Assessing Multi-Agent Reinforcement Learning Algorithms for Autonomous Sensor Resource Management

Trevor J. Bihl, Aaron Jones
Air Force Research Laboratory, USA
{Trevor.Bihl.2, aaron.jones.41}@us.af.mil

Patrick Farr
Applied Research Solutions, USA
pfarr@appliedres.com

Kayla Straub, Brian Bontempo, Frank Jones
The Stratagem Group
{kayla.straub, frank.jones, brian.bontempo}@stratagemgroup.com

Abstract

Unmanned aerial vehicles (UAVs) have applications in search and rescue operations and such operations could be more efficient by using appropriate artificial intelligence (AI) to enable a UAV agent to operate autonomously. Sensor resource management (SRM), which leverages capabilities across location intelligence, facilitates the efficient and effective use of UAVs and their sensors to complete a set of tasks. Furthermore, multiple UAVs, each with different sensor configurations, must be considered when maximizing mission effects. Instantiating operational autonomy for such teams requires considerable coordination. One AI approach relevant to this task is multi-agent reinforcement learning (MARL). However, MARL has seen limited prior use in SRM. This work evaluates the trade-space of MARL algorithms with respect to performing heterogeneous sensor resource management (SRM) tasks, considers the concept of evaluating MARL in a test and evaluation framework and compares a suit of algorithms with random and Bayesian hyperparameter optimization methods.

1. Introduction

Location intelligence involves the complex intersection of fields such as analytics, geographic information science, routing/scheduling, remote sensing, visualization, operations research, and computer science to solve different problems [1]. One particular problem that overlaps these areas is the tasking and routing of sensor platform for search and rescue (SAR). where big data sources, remote imagery, and unmanned systems must find the target in a timely manner. Inherently, SAR tasks involve the interaction between and tasking of sensors and unmanned assets, particularly unmanned aerial vehicles (UAVs). Such applications require sensor resource management (SRM) which involves data exploitation (e.g., remote sensing object recognition), tasking, scheduling, planning, routing, and optimization [2].

SAR with UAVs has the potential to revolutionize the location intelligence field through edge processing for the real-time use of multiple sources of data to quickly find individuals in need [3]. However, challenges exist in SRM which enables the efficient use of data, coordinating operations when developing or deploying appropriate search algorithms for a region of interest that has minimal manual/human intervention.

While SAR actions can be considered for a single UAV, to quickly search a large area multiple assets are needed. Thus, multiple-agent SRM is needed. SRM involves developing a decision approach to create an effective policy for any given sensor operation. Reinforcement learning (RL) enables an agent to gather data through interaction with their environment with rewards given based on choice outcomes. From their collected experience, agents learn to maximize positive rewards and avoid negative rewards. RL has been applied to real world robotics controls [4], video games [5], and even financial markets [6]. UAV and search and rescue operations are conceptually similar to both robotics controls and video game operations whereby both learn from sparse and/or time-delayed rewards with actions coming from a large or complex space. Additionally, UAV and SAR operations often balance exploration and exploitation of data gathered as well as handle imperfect/incomplete information, similar to many video games in which RL has been successfully applied.

A key tradeoff of RL algorithms is exploration and exploitation. Exploration describes the process of trying new actions to learn more about the reward function and potentially find better solutions, while exploitation describes the process of using previously gained knowledge gained by exploring to make the current best choice given what is known about the options [7]. While such a problem has direct parallels to typical RL problems, RL has seldom been applied to sensor resource management, e.g. [8].

Many search and rescue operations are instantiated with multiple UAVs, and thus, are beyond simple RL. In such cases, multi-agent RL (MARL), see [9], is needed whereby the swarm/group of UAVs gains

URI: https://hdl.handle.net/10125/80034
978-0-9981331-5-7

rewards as they operate in a collaborative manner. UAVs can be outfitted with different sensors and capabilities, for instance consider two UAVs operating together: one UAV with a high resolution narrow field of view camera and another with a low resolution and wide field of view camera. Thus, of particular importance to UAV search and rescue operations are MARL approaches which can permit cooperation between heterogeneous agents.

Prior work, c.f. [10] [11], proposed RL for SAR applications. However, multiple heterogeneous agents trained via MARL and/or a broad understanding of the performance of the currently available MARL algorithms were not considered. Prior MARL comparisons are limited, c.f. [12] [13] [14] [15], consider only a small number of MARL algorithms [12] [14], focus on deep learning MARL algorithms only [13], or focus on only a small set of not commonly available algorithms [15]. Additionally, none of these MARL comparisons considered SAR tasks.

This work primarily aims to answer the following questions:

R1) Can MARL algorithms be applied to the general SRM for SAR task?

R2) What MARL algorithms provide utility for SAR and SRM tasks?

R3) Can an experimental framework be created to test and evaluate MARL algorithms as they become available?

The contributions of our paper are thus, as follows. This work introduces the concept of MARL for autonomous SRM with heterogeneous sensor agents which enable SAR through UAVs for location analysis. This work expands upon prior studies by comparing 9 MARL algorithms for SRM of heterogeneous agents, encompassing all of the readily available model-free MARL algorithms in code. The importance of the study is that it demonstrates the utility of model-free MARL algorithms for SRM in unknown environments. To evaluate MARL algorithms, this work further develops a repeatable experimental design framework and ecosystem to compare these algorithms with hyperparameter optimization (HPO) to provide effective algorithm performance evaluation and facilitating future comparisons as both new algorithms and scenarios come available.

2. Background

Control of UAVs ranges from full manual operations via remote control to a potential future of fully autonomous operations [16]. Higher levels of UAV autonomy provide a reduction in operator demands and could enable an operator to control multiple UAVs and/or enable UAVs to complete missions with limited oversight. As UAVs increase in application and use, autonomy is central to providing rapid responses to emerging events in a complex operational space, such as SAR. Critical to enabling fully autonomous UAV operations are the location intelligence capabilities that include planning, sensor data exploitation, and sensor data fusion. Collectively, the intersection of these areas is known as SRM.

2.1. Sensor Resource Management (SRM)

SRM encompasses sensor tasking which considers the resources and capabilities of each sensor that is available for a sensing task [17]. This includes sensor scheduling, planning, task oversight and associated sensor platforms [17] [18]. SRM is conceptualized in Figure 1 whereby high-level tasks are determined by a human operator. As discussed in [18], tasks from the operator are processed by the sensor manager, which then compares the tasks against the current system state(s). The prioritized tasks are then scheduled with appropriate sensor/platform plans. Sensor data is processed by executing these plans/actions and the processed sensor data is used to generate/update an operating picture to both the system and the operator.

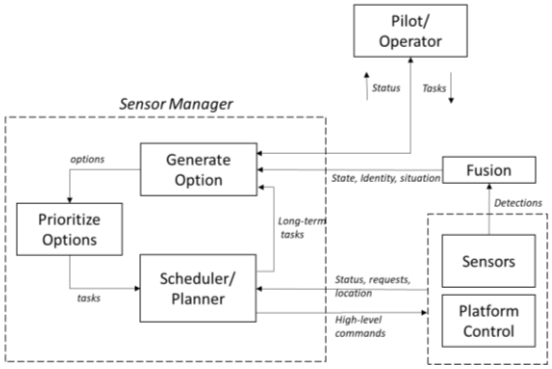

Figure 1. Sensor Resource Management (SRM) in a nutshell, adapted from [18]

A wide variety of approaches to SRM exist, including operations research and decision theoretic approaches [2] [17]. One way of considering the SRM problem is the need to use a computational decision approach to create an effective policy or decision for a given sensor exploitation task [8]. When stated this way, SRM sounds akin to a typical RL problem, but RL has seen limited prior use in SRM [8] [17].

2.2 Reinforcement Learning

The general goal, as conceptualized in Figure 2, of RL is to learn a policy that maximizes a given reward, R_t, based on an agent's action, A_t, and its impact on the

measured environmental state, S_t for t time [19]. This is accomplished by defining the reward function and then allowing an agent to collect data against this reward function by interacting with the environment; the agent is trained to adjust its policy to maximize the reward [20]. The reward is a function of the action, selected by a policy, and the state.. The formulation of the problem and the optimization methodologies applied to solve it are where approaches to RL generally differ.

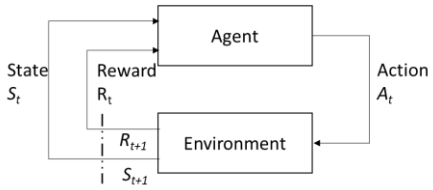

Figure 2. General Conceptualization of RL

2.3. Multi-Agent Reinforcement Learning

Multi-agent RL (MARL) extends RL to consider multiple agents that each receive a reward [9]. Often such agents are required to cooperate to accomplish a task [9]. A common convention is to include information about the other agents' states in the state information passed to any given agent participating in the overall task. This significantly extends general RL whereby MARL considers multiple agents and maximization of the shared rewards within a shared environment. The advantage of MARL is that a decentralized team of agents can be created to independently work towards collective goals. This is conceptualized in Figure 3 where there are individual states, rewards, and actions for each agent, but a shared environment. However, training MARL agents introduces obvious complexities when compared to single-agent RL, i.e. partial observability and environment nonstationarity [9].

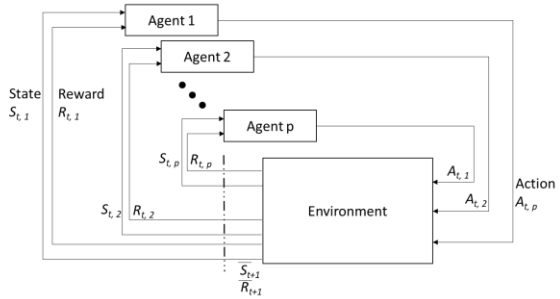

Figure 3. General Conceptualization of Multi-Agent RL (MARL) Problems

To simplify the solution space, MARL algorithms often assume that the agents have similar in capabilities. However, Real world UAV assets often have different sensors and capabilities and thus MARL for heterogeneous assets is of concern for real-world SAR applications [9]. Thus, due to the sequentially added complexity, not all RL algorithms can learn MARL problems without modifications, and fewer still can consider heterogeneous agents.

2.4. RL Algorithms

Extending beyond these general concepts are the algorithmic approaches to RL. As conceptualized in Figure 4, RL methods are generally divided into two high-level categories: model-based and model-free. Model-free learning focuses solely on maximizing the reward by learning to choose the optimal action for a given observation of the environment; model-based RL attempt to predict the output from the environment that they act in, rather than directly training to maximize reward from actions [20]. Due to these expectations and a desire for general applications to any environment, this work does not consider model-based approaches and focuses on mode-free approaches. While other taxonomies of RL exist, see [9], this general taxonomy will facilitate the MARL relevant algorithm space discussed herein.

Model-free approaches can be divided into value-based, policy-based, and hybrid (both value and policy) methods. Under this taxonomy, many RL algorithms exist, as seen in Figure 4 and undoubtedly more RL algorithms exist than are seen in this figure. The selection in Figure 4 is expansive, but not exhaustive; we highlight the examples that are both more cited and more accessible in software form.

Within the taxonomy of Figure 4, not all RL algorithms are equally capable when extended to multi-agent considerations. Some RL algorithms, e.g. CMA-ESX, cannot handle multiple agents. Then, of the remaining RL algorithms that can handle multiple agents, only some can handle heterogeneous agents; these are indicated in Figure 4 with an asterisk.

2.5. MARL with Heterogeneous Agents

Considering the available algorithms which meet our constraints, we are left with the following possible algorithms for consideration:

1. Value-based: DQN, DDQN, C51/Rainbow
2. Policy-based: PPO, PG, CMA-ES
3. Hybrid: DDPG, TD3, SAC, A2C, A3C, IMPALA
4. Model-based: Dyna, PlaNet

Model-based methods were removed from consideration because of their lack of availability or inability to handle discrete action spaces. TD3 and DDPG are designed to work in continuous action spaces only, for some applications operating in a

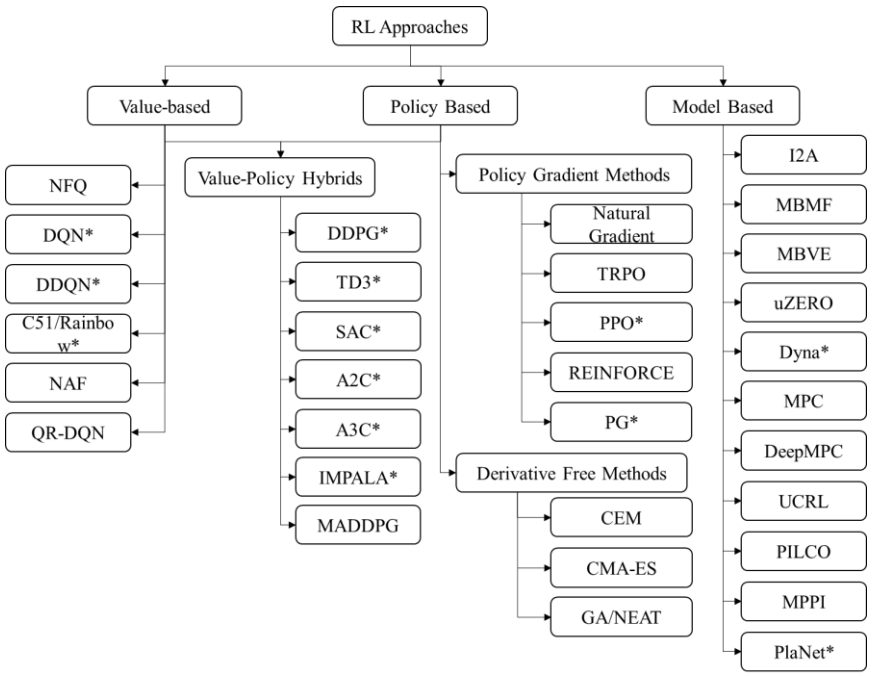

Figure 4. General taxonomy of families of RL approaches, annotated (*) for algorithms with both heterogeneous MARL capability and availability.

continuous vs discrete space matters and these algorithms were not included. CMA-ES is available in common algorithm repositories, e.g. RLLib, but not for multi-agent environments. Thus, we considered the nine following algorithms in this study: PG, Rainbow, A2C, A3C, SAC, DQN, APPO, and IMPALA.

2.5.1. Value-based Methods All value-based models considered in the selection in Section 2.5 were variants of Q-learning. Q-learning uses the reward function to optimize predictions regarding the value of each state action pair [20]. The agent then selects the action given the state as the input to maximize reward. In the Deep Q-learning Network, the Q-learning algorithm employs an underlying neural network structure [21].

Many modifications to the base DQN model have been proposed and both [22] and [23] review this literature further. Of interest herein is Dueling DQN (DDQN), which look at both the value of being in a particular state and the advantage of a given action vs a different action [24]. This allows the network to contain a representation that assesses whether an action given a particular state leads to greater reward, or if the state itself is what leads to greater reward. The two DDQN variants we will consider are Rainbow DQN, which applies multiple RL variants jointly [25], and Ape-X DQN, which uses an experience replay to allows for scalable, distributed prioritization of experience use in training [26].

2.5.2. Policy-based methods. In contrast to value-based methods, policy-based methods view the policy as a function with an environment episode as input, and reward as output. Over the course of an episode, the policy receives reward for certain actions in given states, and policy-base algorithms use gradient ascent on the entire policy to update the policy's tunable parameters [27]. A simple policy-based method, Policy Gradient (PG), implements this approach. PG methods of interest herein are Proximal Policy Optimization (PPO) which optimizes the policy with respect to its local surroundings, rather than the global maximum [28], and Asynchronous PPO (APPO), with an asynchronous architecture applied to PPO [29].

2.5.3. Value Policy Hybrid Models. Hybrid models use both a value-based component and a policy-based component. All hybrid models tested in this paper were Actor-Critic models. Actor-Critic methods use a value-based critic to analyze different states and actions, while simultaneously having a policy-based actor that finds a policy with the best reward based on the critic's estimate. This way, the value function doesn't get stuck, and the policy can have a better/faster direction to optimize rewards [30]. Methods of interest are Advantage Actor-Critic (A2C), which modifies the Q-learner by training it to predict the advantage of a given action in a given state [31], Asynchronous Advantage Actor-Critic (A3C), which runs different versions of the actor parameters

asynchronously in in identical environments [31], Soft Actor-Critic (SAC), which focuses on maximizing the entropy object, rather than reward directly [32], and IMPALA, which introduces a training scheduler, V-trace, to improve agent learning [29].

2.6. Hyperparameter Optimization (HPO)

As it becomes increasingly difficult to hand-craft algorithms which yield reliable results, see the variety of hyperparameters for the MARL algorithms in Table 1, hyperparameter optimization (HPO) is of interest [33]. A general taxonomy of HPO is presented in [33], which is largely separated into model-free and model-based approaches. Model-free approaches range from 1) grid searches, 2) haphazard/expertise based, to 3) random searches [33]. Due to its simplicity, the random HPO method can provide a baseline comparison [34]. In contrast to model-free, model-based approaches employ a wrapper with a greedy search approach around the algorithm of interest [33].

Of particular interest in model-based HPO is Bayesian Optimization (BO) which is one of the most effective and efficient HPO methods [33]. One variation is known as Sequential Model-based Bayesian Optimization (SMBO) which uses data from previously tested parameters whereby it estimates the probability that a set of parameters will result in a high-scoring model; parameters are selected based on an acquisition function which evaluates expected improvement over the prior best result [35].

Additionally, and of critical interest to RL, is how long HPO trials are allowed to run. Thus, stopping a trial that is poorly performing is beneficial and HPO approaches that enable such assessments are of interest. The successive halving algorithm (SHA) is one such approach whereby parallelization is used to evaluate multiple parameter combinations in parallel with under-performing candidates terminated early in processing [36]. Asynchronous SHA (ASHA) extends SHA, which has been shown to have performance advantages over SHA [37].

Of interest herein is Bayesian ASHA (BASHA), which combines the advantages of both SMBO and ASHA whereby the informed SMBO searches find good settings and ASHA quickly terminates poor-performing candidates; in operation BASHA functions by iterating between SMBO and ASHA.

2.7. Quantifying Algorithm Performance

RL algorithm performance is based on maximizing the reward received, for instance, getting the highest score in a game. However, RL has been known to result in unsafe and useless actions which, nevertheless, maximize rewards [38]; such approaches are akin to a toddler following the rule "don't get your clothes dirty" by taking off said clothes and then getting themselves dirty. Thus, beyond the primary stated reward, ancillary metrics, such as raw performance values, are also of interest in order to provide another assessment of overall performance.

2.7.1. Observational Metrics. Fundamental to RL are observed rewards that correspond to agent state. In many applications, time is also an observed reward component, i.e. completion time and total time used.

2.7.1. Reward Metrics. RL algorithm performance is based on maximizing the reward received. In some instances, i.e. games, this is predefined based on score and thus RL involves maximizing this score. Rewards for robotics or real-life scenarios can be tricky since simulations may not have all the nuances of the real world and similarly all possible outcomes might not be predictable. When considering multiple agents, the problem is compounded. Current work involves understanding the action and reward spaces and developing metrics that can include multiple conditions.

3. Evaluation Ecosystem

To facilitate the current and future evaluation of RL and MARL algorithms for sensor resource management, an evaluation ecosystem was developed. This ecosystem consists of general agents, worlds to explore, and a quickly integrateable interface to try new algorithms as they become available, conceptualized in Figure 5. The environment was built from the OpenAI Gym base environment class [39].

3.1. Repeatable Ecosystem

The general approach is further wrapped up into a repeatable ecosystem to address R3. This ecosystem is conceptualized in Figure 5. The ecosystem allows one to select various tunable parameters in a study, including 1) environmental/experimental settings, then 2) the algorithm and its associated libraries, and 3) the hyperparameter optimization and research design process. The ecosystem allows for an arbitrary scaling of grid size, duration of the scenario, number and types of agents in the scenario, and number of targets.

A custom grid-world environment is built from the OpenAI Gym base environment class. The time-step corresponds to the duration of time that passes between two actions that a given agent takes. In the current instantiation, only discrete action spaces are considered

Table 1. Hyperparameters Associated with RL Algorithms of Interest

	ALGORITHM								
HYPERPARAMETERS	DQN w/ Ape-X	RAINBOW DQN	PPO	IMPALA	APPO	PG	A2C, A3C	SAC	DESCRIPTION
ANN Architecture	X	X	X	X	X	X	X	X	Number and size of layers in the network
ANN Learning Rate	X	X	X	X	X	X	X	X	Training step size
RL Discount factor, γ	X	X	X	X	X	X	X	X	Weight assigned to future rewards
RL Gradient Clip	X	X	X	X	X		X	X	Maximum global norm for each gradient calculated
Value function and Entropy Coefficients			X	X	X		X		Number and size of layers in the network
Memory Buffer Size	X	X							Number of experiences to save and sample from for learning
Epsilon Time-steps	X	X							Time-steps over which to anneal epsilon for exploration
Learning Start Time-step	X	X							Number of steps to infer before training
Prioritized replay	X	X							Boolean to use the prioritized replay strategy [40]
Target Network update frequency	X	X							Number of steps between updates to the target network
Clip Parameter			X		X				Parameter to set maximum clip of the objective function
Use V-trace				X	X				Boolean to use V-trace algorithm [29]

and it is a simplification that allows for quick prototyping of possible scenarios.

The custom environment logs information about reward, agents, time spent training, hyperparameters selected, etc. HPO can then monitor the environment state for algorithms that may require information about reward received for any given trial in order to select the next hyperparameter(s) iteration. The process in Figure 5 can be used in various means: single runs can be explored to test the feasibility of concepts, or an experimental design can be wrapped around the ecosystem.

3.2. Custom Environments and Agents

A general UAV search and rescue operation can be visualized in terms of a grid-based world. This mimics the general representation provided by UAV sensor feeds after processing and include both road segments and targets/objects of interest. The two types of agents of interest were developed for the ecosystem:

1) UAV_{LR}—a low resolution screening agent which searches a large area.
2) UAV_{HR}—a high resolution agent which has a smaller sensor collection area.

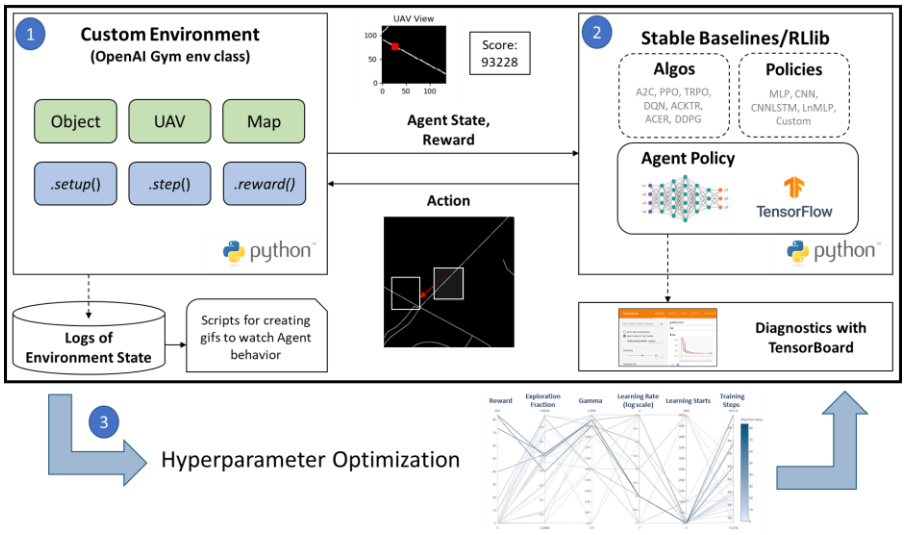

Figure 5. Developed ecosystem to train, evaluate, optimize, and assess RL agents

A general grid world road network is presented in Figure 6 with the road network in white, the target as a red square, and the agents represented by their fields of view: orange for the low resolution agent and blue for the high resolution agent.

Figure 6. Grid environment with examples of sensor footprints of two types of scanning UAVs.

Observations in this world are composed of two parts. The first part is the coordinate position of a UAV on top of a map. The second part of the observations is a binary flag that indicates if the target is in the agent's field of view. In the experimental framework, the target can be instantiated randomly or in a specific location; similarly, the agents can be specified to start at a specific point, notionally the center of the map. Thus, multiple replications could be explored for the same grid world but with the target appearing at a random point for each iteration.

4. Experimentation and Results

The experimentation's general ecosystem follows the ecosystem in Figure 5 where environments can be setup quickly (1) with considerations made to ensure a broad comparison of the MARL algorithms. Computationally, in Python, Rllib [41] agents were used for training, and HyperOpt was used for HPO.

4.1. Experimental Setup

To test R1 and R2, an experiment was constructed that employed the selected MARL algorithms and employed the two types of UAV agents. The challenge is to properly balance reward for finding the target with UAV_{LR}, with reward for finding the target with UAV_{HR}.

4.1.1. Reward Function. The reward function needs to be designed with the general goal to test each MARL algorithm's ability to learn strategies for heterogenous sensor agents. This was designed to balance the advantages of both agents with a small, dense reward for each timestep for UAV_{LR} with the primary objective for this team being for the UAV_{HR} to locate and verify the target, either randomly or after being cued by the $UAV_{LR.}$ The reward given for completing that task is greater but is only provided once. When the target is found, i.e. the task is accomplished, the episode ends. The overall reward is equal to the number of steps taken to locate the target subtracted from the maximum episode length. This is presented as:

$$R_t = (L_e - t) \parallel (UAV_{HR}) + 0.25 \parallel (UAV_{LR}) \quad (1)$$

where L_e is the maximum episode length, t is the current timestep, and UAV_{HR} and UAV_{LR} are binary when the respective agent has the target in view. For each timestep the target goes undetected, this potential reward value in (1) is decremented by one. The UAV_{HR} reward is implicitly 1.0 and the UAV_{LR} reward is set to less than 1.0 for reasons mentioned above; through hueristic-based tuning 0.25 was found to provide reasonable resutls.

4.1.2. Grid World Scenario. The grid world employed for this experiment was a 1609x1609 pixel map. While both UAVs are searching the entire area, if the UAV_{LR} finds a target, the binary target flag is used to communicate its presence to the UAV_{HR}. The goal is for the UAV_{LR} to search the area efficiently, then stay at the target's location when found. The UAV_{HR} should learn to search efficiently, but then learn to approach the UAV_{LR} if the UAV_{LR} spots the target. The episode ends if UAV_{HR} finds the target, and the agents receive a final large reward based on how quickly UAV_{HR} found the target.

4.1.3. Experimental Considerations. The overall experiment aimed for repeatability as laid out by [42]. The process in Figure 5 was employed throughout with a common reward function (4.1.1) and common scenario for all experiments (4.1.2). Each selected MARL algorithm (2.5) performed for 100 different trials (a trial being RL training with a given selection of hyperparameter values) for both random and BASHA. Each trial was then trained on the scenario for 200 environment runs (a run being the environment started, the target randomly placed). Subsequent to 100 runs per algorithm, qualitative and quantitative analysis leads to a threaded discussion and conclusion where both note many significant insights of the research.

For this experimental study, the sensor footprint of UAV_{LR} was set to 320x160 pixels and the sensor footprint of UAV_{HR} was set to 80x80 pixels. Both

UAVs are permitted to move in one of 4 directions, with a movement of 80 pixels per timestep in the direction they choose. For the grid environment under consideration, 1609x1609 pixels, this movement corresponds to the map being a 20x20 grid. Thus, the maximum episode length, L_e, was set to 70 in (1), which is slightly fewer timesteps than needed for the UAV_{HR} to individually search the whole space. The system is further randomly initiated with the target being randomly placed in each episode. Since the target location is unknown, logically, the UAV_{LR} should search possible locations first with the UAV_{HR} searching towards a middle region of the map.

4.2. Hyperparameter Design Region Considerations

As noted in [33], while HPO removes the general problem of finding effective algorithm settings, manual intervention is needed to bound the search region. Table 2 presents the general search regions considered for all algorithms. HPO can find continuous valued parameters anywhere within the specified intervals whereas categorical parameters must be one of the specified search values.

Table 2. General Hyperparameter Search Region for MARL Algorithms

Parameter	Interval Scale	Initial Search Interval
ANN Learning Rate	Logarithmic	$[10^{-7}, 10^{-2}]$
ANN Architecture: Number of Hidden Layers	Categorical	[1, 2, 3]
ANN Architecture Layer Size (selected per layer)	Categorical	[64, 128, 256, 512]
Discount Factor	Linear	[0.9, 0.9999]
Gradient Clip	Linear	[1, 100]
Value Function Coefficient	Linear	[0, 1]
Entropy Coefficient	Logarithmic	[0, 0.5]
Memory Buffer Size	Categorical	$[1x10^4, 2.5x10^4, 5x10^4, 1x10^5]$
Epsilon Time-steps	Categorical	$[5x10^3, 1x10^4, 5x10^4]$
Learning Start Time-step	Linear	[1, 5000]
Prioritized Replay	Categorical	[true, false]
Target Network Update Frequency	Linear	[1, 1000]
Use V-trace	Categorical	[true, false]
Clip Parameter	Linear	[0, 1]

4.3. Results and Discussion

The 9 algorithms were run with the two 2 HPO methods (BASHA and Random). A box plot of their performance on the observational metric of episode length and the reward function are presented in Figures 7 and 8, respectively. Additionally, since box plots present median values, Table 3 presents the mean values along with standard error computations. In total, 100 runs were computed for all algorithms.

For each algorithm, a BASHA hyperparameter search yielded parameters that trained superior or comparable models compared to random search. However, significant performance differences exist across algorithms. What is noticed is that PG, while being a basic RL algorithm, performed relatively well compared to several other algorithms. The tradeoff between reward and episode length is seen in the RAINBOW algorithm results where it consistently learned to find the target, but never learned to find the target quickly. A2C and A3C struggled to perform well, while A3C is known to have training stability issues, more work is needed to understand why it almost never learned to locate the target. Interestingly, while being less consistent than some algorithms, IMPALA performed significantly better than A2C and A3C which shows the effiicacy of the modifications it includes over A3C to improve training stability. Among the actor-critic algorithms, SAC appears to have performed the most consistently. Interestingly, APPO was able to have a few exceptional runs, but most APPO runs timed out; whereas PPO had both the highest mean and median values of the policy gradient methods.

Table 3. Mean Episode Length and Mean Reward with Standard Error: top 5 (and identical/equivalent) in bold in both columns

Algorithm	Mean Episode Length ± Standard Error	Mean Reward ± Standard Error
BASHA PG	**66.97 ± 0.58**	**6.44 ± 1.04**
Random PG	68.16 ± 0.50	4.26 ± 1.01
BASHA Rainbow	67.73 ± 0.10	**8.51 ± 0.22**
Random Rainbow	**67.49 ± 0.10**	**8.08 ± 0.25**
BASHA A2C	**67.48 ± 0.57**	5.05 ± 1.08
Random A2C	68.89 ± 0.39	2.86 ± 0.87
BASHA A3C	70 ± 0.0	0 ± 0
Random A3C	69.99 ± 0.01	0.06 ± 0.06
BASHA SAC	68.31 ± 0.18	**6.49 ± 0.38**
Random SAC	69.12 ± 0.13	2.72 ± 0.35
BASHA PPO	67.77 ± 0.45	**7.28 ± 1.00**
Random PPO	69.02 ± 0.23	3.27 ± 0.62
BASHA DQN with Ape-X	**60.60 ± 1.46**	**10.63 ± 1.59**
Random DQN with Ape-X	67.79 ± 0.75	2.55 ± 0.82
BASHA APPO	**65.91 ± 0.68**	5.41 ± 0.76
Random APPO	**67.34 ± 0.64**	3.52 ± 0.70
BASHA IMPALA	67.61 ± 0.59	4.49 ± 0.75
Random IMPALA	69.65 ± 0.21	1.28 ± 0.31

Overall, DQN with Ape-X resulted in the best possible agents, but only for a small number of runs. This indicates that such an algorithm would require extensive hyperparameter optimization in order to be used effectively. Interestingly, methods that employed asynchronous scheduling (APPO, IMPALA, Ape-X

DQN) appeared to more sensitive to hyperparameter selection, i.e. random vs BASHA results showed a large divergence, compared to synchronous scheduling algorithms (RAINBOW, A2C, SAC, PPO).

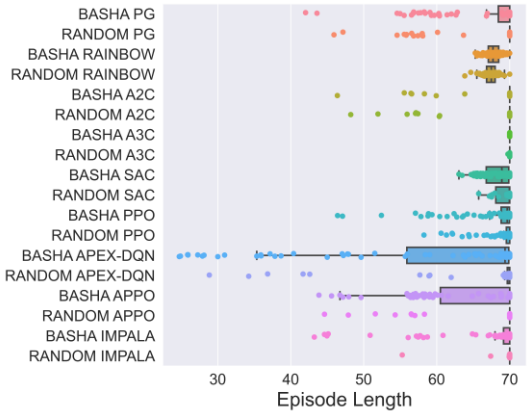

Figure 7. Distribution of raw episode length values from all reward values for models trained during HPO, lower scores are better.

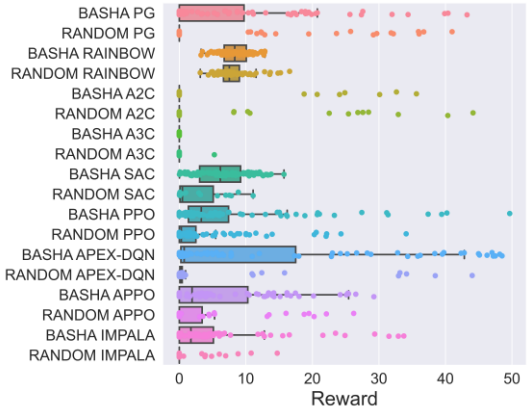

Figure 8. Distribution of rewards received by each model, higher scores are better.

5. Conclusions

The authors presented a systematic comparison of reinforcement learning to a general location intelligence routing task through sensor resource management. Of particular interest were multi-agent RL algorithms and agents which were heterogeneous in sensing capabilities. This study further developed a repeatable ecosystem for test and evaluation and a consistent experimental approach to yield a comparison of MARL methods which had algorithm setting determination through hyperparameter optimization. This application of MARL was both new to SRM and in total, 9 MARL algorithms were considered.

In contrast to past work, e.g. [12], which occurred on different applications, this work found that value-based methods (Ape-X with DQN) provided the best performance and policy-based methods provided the second best; interestingly, Rainbow and SAC provided similar performance to PG. Thus, this study facilitates future developments, selection, and applications of MARL with respect to SRM. This work further illustrated the necessity for selecting appropriate algorithms for MARL tasks as well as the key importance of determining appropriate algorithm settings, since some RL algorithms were unable to provide consistent results. Overall, the Bayesian HPO process employed improved agent reward over random hyperparameter selection.

The theme of future work, in general, will be investigation of RL methods that reduce unsafe and useless results, and which can generalize to ambiguous environments. Additionally, while it is reasonable to expect that the results could translate to continuous action spaces, this needs to be explored. Additionally, broader comparisons with other optimization methods need to be investigated.

6. Acknowledgements

The views expressed in this article are those of the authors and do not reflect the official policy of the United States Air Force, Department of Defense, or the U.S. Government. This work was cleared for unlimited release under case number AFRL-2021-3140.

7. Bibliography

[1] H. Smith, et al., "Locational analysis: highlights of growth to maturity," *Journal of the Operational Research Society*, 60(1), pp. S140-S148, 2009.

[2] R. Malhotra, et al., "Decentralized asset management for collaborative sensing," *SPIE Proceedings*, 2017.

[3] S. Waharte and N. Trigoni, "Supporting search and rescue operations with UAVs," *International Conference on Emerging Security Technologies*, pp. 142-147, 2010.

[4] J. Kober, et al., "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, 32(11), pp. 1238-1274, 2013.

[5] K. Shao, et al., "A survey of deep reinforcement learning in video games.," arXiv preprint arXiv:1912.10944., 2019.

[6] N. Pröllochs, et al., "Detecting negation scopes for financial news sentiment using reinforcement learning," *Hawaii International Conference on System Sciences (HICSS)*, pp. 1164-1173, 2016.

[7] E. Lindwurm, "Intuition: Exploration vs Exploitation," 20 March 2021. [Online]. Available: https://towardsdatascience.com/intuition-exploration-vs-exploitation-c645a1d37c7a.

[8] R. Malhotra, et al., "Learning sensor-detection policies," *National Aerospace and Electronics Conference (NAECON)*, pp. 769-776, 1997.

[9] L. Busoniu, et al., "Multi-agent reinforcement learning: A survey.," *9th International Conference on Control, Automation, Robotics and Vision*, pp. 1-6, 2006.

[10] C. Sampedro, et al., "A fully-autonomous aerial robot for search and rescue applications in indoor environments using learning-based techniques," *Journal of Intelligent & Robotic Systems*, 95(2), pp. 601-627, 2019.

[11] C. Wu, et al., "UAV autonomous target search based on deep reinforcement learning in complex disaster scene.," *IEEE Access*, 7, pp. 117227-117245, 2019.

[12] J. Gupta, et al., "Cooperative multi-agent control using deep reinforcement learning," *Int'l Conf. on Autonomous Agents and Multiagent Systems*, pp. 66-83, 2017.

[13] G. Papoudakis, et al., "Comparative evaluation of multi-agent deep reinforcement learning algorithms.," arXiv preprint arXiv:2006.07869., 2020.

[14] R. Grunitzki and A. Bazzan, "Comparing two multiagent reinforcement learning approaches for the traffic assignment problem," *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pp. 139-144, 2017.

[15] R. Ranjan Kumar and P. Varakantham, "On Solving Cooperative MARL Problems with a Few Good Experiences.," arXiv e-prints, pp.arXiv-2001., 2020.

[16] NRC, *Autonomous vehicles in support of naval operations*, National Academies Press, 2005.

[17] A. Hero and D. Cochran, "Sensor management: Past, present, and future.," *IEEE Sensors Journal*, 11(12), pp. 3064-3075, 2011.

[18] S. Musick and R. Malhotra, "Chasing the elusive sensor manager," *Proceedings of National Aerospace and Electronics Conference (NAECON'94)*, pp. 606-613, 1994.

[19] A. G. Barto, "Reinforcement Learning," in *The Handbook of Brain Theory and Neural Networks*, Cambridge, MA, The MIT Press, 2003, pp. 963-972.

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

[21] V. Mnih, et al., "Playing Atari with deep reinforcement learning," *NIPS Deep Learning Workshop*, 2013.

[22] A. Mosavi, et al., "Comprehensive review of deep reinforcement learning methods and applications in economics.," *Mathematics*, 8(10), p. 1640, 2020.

[23] T. Nguyen, et al., "Deep reinforcement learning for multiagent systems: A review of challenges, solutions, and applications.," *IEEE transactions on cybernetics*, 50(9), pp. 3826-3839, 2020.

[24] Z. Wang, et al., "Dueling network architectures for deep reinforcement learning," *International conference on machine learning*, pp. 1995-2003, 2016.

[25] M. Hessel, et al., "Rainbow: Combining improvements in deep reinforcement learning," *AAAI Conf. on Artificial Intelligence*. pp.3215-3222

[26] D. Horgan, et al., "Distributed prioritized experience replay," *Int'l Conf Learning Representations (ICLR)*, 2018.

[27] R. Sutton, et al., "Policy gradient methods for reinforcement learning with function approximation," *Neural Information Processing Systems (NIPS)*, pp. 1057-1063, 1999.

[28] J. Schulman, et al., "Trust region policy optimization," *Int'l conference on machine learning*, pp. 1889-1897, 2015.

[29] L. Espeholt, et al., "Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures," *Int'l Conference on Machine Learning*, pp. 1407-1416, 2018.

[30] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Adv. neural inform. processing sys.*, pp. 1008-1014, 2000.

[31] V. Mnih, et al., "Asynchronous methods for deep reinforcement learning," *International conference on machine learning*, pp. 1928-1937, 2016.

[32] T. Haarnoja, et al., "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," *International Conference on Machine Learning (ICML)*, pp. 1861-1870, 2018.

[33] T. Bihl, et al., "Easy and Efficient Hyperparameter Optimization to Address Some Artificial Intelligence "ilities"," *Hawaii International Conference on System Sciences*, pp. 943-952, 2020.

[34] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, 13(2), p. 281–305, 2012.

[35] B. Shahriari, et al., "Taking the human out of the loop: A review of Bayesian optimization.," *Proceedings of the IEEE*, 104(1), pp. 148-175, 2015.

[36] K. Jamieson and A. Talwalkar, "Non-stochastic best arm identification and hyperparameter optimization," *Artificial Intelligence and Statistics*, pp. 240-248, 2016.

[37] L. Li, et al., "A system for massively parallel hyperparameter tuning," arXiv:1810.05934, 2018.

[38] J. Garcıa and F. Fernández, "A comprehensive survey on safe reinforcement learning," *Journal of Machine Learning Research*, 16(1), pp. 1437-1480, 2015.

[39] G. Brockman, et al., "Openai gym," arXiv preprint arXiv:1606.01540., 2016.

[40] T. Schaul, et al., "Prioritized experience replay," *Int'l Conf. on Learning Representations (ICLR)*, 2015.

[41] E. Liang, et al., "RLlib: Abstractions for distributed reinforcement learning," *Int'l conference on machine learning*, pp. 3053-3062, 2018.

[42] G. Zhang, "Avoiding pitfalls in neural network research," *IEEE Trans. on Systems, Man, and Cybernetics*, Part C, 37(1), pp. 3-16, 2006.