments*: In fully competitive environments, the success of one agent means failure for the others. Therefore, such settings are modeled as zero-sum games: $\sum_{i=1}^{Agents} g_i = 0$, where g is the reward.
- *Mixed cooperative-competitive environments*: The last type of environment involves both collaboration and cooperation between agents. These environments are usually modeled as general sum games: $\sum_{i=1}^{Agents} g_i = R$, where g is the reward and R is a type of reward that all agents can collect.

A few complex and challenging issues related to MARL are discussed below.

- *Non-stationarity*: The mathematical basis for single-agent RL is the Markov Decision Process (MDP), which assumes that the environment's dynamics depend only on the current state, not on history. This stationary environment assumption is crucial for many RL methods to converge. In MARL, the simultaneous learning and behavioral changes of multiple agents invalidate this assumption, complicating the analysis compared to single-agent RL.
- *Scalability*: The non-stationarity issue admits one possible solution, which is to take into account the actions of the other agents, such as using a joint action space. As the number of agents increases, this becomes increasingly intractable, which makes scalability an issue in MARL.
- *Unclear reinforcement learning objective*: In single-agent RL, the objective is maximizing the expected cumulative return. On the other hand, there is no such unique objective defined by MARL.
- *Information sharing*: Another important challenge in MARL is to design the information-sharing structure between all agents. There are three alternative information structures that we can consider:
  - *Fully centralized*: All the information collected by the agents is processed by a central mechanism, and the local policies leverage this centralized knowledge. The advantage of this structure is the full coordination between the agents. On the other hand, this could lead to an optimization problem that won't scale as the number of agents grows.
  - *Fully decentralized*: No information is exchanged between the agents and each agent would act based on their local observations. The obvious benefit is the absence of a centralized coordinator. On the flip side, the actions of the agents would be suboptimal due to their limited information about the environment. In addition, RL algorithms might have a hard time converging when the training is fully independent due to high partial observability.
  - *Decentralized but networked agents*: This structure would allow information exchange between small groups of (neighbor) agents. In turn, this would help the information spread among them. The challenge here would be to create a robust communication structure that would work under different conditions of the environment.

A. A Multi-Agent Environment Challenge

In the MARL context, it is possible to overcome the non-stationarity of the environment by taking into account the actions of the other agents. A Q function in a Single-Agent Reinforcement Learning (SARL) context is defined as $Q(s, a) : \mathbf{S} \times \mathbf{A} \rightarrow \mathbb{R}$, where $\mathbf{S}$ and $\mathbf{A}$ are the *state space* and the *action space* respectively, which is a function from a state-action pair to a real number that represents the agent reward. The aforementioned non-stationary problem can be overcome by defining a slightly more sophisticated Q function that incorporates knowledge of the actions of other agents. That said, it is possible to define a Q function defines as follows:

$$Q_j(s, a_j, a_{\neq j}) : \mathbf{S} \times \mathbf{A}_j \times \mathbf{A}_{\neq j} \rightarrow \mathbb{R}. \tag{1}$$

The MARL Q function for the j^{th} agent takes a tuple of the state, agent j's action and all the other agents' actions (denoted with $\neq j$) to the predicted reward for this tuple. This Q function guarantees convergence, eventually learning the optimal value and policy functions, resulting in improved

performance. However, this Q function becomes intractable as the number of agents increases due to the exponential growth of the joint action space $\mathbf{A}_{\neq j}$. The exponential growth is due to the encoding of actions using one-hot vectors. For instance, with four available actions (UP, DOWN, LEFT, RIGHT), the action UP is encoded as $[1, 0, 0, 0]$.

The agent policy, defined as $\sigma(s) : \mathbf{S} \rightarrow \mathbf{A}$, is a function that takes a state as input and returns an action. The exponential growth is due to the representation of the joint action. In an environment with two agents and four actions available for each agent, the joint action length is $4^2 = 16$. Generally, the size of the joint action vector is $|A|^N$, where $|A|$ is the cardinality of the action space and N is the number of agents. This results in a vector of exponential growth in the number of agents, which becomes impractical and unmanageable as the number of agents increases.

B. A Multi-Agent Environment for a Swarm of Drones

In multi-agent environments, interactions among autonomous agents present unique challenges and opportunities. These environments often involve multiple agents with different goals, capabilities, and information, necessitating sophisticated coordination and negotiation mechanisms for effective collaboration. Key considerations include:

- (i) how agents perceive and reason about their environment and neighboring agents,
- (ii) how they communicate and exchange information, and
- (iii) how they adapt their strategies in dynamic and uncertain settings.

The design and analysis of multi-agent systems are further complicated by the emergence of complex behaviors and collective intelligence arising from agent interactions. Tackling these challenges demands a thorough understanding of agent behaviors, interactions, and environmental dynamics, as well as the development of innovative algorithms and frameworks for efficient, scalable coordination.

In the considered scenario, the environment contains both the UAV agents and the MBS agents. For the former agents, the environment has been created by dividing the area to cover as a grid of dimension $N \times M$, where N and M represent respectively the number of rows and columns. The grid world contains unexplored cells marked in white and explored ones marked in black. The first simulation has been put into effect using three UAVs, represented by the red, green, and blue circles as shown in Fig. 1. For the MBS agents, the scenario remains completely unchanged. It is worth noting that the Q function related to the MBS agents needs to take into account the information concerning all the MBS agents more than the information related to all the UAVs. This necessity significantly complicated the simulation, limiting the number of UAVs used in the simulation itself.

The agents provide one of the possible movement actions to their respective UAVs. The agents decide whether an action is legal or illegal. For example, an action of moving `LEFT` when the UAV is located next to the left boundary of the environment is deemed illegal. Similarly, actions for colliding against obstacles and other agents in the environment are illegal actions and hence draw penalties. The overall goal is to explore all cells within a fixed time frame.

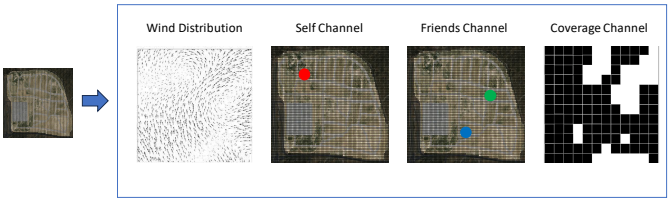

Fig. 1: Drone's agent input.

At each time step, a UAV agent observes the environment using an $N \times M \times 4$ image. Each of the four channels in the image captures the following information in order:

- Cells containing wind information that has been generated by using a numerical simulation;
- Current position of the UAV (self) being controlled;
- Position of other (friend) UAVs;
- Cells that have been explored during the episode.

At each time step, UAV agents receive the following rewards:

- +2 for moving to a previously unexplored cell (white in Fig. 1);
- -3 for an illegal action (attempt to move outside the boundary or collide against other UAVs or obstacles);
- -1 for an action that results in no motion (lazy penalty);
- -0.5 for moving to an explored cell;
- +100 if the grid is fully explored.

At the same time, any MBS agent receives as input a vector of numbers containing all the information related to the actual state of charging of all UAV agents plus information relative to the UAV agent's position. At each time step, MBS agents receive the following rewards:

- +2 for moving towards the UAV agent with the minimum state of charging;
- -2 for moving away from the UAV agent with the minimum state of charging;
- -1 for moving towards any other MBS agent;
- -3 for an illegal action (attempt to move outside the boundary or collide against other MBS agents);
- +100 if a UAV agent can reach the closest MBS agent with the charging state remaining.

For the MARL agent simulation only, various algorithms designed for discrete action spaces have been tested. The Advantage Actor-Critic (A2C) algorithm was initially employed to establish a benchmark. Subsequently, the Asynchronous Advantage Actor-Critic (A3C) was utilized for further evaluation. Among the tested algorithms, the Proximal Policy Optimization (PPO) achieved the best performance.

C. MARL Simulation Results

As previously mentioned, the simulation has been realized by using three UAV agents, as illustrated in Fig. 2, by using a

red, green, and blue dot and two MBS agents. The environment has been created by using OpenAI's Gym library called Gymnasium [14], whose interface is simple and capable of representing general RL problems, and has a compatibility wrapper for old Gym environments. All the other components have been put into effect using Tianshou [15], which is an RL platform based on pure PyTorch. Tianshou provides a fast-speed framework and Pythonic API for building the deep reinforcement learning agent.

At the start of the simulation the ϵ parameter, which governs the agent's exploration strategy, may lead to some invalid moves as the agent relies on random actions to gather initial information. Over time, ϵ is adjusted dynamically based on problem-specific trends to refine the agent's policy.

Tab. I provides a summary of the average MARL training times for the involved agents as the dimension of the *state space* increases. Specifically, the board on which the UAV drones operate is divided into cells along the x and y axes, with the number of cells corresponding to the values listed in the aforementioned table.

TABLE I: Execution time of the MARL training.

Scenario	Exec. time (s)
Grid $(16 * 10^3)$	0.0648
Grid $(24 * 10^3)$	0.1269
Grid $(32 * 10^3)$	0.3402
Grid $(36 * 10^3)$	2.17755
Grid $(48 * 10^3)$	58.8573
Grid $(56 * 10^3)$	339.7351
Grid $(60 * 10^3)$	2055.7082
Grid $(64 * 10^3)$	3354.00

Fig. 2 shows a result of the MARL simulation. The UAV used in the simulation have been capable of carrying out their mission covering the entire initial area to explore.

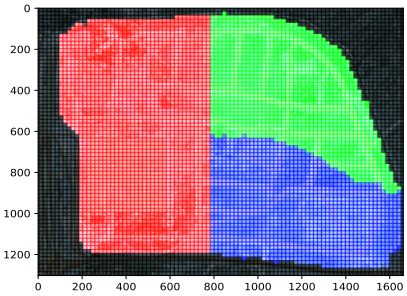

Fig. 2: Multi-Agent covered area.

Fig. 3 illustrates the convergence behavior of the three drone agents during the MARL simulation, comparing the results obtained from the three considered algorithms. The simulation spans 500 episodes, providing sufficient time for the agents to converge. Fig. 4 shows the variation in episode length throughout training. This metric converges to a minimum by the end of training, depending on the algorithm used. Initially, agents rely heavily on exploration, navigating to gather information. Over time, as experience accumulates from

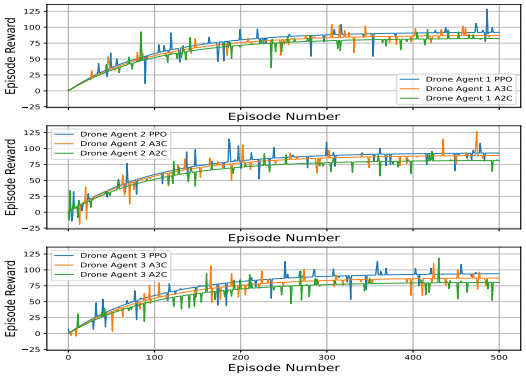

Fig. 3: Episode reward for each drone agent.

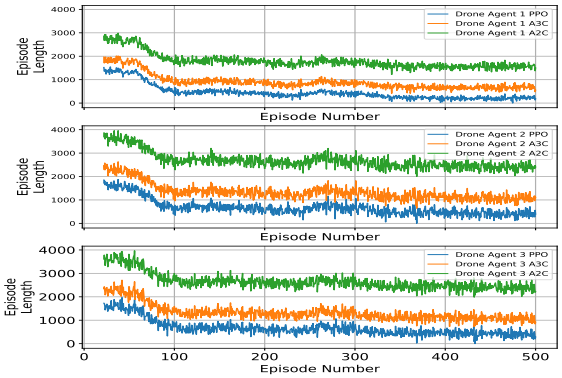

Fig. 4: Episode length for each drone agent.

previous episodes, agents become more efficient, navigating with greater awareness and making fewer errors.

Fig. 5 shows the average cumulative energy used by UAVs, while Fig. 6 shows the average mission time. Both previous figures compare the three algorithms against the number of involved UAVs.

Finally, Fig. 7 highlights the convergence patterns of the two MBS agents operating on the ground, while Fig. 8 depicts the episode lengths during their training. These results reflect the performance differences across the applied algorithms.

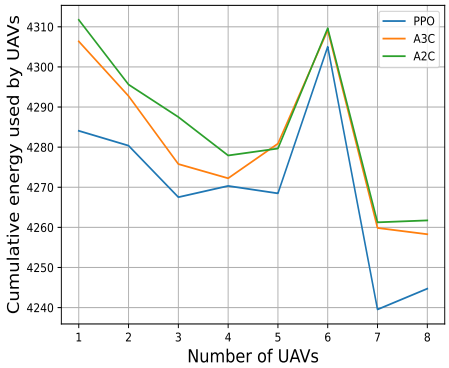

Fig. 5: MARL UAVs cumulative energy.

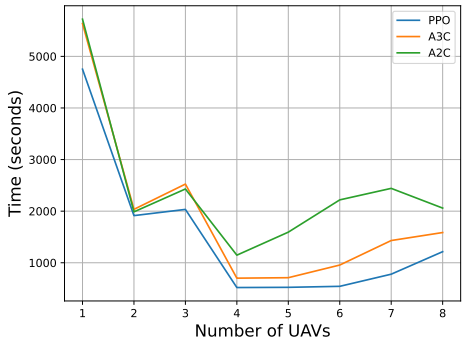

Fig. 6: MARL UAVs mission time.

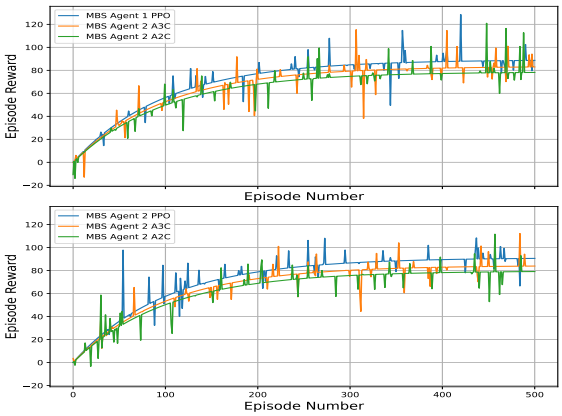

Fig. 7: MBS episode reward.

III. Conclusions

Regarding the resolution of the problem using a MARL approach, it uses a stochastic approach. The training stage requires a set of parameters, a.k.a. *hyperparameters*, which have to be tuned to find the correct set of values that permits obtaining the best performance. For the training stage, a valid dataset has been created by using our simulation framework [16], to permit the MARL system to properly generalize the flying and ground conditions. As mentioned in Sec. II-A, the exponential growth depends on the way the joint action is represented. Hence, the size of the joint action vector will be $|A|^N$, where $|A|$ represents the cardinality of the action space, while N is the number of agents. This is a vector of exponential growth against the number of agents, which is impractical and almost unmanageable for any significant number of agents.

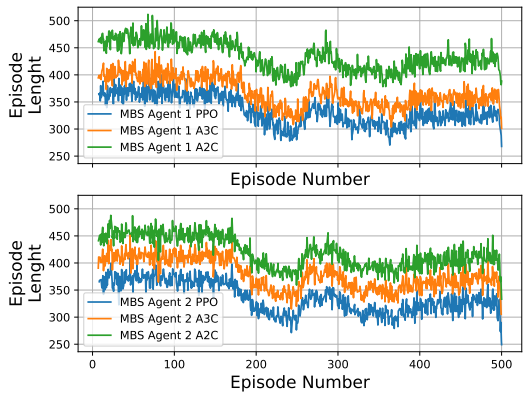

Fig. 8: MBS episode length.

Acknowledgments

This work was supported by project SERICS (PE00000014) under the NRRP MUR program funded by the EU - NGEU.

References

[1] Guerrero, J.A., Bestaoui, Y.: Uav path planning for structure inspection in windy environments. Journal of Intelligent & Robotic Systems **69**, 2013, pp. 297–311.

[2] Nikhil, N., Shreyas, S., Yadav, S.: Unmanned aerial vehicles (UAV) in disaster management applications. In Proc. of the 3th Int. Conf. on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 140–148.

[3] Song, B.D., Park, K., Kim, J.: Persistent UAV delivery logistics: Milp formulation and efficient heuristic. Computers & Industrial Engineering, vol. 120, 2018, pp. 418–428.

[4] Ficco, M., Palmiro, R., Rak, M., Granata, D.: Mavlink protocol for unmanned aerial vehicle: Vulnerabilities analysis. In the IEEE Int. Conf. on DASC/PiCom/CBDCom/CyberSciTech, 2022, pp. 1–6.

[5] Boukoberine, M.N., Zhou, Z., Benbouzid, M.: A critical review on unmanned aerial vehicles power supply and energy management: Solutions, strategies, and prospects. Applied Energy, vol. 255, 2019, 113823.

[6] De Silva, S.C., Phlernjai, M., Rianmora, S., Ratsamee, P.: Inverted docking station: A conceptual design for a battery-swapping platform for quadrotor UAVs. Drones **6**(3), 56, 2022.

[7] Porcelli, L., Ficco, M., D'Angelo, G., Palmieri F.: Context-Aware Coverage Path Planning for a Swarm of UAVs using Mobile Ground Stations for Battery-Swapping. Soft Computing, 2025, Accepted, in press.

[8] Braekers, K., Ramaekers, K., Van Nieuwenhuyse, I.: The vehicle routing problem: State of the art classification and review. Computers & industrial engineering, vol. 99, 2016, pp. 300–313.

[9] Dorling, K., Heinrichs, J., Messier, G.G., Magierowski, S.: Vehicle routing problems for drone delivery. IEEE Transactions on Systems, Man, and Cybernetics: Systems, **47**(1), 2016, pp. 70–85.

[10] Coutinho, W.P., Battarra, M., Fliege, J.: The unmanned aerial vehicle routing and trajectory optimisation problem, a taxonomic review. Computers & Industrial Engineering, vol. 120, 2018, pp. 116–128.

[11] Maini, P., Sundar, K., Singh, M., Rathinam, S., Sujit, P.: Cooperative aerial–ground vehicle route planning with fuel constraints for coverage applications. IEEE Transactions on Aerospace and Electronic Systems, **55**(6), 2019, pp. 3016–3028.

[12] Zhen, L., Li, M., Laporte, G., Wang, W.: A vehicle routing problem arising in unmanned aerial monitoring. Computers & Operations Research **105**, 2019, pp. 1–11.

[13] Ribeiro, R.G., Cota, L.P., Euzébio, T.A., Ramírez, J.A., Guimarães, F.G.: Unmanned-aerial-vehicle routing problem with mobile charging stations for assisting search and rescue missions in postdisaster scenarios. IEEE Transactions on Systems, Man, and Cybernetics: Systems, **52**(11), 20121, pp. 6682–6696.

[14] Towers, M., et al. Imitation Learning Datasets: A Toolkit For Creating Datasets, Training Agents and Benchmarking, 2024.

[15] Weng, J., Chen, H., Yan, D., You, K., Duburcq, A., Zhang, M., Su, Y., Su, H., Zhu, J.: Tianshou: A highly modularized deep reinforcement learning library. Journal of Machine Learning Research **23**(267), 2022, pp. 1–6.

[16] Rimoli, G.P., Ficco, M., Pascarella, D., Castrillo, V.U.: Security Assessment of Drone Teams and Swarms Using an Extended SecRAM Methodology: In Proc. of the 14th Int. Defense and Homeland Security Simulation Workshop (DHSS 2024), 2024, pp. 1-5.