Using Enhanced Simulation Environments to Improve Reinforcement Learning for Long-Duration Satellite Autonomy*

Mark Stephenson†
University of Colorado, Boulder, Colorado, 80304

Lorenzzo Mantovani‡
University of Colorado, Boulder, Colorado, 80304

Sean Phillips§
Air Force Research Laboratory, Kirtland AFB, New Mexico, 87117

Hanspeter Schaub¶
University of Colorado, Boulder, Colorado, 80304

Recent research has demonstrated that deep reinforcement learning is effective for onboard tasking of Earth-observing satellites. In Earth-observing satellite scheduling problems, an important consideration is balancing mission operations with safety-critical tasks, such as reaction wheel desaturation and maintaining battery charge. While critical to mission success, safety actions are relatively uncommon, leading to relatively sparse training data in important regimes. This work introduces various enhancements to the training environment that induce dangerous edge cases more frequently so that safety-critical regimes can be learned. The policies resulting from enhanced training are deployed in the nominal environment for long durations, showing safer and improved performance than policies trained in the standard environment. The robustness of the enhanced policies to uncertain environments is also shown to be superior to that of the standard policies.

I. Introduction

Effective scheduling of a satellite's objectives and resource management tasks is critical for a mission's success. While activity planning has traditionally been completed as an offline, ground-based process guided by operators, advancements in autonomy have made onboard autonomous tasking an increasingly appealing option. Autonomous policies for tasking are flexible to changing environments, unlike brittle preplanned task sequences, and require less input from operators, which is increasingly necessary as the sizes of satellite constellations increase. A commonly considered class of spacecraft are Earth-observing satellites (EOSs), which have an Earth-data collection mission objective and — like any satellite — must also maintain vehicle safety [1]. For such satellites, planning is often posed as a mode-based problem, where tasking takes the form of deciding what mission objective to work towards or what resource management action (e.g. charging, reaction wheel desaturation, etc.) to take for the next period of time, as opposed to lower-level direct control of actuators.

Variations of the EOS scheduling problem and other spacecraft tasking problems have been formulated as Markov decision processes (MDPs), with early examples in [2–4] allowing the use of reinforcement learning (RL) to find autonomous tasking policies. A variety of algorithms have been explored, ranging from problem-specific architectures [5] to off-the-shelf algorithms including Q-learning [6, 7] and proximal policy optimization (PPO) [8, 9]. In some papers, optimal scheduling of complex mission objectives is considered as an alternative to computationally expensive, offline methods like linear programming [10, 11] or iterative local search [12, 13]. These include Wei et al., which

*Approved for public release; distribution is unlimited. Public Affairs approval #AFRL-2023-6098

†PhD Student and NSTGRO Fellow, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80303. AIAA Member.

‡PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80303. AIAA Member.

§Research Mechanical Engineer, Space Vehicles Directorate, Kirtland AFB, New Mexico, 87117. AIAA Member

¶Professor and Department Chair, Schaden Leadership Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AIAA Fellow.

uses RL with transfer learning to optimize for timeliness and success rate of observation tasks [14], and Herrmann et al.'s work which considers online RL-based planning in various spacecraft domains [9, 15]. Ensuring safe resource management while optimizing completion of a task is another important aspect of spacecraft operations considered by various RL-based approaches to scheduling. In some solutions, resource management is learned by the policy by directly penalizing failures so that the policy avoids them [4, 8, 9]. In others, a shield (i.e. an expert-designed policy of responses to safety-critical states [16]) is deployed during or after learning to guarantee the safety of the satellite during operations [8, 17].

In this work, methods of enhancing the training environment for an EOS are considered to increase the robustness of the satellite when deployed for long-term operations. Under nominal operating conditions, resource management tasks occur relatively infrequently for most satellites (especially for cube- and small- satellites that cannot multitask, as considered in this paper); because of this sparsity of safety-marginal data, training over relatively short horizons and reasonable domains of initial conditions results in policies derived from little exposure to unsafe states. This paper proposes the use of artificially wide training domains and artificially difficult ("augmented") scenarios to increase the incidence of safety interactions when training, so that long-term deployment is more successful. The impacts of lengthening training episodes are also considered.

By demonstrating these techniques, results from prior papers are generalized by validating the assumption that the performance seen in enhanced EOS environments translates to realistic, long-term mission scenarios. Prior work features enhancements to create a more challenging learning environment in various single-agent [2, 9, 18] and multiagent [19] EOS scenarios. In these papers by Harris et al. and Herrmann et al., augmented satellite parameters and external torque were used in both the training and evaluation environment without considering performance on a more realistic set of simulation parameters; in comparison, this work explores how enhanced policies generalize to more realistic environments, and if they provide the hypothesized performance benefits.

In section II, the nadir scanning EOS simulation environment is described and formalized as a MDP. Section III summarizes the PPO training algorithm and explains the rationale behind three enhancements to the training environments. Finally, section IV studies the behavior of the RL algorithms and uses of long-duration mission simulations (both with a nominal satellite and with degradation) to compare the performance of policies trained in enhanced environments to those trained in the standard environment. The limitations of enhancement are also explored.

II. Problem Statement

A nadir-pointing EOS scheduling problem is posed as an environment to demonstrate enhanced training methods. In the environment, the satellite collects data by scanning nadir with an imaging sensor. The satellite must manage the momentum of the ADCS subsystem and remain power positive. First, the problem objectives and underlying simulation models are described in detail. Then, the environment is formalized as a partially-observable Markov decision process (POMDP) for learning with RL.

A. Nadir Scanning Environment

In the nadir-scanning EOS scheduling problem, a satellite in a low-Earth orbit operates in the four modes depicted in Figure 1: nadir-scanning data collection, charging, reaction wheel desaturation, and a data downlink. The mission objective is to maximize the amount of data collected. To collect data, the scanning mode must be active, the instrument within some angle $\theta \leq \theta_{\text{scan}}$ of nadir, and space must be available in the data buffer. While these conditions are met, the instrument continously images, filling the buffer at a constant rate. This objective must be balanced against resource management modes, which are necessary to prevent failure but take time away from data collection.

The satellite is required to stay powered over the course of operations, maintaining a positive battery charge z out of a capacity z_{cap}. The battery is always subject to a baseline power draw $\dot{z}_{\text{base}}$ and reaction wheel power draw $\dot{z}_w(\mathbf{\Omega})$ as a function of reaction wheel speeds when requesting torques. When the scanning mode is active, there is an additional instrument power draw $\dot{z}_{\text{inst}}$; when downlinking, a transmitter power draw $\dot{z}_{\text{down}}$; and when desaturating reaction wheels, a thruster power draw $\dot{z}_{\text{thr}}$. To generate power, the satellite has solar panels with a power generation model $\dot{z}_{\text{sun}}(\phi, e)$ that accounts for solar incidence angle θ and eclipse e; the charging mode points the body-fixed solar arrays at the sun.

The satellite must not allow the attitude control system's reaction wheels to speed saturate, maintaining $\|\mathbf{\Omega}\|_\infty < \Omega_{\max}$. The reaction wheels are modeled on a pyramid of Honeywell HR 16 devices. The wheels gain speed via attitude maneuvers and by counteracting a external torques $\boldsymbol{\tau}_{\text{ext}}$. To desaturate the wheel speeds, the satellite is equipped with RCS thrusters to dump momentum. The wheel model is coupled to power management because higher wheel speeds lead to a greater $\dot{z}_w$, so wheel desaturation helps reduce long-term wheel power draw, at the cost of pausing science

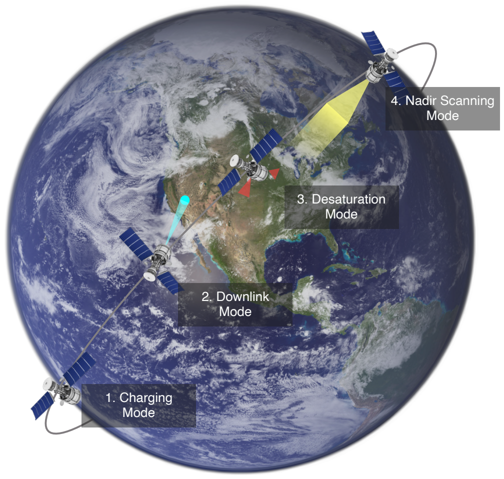

Fig. 1 Action modes in the nadir scanning EOS problem.

operations and drawing power to operate the thrusters.

Finally, the satellite has a data buffer that must have available space to collect data while scanning, satisfying $b < b_{\text{cap}}$. This is not a strict constraint: Attempting to collect data with a full buffer does not lead to mission failure, but it does inhibit the mission objective. If the buffer is not full when scanning is tasked and the other requirements for imaging are satisfied, the buffer fills at a constant rate $\dot{b}_{\text{fill}} > 0$. The satellite can empty the buffer by downlinking data at a rate $\dot{b}_{\text{down}}$ by using the downlink mode, as long as it is within view of a ground station with elevation $\psi \geq \psi_{\text{down}}$.

The environment and satellite are modeled using Basilisk [20]*, a high-performance, high-fidelity spacecraft simulation framework. Using Basilisk's modular architecture, the described scenario is constructed and features realistic models of spacecraft dynamics, power systems, flight software, and the space environment. See [9] for a diagram of Basilisk simulation modules used in a similar simulation.

B. Markov Decision Process Formulation

A POMDP is formulated to represent the scheduling problem for the mission described in the previous section. In the POMDP representation of the scenario, each action enables a corresponding flight mode, state transitions are found by propagating the simulation for some duration, and reward is given for collecting data. The task-based POMDP formulation of the EOS scheduling environment is inspired by previous work from Harris et al., Eddy and Kochenderfer, and Herrmann and Schaub [2, 4, 9]. The POMDP is implemented using the open-source `bsk-rl`† package, which constructs a Basilisk simulation within a standard Gymnasium-API environment [21]. `bsk-rl` is designed to allow for easy configuration of the scenario and repeatable randomization of simulation parameters.

Formally, a POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, O, Z)$. For this environment, some elements are implicitly and trivially defined by the deterministic generative model $G(s, a) = s'$ provided by propagating the simulator for a duration Δt of one decision interval, while others are designer-selected:

- **State Space $\mathcal{S}$:** The entire state used within the simulator. This include both problem-relevant states, such as battery charge and attitude, and hidden states that are require to maintain the Markov property of the full simulation but are near-negligible to the scheduling problem, such as internal integrator states in low-level controllers or epoch to the extent it affects gravity in the Sun-Earth Hill frame.
- **Observation Space O and Observation Probability Function Z:** The observation space O is a selected subset of the dimensions of the state space presumed to be relevant to decision-making: if $\mathcal{H}$ are the hidden dimensions necessary for the Markov property but unnecessary for decision making, $\mathcal{S} = O \times \mathcal{H}$ and the observation probability $Z(o|s) = 1$ for o that is a subset of dimensions of $s = [o, h]$. This indicates that the observation is directly taken from the state without noise, though other work has shown that noise can be added to the

*http://hanspeterschaub.info/basilisk/
†https://github.com/AVSLab/bsk_rl/

Quantity	Normalization	Description
$\boldsymbol{\Omega}$	Ω_{max}	Reaction wheel speed fraction
z	z_{max}	Charge fraction
θ	π	Nadir-pointing angle error [rad]
ϕ	π	Sun-pointing angle error [rad]
b	b_{max}	Data buffer fraction
τ_o^{GS}, τ_c^{GS}	T	Next ground station opportunity, orbital period-normalized
$\tau_o^{Ecl}, \tau_c^{Ecl}$	T	Next eclipse transitions, orbital period-normalized

Table 1 Elements in the observation o, defined by a quantity divided by some normalization constant.

observation without impacting performance as long as the noise was modeled in training and deployment [22]. Deciding which elements of the state are observed relies on the designer's knowledge of which factors could be relevant to the scheduling problem; for this environment, the observation in Table 1 is used. The observations are nondimensionalized and normalized where possible, both to allow for generalization of policies within a reasonable domain of satellite parameters and to improve numerical conditioning of the problem. Observation elements that favor generality are preferred: for example, the times to next eclipse and downlink opportunity are given rather than planet or inertially-fixed position [17].

- **Action Space $\mathcal{A}$:** The action space consists of four discrete actions, each corresponding to switching to one of the flight modes in Figure 1: a_{charge} points the solar panels at the sun; a_{down} turns on the transmitter and, if available, downlinks data to a ground stations; a_{desat} fires thrusters while despinning reaction wheels; and a_{scan} turns on the instrument and, if within θ_{scan} of nadir, logs data to the buffer.
- **Transition Probability Function $T(s'|s,a)$:** Following from the deterministic generative simulator, the transition is deterministically $T(G(s,a)|s,a) = 1$. Two sets of terminal states are defined: timeout states which are reached when an episode successfully completes at $t \geq t_{\text{max}}$, and failure states $\mathcal{S}_{\text{fail}}$ in which a resource constraint (power or wheel speeds) is violated:

$$\mathcal{S}_{\text{fail}} = \{s \mid z = 0 \ \wedge \ \|\boldsymbol{\Omega}\|_\infty \geq \Omega_{\text{max}}\} \tag{1}$$

- **Reward Function $R(s,s')$:** Reward is gathered at a constant rate for adding data to the buffer, normalized such that the theoretical maximum reward is 1 for imaging the entire episode and minimum is -1 for immediately failing.

$$R(s,s') = \begin{cases} \frac{b'-b}{b_{\text{fill}} t_{\text{max}}} & \text{if } b' > b \\ -1 & \text{if } s' \in \mathcal{S}_{\text{fail}} \\ 0 & \text{else} \end{cases} \tag{2}$$

While downlinking data is the ultimate goal of the mission, a downlink-based reward function would be sparse and thus harder to learn than a reward-rich environment. In this formulation, downlink is incentivized implicitly because the the agent needs to free space to collect more data.

III. Methods

Formulating the problem as an POMDP allows an online scheduling policy to be found using RL. PPO is selected for training policies for this environment; a summary of PPO is given. Three methods for enhancing the simulation environment to increase experience gained in unsafe regimes are proposed: parameter augmentation, wide initialization, and longer horizon episodes.

A. Reinforcement Learning

RL refers to a class of optimization algorithms that find a policy — a mapping from states to actions that optimizes reward — for an MDP [23]. Formally, the objective is to find a policy $a = \pi(s)$ that maximizes

$$U^\pi(s) = R(s,\pi(s)) + \gamma \sum_{s'} T(s'|s,a) U^\pi(s'), \tag{3}$$

the discounted ($0 < \gamma < 1$) sum of future rewards, for all $s \in \mathcal{S}$. The algorithm does not have knowledge about the underlying model or reward structure of the MDP; instead it must interact with the MDP to infer a relationship between states, actions, and future rewards. Generally, this occurs through an iterative process of environment interaction and policy improvement: the learning agent collects experience in the environment and uses it to optimize the policy.

B. Proximal Policy Optimization

Deep reinforcement learning (DRL) is a class of RL that uses neural networks to represent the policy and other learned function; among their advantages, DRL algorithms tend to generalize well and work on continuous state and action spaces. PPO is a state-of-the-art policy gradient-based algorithm that restricts the size of policy updates to maintain stability while training [24]. In this work, the RlLib implementation of asynchronous proximal policy optimization (APPO) is used, which utilizes the PPO-clip variation of the algorithm [25]. Like any policy gradient method, PPO optimizes some policy $\pi_\theta(s)$ by improving the policy's parameterization θ. In PPO, this is achieved through gradient descent over a loss function. The PPO-clip loss function is given in Equation 4, where ϵ is a small hyperparameter that clips the change in policy at a given update step:

$$L\left(s, a, \theta_k, \theta\right) = \min\left(\frac{\pi_\theta(a \mid s)}{\pi_{\theta_k}(a \mid s)} A^{\pi_{\theta_k}}(s, a),\ \ g\left(\epsilon, A^{\pi_{\theta_k}}(s, a)\right)\right) \tag{4}$$

where the clipping function g is

$$g(\epsilon, A) = \begin{cases} (1+\epsilon)A & A \geq 0 \\ (1-\epsilon)A & A < 0 \end{cases} \tag{5}$$

Equation 4 uses advantages estimates $A^{\pi_{\theta_k}}(s, a)$ collected from on-policy trajectories to compute loss. By repeating the process of trajectory generation, advantage estimation, and policy update steps, the policy is iteratively improved towards an optimal policy, as described in Algorithm 1[‡].

Algorithm 1 Proximal Policy Iteration

```
for $k = 1, 2, 3, ...$ do
    Collect trajectories by running policy $\pi_{\theta_k}$ in the environment
    Estimate advantages $A^{\pi_{\theta_k}}(s, a)$ for each step in trajectories
    $\theta_{k+1} \leftarrow$ Update the policy parameterization using the PPO-Clip loss function (Equation 4) and the value function
end for
```

PPO is selected for this work as it is known to be performant across a wide range of MDPs. In particular, Herrmann and Schaub have shown that it is effective for a variety of similar spacecraft scheduling problems [9].

C. Enhanced Training Environments

Training a satellite tasking policy using RL inherently introduces a gap between the training and deployment environment: Training is completed on finite-horizon episodes (often of a few orbits), while satellites in orbit tend to operate for years or decades. A few challenges commonly emerge when training policies on episodes simulating a nominal satellite's behavior:

For a satellite designed with comfortable margins, the frequency of safety interactions (e.g. charging, desaturating reaction wheels) is relatively low when the vehicle is in a nominal operating range. This results in sparse training data in safety-critical regimes, which means the resulting policy is less aware of optimizing mission objectives with respect to safety actions and potentially less safe overall when deployed in a long-term setting. Also, the resulting policy may not be robust to "unexpected unexpecteds"; that is, while the policy may be safe and performant even in the most difficult cases that appear in the randomized training environment such as long eclipses, it has no incentive to remain any safer than necessary to survive those cases. For a concrete example, it may take little to no extra effort to maintain the battery at an average level of 80% instead of 40%, but if the worst case encountered in training only requires 40% battery, the policy will exhibit the latter behavior even though the former is more robust to unexpected events like degraded or malfunctioning equipment.

Three methods of enhancing the training environment to expose the satellite to a wider range of states and increase learning near unsafe domains are proposed:

[‡]Adapted from spinningup.openai.com/en/latest/algorithms/ppo.html#pseudocode

1) **Wide Episode Initialization:** In training, it is typical to initialize the satellite in states that are expected to be encountered in deployment. However, there is no theoretical justification for restricting initialization to a "reasonable" domain. Increasing the domain to all possible states (even those that are unrecoverable, such as a near-zero battery at the start of an eclipse) exposes the learning agent to more unsafe states without relying on exploration to encounter them. Unrecoverable initializations provide the agent with information about parts of the domain that are certainly unsafe, which can be usefully extrapolated from by RL algorithms.
2) **Longer Episodes:** Longer episodes allow the agent to fully realize the impacts of its actions, at the expense of fewer unique initial conditions experience over the same number of steps of training. Effects of the closed loop environment-policy system that take many orbits to manifest — including negative effects that lead to reduced reward or failure — can be properly accounted for by the agent.
3) **Environmental Augmentation:** With nondimensionalized observations, the policy can be effective across a range of similar problems; for example, since battery percentage is reported as opposed to absolute charge, a policy should reasonably generalize to other satellites with similar power system dynamics. This can be leveraged in by training in an environment that is more difficult than the nominal mission environment, i.e. one that leads to more safety interactions. Parameters can be adjusted for lower resource capacities and less favorable environmental conditions to make an augmented environment. Not only does this increase safety interactions, it encourages maintenance of resources at more favorable levels in the nominal case without reward function tuning.

While each of these methods is likely to increase exposure to safety-critical states, the third, augmentation to the environment parameters, offers the greatest potential benefits but risks drawbacks. Since the policy is trained for a harder-than-expected environment, the policy is more robust to departures from the nominal system than a policy trained in the standard environment. This is desirable behavior for many systems, where modeling errors or component damage and degradation can result in control and tasking problems that are more challenging than the nominal case. However, this method risks decreasing the performance of the agent with respect to the science objective: too augmented of an environment in training may lead to an overconservative policy for the nominal environment that depresses reward in favor of over-safe actions.

IV. Results

Parameters for training and deployment environment are selected and the results of training with and without enhancements are compared in long-term deployed simulations to evaluate true performance. To test the robustness of enhanced training, the same policies are tested in a more challenging environment with battery and solar panel degradation. Finally, a scenario demonstrating the limits of augmentation on already-hard environments is considered.

A. Experimental Parameters

Decision-making agents are trained on each combination of enhancements: standard and augmented environment parameters, narrow and wide initialization domains, and short and long episode horizons. Most parameters are constant across all training and deployment environments, including those given in Table 2, ground station locations (Boulder, Merritt Island, Singapore, Weilheim, Santiago, and Hawaii), and 500 km altitude and 45° inclination orbital parameters. Parameters changed between the standard and augmented environments are as follows:

- **Standard** policies are trained using standard parameters in Table 3, which are representative of a typical small satellite. These standard parameters are the same as those in the nominal, long-term deployment simulations used to test the policies.
- **Augmented** policies are trained using augmented parameters in Table 3, which reduce resource capacities and increase external torques.

Other parameters are randomized per-episode to ensure a complete training domain, such as epoch and ascending node. Two different episode initialization ranges for safety-critical state elements given in Table 4 are compared when training policies:

- The **narrow** domain initializes the satellite within a feasible, nominal range of states. This range is also used when benchmarking policies in long-term deployment to ensure that the test is survivable.
- The **wide** domain initializes the satellite both in feasible and challenging scenarios. Some initializations may lead to near-immediate failure, but overall the satellite is exposed to more difficult scenarios. Impossible initialization are not inherently bad for training, as the agent can quickly correlate those states to failure without much exploration.

Agents train on episodes of one of two different lengths:

Table 2   Fixed satellite parameters.

Quantity	Value	Description
$\dot{z}_{\text{base}}$	10.0 W	Base power draw
$\dot{z}_{\text{thr}}$	80.0 W	Thruster power draw
$\dot{z}_{\text{inst}}$	30.0 W	Instrument power draw
$\dot{z}_{\text{down}}$	25.0 W	Transmitter power draw
η_w	0.5	Reaction wheel power efficiency for $\dot{z}_w(\mathbf{\Omega})$
A_B	1 m^2	Solar panel area
η_B	0.2	Solar panel efficiency
$\dot{b}_{\text{fill}}$	0.5 MBd	Data collection baud
$\dot{b}_{\text{down}}$	$-0.0224 b_{\text{cap}}$ /s	Downlink baud
Ω_{max}	6000 rpm	Maximum wheel speed
θ_{scan}	22.8°	Max. angle from nadir for scanning
ψ_{down}	10°	Min. elevation angle from ground station for downlink
$\boldsymbol{I}$	$[121, 98, 82]$ kg m^2	Moments of inertia
Δt	3 min	Tasking decision interval

Table 3   Satellite parameters varied between the standard and augmented training environments.

Quantity	Standard	Augmented	Description
z_{cap}	400 W hr	80 W hr	Battery capacity
b_{cap}	5 Gb	1 Gb	Data buffer capacity
$\lvert\boldsymbol{\tau}_{\text{ext}}\rvert$	0.2 mN m	2.0 mN m	External torque magnitude

Table 4   Initialization ranges of satellite parameters.

Quantity	Narrow	Wide	Description
z	$[37.5, 62.5]$%	$[0.0, 100.0]$%	Battery level
b	$[37.5, 62.5]$%	$[0.0, 100.0]$%	Data buffer level
Ω_i	$[-4000, 4000]$ rpm	$[-6000, 6000]$ rpm	Wheel speeds

Table 5 Average probability of an agent failing (due to battery discharge or reaction wheel desaturation) per step over the course of training [% chance per step]. Top three in bold.

	Standard				Augmented			
	Narrow		Wide		Narrow		Wide	
Failure	Short	Long	Short	Long	Short	Long	Short	Long
Battery	0.03	0.00	0.07	0.02	**0.55**	0.20	**0.71**	**0.42**
Wheel Sat.	0.01	0.00	**0.42**	0.11	0.03	0.00	**0.51**	**0.13**
All	*0.05*	*0.01*	*0.49*	*0.12*	***0.57***	*0.20*	***1.20***	***0.55***

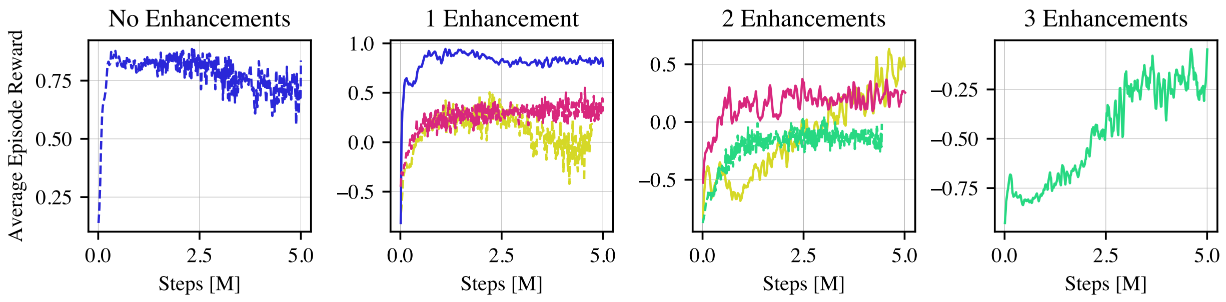

Fig. 2 Comparison of training curves with different enhancements.

- **Short** episodes have a duration of 3 orbits (90 decision intervals).
- **Long** episodes are simulated for 15 orbits (450 decision intervals, approximately one day), giving the agent more time to explore if the policy leads to an unsafe or unrewarding state.

In addition to the RL-based policies, two other policies are considered in comparisons: A random policy, for which $P(a_{\text{charge}}) = P(a_{\text{down}}) = P(a_{\text{desat}}) = P(a_{\text{scan}}) = 0.25$, and a deterministic hand-tuned policy described by Algorithm 2.

Algorithm 2 Hand-Tuned Policy

```
if $z \leq 25\%$ then
    return $a_{\text{charge}}$
else if downlink window available then
    return $a_{\text{down}}$
else if $|\Omega_i| \geq \Omega_{\max} \cdot 80\%$ then
    return $a_{\text{desat}}$
else
    return $a_{\text{scan}}$
end if
```

B. Training Results

Each combination of environment, initialization, and episode horizon is trained using APPO. Table 5 shows that the incidence of safety interactions in training increases with the use of wide initialization and augmented parameters, but decreases with longer episodes. Wide initialization and augmented parameters intuitively lead to more safety interactions in training, because both are more likely to put the satellite in a scenario from which it is difficult or impossible to escape. Longer episodes decrease the chance of the satellite failing on a given step of training: While this gives the satellite more time to fail due to a bad policy, once a good policy has been learned the satellite will survive nearly indefinitely as long as the initial conditions were survivable. However, with long episodes the rate of failure is still high at the start of training, so the lower failure rate overall does not necessarily indicate that less safety information was learned.

A surprising result shown in Figure 2 is that enhancements lead to better training stability. This is a side effect of the

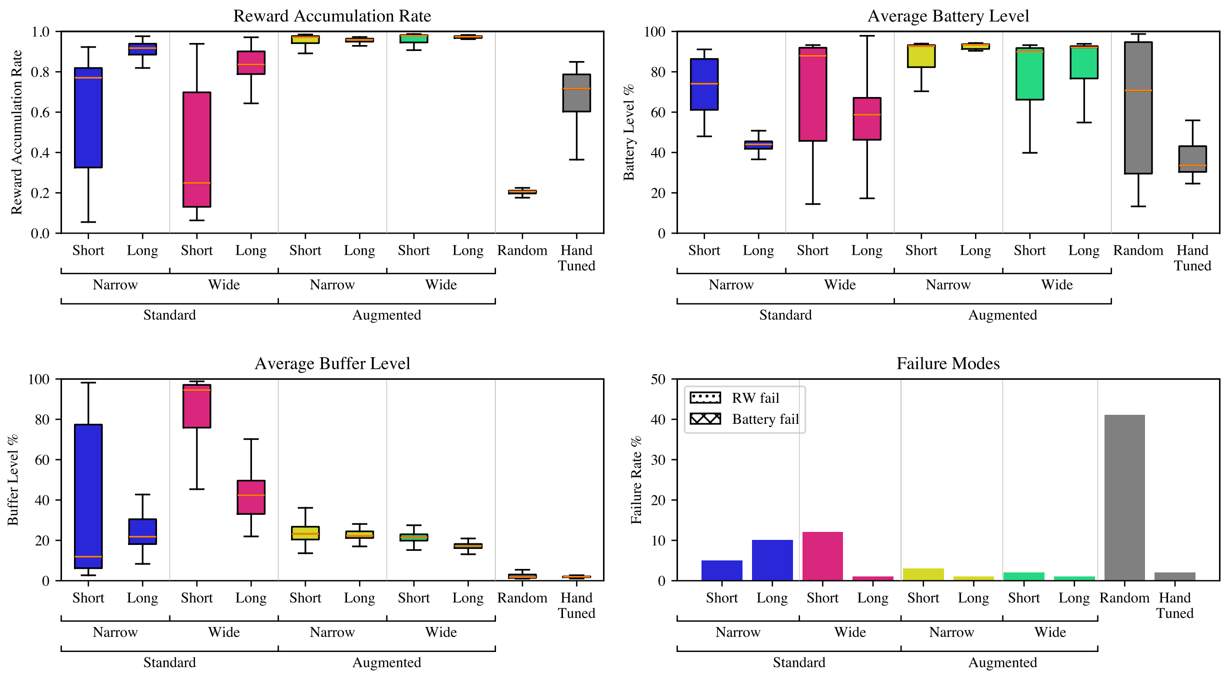

Fig. 3 Statistics of each policy over 100 trials in the nominal environment in long-term deployment. For reward accumulation rate: 0 = no data collection; 1 = continuous data collection.

increased frequency of safety interactions with enhancements: In the non-enhanced cases, the sparsity of failures causes the network to overcorrect when a failure is encountered because the value function is largely unaware of their existence; unlearning due to this is observed in both the no enhancements and one enhancement case. By increasing the frequency of those interactions, the learning agent interacts with a more apparently uniform problem over time and stability is improved as seen in the majority of one and two enhancement cases. A drawback of enhancements is exposed here: The narrow- and wide- long augmented cases display slower learning per-step due to the longer episodes, though this is still preferable to unlearning.

C. Long-Duration Deployment of Policies

Each policy is deployed in the nominal mission environment, which uses standard parameters and the narrow initialization range for one week of simulation time (3150 decision intervals), intended to mimic long-term operations. Results for each policy are given in Figure 3.

In the upper right subplot, it is observed that parameter augmentation and episode duration have the two greatest impacts on the rate at which data is collected and thus maximize the reward earned. The combination of all three enhancements (augmented parameters, wide initialization, and long-horizon training) is best; in general, more enhancements improves performance. Learned policies with a low data collection rate are indicative of a lack of exposure in training to the data buffer filling and preventing data collection. While opportunities for downlink are guaranteed to appear even in short-horizon training environments, it is not necessary to empty the buffer to make space for more data in short-horizon, large buffer cases; however, when deployed in a long horizon the satellite continuously attempts to collect data without ever freeing space by downlinking.

Enhancements to training also improve the robustness of policies. In the upper right plot, enhancements lead to higher average battery levels in deployment; in particular, the use of augmented parameters and long-horizon environments leads to this behavior. Enhancements likewise lead to better buffer dynamics, as shown in the lower right plot. The buffer is on average held at 20% full when trained with enhancements, as opposed to other policies with underperform by limiting data collection with a full buffer or by wasting power and time by downlinking even when the buffer is relatively empty.

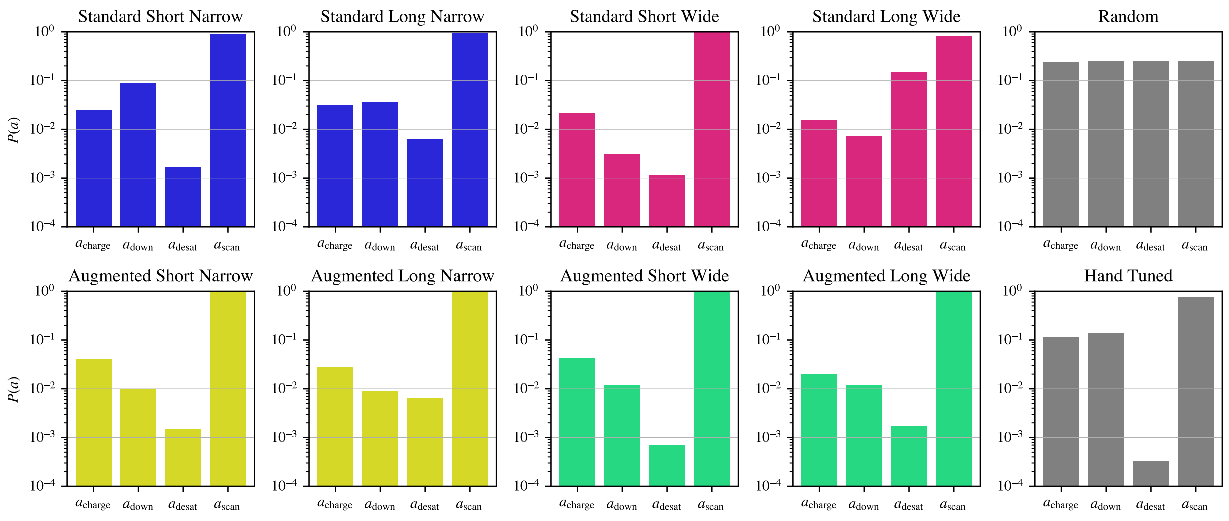

Fig. 4 Distribution of actions taken by each policy.

Table 6 Satellite parameters in the degraded environment.

Quantity	Standard	Degraded	Description
z_{cap}	400 W hr	200 W hr	Battery capacity
η_B	0.20	0.15	Solar panel efficiency

These improvements to average resource levels are reflected in the failure rates of each policy, lower right: Enhanced training environments failed less. Most notably, the enhancements eliminate failures due to reaction wheel saturation which are relatively hard to encounter in standard training. For example, the standard wide long policy frequently fails due to reaction wheel saturation because it was never encountered in training, as Table 5 shows.

Figure 4 shows the distributions of actions taken by each policy. Among the enhanced policies, distributions are remarkably similar despite the differing performance, which indicates that careful sequencing is critical for maximizing reward. High-performing policies learn to only downlink when necessary, as opposed to at every opportunity, saving time and power. The frequency of desaturation varies between policies, ranging from desaturating frequently to minimize power draw to only desaturating to prevent wheel saturation.

D. Deployment in Unexpected Environments

To test robustness of these policies, the nominal environment parameters are modified to represent a satellite with degraded performance; each of the prior policies is deployed on the degraded satellite for week-long trials. Degradation parameters are given in Table 6, which model reduced battery capacity and solar panel efficiency, both issues commonly observed on space systems. Observations reflect the lower z_{cap} when normalizing.

Figure 5 shows the performance of each policy on the degraded system. Unsurprisingly, the standard parameter policies perform poorly when extrapolating to a more difficult environment; those policies either fail very frequently or accumulate reward very slowly (as is the case for the short narrow standard policy). Failures are almost all due to battery discharge, which makes sense given the more challenging power environment. In comparison, the augmented parameter policies maintain high reward accumulation and very low failure rates, following trends seen in the nominal benchmarks. Lower failure rates are correlated with the augmented policies' tendency to maintain battery charge at a higher level.

Figure 8 shows that all of the policies — even those with fewer enhancements — adjust their action distributions to respond to the new environment; RL tends to be inherently good at generalizing even without generalization as a specific objective. Again, the similar distributions of actions across all policies show that the specific sequencing that each policy executes is critical to success and performance.

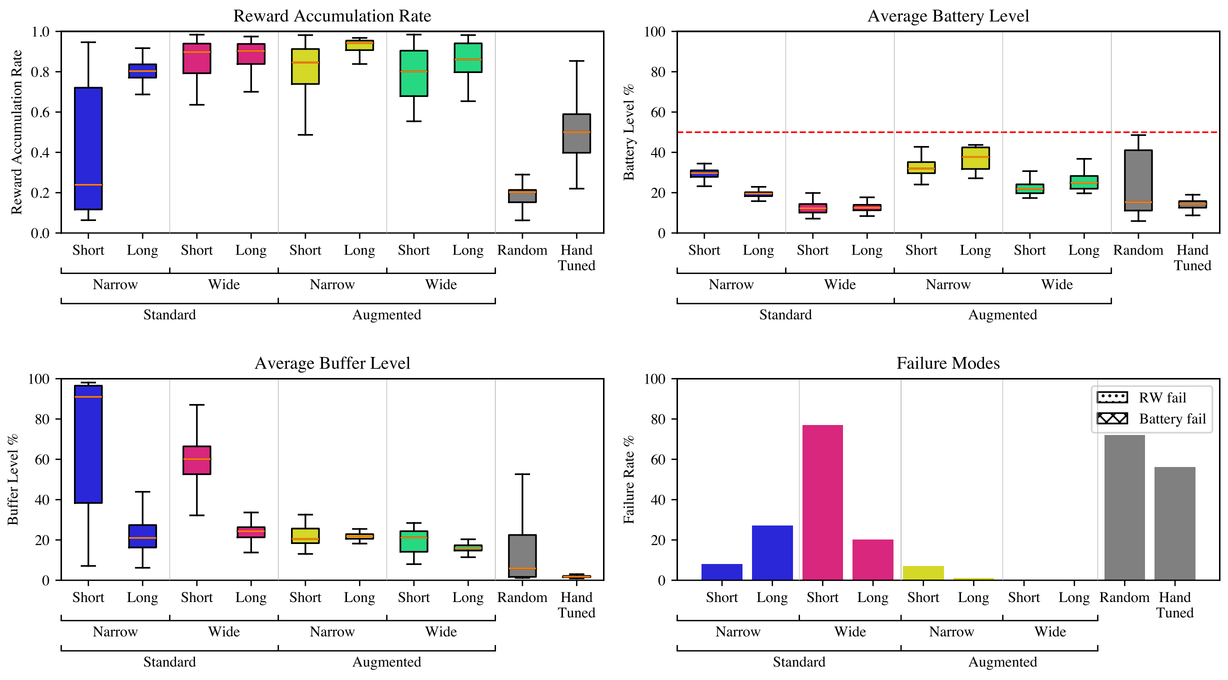

Fig. 5 In the degraded environment, statistics for each of the prior policies over 100 trials in long-term deployment.

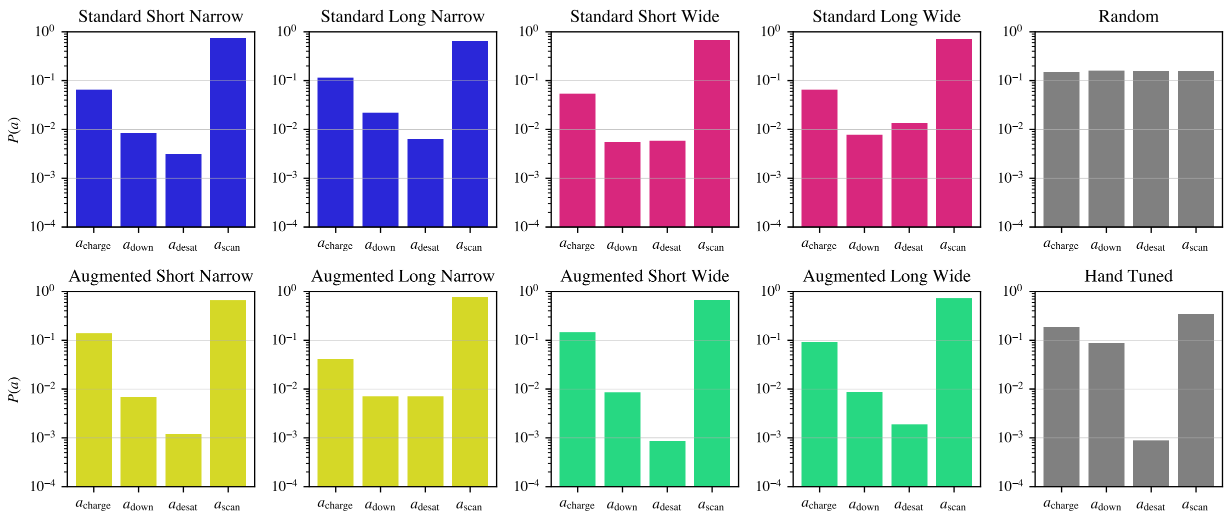

Fig. 6 In the degraded environment, distribution of actions taken by each policy.

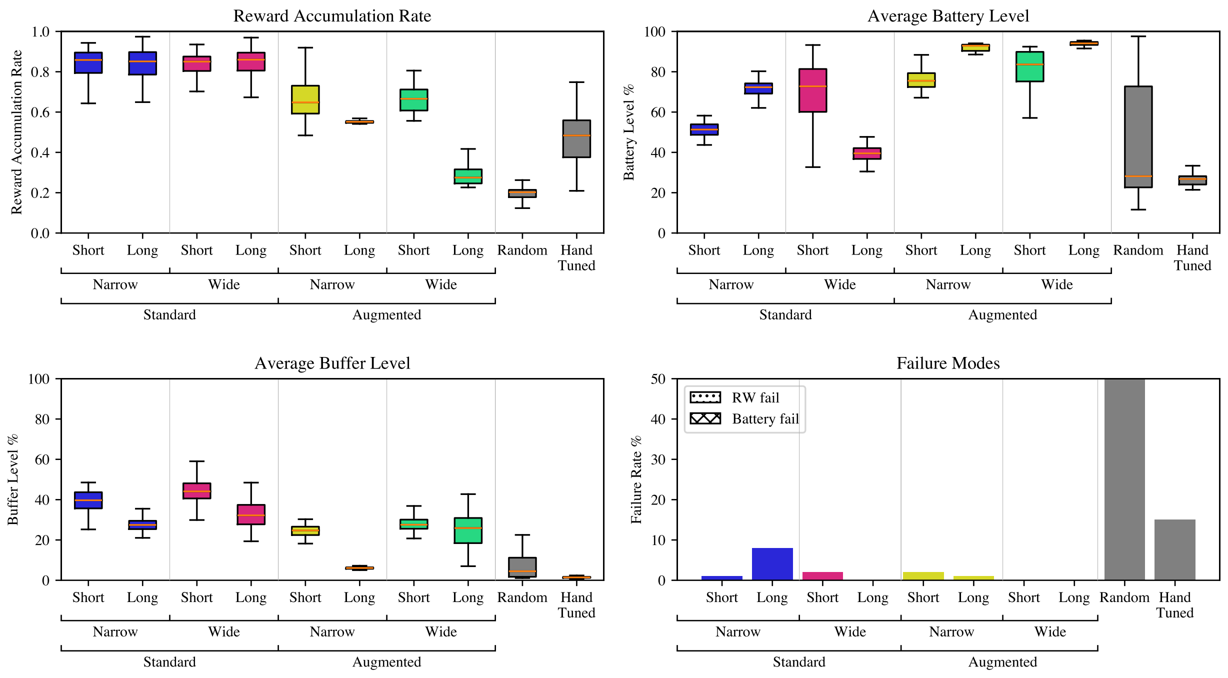

Fig. 7 Statistics for each policy trained and benchmarked in an environment with $\dot{z}_{\text{base}} = 20$ W **in long-term deployment over 100 trials.**

E. Limitations to Environment Augmentation

As one may suspect, parameter augmentation for training environments are not universally applicable. If the environment is augmented too much, it may no longer sufficiently resemble the original environment to provide benefit to a nominal deployment and may even be infeasible. Similarly, environments that are already hard may not benefit from augmentation, resulting in an overconservative policy that reduces data collected.

An example of the latter phenomenon can be demonstrated by increasing the baseline power draw of the nominal environment from 10 W to 20 W, making power management significantly harder: The agent must charge >10% of the time to maintain power positivity. Figure 7 shows the augmented-parameter policies underperforming compared to standard polices because are too conservative. This conservatism is reflected in resource management: Power levels are higher and buffer levels are lower for the augmented policies at the cost of reward. As a result, the failure levels are lower with enhancements. The other enhancements (wide initialization and long episodes) have the same impacts on the policy as in previous cases: Resource management is generally improved and reward is slightly increased. However, these effects are not as pronounced as in other examples because the harder environment allows for a smaller range of good policies.

V. Conclusion

This work demonstrates three enhancements to training environments for satellite tasking that improve robustness and performance in long-term deployment: wider domains for initial states, longer horizons for training episodes, and augmentations to the environment's parameters to make the learning environment more challenging. These methods expose the satellite to a larger domain of states while training, allowing the agent to more thoroughly learn unsafe states and long-term system dynamics. These methods are shown to be effective at reducing failures and improving performance on an EOS tasking problem. Furthermore, policies trained with these methods perform better than their non-enhanced counterparts when running on degraded hardware.

These methods raise a number of questions to be explored in future work. As noted in the results, it is possible to over-enhance the environment to the point of degrading performance with respect to the mission objective; further study into this phenomenon would be valuable so that the proper amount of augmentation can be found efficiently. The results

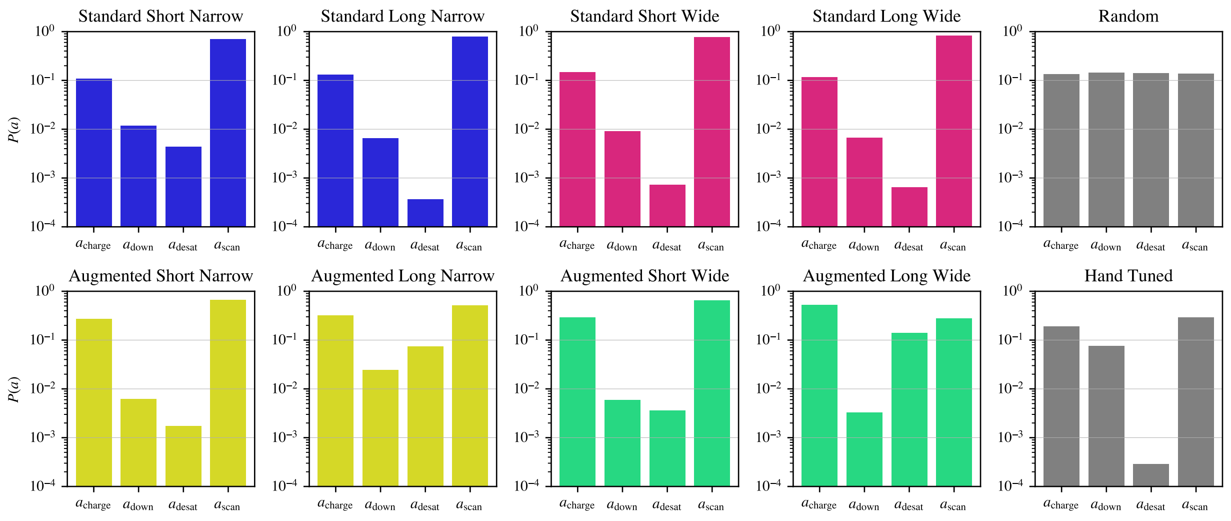

Fig. 8 In the $\dot{z}_{\text{base}} = 20$ W environment, distribution of actions taken by each policy.

also show enhanced policies maintaining resources at levels that are preferable to an operator; a method to induce specific operator preferences through parameter augmentation may exist. Finally, effective ways to combine this work with shielding, as demonstrated for spacecraft tasking in [8], should be explored; these methods would .

Acknowledgments

This work is partially supported by the Air Force Research Lab grant FA9453-22-2-0050. Approved for public release; distribution is unlimited. Public Affairs approval #AFRL-2023-6098. This work is also partially supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) grant, 80NSSC23K1182. Special thanks to Adam Herrmann for formulating and implementing earlier versions of the training environment.

This work utilized the Alpine high performance computing resource at the University of Colorado Boulder. Alpine is jointly funded by the University of Colorado Boulder, the University of Colorado Anschutz, Colorado State University, and the National Science Foundation (award 2201538).

References

[1] Wang, X., Wu, G., Xing, L., and Pedrycz, W., "Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions," *IEEE Systems Journal*, Vol. 15, No. 3, 2021, pp. 3881–3892. https://doi.org/10.1109/JSYST.2020.2997050, URL https://ieeexplore.ieee.org/document/9115252/.

[2] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, Maui, Hawaii, 2019.

[3] Harris, A., and Schaub, H., "Deep On-Board Scheduling For Autonomous Attitude Guidance Operations," *AAS Guidance, Navigation and Control Conference*, Breckenridge, CO, 2020.

[4] Eddy, D., and Kochenderfer, M., "Markov Decision Processes For Multi-Objective Satellite Task Planning," *2020 IEEE Aerospace Conference*, IEEE, Big Sky, MT, USA, 2020, pp. 1–12. https://doi.org/10.1109/AERO47225.2020.9172258, URL https://ieeexplore.ieee.org/document/9172258/.

[5] Zhao, X., Wang, Z., and Zheng, G., "Two-Phase Neural Combinatorial Optimization with Reinforcement Learning for Agile Satellite Scheduling," *Journal of Aerospace Information Systems*, Vol. 17, No. 7, 2020, pp. 346–357. https://doi.org/10.2514/1.I010754, URL https://arc.aiaa.org/doi/10.2514/1.I010754.

[6] He, Y., Xing, L., Chen, Y., Pedrycz, W., Wang, L., and Wu, G., "A Generic Markov Decision Process Model and Reinforcement Learning Method for Scheduling Agile Earth Observation Satellites," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, Vol. 52, No. 3, 2022, pp. 1463–1474. https://doi.org/10.1109/TSMC.2020.3020732, URL https://ieeexplore.ieee.org/document/9200466/.

[7] Piccinin, M., Lunghi, P., and Lavagna, M., "Deep Reinforcement Learning-based policy for autonomous imaging planning of small celestial bodies mapping," *Aerospace Science and Technology*, Vol. 120, 2022, p. 107224. https://doi.org/10.1016/j.ast.2021.107224, URL https://linkinghub.elsevier.com/retrieve/pii/S1270963821007343.

[8] Harris, A., Valade, T., Teil, T., and Schaub, H., "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 611–626. https://doi.org/10.2514/1.A35169, URL https://arc.aiaa.org/doi/10.2514/1.A35169.

[9] Herrmann, A., and Schaub, H., "A Comparison Of Deep Reinforcement Learning Algorithms For Earth-Observing Satellite Scheduling," *AAS/AIAA Spaceflight Mechanics Meeting*, Austin, TX, 2023.

[10] Spangelo, S., Cutler, J., Gilson, K., and Cohn, A., "Optimization-based scheduling for the single-satellite, multi-ground station communication problem," *Computers & Operations Research*, Vol. 57, 2015, pp. 1–16. https://doi.org/10.1016/j.cor.2014.11.004, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054814002809.

[11] Cho, D.-H., Kim, J.-H., Choi, H.-L., and Ahn, J., "Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation," *Journal of Aerospace Information Systems*, Vol. 15, No. 11, 2018, pp. 611–626. https://doi.org/10.2514/1.I010620, URL https://arc.aiaa.org/doi/10.2514/1.I010620.

[12] Liu, X., Laporte, G., Chen, Y., and He, R., "An adaptive large neighborhood search metaheuristic for agile satellite scheduling with time-dependent transition time," *Computers & Operations Research*, Vol. 86, 2017, pp. 41–53. https://doi.org/10.1016/j.cor.2017.04.006, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054817300977.

[13] Peng, G., Dewil, R., Verbeeck, C., Gunawan, A., Xing, L., and Vansteenwegen, P., "Agile earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times," *Computers & Operations Research*, Vol. 111, 2019, pp. 84–98. https://doi.org/10.1016/j.cor.2019.05.030, URL https://linkinghub.elsevier.com/retrieve/pii/S0305054819301510.

[14] Wei, L., Chen, Y., Chen, M., and Chen, Y., "Deep reinforcement learning and parameter transfer based approach for the multi-objective agile earth observation satellite scheduling problem," *Applied Soft Computing*, Vol. 110, 2021, p. 107607. https://doi.org/10.1016/j.asoc.2021.107607, URL https://linkinghub.elsevier.com/retrieve/pii/S1568494621005287.

[15] Herrmann, A., and Schaub, H., "Reinforcement Learning for Small Body Science Operations," *AAS Astrodynamics Specialist Conference*, Charlotte, North Carolina, 2022.

[16] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U., "Safe Reinforcement Learning via Shielding," , Sep. 2017. URL http://arxiv.org/abs/1708.08611, arXiv:1708.08611 [cs].

[17] Nazmy, I., Harris, A., Lahijanian, M., and Schaub, H., "Shielded Deep Reinforcement Learning for Multi-Sensor Spacecraft Imaging," *2022 American Control Conference (ACC)*, IEEE, Atlanta, GA, USA, 2022, pp. 1808–1813. https://doi.org/10.23919/ACC53348.2022.9867762, URL https://ieeexplore.ieee.org/document/9867762/.

[18] Herrmann, A., and Schaub, H., "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, 2023, pp. 1–13. https://doi.org/10.1109/TAES.2023.3251307, URL https://ieeexplore.ieee.org/document/10058020/.

[19] Herrmann, A., Stephenson, M., and Schaub, H., "Reinforcement Learning For Multi-Satellite Agile Earth Observing Scheduling Under Various Communication Assumptions," *AAS Rocky Mountain GN&C Conference*, Breckenridge, CO, 2023.

[20] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 496–507. https://doi.org/10.2514/1.I010762, URL https://arc.aiaa.org/doi/10.2514/1.I010762.

[21] Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., de Cola, G., Deleu, T., Goulão, M., Kallinteris, A., KG, A., Krimmel, M., Perez-Vicente, R., Pierré, A., Schulhoff, S., Tai, J. J., Tan, A. J. S., and Younis, O. G., "Gymnasium," , Oct. 2023. URL https://github.com/Farama-Foundation/Gymnasium, original-date: 2022-09-08T01:58:05Z.

[22] Brandonisio, A., Capra, L., and Lavagna, M., "Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target," *Advances in Space Research*, 2023, p. S0273117723005276. https://doi.org/10.1016/j.asr.2023.07.007, URL https://linkinghub.elsevier.com/retrieve/pii/S0273117723005276.

[23] Sutton, R. S., and Barto, A., *Reinforcement Learning: An Introduction*, second edition ed., Adaptive computation and machine learning, The MIT Press, Cambridge, Massachusetts London, England, 2018.

[24] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., “Proximal Policy Optimization Algorithms,” , Aug. 2017. URL http://arxiv.org/abs/1707.06347, arXiv:1707.06347 [cs].

[25] Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I., “RLlib: Abstractions for Distributed Reinforcement Learning,” , Jun. 2018. URL http://arxiv.org/abs/1712.09381, arXiv:1712.09381 [cs].