(Preprint) AAS 25-27

IMPROVING ROBUSTNESS OF AUTONOMOUS SPACECRAFT SCHEDULING USING CURRICULUM LEARNING

Lorenzzo Quevedo Mantovani[*] and Hanspeter Schaub[†]

This paper investigates the use of curriculum learning (CL) to enhance the robustness of policies obtained by deep reinforcement learning (DRL) for agile Earth observing satellite (AEOS) tasking. While the commonly used optimization methods tend to be brittle to initial conditions changes and require partial or total replan when unexpected events occur, DRL can account for complex dynamics and constraints and provide solutions in real-time for on-board decision-making. Still, the obtained policy may not be able to generalize to longer episodes or different than nominal conditions. The problem intensifies when working with multi-spacecraft systems in formation flying or constellation configurations, where the battery capacity and solar panel efficiency of initially identical agents can diverge over time due to external factors. While one policy could be used for all spacecraft in these scenarios, the policy's robustness is essential to account for agents' differences and ensure the mission's success. Previous research has shown that using enhanced training environments with degraded battery capacity and increased external torque can lead to more robust policies, but the full scope of the training environment impact on the policy was never comprehensively addressed. This paper studies the impact of curriculum learning in detail by testing four different training approaches with different intensities and different training episode lengths. The training approaches are based on curriculum learning, which follows a structured training sequence, and domain randomization. The policies are tested with episode up to 125 times longer than training episodes to evaluate robustness and safety in nominal and degraded conditions. As an application example, policies obtained with enhanced training are tested in a heterogeneous multi-spacecraft system. The results indicate that policies trained with the investigated approaches are more robust and can be deployed for longer periods with few failures.

INTRODUCTION

Earth observing satellites (EOSs) are equipped with instruments to acquire information about the Earth. EOSs are tasked with different activities, such as charging, downlinking data, momentum management, and science acquisition modes. The tasking of EOS has the goal of defining a sequence of actions to maximize the mission outcome while preserving the spacecraft's health and safety given by the system constraints. Traditionally, spacecraft tasking has been performed by humans or optimization algorithms. The advent of agile EOS (AEOS), capable of maneuvering along and across track increases the complexity of the solution space and problem. The increasing complexity of the problem and the need for autonomy with on-board tasking in spacecraft have motivated the use of deep reinforcement learning (DRL) to obtain the policy.

[*]PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80303.

[†]Distinguished Professor and Department Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, 431, Colorado Center for Astrodynamics Research, Boulder, CO, 80309.

The AEOS scheduling problem has been shown to be NP-hard (Reference 1) and has been tackled using optimization methods. Techniques such as greedy algorithms, dynamic programming, constraint programming, and local search were investigated to solve the problem (Reference 1) as well as heuristics (Reference 2). Other methods showed the use of infeasibility-based graphs to solve problems with up to 10,000 requests and 24 satellites (Reference 3). The use of budgeted uncertainty to deal with cloud coverage uncertainty was also proposed (References 4 and 5) as well as methods to perform on-board re-plans to cloud coverage information (Reference 6).

Still, optimization algorithms tend to be brittle to initial conditions changes and require partial or total replan when unexpected events occur or new targets are added to the target list. Additionally, many studies adopt approximations to the problem, such as assuming the satellite to slew at a constant rate or simplified power generation and consumption models, increasing the gap between simulation and reality. This gap can decrease the performance of the obtained solutions when deployed in the real world. The use of DRL has been proposed in the context of spacecraft autonomy to mitigate these problems. More specifically, DRL can account for complex dynamics and constraints. After training, the deep neural network (DNN) can provide solutions in real-time and closed-loop format due to its fast evaluation.

For example, Monte Carlo Tree Search combined with DNN has been used to solve the spacecraft tasking problem, showing good performance compared to a genetic algorithm (References 7 and 8). Comparisons among different DRL algorithms, such as A2C, DQN, and PPO were performed, as well as comparisons with optimization techniques such as Mixed Integer Linear Programming (References 9 and 10), showing the potential of DRL for on-board autonomy in different scenarios and how close it can get to the pseudo-optimal solution. The use of DRL has also been investigated in multi-satellite scenarios (References 11 and 12) and with uncertainty in cloud coverage (Reference 13).

However, using on-board planning without human supervision can bring concerns about the reliability of the policy. Moreover, the deployment conditions can differ from the training environment or change over time; battery degradation, reaction wheel wear, reduced solar panel performance, and other factors can affect the policy's performance. While training tends to use shorter episodes with varying initial conditions to introduce the agent to different scenarios, the obtained policy may not be able to generalize to longer episodes or different conditions. These effects can be amplified when working with multi-spacecraft systems in formation flying or constellation configurations. Even starting with similar initial conditions and parameters, the agents may diverge over time due to partial failures or degradation. Different policies tailored for each spacecraft can be used, but it is not practical for large constellations. While a single policy can be used for all spacecraft in these scenarios, the policy's robustness is essential to ensure the mission's success. Figure 1 provides an illustration of identical spacecraft being tasked different actions, which could be running the same robust policy.

To overcome this problem, Reference 14 proposed the use of enhanced training environments to improve the robustness of the policy. In addition to the nominal environment, policies were trained in an enhanced environment with less battery capacity, more external torque, and less storage space. These policies were tested in the nominal and degraded versions of the environment with longer-than-training episodes to mimic long-term deployment. The results indicate that agents trained in the enhanced environment showed better performance and fewer failures than agents trained in the nominal environment; the performance was also superior in the degraded version of the environment. Still, only a few levels of training environment enhancements were tested.

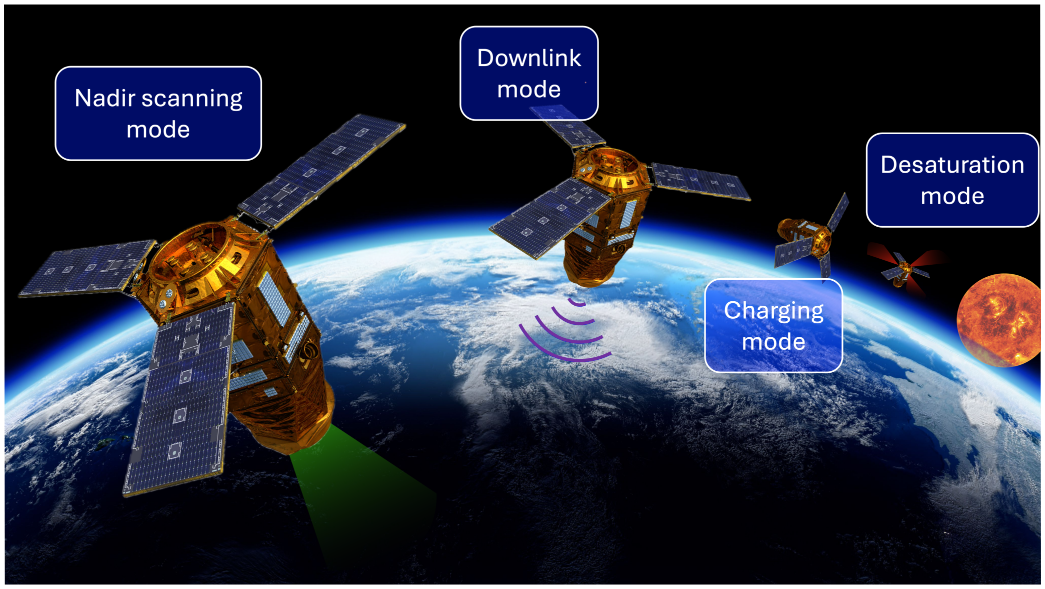


Figure 1: Constellation of identical spacecraft being tasked different actions. The spacecraft can have different parameters due to partial failures or degradation. The same robust policy can be used for all spacecraft.

Variable enhancements in training can also be explored in addition to the fixed enhancements. Reference 15 introduced the concept of curriculum learning (CL) in machine learning, where the training environment is shaped to help the agent learn more complex tasks, similar to how humans learn, in a gradual fashion (a form of continuation method). It was hypothesized that using a curriculum could improve convergence speed and obtain better local minima. Many papers have investigated the use of CL, ranging from pre-defined curriculums to automated curriculums and student-teacher curriculums, leading to increased convergence speed during training and better accuracy (Reference 16). For example, CL showed improved training speed when a reverse curriculum was applied, where the agent was placed close to the final goal and gradually moved away (Reference 17), and also in games when using an end-game-first approach (Reference 18). CL has proven effective in transfering skills from simulation to real-world robotic control in visuomotor tasks by adjusting training difficulty according to success rate (Reference 19). It also has been employed to aid in the learning of complex behavior, facilitate the learning of a bipedal robot to walk over complex terrain through the use of guiding forces to shape the training environment (Reference 20), as well as the learning of a policy to control a quadrotor through small gaps, where the agent could not learn without the structured progression provided by CL (Reference 21). In a spacecraft context, CL has been used to improve the results obtained when using RL to solve a fuel-optimal guidance and control problem (Reference 22).

Automated CL has also been shown to be important in the training of quadrupedal robots, where agents fail to learn if no curriculum is used (Reference 23). Curriculum changing the velocity commands during training using a Grid Adaptive Curriculum based on success threshold has been explored (Reference 23), as well as changing the terrain difficulty (References 24 and 25). Navigation with obstacle avoidance has also benefited from the use of CL, leading to more robust policies that performed well in unseen simulated and real environments (Reference 26). CL has also been explored to ensure safety during training in a student-teacher framework (Reference 27). Reference

28 have a comprehensive review of the use of CL in reinforcement learning while Reference 29 focus on automated curriculum. Still, the use of CL to improve training in the spacecraft tasking context has not been explored, except for the study conducted by Reference 14.

Therefore, this paper aims to investigate the use of CL to enhance the robustness of the policy obtained by DRL for spacecraft tasking. Depending on the curriculum structure, the agent can be presented with an easier-to-survive environment where it can first learn how to maximize rewards and over time learn how to survive. If an initially harder environment is presented, the agent has a stronger incentive to avoid terminal states, learning first how to survive and later how to optimize for rewards. These different approaches can lead to different policies and, consequently, different robustness levels.

The novel contribution of this work is researching how CL can be applied to the AEOS problem by analyzing how different curriculums affect policy robustness. It provides a comprehensive study in contrast to the initial results presented in Reference 14. Three different curriculums were tested, as well as a Domain Randomization (DR) approach. The policies are tested with different episode up to 125 times longer than training episodes to evaluate robustness and safety. Additionally, policies were tested with nominal and degraded environmental conditions. As an application example, policies obtained with CL are tested in a multi-agent system with different parameters to evaluate their robustness.

PROBLEM FORMULATION AND SIMULATION ENVIRONMENT

Partially Observable Markov Decision Process formulation

The tasking of spacecraft can be formulated as a decision process using a partially observable Markov decision process (POMDP). A POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, R, Z, \gamma)$ such that:

- $\mathcal{S}$ is the state space, which is the space of simulator states required to preserve the Markov property. It includes states such as satellite's battery charge, position, orientation and internal flight software states.
- $\mathcal{A}$ is the set of actions that the agent can take. In this problem, the agent can choose between four actions: charge, downlink, reaction wheel momentum management, and nadir scanning.
- $\mathcal{O}$ is the set of observations space, which is a subset of the dimensions in $\mathcal{S}$, and includes information available for the agent and useful for the decision-making process, such as battery charge fraction, sensing instrument orientation, and time to next eclipse.
- $T(s'|s,a)$ is the transition function that defines the probability of transitioning to the next state s' given the current state s and action a. In this problem, the transition function is deterministic and a generative model G is used, provided by a simulation engine, such that $s' = G(s,a)$.
- $R(s,a,s')$ is the reward function that defines the reward obtained by the agent. Rewards are awarded for the time spent in nadir scanning mode when within the relative angle and angular velocity limits during each time step k, given by $t_{\text{nadir}}^{(k)}$. The reward function is defined as

$$R(s,a,s') = \begin{cases} \frac{t_{\text{nadir}}}{T_{\text{sim}}}(k) & \text{if } a = \text{nadir scanning}, \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

where T_{sim} is the total simulation time of the episode. Therefore, cumulative rewards are between 0 and 1, where 1 is the maximum reward obtained when the agent is in nadir scanning mode for the entire episode.

- $Z(o|s', a)$ is the observation function that defines the probability of observing o given the current state s and action a. In this problem, the observation function is deterministic without uncertainty about its observations.
- γ is the discount factor that defines the importance of future rewards.

The agent interacts with the environment by taking actions, receiving observations, and obtaining rewards. The agent's policy is a mapping from states to actions, and the goal is to find the optimal policy that maximizes the expected return. Table 1 summarizes the observations available to the agent and the normalization factors used.

Table 1: Observation space

Parameter	Description
$\hat{c}_{\mathcal{B}/\mathcal{P}}$	Satellite's sensing instrument orientation with respect to nadir, normalized by π.
P_f	Battery charge fraction.
S_f	Buffer storage space fraction.
$\hat{W}_f$	Wheel speeds normalized by the maximum wheel speed.
Φ_s	Angle between solar panels and sun vector. Normalized by π.
e_b	Time until next eclipse. Normalized by orbital period.
e_f	End of next eclipse. Normalized by orbital period.
g_b	Time until next ground station pass. Normalized by orbital period.
g_f	End of next ground station pass. Normalized by orbital period.

Simulation environment

The simulation was implemented in BSK-RL[*], which is a Python package for spacecraft tasking studies using RL (Reference 30). BSK-RL combines the Basilisk[†] software with Gymnasium, a widely adopted RL library. Basilisk is used as the generative model for the simulation environment, which is a spacecraft simulation software written in C and C++ with a Python interface, making it fast for training and testing and easy to interact with (Reference 31). The simulation accounts for the spacecraft dynamics, reaction wheel subsystems, power generation, and storage subsystems.

The goal of the agent is to maximize the time spent imaging nadir. Therefore, the scanning action tracks the nadir, pointing the camera to the Earth's surface. The scanning action obtains information from the Earth's surface when the relative attitude and angular velocity criteria are met and adds that to the storage subsystem. If the storage unit is full, no more data can be added and no more reward can be awarded. To account for this, action downlink points the satellite's antenna to a ground

[*] `https://avslab.github.io/bsk_rl/`
[†] `https://avslab.github.io/basilisk/`

Table 2: Nominal satellite parameters

Battery capacity	400 W
Base power consumption	10 W
Data storage capacity	5 GB
Relative angle limit for imaging	$28°$
Relative angular rate limit for imaging	0.1 rad/s
Minimum image elevation	$45°$
Reaction wheel maximum speed	6000 RPM
External torque magnitude	0.2 mNm
Thruster power draw	80 W
Instrument baud rate	0.5 MB/s
Instrument power draw	30 W
Transmitter baud rate	112 MB/s
Transmitter power draw	25 W
Solar panel efficiency	20%
Initial battery charge fraction	$[0.375, 0.625]$
Initial storage space fraction	$[0.0, 1.0]$
Initial reaction wheel speeds	$[-4000, 4000]$ RPM

station to downlink data stored in the satellite, accounting for transmission baud rate, amount of stored data, and access to a ground station.

In addition to consuming power at all times due to baseline power consumption (to represent essential subsystems), reaction wheels follow a non-linear power consumption law that is a function of their angular velocity. The scanning instrument also consumes energy when in use. When tasking the action charge, the satellite maneuvers to align its solar panels with the sun vector to maximize its power generation. Power generation is calculated by the underlying simulation in Basilisk, which accounts for eclipse and incidence angle. Therefore, the satellite can passively charge even when taking other actions.

Satellite attitude is controlled by a Modified Rodrigues Parameters (MRP) steering law using three reaction wheels positioned orthogonal to each other. As the reaction wheels accumulate momentum due to external torque, leading to higher angular velocities and higher power consumption, a momentum management action was also added to the agent. The momentum management action aligns the spacecraft with the nadir and fires thrusters to reduce the momentum accumulated by the reaction wheels. The agent reaches a terminal state (dies) when the battery level is below zero or any of the three reaction wheels exceed their maximum angular speed. Each action has a fixed duration of 180 seconds.

Table 2 shows the satellite parameters used in the simulation. Parameters not listed here are set to the default values in BSK-RL. An example script of a nadir scanning environment for spacecraft tasking is available at the BSK-RL website[*].

A degraded version of the environment was also used to test policy's robustness. The degraded environment was obtained by reducing the battery capacity in 50%, the solar panel efficiency in 25%, and setting the external torque as three times the nominal value.

[*]`https://avslab.github.io/bsk_rl/examples/rllib_training.html`

TRAINING WITH CURRICULUM LEARNING

Training setup

The policy was obtained using the Asynchronous Proximal Policy Optimization (APPO) algorithm provided by the RLlib package. The training was performed at the University of Colorado Research Computing (CURC) using 32 cores and limited to 5 million steps. Table 3 provides more information about the used training hyperparameters (parameters not listed here were set as the default values in RLlib). For a comprehensive hyperparameters analysis on DRL algorithms for spacecraft tasking, refer to Reference 9.

Table 3: Training parameters

Number of workers	32
Number of Training steps	$5 \cdot 10^6$
Learning rate	$3 \cdot 10^{-5}$
Training batch size	$10,000$
Minibatch size	250
Number of SGD iterations	50
Neural Network	2 layers with 512 neurons each
Discount factor	0.999
Failure penalty	-1

A failure penalty of -1 was used, being added to Eq. 1 when the satellite reaches a terminal state. The value of -1 was used to limit the cumulative reward range to $[-1, 1]$ (Reference 14).

Curriculum Learning

Curriculum learning provides a methodology to shape the training environment to help the agent learn more complex tasks (Reference 15). It starts with simple tasks and gradually increases the difficulty of the environment. In this work, three different curriculums were tested: constant (CC), direct (DC), and inverse (IC). The constant curriculum sets the difficulty of the environment constant throughout the training process. The direct curriculum starts with a nominal environment and gradually increases the difficulty towards a harder environment. The inverse curriculum starts with a harder environment and gradually decreases the difficulty towards the nominal case. Defining the difficulty of an environment can be subjective and usually requires expert knowledge of the problem. In this work, difficulty is associated with the satellite's battery capacity and external torque. A difficulty of 1 corresponds to what is considered the nominal environment; a difficult higher than one corresponds to reduced battery capacity and, when applicable, increased external torque.

In addition to the curriculum cases, a fourth training method using Domain Randomization (DR) was tested. When using DR, the agent is trained in a variety of environments with different parameters, such as battery capacity and external torque, which are uniformly sampled at each environment reset. Therefore, the main difference between DR and CL is that in the DR the agent does not have a structured progression of difficulty. Figure 2 provides an illustration of the three different curriculum approaches, as well as the DR method.

Because the direct CL initially presents the agent with the nominal environment, in which is easier to survive, it is expected that the agent will first learn how to obtain rewards through scanning

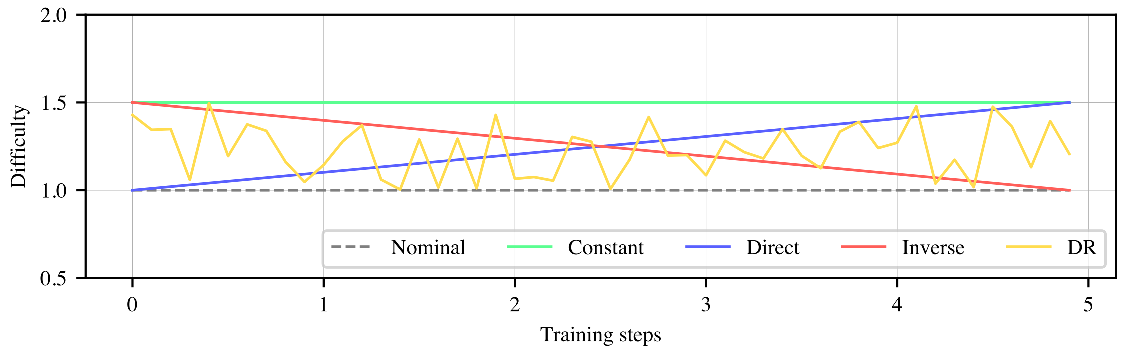


Figure 2: Illustration of the four investigated training approaches.

Table 4: Curriculums and intensities tested

Decision intervals	Number of Intensity Levels							
	CC		DC		IC		DR	
	B	BT	B	BT	B	BT	B	BT
90	3	3	3	3	3	3	3	3
450	-	3	-	3	-	3	-	3

actions. Later, as the environment gets more difficult, the agent will improve its survivability to maximize cumulative rewards. On the other hand, the inverse CL presents the agent with a harder environment at the beginning, where the agent has a stronger incentive to avoid terminal states, learning how to survive. As the number of training steps increases and the environment gets easier, the agent will learn how to optimize for rewards through imaging actions. IC can be seen similarly to the reverse curriculum (Reference 17) or the end-game-first approach (Reference 18), where terminal states are presented first, making the agent learn how to achieve or avoid them. While the structured sequence of CL can help the agent learn more complex tasks, tasks learned in the early stages of training may be forgotten as the environment gets more complex. Therefore, the agent may not be able to generalize well. DR, on the other hand, can help the agent learn a policy that generalizes better to different environments and will be used for comparison.

Each of the four approaches was tested at different intensities, defined by their initial and final difficulties. Additionally, each combination of curriculums and intensities was tested with different episode lengths in training and in a narrow (N) and wide (W) initialization range. Table 4 summarizes the different curriculums and intensities used for training, where "B" indicates the environment where the battery capacity was varied, "T" for environments with modified external torque, and "BT" for both. Each combination of curriculum and intensity was trained three times to account for the randomness in the training process, resulting in 114 trained policies.

The three intensity levels of curriculum applied to battery capacity correspond to decreases of 20% (B20), 40% (B40), 60% (B60) in battery capacity. When the battery capacity change was combined with an increase in external torque, the three intensity levels correspond to increases of 2 (B20-T02), 4 (B40-T04), and 6 (B60-T06) times the nominal external torque.

The curriculum cases varying only in the battery capacity were tested exclusively for the case with 90 decision intervals (time steps). The cases with both battery and torque were tested with both 90 and 450 decision intervals.

EFFECT OF CURRICULUM LEARNING ON POLICY ROBUSTNESS AND PERFORMANCE

The following section includes comparisons between policies trained with and without CL. The policies were tested in the nominal and degraded versions of the environment. They were also tested with different episode up to 125 times longer than training episodes to evaluate robustness and safety aspects. Initially, the cases with battery variation in the curriculum will be presented and compared to the baseline. Then, the cases varying only battery capacity will be compared to the cases varying both battery capacity and external torque. Later, the impact of using longer training episodes and narrower and wider initialization ranges will be discussed.

Curriculum comparison

The 114 policies were tested in environments with 90, 450, 2250, and 11250 decision intervals, corresponding to 2.84, 14.2, 71.05, and 355.26 orbits, respectively. Each policy was tested in the nominal environment and in the degraded version of the environment (discussed in Simulation environment section). Each policy was tested in 100 simulation runs with different random initial conditions for each combination of episode length and environment (nominal and degraded). The results show violin plots, where each violin contains 300 test runs (100 for each of the 3 policies for the same curriculum and intensity). The combined results correspond to 2813 years of simulation time.

The impact of curriculum learning Initially, the impact of the four different training approaches was investigated when considering only variation in battery in the curriculum. For this case, the policies were trained with 90 decision intervals. While very few differences are seen when testing these policies in environments with 90, 450, and 2250 decision intervals, these differences become more visible in the longest test case with 11250 decision intervals.

Figure 3 reports the cumulative rewards obtained by the policies during the episode with 11250 decision intervals, the number of orbits completed per case, percent of battery or reaction wheel (RW) related failure, and the percent of each of the four actions taken by the agents across all 300 test runs. These results do not consider the failure penalty of -1 used during training. Charge, Downlink, and Desat (reaction wheel momentum management) actions are indicated by color bars with different patterns; the nadir scanning action percentage corresponds to the remaining percentage to get to 100%. While gray bars indicate the baseline case trained in the nominal environment without curriculum, green represents the CC cases, blue the DC, red the IC, and yellow the DR cases.

The results in Figure 3 show that the failure rate of policies trained without curriculum is higher than those trained with CL or DR. Additionally, the cumulative reward obtained by policies trained with curriculum is higher than the baseline. While the higher reward can be linked to higher survivability, the bar plot of action percentage shows that the baseline policy takes a similar percentage of charge and desaturation actions as the other policies. This indicates that the policies trained using curriculum learned better how to avoid unsafe states without significantly compromising the reward rate.

Different from Figure 3, Figure 4 shows the results for the degraded version of the environment. In this case, the baseline showed a failure rate of more than 40%, as well as the CC-B20, DC-

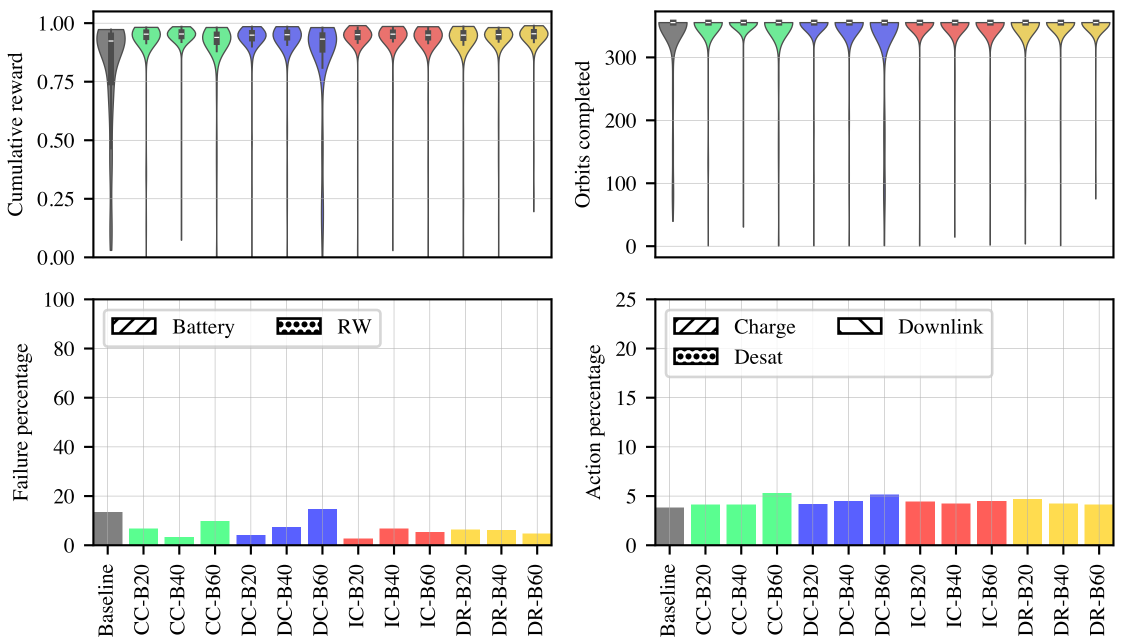


Figure 3: Performance of policies trained with battery variation in 90 decision intervals and tested in the nominal environment with 11250 decision intervals. Cumulative reward (without failure penalty), number of orbits alive, failure type and percentage, and percentage of actions (nadir scanning corresponds to the remaining percentage to reach 100%) taken by the agents are reported.

B20, IC-B20, and DR-B40. These same policies also showed lower cumulative rewards. The three best-performing cases are CC-B40, CC-B60, and DR-B60, with failure rate below 30%. Again, the percentage of scanning actions taken by the policies is similar.

While the cases trained with IC showed a balanced number of battery and torque failures, the cases trained with CC, DC, and DR showed mostly reaction wheels related failures. This indicates that there is room to add torque to the curriculum to help the agent learn how to avoid unsafe states related to maximum wheel speeds. This is investigated next.

Battery vs Battery and Torque Curriculum Varying the battery capacity with the difficulty of the environment is expected to enhance the policy robustness with respect to degraded battery capacities. When also adding the external torque to the curriculum, it is expected that the policy will be more robust to external disturbances too, leading to fewer reaction wheel related failures. To investigate this, the policies trained with battery variation in the curriculum (discussed in the previous section) were compared to the cases trained with both variations. Figure 5 shows the results for policies trained with DC and 90 decision intervals and tested in the degraded environment with 11250 decision intervals. All three cases with battery and torque curriculum showed fewer failures than the cases with similar intensity but only battery curriculum.

While DC seems to benefit from the addition of torque to the curriculum, the same is not observed for the policies trained with IC. Instead, Figure 6 shows that adding torque to the IC leads to slightly more failures in all three cases. The cases with CC and DR also showed improved results when adding torque to the curriculum, although not as much as DC. The results for this comparison with CC and DR won't be presented here for brevity. Still, since most of the cases trained with battery

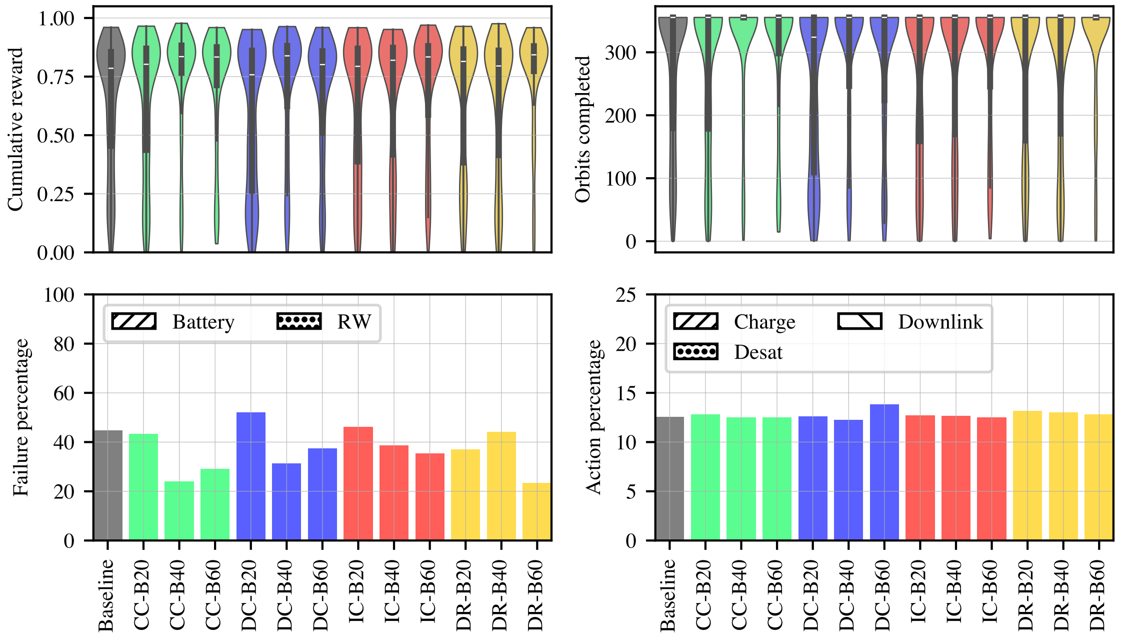


Figure 4: Performance of policies trained with battery variation in 90 decision intervals and tested in the degraded environment with 11250 decision intervals. While similar percentage of nadir scanning actions are taken, differences are seen in cumulative reward and failure percentage.

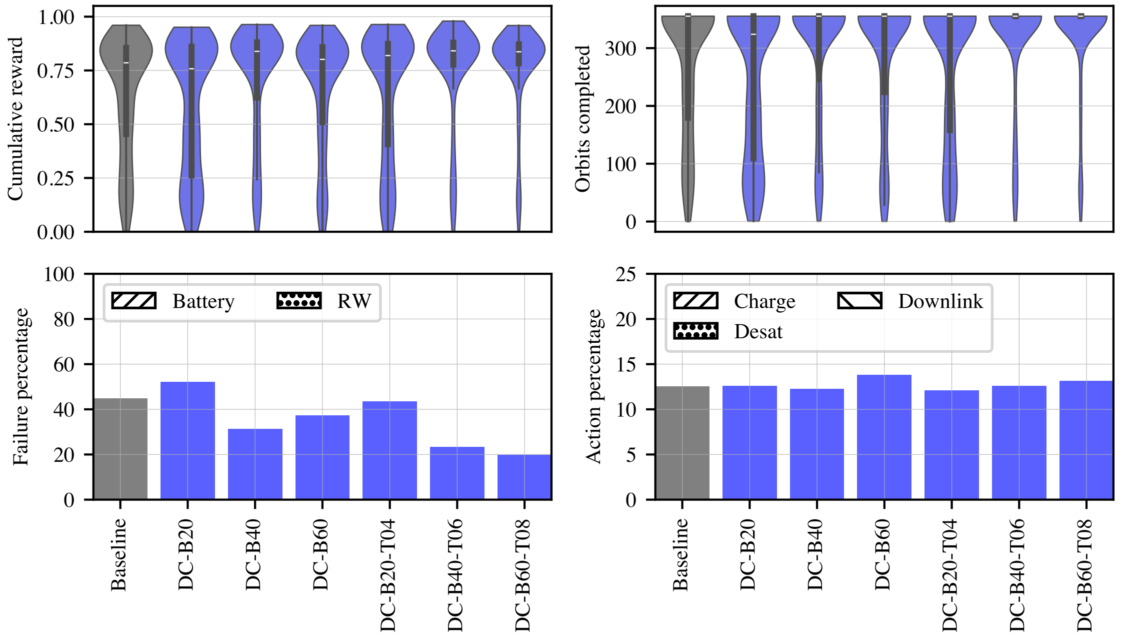


Figure 5: Comparison between policies trained with only battery and battery and torque variation in direct curriculum (DC). Adding torque to the curriculum lead to reduced battery failures. Policies were tested in the degraded environment with 11250 decision intervals.

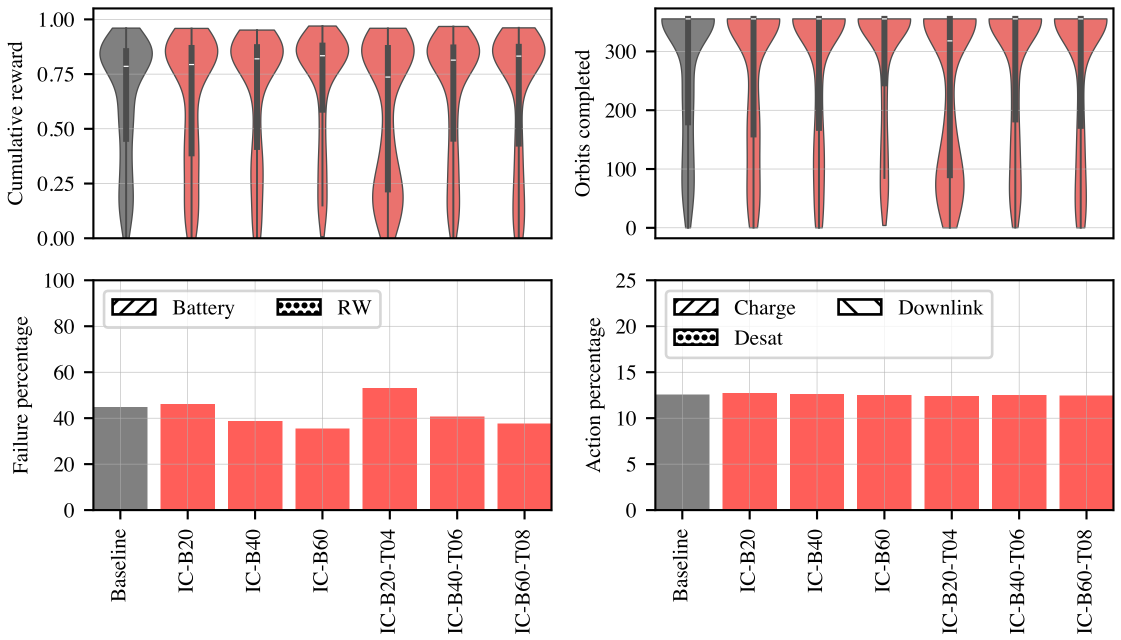
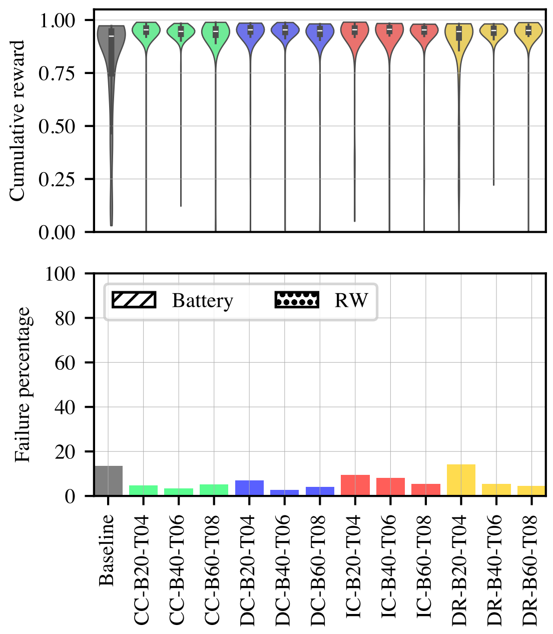


Figure 6: Comparison between policies trained with only battery and battery and torque variation in inverse curriculum (IC). Adding torque to the curriculum didn't change the reaction wheels related failure significantly. Policies were tested in the degraded environment with 11250 decision intervals.

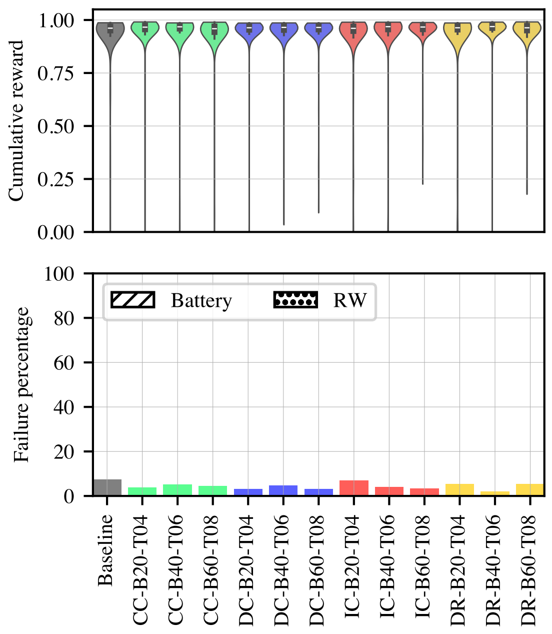
and torque curriculum showed fewer failures, only cases containing both battery and torque in the curriculum will be considered for the remaining analysis.

Episode length in training While the agents are trained with three orbit long episodes, they are deployed for many more orbits as satellite missions usually range from months to years. Still, it is impractical to train with such long episodes and randomize the environment enough to ensure generalization due to computational costs. While the training algorithm should use the estimate of the value function to bootstrap the return in shorter environments (truncated), training with longer episodes should provide the agent with more experience in the region of the state space closer to its nominal deployment. To investigate the impact of training with longer episodes, policies trained with battery and torque were trained for 90 and 450 decision intervals. Figure 7 shows the obtained results when deploying in the nominal environment for 11250 decision intervals. The results show that the policies trained with 450 decision intervals obtained higher rewards and had fewer failures. Improvements were also seen in the baseline policy trained for 450 episodes when compared to the case trained with 90.

More significant differences are seen when comparing those policies in the degraded environment, as shown in Figure 8. Extending the training episode length from 90 to 450 decision intervals leads to reduced failure rates in more than half in some cases. The best-performing policies trained with 450 decision intervals were trained with CC-B40-T06, CC-B60-T08, DC-B40-T06, DC-B60-T08, IC-B60-T08, and DR-B60-T08. The decrease in failure rates was also accompanied by higher cumulative rewards (higher median value and less variance, as seen in the violin plots being more concentrated at the top). The best policy with curriculum, DC-B60-T08, showed less than 10% failure rate.

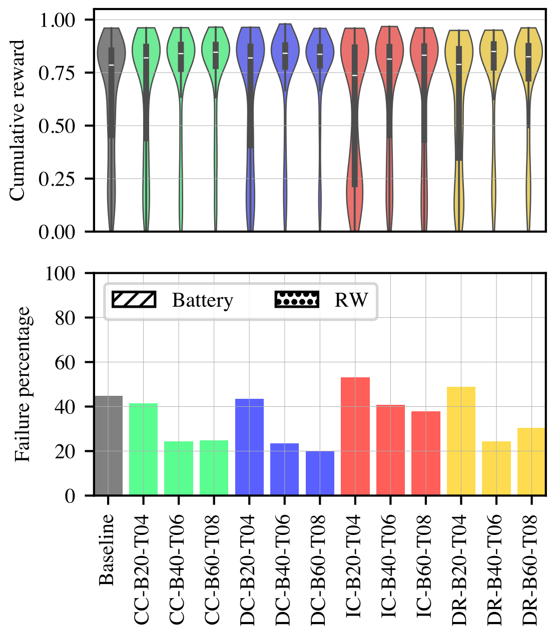


(a) Policies trained with 90 decision intervals.

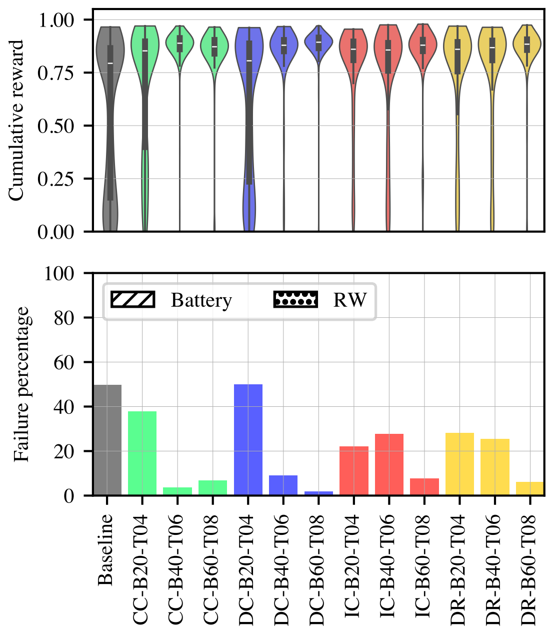


(b) Policies trained with 450 decision intervals.

Figure 7: Results comparing the effect of training episode length in the nominal environment with 11250 decision intervals. The baseline policy showed higher cumulative reward and fewer failures when compared when trained with longer episodes.

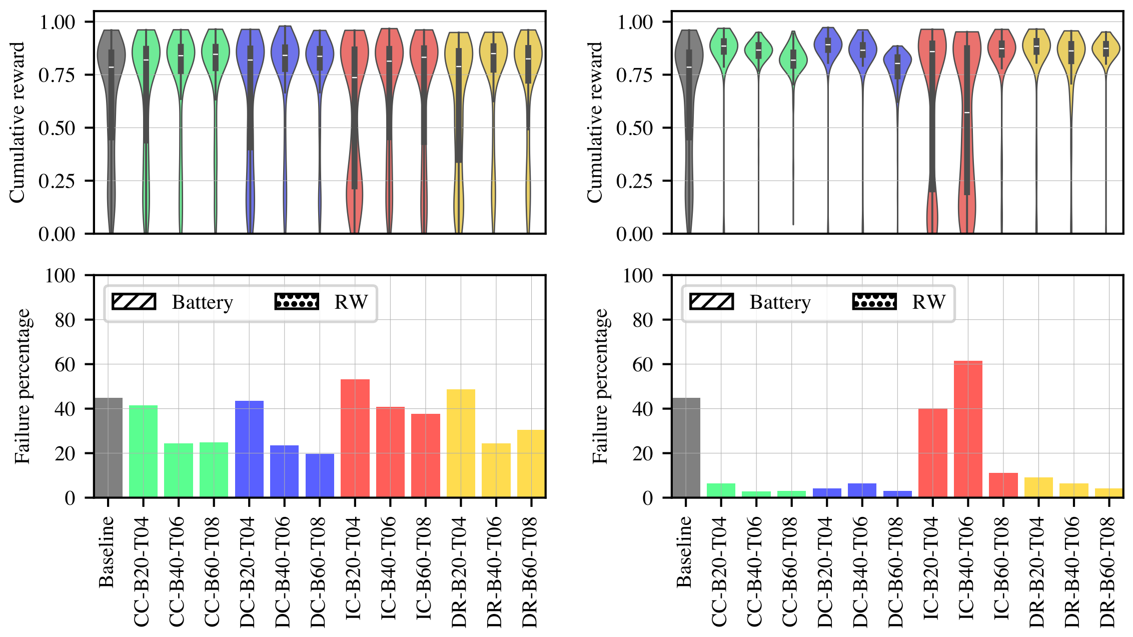


(a) Policies trained with 90 decision intervals.



(b) Policies trained with 450 decision intervals.

Figure 8: Results comparing the effect of training episode length in the degraded environment with 11250 decision intervals. Policies trained with curriculum showed significantly fewer failures when trained with longer episodes.



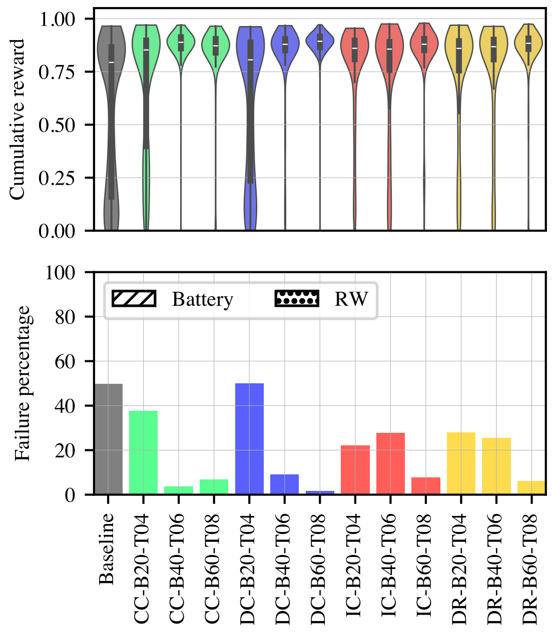
(a) Policies trained with narrow initialization range. (b) Policies trained with wide initialization range.

Figure 9: Comparison between policies trained with 90 decision intervals and narrow and wide initialization range in the degraded environment with 11250 decision intervals. A wider initialization lead to fewer failures; it was also associated with a decrease in cumulative rewards when combined with more intense curriculums.

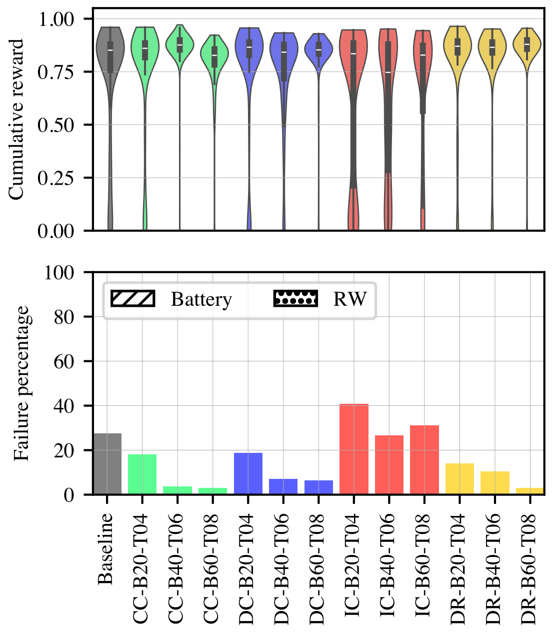
Narrow vs Wide initialization range While extending the length of training episodes can return better experience in terms of the learning perspective, initializing the environment over a wider range of initial conditions can help to estimate the value function better across the state space. To investigate the impact of the initialization range in the training process, policies were trained with the narrow initialization range where the battery fraction and wheel speeds were initialized as indicated in Table 2, and with the wide initialization, where battery level fraction can vary from 0.0 to 1.0 and wheel speeds from -6000 to 6000 RPM. Figure 9 shows the results for policies trained with 90 decision intervals with both narrow and wide initialization. While using a wide initialization range leads to a significant decrease in the failure rate, it is associated with a decrease in cumulative reward. These results indicate that the policies obtained with the wide initialization range are more robust but also overly conservative.

Figure 10 shows the results when combining longer training episodes with a wide initialization range. While most cases benefit from the wide initialization range in terms of failure rate, the cumulative reward is affected. Interestingly, policies that suffered the largest decrease in cumulative reward tend to be policies with more intense curriculums. This outcome indicates that the training environment may become too hard for the agent to generalize well. On the other hand, the baseline case significantly benefits from the wide initialization, showing fewer failures and higher mean cumulative rewards when compared to the narrow case.

While using the wide initialization helps expose the agent to more diverse parts of the state space, it also leads to more failures. The agent ends up being overexposed to terminal states, resulting in a more conservative policy - showing great robustness but also a decrease in cumulative reward.



(a) Policies trained with narrow initialization range.



(b) Policies trained with wide initialization range.

Figure 10: Comparison between policies trained with 450 decision intervals and narrow and wide initialization range in the degraded environment with 11250 decision intervals. A wider initialization lead to fewer failures, especially in less intensive curriculums.

Finding an adequate initialization range can be challenging, but also improve the policy's robustness and generalization.

Despite the results indicating that the use of curriculum leads to more robust policies, all cases showed at least one failure when deployed for 11250 decision intervals. In a real case, the selected policy could be deployed with shields, a technique that monitors the actions selected by the agent and can provide safety guarantees (see Reference 32).

Training time

While different curriculums show different performance in terms of rewards and failures, it is important to consider the training time required for each policy. If a given curriculum takes much longer to finish training, the case without a curriculum could be trained for more steps instead. Figure 11 shows the training time for the policies grouped by episode length and initialization range. The results indicate that policies trained with longer episodes take more time to finish training. Training with a wider initialization range takes longer than with a narrow initialization, which is linked to more satellites starting with initial conditions close to terminal states, more likely to lead to a failure and a reset of the environment. Still, the relation between the time to propagate a step and perform a reset is heavily dependent on the simulation environment.

POLICY ROBUSTNESS IN A HETEROGENEOUS MULTI-AGENT SYSTEM

Robust policies are useful in scenarios where satellite parameters vary over time due to unexpected events or natural degradation. This effect is pronounced when considering a system of mul-

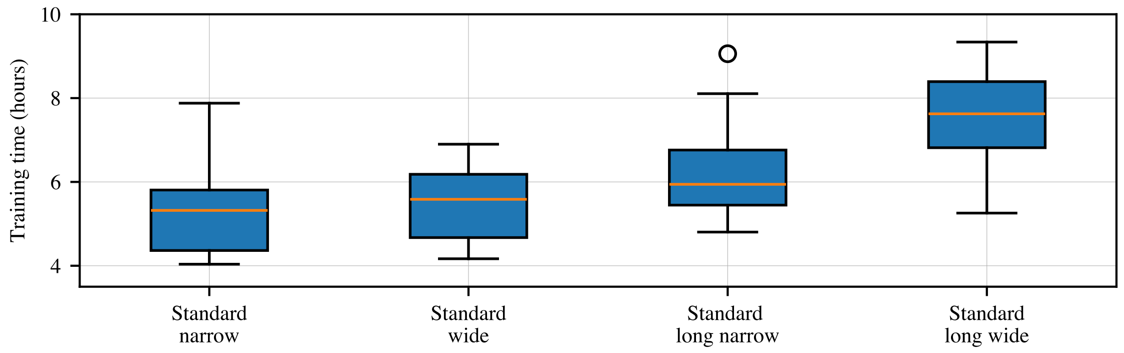


Figure 11: Training time for policies grouped by episode length and initialization range.

tiple agents with ideally identical spacecraft. In reality, each spacecraft will have slightly different parameters, with differences increasing over time. While training one policy per agent might be unfeasible in this case, these agents could use an identical and robust policy.

To demonstrate the usefulness of such robust policies in multi-agent heterogeneous systems, five different satellites were considered, each with a unique combination of battery capacity and external torque. While the first four satellites have fixed parameters, the fifth satellite has time-varying battery capacity and external torque to represent progressive degradation. Table 5 shows the parameters used for each satellite with respect to the nominal values shown in Table 2.

Table 5: Satellite parameters for the multi-agent system

Parameter	Satellite				
	EO-0	EO-1	EO-2	EO-3	EO-4
Battery capacity	100%	100%	50%	80%	100% $\rightarrow$ 40%
External torque	100%	400%	100%	200%	100% $\rightarrow$ 1,000%

The best policy trained without a curriculum was selected, as well as the best policy trained with a curriculum. These policies were selected based on cumulative rewards and failure percentage on the degraded environment with 11250 decision intervals. The best-performing policy without curriculum was the third run trained with 450 decision intervals episodes and wide initialization ranges (0.75 mean reward and 312.87 mean orbits alive). The best policy with a curriculum was the third run of the case trained with the longer episodes (450 decision intervals) and with the DC-B60-T08 curriculum (0.88 mean reward and 351.71 mean orbits alive) with narrow initialization range.

Figure 12 shows the results obtained for the heterogeneous multi-agent system in 100 test runs. The policy in blue represents the DC-B60-T08 case, while the policy in gray represents the baseline. The case trained with curriculum showed significantly fewer failures. Besides, most of its failures occurred close to the end of the test episodes. Higher cumulative rewards were also seen in the case trained with CL. It is relevant to note that, if a satellite fails, the episode is still propagated forward, until the failure of all satellites or the maximum episode duration is reached.

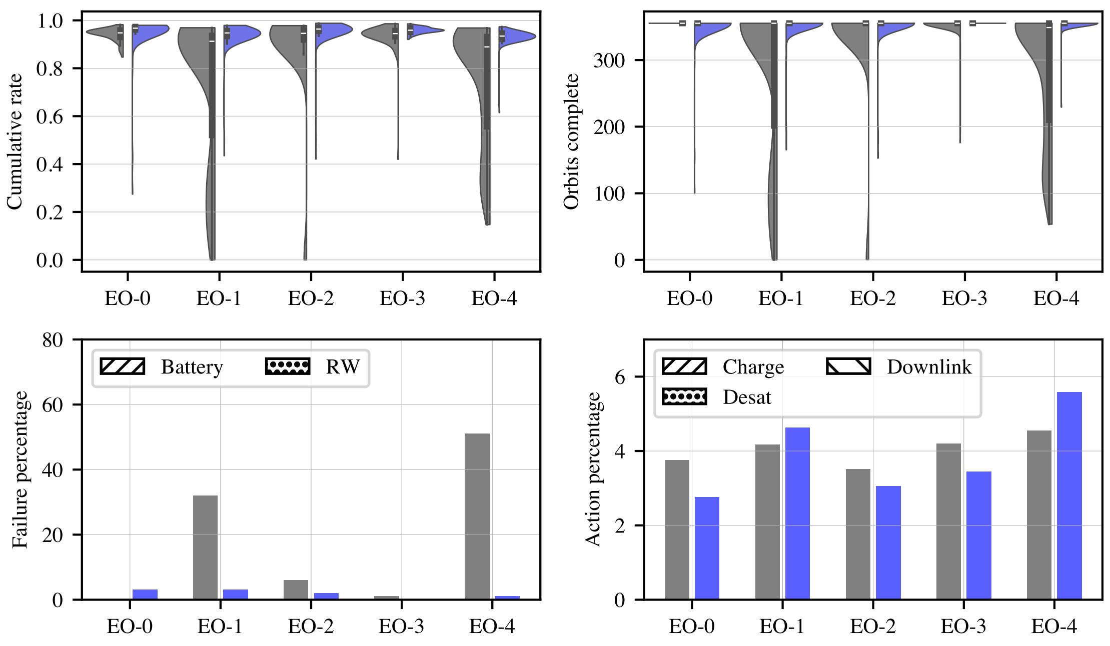


Figure 12: Results for a heterogeneous multi-agent system. Grey represents the best baseline policy (trained for 450 decision intervals with wide initialization), while blue represents the best policy trained with curriculum (direct curriculum with 450 decision intervals, DC-B60-T08).

CONCLUSION

This paper investigates the impact of structured training on enhancing the robustness of DRL-based policies for spacecraft tasking. We explored three curriculum learning (CL) and a domain randomization approaches. The results demonstrate that CL significantly improves policy robustness to battery and reaction wheel failures while maintaining or enhancing cumulative rewards.

Among the approaches tested, direct, constant, and domain randomization yielded the best results. Adding battery and torque to the curriculum decreased the number of associated failures, with intermediate and intense curriculums achieving the best performance. Also, extending the training episodes' length by five times is beneficial for both policy robustness and performance. Still, continuously extending the length in unlikely to lead to endlessly improvements as it happens in detriment of randomization when preserving the same number of training steps. Additionally, increasing randomization intervals lead to more robust policies, but also more conservative.

In this study, the direct and constant curriculums showed the best performance when combined with narrow initialization range, intermediate intensity, and longer training episodes. While some training approaches were explored, the most adequate curriculum, curriculum intensity, training episode length, and parameter randomization range should be treated as additional training parameters to be tuned, being environment dependent. While the trained policies are robust to environmental variations, they currently lack adaptive capabilities to optimize performance based on environment difficulty. If observed or estimated, battery health and external torque information could be passed to the agent in order to have better informed decision-making. This is left as future work.

ACKNOWLEDGMENT

This work is partially supported by the Air Force Research Lab grant FA9453-22-2-0050. Also, 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] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, “Selecting and Scheduling Observations of Agile Satellites,” *Aerospace Science and Technology*, Vol. 6, No. 5, 2002, pp. 367–381, https://doi.org/10.1016/S1270-9638(02)01173-2.

[2] N. Bianchessi, J.-F. Cordeau, J. Desrosiers, G. Laporte, and V. Raymond, “A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites,” *European Journal of Operational Research*, Vol. 177, Mar. 2007, pp. 750–762, 10.1016/j.ejor.2005.12.026.

[3] D. Eddy and M. J. Kochenderfer, “A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations,” *Journal of Spacecraft and Rockets*, Vol. 58, Sept. 2021, pp. 1416–1429, 10.2514/1.A34931.

[4] X. Wang, G. Song, R. Leus, and C. Han, “Robust Earth Observation Satellite Scheduling With Uncertainty of Cloud Coverage,” *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, June 2020, pp. 2450–2461, 10.1109/TAES.2019.2947978.

[5] X. Wang, Y. Gu, G. Wu, and J. R. Woodward, “Robust scheduling for multiple agile Earth observation satellites under cloud coverage uncertainty,” *Computers & Industrial Engineering*, Vol. 156, June 2021, p. 107292, 10.1016/j.cie.2021.107292.

[6] C. Zhang, L. Yuan, M. Xie, S. Zhang, and J. Li, “Autonomous mission planning of Earth observation satellite based on onboard cloud detection,” *Advances in Space Research*, Vol. 70, Oct. 2022, pp. 2178–2194, 10.1016/j.asr.2022.07.007.

[7] A. P. Herrmann and H. Schaub, “Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem,” *Journal of Aerospace Information Systems*, Vol. 19, Jan. 2022, pp. 70–82, 10.2514/1.I010992.

[8] A. Herrmann and H. Schaub, “Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem,” *IEEE Transactions on Aerospace and Electronic Systems*, 2023, pp. 1–13, 10.1109/TAES.2023.3251307.

[9] A. Herrmann and H. Schaub, “A comparative analysis of reinforcement learning algorithms for earth-observing satellite scheduling,” *Frontiers in Space Technologies*, Vol. 4, Nov. 2023, p. 1263489, 10.3389/frspt.2023.1263489.

[10] M. Stephenson and H. Schaub, “Reinforcement Learning for Earth-Observing Satellite Satellite Autonomy with Event-Based Task Intervals,” *AAS Guidance and Control Conference*, Breckenridge, CO, Feb. 2–7 2024. Paper No. AAS 24-012.

[11] A. Herrmann, M. A. Stephenson, and H. Schaub, “Single-Agent Reinforcement Learning for Scalable Earth-Observing Satellite Constellation Operations,” *Journal of Spacecraft and Rockets*, Nov. 2023, pp. 1–19, 10.2514/1.A35736.

[12] M. Stephenson, L. Q. Mantovani, and H. Schaub, “Intent Sharing For Emergent Collaboration In Autonomous Earth Observing Constellations,” *AAS Guidance and Control Conference*, Breckenridge, CO, Feb. 2–7 2024. Paper No. AAS 24-192.

[13] L. Q. Mantovani, Y. Nagano, and H. Schaub, “Reinforcement Learning for Satellite Autonomy Under Different Cloud Coverage Probability Observations,” *AAS Astrodynamics Specialist Conference*, Broomfield, CO, Aug. 11–15 2024. Paper No. AAS 24-189.

[14] M. Stephenson, L. Q. Mantovani, S. Phillips, and H. Schaub, “Using Enhanced Simulation Environments to Improve Reinforcement Learning for Long-Duration Satellite Autonomy,” *AIAA Science and Technology Forum and Exposition (SciTech)*, Orlando, FL, Jan. 8–12 2024, 10.2514/6.2024-0990.

[15] Yoshua Bengio, Y. Bengio, Jérôme Louradour, J. Louradour, Ronan Collobert, R. Collobert, Jason Weston, and J. Weston, “Curriculum learning,” June 2009, pp. 41–48. MAG ID: 2296073425, 10.1145/1553374.1553380.

[16] P. Soviany, R. T. Ionescu, P. Rota, and N. Sebe, “Curriculum Learning: A Survey,” Apr. 2022. arXiv:2101.10382 [cs].

[17] C. Florensa, D. Held, M. Wulfmeier, M. Zhang, and P. Abbeel, “Reverse Curriculum Generation for Reinforcement Learning,” July 2018. arXiv:1707.05300 [cs].

[18] J. West, F. Maire, C. Browne, and S. Denman, “Improved reinforcement learning with curriculum,” *Expert Systems with Applications*, Vol. 158, Nov. 2020, p. 113515, 10.1016/j.eswa.2020.113515.

[19] Lukás Hermann, L. Hermann, Max Argus, M. Argus, Andreas Eitel, A. Eitel, Artemij Amiranashvili, A. Amiranashvili, Wolfram Burgard, W. Burgard, Thomas Brox, and T. Brox, “Adaptive Curriculum Generation from Demonstrations for Sim-to-Real Visuomotor Control.,” *arXiv: Robotics*, Oct. 2019. MAG ID: 2980595763, 10.1109/ICRA40945.2020.9197108.

[20] B. Tidd, N. Hudson, and A. Cosgun, “Guided Curriculum Learning for Walking Over Complex Terrain,” Feb. 2021. arXiv:2010.03848 [cs].

[21] C. Xiao, P. Lu, and Q. He, “Flying Through a Narrow Gap Using End-to-End Deep Reinforcement Learning Augmented With Curriculum Learning and Sim2Real,” *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 34, May 2023, pp. 2701–2708, 10.1109/TNNLS.2021.3107742.

[22] L. Federici and A. Zavoli, “A Curriculum Learning Approach for Improving Constraint Handling in Reinforcement Learning Applications to Spacecraft Guidance and Control,” Oct. 2024. Publisher: Zenodo Version Number: 2024-10-03, 10.5281/ZENODO.13885667.

[23] G. B. Margolis, G. Yang, K. Paigwar, T. Chen, and P. Agrawal, “Rapid Locomotion via Reinforcement Learning,” May 2022. arXiv:2205.02824 [cs].

[24] N. Rudin, D. Hoeller, P. Reist, and M. Hutter, “Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning,” Aug. 2022. arXiv:2109.11978 [cs].

[25] I. M. Aswin Nahrendra, B. Yu, and H. Myung, “DreamWaQ: Learning Robust Quadrupedal Locomotion With Implicit Terrain Imagination via Deep Reinforcement Learning,” *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, United Kingdom, IEEE, May 2023, pp. 5078–5084, 10.1109/ICRA48891.2023.10161144.

[26] H.-C. Wang, S.-C. Huang, P.-J. Huang, K.-L. Wang, Y.-C. Teng, Y.-T. Ko, D. Jeon, and I.-C. Wu, “Curriculum Reinforcement Learning From Avoiding Collisions to Navigating Among Movable Obstacles in Diverse Environments,” *IEEE Robotics and Automation Letters*, Vol. 8, May 2023, pp. 2740–2747, 10.1109/LRA.2023.3251193.

[27] M. Turchetta, A. Kolobov, S. Shah, A. Krause, and A. Agarwal, “Safe reinforcement learning via curriculum induction,” *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS ’20, Red Hook, NY, USA, Curran Associates Inc., 2020. event-place: Vancouver, BC, Canada.

[28] S. Narvekar, B. Peng, M. Leonetti, J. Sinapov, M. E. Taylor, and P. Stone, “Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey,” Sept. 2020. arXiv:2003.04960 [cs, stat].

[29] R. Portelas, C. Colas, L. Weng, K. Hofmann, and P.-Y. Oudeyer, “Automatic curriculum learning for deep RL: a short survey,” *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI’20, 2021. Place: Yokohama, Yokohama, Japan.

[30] M. Stephenson and H. Schaub, “BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking,” *International Astronautical Congress*, Milan, Italy, Oct. 14–18 2024.

[31] P. W. Kenneally, S. Piggott, and H. Schaub, “Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework,” *Journal of Aerospace Information Systems*, Vol. 17, Sept. 2020, pp. 496–507.

[32] R. Reed, H. Schaub, and M. Lahijanian, “Shielded Deep Reinforcement Learning for Complex Spacecraft Tasking,” Mar. 2024. arXiv:2403.05693 [cs].