Performance Evaluation of Shielded Neural Networks for Autonomous Agile Earth Observing Satellites in Long Term Scenarios

Lorenzzo Quevedo Mantovani[1] **and Hanspeter Schaub**[1]
[1]Ann and H.J. Smead Aerospace Engineering Sciences Department
University of Colorado Boulder
429 UCB, 3775 Discovery Drive
Boulder, Colorado 80303 USA
lorenzzo.quevedomantovani@colorado.edu

Abstract

This paper investigates the impact of auto-generated shields on the performance of policies trained using Deep Reinforcement Learning (DRL) in the context of autonomous scheduling for Agile Earth Observing Satellites. The planning and scheduling phase during spacecraft operations determines the sequence of actions the satellite should take to meet the mission requirements while adhering to the system's constraints. Traditionally, the scheduling has been handled by human operators or by offline optimization techniques. However, the increasing number of satellites and the growing demand for their services make the problem more challenging. Additionally, the need for fast re-planning due to unexpected events or additional requests pushes for the need for fast schedulers that can provide solutions in real time. Recently, DRL has shown promise for on-board scheduling, but deploying neural networks in critical systems, such as satellites, raises reliability concerns. To overcome this problem, recent research has proposed using Shielded Neural Networks to provide safety guarantees. This work provides a comprehensive analysis of the impact of different shield formulations on agent performance, considering agents trained in 3-orbit episodes and deployed in 90-orbit episodes. Results show that despite the decrease in the reward rate in shielded agents, the cumulative reward is higher due to their longer survivability. Moreover, shields provide safety guarantees, allowing agents to be deployed in episodes thirty times longer than their training episodes. While shields can prevent agents from dying, policies trained with safety concerns can perform better than policies trained without safety concerns, especially for higher target densities.

Introduction

Earth Observing Satellites (EOS) are equipped with sensing instruments to acquire images of Earth's surface, which can be used for multiple purposes such as crop monitoring, intelligence gathering, and disaster response. During the spacecraft operation, there is a planning and scheduling phase responsible for providing the sequence of actions the satellite should take to meet the mission requirements while respecting the system's constraints. The advent of agile EOSs (AEOSs) adds complexity to the problem due to their extra maneuverability capabilities, both along- and across-track, increasing the solution space. The growing demand for satellite imagery has led to oversubscribed systems, making the scheduling problem increasingly complex.

Originally, optimization techniques were used to solve the EOS scheduling problem. The AEOS scheduling problem was shown to be NP-hard, and different optimization techniques were investigated to solve the problem, such as greedy algorithms, dynamic programming, constraint programming, and local search (Lemaître et al. 2002). Heuristics were employed in the context of multi-satellite systems (Bianchessi et al. 2007), and a local search algorithm to maximize the total reward obtained by an AEOS while minimizing the reward difference of users sharing the platform (Tangpattanakul, Jozefowiez, and Lopez 2015). More recently, the use of an infeasibility-based graph was proposed, where the best solution corresponds to the maximum independent set of the graph (Eddy and Kochenderfer 2021). The method was investigated with up to 10,000 requests and 24 satellites, assuming the satellite to slew at a constant rate and not considering power constraints. Other papers proposed the use of budgeted uncertainty to deal with cloud coverage uncertainty and showed the possibility of solving the problem using column generation and simulated annealing methods (Wang et al. 2020, 2021a). On-board re-plans based on cloud coverage information were also proposed (Zhang et al. 2022). Wang et al. (2021b) have an overview of the AEOS scheduling problem and the methods proposed to solve it.

Despite advances in optimization techniques, the AEOS scheduling problem remains challenging, especially when accounting for the need for fast re-planning due to unexpected events, such as cloud coverage or additional requests. This pushes for the need for fast schedulers that can provide solutions in real-time and closed-loop format. The use of Reinforcement Learning combined with Deep Neural Networks (DNNs), known as Deep Reinforcement Learning (DRL), has shown the ability to effectively solve the scheduling problem under complex dynamics constraints while presenting high adaptability. The fast evaluation of the DNNs after training shows the potential of DRL for on-board autonomy. In these methods, the problem is formulated as a Markov Decision Process (MDP) or as a Partially Observable MDP (POMDP) when only partial information about the states is available. Techniques such as Monte Carlo Tree Search (MCTS) or Proximal Policy Op-

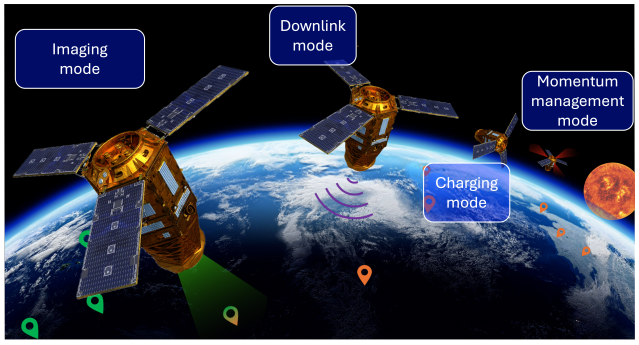

Figure 1: Illustration of AEOS running DRL-based policies for on-board closed-loop decision-making.

timization (PPO) are then used to obtain a policy.

For example, MCTS with Neural Networks has been tested to solve the scheduling problem for EOS and AEOS, performing similarly to a genetic algorithm (Herrmann and Schaub 2022, 2023b). An in-depth comparison of different DRL algorithms for EOS scheduling, such as A2C, DQN, and PPO, is provided in (Herrmann and Schaub 2023a). Performance comparison of DRL with PPO and Mixed Integer Linear Programming (MILP) under different target distributions was also investigated, showing promising results (Stephenson and Schaub 2024b). DRL has also been shown to be effective in multi-satellite deployment when communication is used to create collaboration among agents (Herrmann, Stephenson, and Schaub 2023; Stephenson, Mantovani, and Schaub 2024). The uncertainty of clouds above targets was also accounted for in the AEOS scheduling problem, and the performance of agents with different observation capabilities was investigated (Mantovani, Nagano, and Schaub 2024). Figure 1 illustrates AEOSs running DRL-based policies in a closed-loop format.

However, the use of such policies obtained from NN for autonomy with minimal human intervention in critical systems, such as satellites, raises concerns about their reliability. To overcome this problem, recent papers proposed the use of Shielded Neural Networks (SNN) to provide guarantees on safety (Alshiekh et al. 2018). Shields are reactive systems that check the action selected by the agent and correct it if it leads to an unsafe state. They can be designed using the temporal logic specification.

Shields have been applied to the satellite scheduling problem, where the spacecraft must maximize its rewards obtained by scanning nadir (Harris and Schaub 2020). The problem is formulated as a POMDP, and the simulation is implemented in Basilisk[1], which provides high-fidelity dynamics and constraints (Kenneally, Piggott, and Schaub 2020). Two safety aspects are considered: battery levels and reaction wheel speeds (since speeds over the limit can drive the system unstable). The authors construct a safety MDP from the original POMDP, which includes unsafe states that the agent should avoid. While the policy is obtained using PPO, the PRISM-games solver is used to solve the safety MDP and obtain the shield; agents are trained with and without shields for comparison. The results indicate that shielded agents effectively avoid unsafe states and return to safe regions of the state space.

SNNs have also been used in the scheduling of AEOS with point targets, and, in addition to battery levels and reaction wheel speeds, the satellite's angular velocity is also considered a safety aspect (Nazmy et al. 2022). Although the agent was trained to image a specific location on Earth, the results indicate that the SNN could still keep the agent safe when deploying in different environments, such as the Moon, due to the use of canonical coordinates. The performance of DRL and a rule-based policy, both shielded, were compared in a multi-satellite system (Bajenaru et al. 2023). The results indicate that the DRL policy outperforms the rule-based policy with more imaged targets while showing less action interference from the shield. The problem considered up to 1,000 requests submitted to the satellites.

The work of shields for autonomous spacecraft operations was expanded using Linear Temporal Logic (LTL) to specify the safety requirements and presents a methodology to create a safety MDP used to obtain the shields (Reed, Schaub, and Lahijanian 2024). Three distinct shield algorithms were presented: One-Step, Two-Step, and Q-Optimal. The One-Step shield provides actions that will lead the agent to a safe state. The Two-Step shield provides actions that will lead the agent to a state that is safe and has an action that will also lead to a safe state. The Q-Optimal shield uses the Q-value of the optimal safety MDP to determine the probability of an action remaining safe under that strategy. It only allows actions that have a safe probability above a threshold. Their results indicate that policies trained without shields in an environment with safety constraints but deployed with shields lead to the best interaction between policy and shield, seen in fewer safety violation and lower shield interference.

Previous work demonstrated efficient ways to construct shields for DNN and their applicability. Still, there is a need to investigate the impact of shields on the performance of the autonomous agent policy in more detail and in more complex scenarios. Therefore, this work provides a comprehensive analysis of the impact of shields on autonomous agent performance and evaluates the trade-offs between safety and performance. The main contribution of this paper is to analyze the impact of different shield formulations introduced by Reed, Schaub, and Lahijanian (2024) on agent's performance with more detailed test cases; agents were trained in 3-orbit episodes and deployed in 90-orbit episodes to analyze long-term survivability. This paper also investigates the performance of different policies with similar shields, introducing an agent trained in an environment without safety concerns as a baseline. The number of requests submitted to the satellite varies from 1,000 to 10,000, providing a more comprehensive analysis than previous related works.

Methodology

Problem Formulation

The AEOS scheduling problem can be formulated as a decision-making process using a POMDP, a version of an

[1]https://avslab.github.io/basilisk/

MDP where the agent does not have full observability of the state space. As an MDP, a POMDP has a reward function R, state space $\mathcal{S}$, action space $\mathcal{A}$, and a transition function T where the next state s' depends only on the current state s and action a (Markov independence (Sutton and Barto 2020)). Additionally, POMDPs possess an observation space $\mathcal{O}$ and observation function Z being defined by a tuple $\langle \mathcal{S}, \mathcal{A}, T, \mathcal{O}, Z, R \rangle$ (Kaelbling, Littman, and Cassandra 1998) where:

- State space $\mathcal{S}$: Contains information about the spacecraft states, targets, and internal variables to the simulator. $\mathcal{S}$ consists of continuous and discrete states.
- Action space $\mathcal{A}$: Contains 35 discrete actions that the agent can take. The satellite can enter charge mode (a_{charge}), downlink mode (a_{downlink}), momentum management mode (a_{desat}), or image one of the 32 upcoming targets ($\mathcal{A}_{\text{image}} = \{a_{\text{im},j} | j \in [1, 32]\}$).
- Transition probability function $T(s'|s, a)$: Transitions are deterministic and generated by a generative model G that simulates the spacecraft dynamics ($s' = G(s, a)$).
- Reward function $R(s, a, s')$: A target request t_i is defined by its priority ρ_i and position $\mathbf{r}_i$ such that $t_i = (\rho_i, \mathbf{r}_i)$. Initially, all targets are in an unfulfilled list of requests $\mathcal{U}$; a target is moved to a fulfilled list F when it is successfully imaged. The agent receives a reward based on the target's priority, ρ_i such that:

$$R(s, a, s') = \begin{cases} \rho_i & \text{if } t_i \in \mathcal{U} \text{ and } a \in \mathcal{A}_{\text{image}} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

- Observation space $\mathcal{O}$: Is a subset of the dimensions in $\mathcal{S}$ and contains information the agent can observe and is useful for the decision-making problem. It includes data on the next 32 upcoming targets in $\mathcal{U}$, while internal simulator variables and targets beyond this set at a given timestep remain unobserved. Table 1 describes the observation space (normalized to be between -1 and 1, a common practice for DNNs).
- Observation function $Z(o|s', a)$: The observation function is deterministic since the satellite is assumed to observe the observation space perfectly with $o \in \mathcal{O}$.

Although all targets are created at the beginning of each episode, only a subset of them is available for imaging and in the observation space at each time step. This subset contains the following N upcoming targets in $\mathcal{U}$ in terms of their imaging window, making the system scalable to any number of targets in the environment. From the agent's perspective, targets are arriving dynamically. Stephenson and Schaub (2024b) investigate the agent's performance with different N in the observation and action spaces, showing $N = 32$ to lead to good performance, and compared it to a MILP solution.

Simulation Environment

The simulation was implemented in BSK-RL[2], a Python package focused on environments for spacecraft planning and tasking (Stephenson and Schaub 2024a). BSK-RL combines the Gymnasium Python package used for RL studies with Basilisk, which is used as the generative model G. Basilisk is an astrodynamics simulation framework that can simulate high-fidelity spacecraft dynamics and its systems and subsystems (Kenneally, Piggott, and Schaub 2020). It is written in C and C++ and has a Python interface, which makes it easy to interact and fast for training and testing.

The simulation environment accounts for the spacecraft dynamics, reaction wheels, battery subsystem, storage, and downlink capabilities. Therefore, if the agent tasks an action to image an incoming target, it is not guaranteed to succeed. The tasked action specifies the desired attitude to the closed-loop control module, which then computes the reaction wheel torques, with the spacecraft's attitude being propagated by the high-fidelity simulator. During the maneuver, the simulator will check if the imaging constraints are met, such as the relative angle between the target and the spacecraft, the relative angle rate, and the observation window. If the constraints are met, the image is taken, and the target is moved to the fulfilled list. When successfully imaging a target, the storage space available is reduced.

Similar to imaging actions, the downlink action is not

Parameter	Description
${}^{\mathcal{P}}\hat{r}_{\mathcal{B}/\mathcal{N}}$	Position of the satellite with respect to the inertial frame $\mathcal{N}$, expressed in the planet frame $\mathcal{P}$ and normalized by the Earth's radius.
${}^{\mathcal{P}}\hat{v}_{\mathcal{B}/\mathcal{P}}$	Velocity of the satellite with respect to $\mathcal{P}$, expressed in $\mathcal{P}$ and normalized by $\lvert{}^{\mathcal{P}}\mathbf{v}_{B/N}\rvert$.
$\hat{c}_{\mathcal{B}/\mathcal{P}}$	Orientation of the satellite's sensing instrument with respect to $\mathcal{P}$, normalized by π.
${}^{\mathcal{P}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{P}}$	Angular velocity of the satellite with respect to $\mathcal{P}$, expressed in $\mathcal{P}$. Normalized by 0.03.
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.
ρ_i	Priority of target i for the 32 upcoming targets in $\mathcal{U}$.
α_i	Angle of target i with respect to the satellite's sensing instrument for the 32 upcoming targets in $\mathcal{U}$. 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.
t	Time since the beginning of the episode. Normalized by the episode duration.

Table 1: Observation space

[2]https://avslab.github.io/bsk_rl/

guaranteed to succeed unless the spacecraft meets the requirements and is within the reach of a ground station. The downlink action frees storage space, accounting for the amount of stored data and downlink rate. Although rewards are not awarded for downlinking information, the agent cannot add more images to storage space if it is full, which requires it to downlink data.

When tasking action charge, the satellite will maneuver to position its solar panels to maximize its energy generation. Still, charging is not guaranteed to be successful, as the satellite must not be in eclipse. Despite the charge action, the satellite will passively charge its batteries whenever there is adequate solar incidence, with the charging rate as a function of the solar incidence angle. At all times, the satellite has a baseline power consumption to represent essential subsystems, in addition to the power consumption of the imaging and downlink instruments when in use and the reaction wheels as a function of their speed.

Lastly, the momentum management action is responsible for desaturating the reaction wheels. Reaction wheels are used to control the spacecraft's attitude and accumulate momentum over time in the presence of external torques. The momentum management action aligns the spacecraft with an inertial reference frame and fires thrusters to desaturate the reaction wheels.

Table 2 shows some spacecraft parameters, such as mass, inertia, and subsystem information, such as storage space. Values not mentioned here are set as the standard values for the modules in BSK-RL.

Altitude	500 km
Inclination	$45°$
Mass	330 kg
Battery capacity	160 W·s
Base power consumption	20 W
Data storage capacity	50 MB
Relative angle limit for imaging	$28°$
Relative angular rate limit for imaging	0.01 rad/s
Minimum target image elevation	$45°$
Reaction wheels maximum torque	0.2 N·m
Reaction wheels maximum speed	1500 RPM
K_1 gain for the MRP steering control	0.5
Initial battery charge fraction	$[0.4, 1.0]$
Initial storage space fraction	$[0.0, 0.75]$
Initial reaction wheels speed fraction	$[-0.5, 0.5]$
Number of requests	$[1000, 10000]$
ρ_i	[0, 1]

Table 2: Spacecraft parameters

The implemented simulation uses an event-based tasking method, where, after the agent takes an action, the simulator will propagate the simulation until certain conditions are met (Stephenson and Schaub 2024b). For imaging actions, the satellite simulator will propagate the dynamics until the target is successfully imaged, the opportunity to image it is missed, or a determined maximum time step is reached. Similarly, downlink actions will be propagated until the data is successfully downlinked, the opportunity is missed, or a determined time step is reached. Charge and momentum management actions will be propagated until the determined time step is reached. The maximum time step for imaging and downlink actions is 300 seconds; for other actions, the maximum time step is 60 seconds.

As safety constraints, the satellite must always keep the battery charge level above zero. Further, the reaction wheels must be below a maximum angular speed threshold. The episode terminates if any of the conditions are not fulfilled.

Target Distribution

The spacecraft was tested in two different scenarios of target distributions, uniform and city-based. Uniformly distributed targets are randomly distributed over the Earth's surface; city-based targets are distributed according to the world's most populated cities (Stephenson and Schaub 2023). The essential difference between the two distributions is the clustering of targets in the city-based distribution. Although tested with both distributions, agents were trained with a Swiss-cheese model, which generates areas of low and high target densities (Stephenson and Schaub 2024b). The Swiss-cheese model is obtained by sampling targets uniformly across the globe and up to fifteen exclusion zones with different radii. Targets inside the exclusion zones are removed, creating empty space regions. The Swiss-cheese presents more diverse cases from which the agent can learn.

Training and Testing Environments

Training Environment

A training environment was set up to train the agent and obtain a policy using DRL. The Asynchronous PPO (APPO) algorithm (Luo et al. 2020) from RLlib (Liang et al. 2018) was used, which is a variant of PPO (Schulman et al. 2017). Table 3 contains the hyperparameters used for training; other parameters were set as the standard values in RLlib version 2.6.3. See Herrmann and Schaub (2023a) for hyperparameter analysis for the AEOS scheduling problem. The training was performed in the University of Colorado Research Computing (CURC) (University of Colorado Boulder Research Computing 2023) using 32 cores and up to 20M steps.

Number of workers	32
Number of Training steps	$20 \cdot 10^6$
Learning rate	$3 \cdot 10^{-5}$
Training bath size	$10,000$
Minibatch size	250
Epochs	50
Neural Network	2 layers, 512 neurons each
Discount factor	0.999
Failure penalty	0

Table 3: Training hyperparameters

The training was performed without any shield mechanisms, so agents would need to learn the safety aspects of the problem related to battery charge and reaction wheels' maximum speed. This work focuses on policies trained without shields because they show better policy-shield interaction

when deployed with shields (after training) (Reed, Schaub, and Lahijanian 2024).

In total, three different policies were trained. Policy π_1 was trained in the nominal environment, while policy π_2 was trained with the external torque set to $1 \cdot 10^{-4}$ N·m to simulate a more challenging environment. External torque leads to higher momentum on the reaction wheels, also increasing power consumption. Due to the more difficult environment, the agent was expected to be more conservative regarding safety and present a higher survivability rate. The third policy, π_3, was trained using a modified simulation environment with excessive battery capacity and reaction wheel speeds so that it would not need to learn the safety aspects of the problem. Therefore, π_3 is extremely greedy, taking only imaging and downlink actions; it is used as a baseline to compare the performance of the safety-aware policies (π_1 and π_2) when deployed with shields.

All policies were trained with zero failure penalty since, as investigated by Stephenson and Schaub (2024b), policies trained with zero failure penalty and no shields but deployed with shields outperform policies trained with a large failure penalty. Still, the discount factor was set close to one, providing a small discount on future rewards and incentivizing survivability. Also, the three-orbit-long episodes are treated as truncated episodes.

Shielded Neural Networks

Shield mechanisms are combined with the policy during deployment to prevent the agent from taking actions that can lead to unsafe states. The framework utilized in this paper is based on the work of Reed, Schaub, and Lahijanian (2024), which uses an SNN to provide guarantees on safety. Therefore, the agent will decide on an action, which is verified by the shield afterward (post-posed) (alternatively, action masking could be combined with APPO, where only safe actions are available to the agent (Tang et al. 2020)). If the shield detects that an action would lead to an unsafe state, it overrides the agent's choice and selects an action most likely to take the agent to a safe region.

In this paper, four different shields are investigated. Three shields are from the work of Reed, Schaub, and Lahijanian (2024), where a lower dimensional safety MDP is abstracted from the original POMDP. The safety MDP is obtained by partitioning safety relevant states, keeping it trackable while preserving the Markov property; transition probabilities are obtained through simulations. The fourth shield is handmade based on expert knowledge of the problem. The shields consider battery charge level and reaction wheel speeds as safety aspects in all cases. The shields are described as follows:

- One-Step: Provides a guarantee that the agent will remain in a safe state for at least one step.
- Two-Step: Provides a guarantee that the agent will remain in a safe state in the next step and will have at least one action leading to a safe state in the second time step.
- Q-Optimal: Checks the Q-value (of the safety MDP, abstracted from the POMDP; see Reed, Schaub, and Lahijanian (2024)) of each action and allows only action that results in a safe probability larger than a threshold
- Handmade: Is based on expert knowledge of the problem, having heuristic rules to enforce safety.

The Handmade shield is based on the works of Herrmann and Schaub (2023a) and Stephenson and Schaub (2024b) and will override the agent's selection to take action charge if it is not in eclipse and the battery charge is below a certain threshold. The battery threshold depends on a minimum energy threshold, P_{min}, and the time until the next eclipse. Therefore, the Handmade shield will obtain the eclipse duration, e_d, and estimate the energy consumption during the eclipse, P_e, in that period with

$$P_e = e_d P_d \tag{2}$$

where P_d is an estimate of the satellite's power consumption. The minimum energy threshold, P_t, is calculated as

$$P_t = P_e + P_{\text{min}} - t_e P_c \tag{3}$$

where P_c is the passive charging rate of the satellite, and t_e is the time to the next eclipse. The Handmade shield will also force the agent to take the momentum management mode to desaturate the wheels if any of the wheels' angular speed fraction is above a threshold $\hat{W}_t$. If both conditions are met, the Handmade shield will select the charge action. Despite using expert knowledge, the Handmade shield is expected to fail in edge cases that are not considered in the design. For this work, $P_{\text{min}} = 0.25$, $P_d = 1.0$, $P_c = 1.0$, and $\hat{W}_t = 0.7$.

These shields were designed to be checked every 60 seconds, which may not meet the retasking time because of the variable event-based tasking method. Therefore, the shields are checked every 60 seconds, even if the agent has an ongoing task. If no correction is needed, the agent continues the ongoing task; otherwise, the shield overrides the ongoing action with a safe action.

Results

The results present the performance comparison of the different policies with different shields. The number of requests, $|R|$, was varied from 1,000 to 10,000 (where $R = \mathcal{U} \cup \mathcal{F}$, with all targets initially in $\mathcal{U}$). Fifty test cases were run for each combination of policy, shielded case, target distributions, and number of requests; results report the average of those runs and the associated standard error of the mean. As aforementioned, policies were trained in 3-orbit-long episodes. However, the final goal is a policy that can be deployed for the mission's lifetime. Therefore, tests were conducted with 90-orbit-long episodes, representing a thirty times increase in episode length, to identify the effect of long-term deployment. First, the performance of each policy with different shields is presented. Later, the performance of different policies with the same shield is investigated.

Performance of Policies with Different Shields

The agent's performance is evaluated in terms of the cumulative reward collected in the episode and the reward per orbit (reward rate), which is the cumulative reward divided by the number of orbits the agent was alive. Additionally, the percentage of shield interference is calculated, which is the

percentage of times the shield corrected the agent's action divided by the total number of actions taken. The number of failures per orbit is also reported.

Figure 2 shows the number of cases alive as a function of the number of orbits for each shielded case for policy π_1. While the number of cases alive at the end of 90-orbit episodes is 49.8% for the unshielded cases, it is 99.67% for the Handmade shield, 99.8% for Q-Optimal and Two-Step shields, and 100% for One-Step shields.

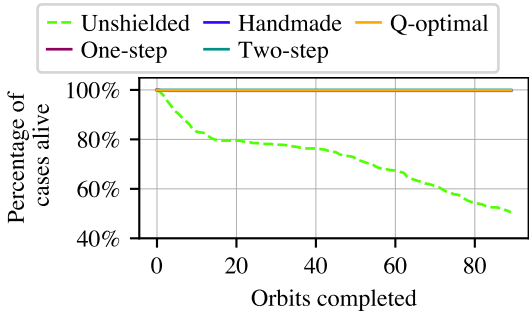

Figure 2: Number of cases alive per orbit for π_1 for city and uniform targets. Lines for shielded cases overlap, showing similar survivability between 99.67% and 100%.

Figure 3 shows the results for policy π_1 with city-based targets. Despite unshielded cases showing the highest reward per orbit, all shielded cases achieve higher cumulative rewards. This result indicates that, despite the lower reward per orbit, the fact that the agent is alive for longer allows it to collect more reward. On average, Q-Optimal, One-Step, and Two-Step shields interfered in actions 6.00%, 5.67%, and 5.68% of the time, respectively, for city-based targets; for uniform targets, the average shield interference is 5.92%, 5.64%, and 5.92%, respectively. The Handmade shield interfered 1.54% with city targets and 1.60% with uniform targets. On the other hand, 0.33% of the Handmade cases failed, while the other shields showed 0.17% or fewer failures. Despite the different interference levels, all shielded cases present similar cumulative rewards.

Since dead satellites affect the total reward in the episode, Fig. 4 shows the cumulative reward obtained in the episode, similar to Fig. 3, but only considering the elite group - cases that survived the 90-orbit episodes for city and uniform targets. It shows that elite unshielded cases have a similar cumulative reward to the shielded cases, indicating that the shields have little effect on the reward collected by the agent while adding safety guarantees.

Policy π_2 was trained in a similar environment as π_1, but with a slightly external torque, which leads to a different and more conservative policy. As expected, π_2 showed better survivability than π_1, with 76.67% of the unshielded cases surviving the 90-orbit episodes and all shielded cases showing 100% survivability. Figure 5 indicates the cumulative reward obtained by the different shielded cases for policy π_2 when using city and uniform based targets. The difference between the unshielded case and shielded cases in terms of rewards is smaller for policy π_2 than for policy π_1 due to its

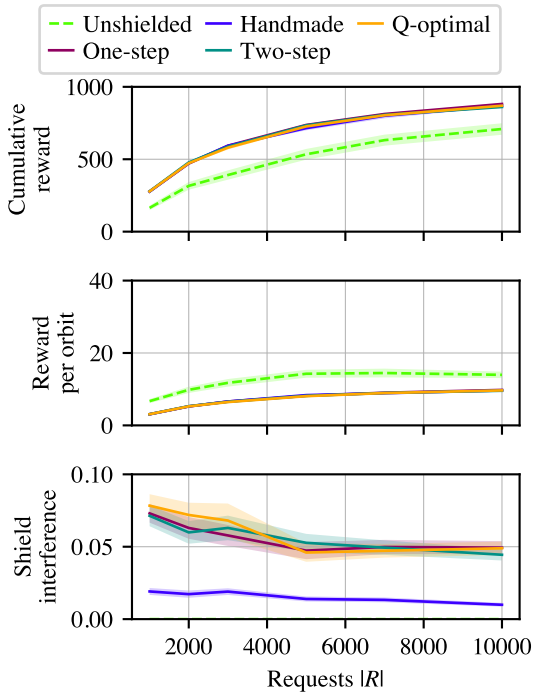

Figure 3: Performance metrics for π_1 for city-based target distribution. While shielded policies cumulative reward overlap, the handmade shield shows less interference.

higher survivability. The average shield interference across city and uniform target distributions are below 3.66% for the Q-Optimal, One-Step, and Two-Step shields. The Handmade shield has an average shield interference of 0.75%. These results indicate that policy π_2 is more conservative than π_1, which is reflected in the higher survivability and lower shield interference compared to π_1.

Policies π_1 and π_2 were trained without a shield and learned the safety aspects of the problem. For comparison, π_3, which was trained in an environment without safety concerns, was also tested. Figure 6 reports the number of cases alive as a function of the number of orbits and shows that all unshielded cases died before the end of the 90-orbit episodes (agents survived for 2.38 orbits on average). In contrast, 99.5% and 99.83% of the cases with the Handmade and Two-Step shields survived, respectively. All agents with Q-Optimal and One-Step shields survived the 90-orbit episodes.

Similarly to Fig. 3, Fig. 7 indicates that shielded cases achieve higher cumulative rewards than unshielded cases. The average shield interference for the Q-Optimal, One-Step, and Two-Step shields is 37.96%, 37.48%, and 37.49%, respectively. The Handmade shield has an average shield interference of 26.77%.

Among shielded cases in Fig. 3, the Handmade shield has the lowest shield interference and tends to show the highest reward per orbit. Still, it presents more failure cases than other shields since it is constructed using expert knowledge but has no mathematical guarantees on safety, still being sus-

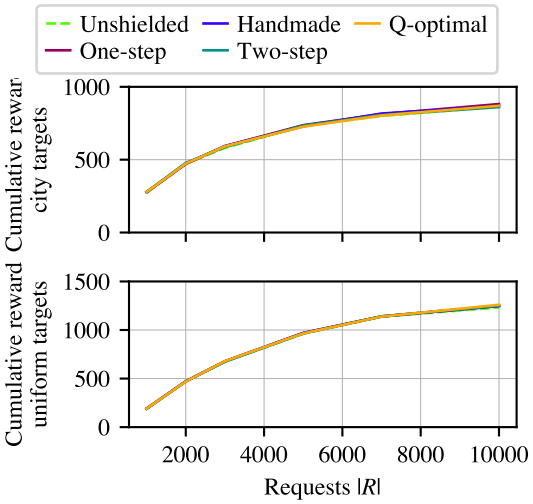

Figure 4: Policy π_1 results for elite groups. City (top) and uniform (bottom) targets. All lines overlap, showing similar performance among the best performing cases.

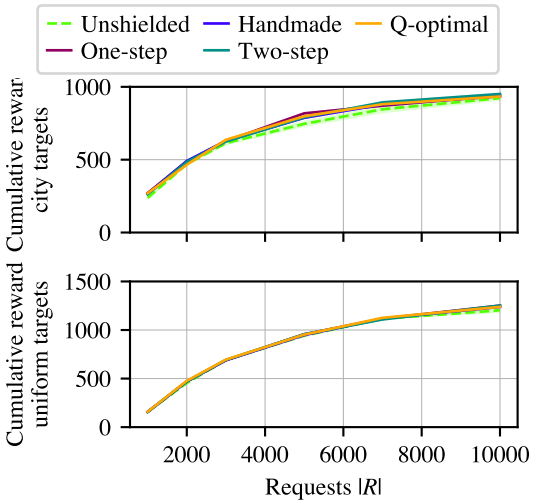

Figure 5: Performance metrics for π_2. City (top) and uniform (bottom) targets. The unshielded performance is closer to the shielded cases when considering a uniform target distribution.

ceptible to edge cases leading to failure. On the other hand, Q-Optimal, One-Step, and Two-Step shields presented the smallest amount of failures in the test cases. As discussed by Reed, Schaub, and Lahijanian (2024), the small differences across these three shields indicate that they are overly conservative, which is an issue to be addressed in future work.

The results of this section indicate that shields can provide safety guarantees for agents deployed in episodes 30 times longer than their training episodes without a significant impact on performance. Small differences are seen when comparing the shielded cases to the elite subset of unshielded cases capable of surviving 90 orbits. Although 90 orbits are far short of a typical mission's lifetime, it provides an initial baseline for deployments exceeding training duration. While one could argue that training with longer episodes might result in policies with better safety aspects, this approach can be computationally expensive due to the remaining need for randomized environments to prevent overfitting. Even if feasible, such policies would still lack the mathematical safety guarantees provided by shields.

Performance of Different Policies with the Same Shield

While the previous section compared the performance of a policy with different shields, this section investigates the performance a shield with different policies.

Figure 8 reports the cumulative reward for unshielded agents and agents with the Q-Optimal shield. Policy π_2 shows the best performance, followed by π_1 in terms of cumulative reward, while the performance of policy π_3 is much lower due to its low survivability rate. On the other hand, the difference in cumulative reward between the shielded cases is smaller; still, π_3 shows the worst performance.

Figure 9 compares the cumulative reward of the cases using the Handmade, One-Step, and Two-Step shields. As shown in Fig. 8, π_3, which was trained without safety con-

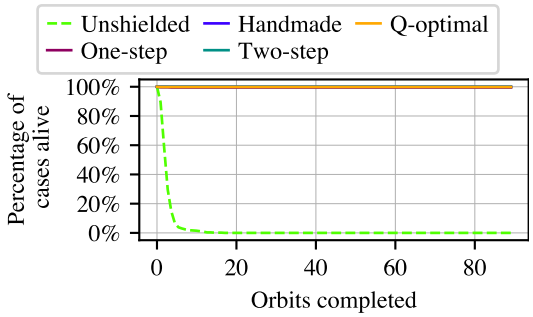

Figure 6: Number of cases alive per orbit for π_3 for city and uniform targets. Shielded cases present a similar survivability close to 100% while the unshielded case has a significant decrease in the first ten orbits.

cerns, achieves a higher cumulative reward than the other two policies for 1,000 and 2,000 targets; however, the other two policies outperform π_3 as the number of targets increases. More targets result in greater target density, especially in the city-based distribution, leading to safety-aware policies to perform better. These safety-aware policies select better moments to charge and desaturate the wheels, leading to higher cumulative rewards. On the other hand, the greedy policy π_3 only charges when forced by the shield, which can occur when flying above regions with high target density, leading to missed imaging opportunities and lower rewards.

Overall, this section's results indicate similar average cumulative rewards for different policies with the same shield. This corroborates the results from the previous section, indicating that shields have little impact on the reward collected by the agent in the long run while providing safety guarantees. Additionally, it indicates that it is possible to train agents in environments without safety concerns and de-

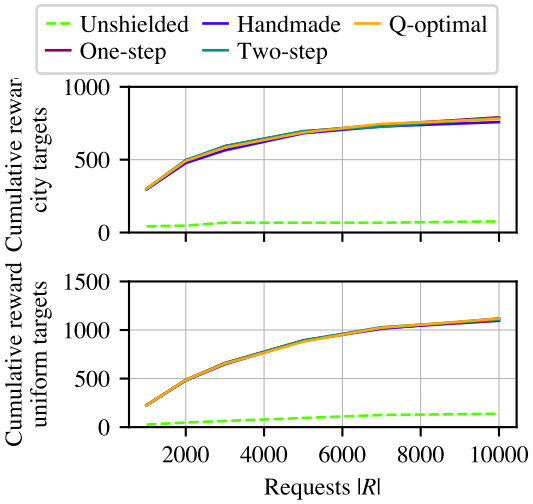

Figure 7: Performance metrics for π_3. City (top) and uniform (bottom) targets. Shielded cases show similar cumulative reward for both target distributions. The unshielded case has low performance due to the high number of failures.

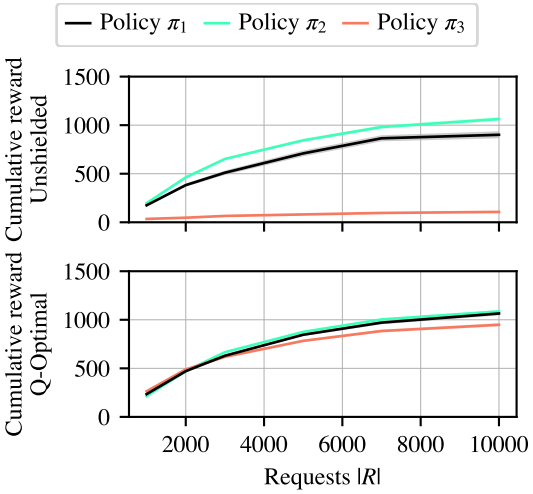

Figure 8: Policies unshielded and with Q-Optimal Shield combining city and uniform targets. Policy π_3 shows similar performance to the other two policies when shielded in environments with low target density.

ploy them with shields to keep them alive. Still, safety-aware agents achieve higher performance as the target density (and the number of targets) increases.

Conclusions

This paper explored the impact of shields on the performance of autonomous agents in the context of AEOS scheduling with tests conducted with different policies and shields across two distinct target distributions. The findings indicate that shields can provide safety guarantees for agents deployed in episodes thirty times longer than their training

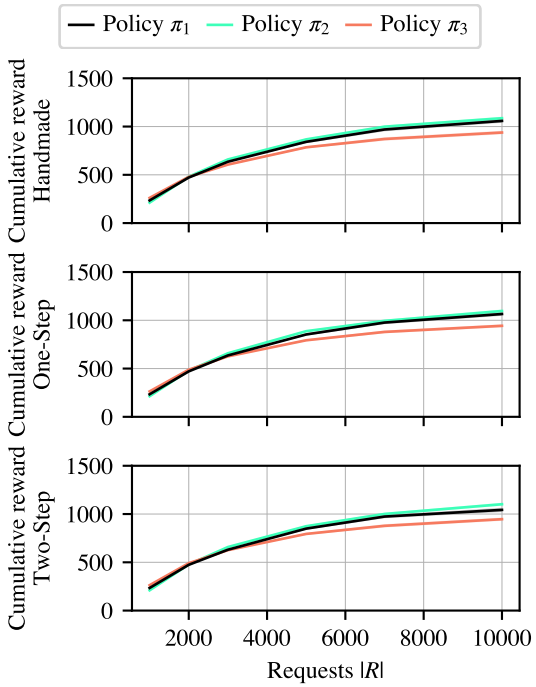

Figure 9: Policies with Handmade, One-Step, and Two-Step shields combining city and uniform targets. Policy π_3 consistently shows lower performance than the other two policies for higher target densities.

episodes without significantly impacting performance. Although shielded cases show higher cumulative rewards than unshielded cases, they present similar performance to the elite group of unshielded cases. This shows that shields have little impact on the reward collected by the agent while providing safety guarantees. Results also indicate that the average cumulative reward of the different shielded policies is similar. Also, it indicates that it is possible to train agents in environments without safety concerns and deploy them with shields to keep them alive. Still, safety-aware agents show better performance in scenarios with higher target density.

Future work will investigate the performance of agents trained with shields in environments with safety concerns and environments with more complex and restrictive safety aspects.

Acknowledgments

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). The authors also would like to acknowledge the support of Robert Reed and Dr. Morteza Lahijanian for their help with the development of the shields and insights.

References

Alshiekh, M.; Bloem, R.; Ehlers, R.; Könighofer, B.; Niekum, S.; and Topcu, U. 2018. Safe Reinforcement Learning via Shielding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Bajenaru, V.; Herrmann, A.; Schaub, H.; Ramirez, J.; and Phillips, S. 2023. Trustworthy Reinforcement Learning For Decentralized Control Of Satellites. In *AAS Guidance and Control Conference*. Breckenridge, CO. Paper No. AAS-23-041.

Bianchessi, N.; Cordeau, J.-F.; Desrosiers, J.; Laporte, G.; and Raymond, V. 2007. A heuristic for the multi-satellite, multi-orbit and multi-user management of Earth observation satellites. *European Journal of Operational Research*, 177(2): 750–762.

Eddy, D.; and Kochenderfer, M. J. 2021. A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations. *Journal of Spacecraft and Rockets*, 58(5): 1416–1429.

Harris, A.; and Schaub, H. 2020. Spacecraft Command and Control with Safety Guarantees using Shielded Deep Reinforcement Learning. In *AIAA SciTech*. Orlando, Florida.

Herrmann, A.; and Schaub, H. 2023a. A comparative analysis of reinforcement learning algorithms for earth-observing satellite scheduling. *Frontiers in Space Technologies*, 4: 1263489.

Herrmann, A.; and Schaub, H. 2023b. Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem. *IEEE Transactions on Aerospace and Electronic Systems*, 1–13.

Herrmann, A.; Stephenson, M. A.; and Schaub, H. 2023. Single-Agent Reinforcement Learning for Scalable Earth-Observing Satellite Constellation Operations. *Journal of Spacecraft and Rockets*, 1–19.

Herrmann, A. P.; and Schaub, H. 2022. Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem. *Journal of Aerospace Information Systems*, 19(1): 70–82.

Kaelbling, L. P.; Littman, M. L.; and Cassandra, A. R. 1998. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1-2): 99–134.

Kenneally, P. W.; Piggott, S.; and Schaub, H. 2020. Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework. *Journal of Aerospace Information Systems*, 17(9): 496–507.

Lemaître, M.; Verfaillie, G.; Jouhaud, F.; Lachiver, J.-M.; and Bataille, N. 2002. Selecting and Scheduling Observations of Agile Satellites. *Aerospace Science and Technology*, 6(5): 367–381.

Liang, E.; Liaw, R.; Moritz, P.; Nishihara, R.; Fox, R.; Goldberg, K.; Gonzalez, J. E.; Jordan, M. I.; and Stoica, I. 2018. RLlib: Abstractions for Distributed Reinforcement Learning.

Luo, M.; Yao, J.; Liaw, R.; Liang, E.; and Stoica, I. 2020. IMPACT: Importance Weighted Asynchronous Architectures with Clipped Target Networks. ArXiv:1912.00167 [cs].

Mantovani, L. Q.; Nagano, Y.; and Schaub, H. 2024. Reinforcement Learning for Satellite Autonomy Under Different Cloud Coverage Probability Observations. In *AAS Astrodynamics Specialist Conference*. Broomfield, CO. Paper No. AAS 24-189.

Nazmy, I.; Harris, A.; Lahijanian, M.; and Schaub, H. 2022. Shielded Deep Reinforcement Learning for Multi-Sensor Spacecraft Imaging. In *American Control Conference*. Atlanta, Georgia.

Reed, R.; Schaub, H.; and Lahijanian, M. 2024. Shielded Deep Reinforcement Learning for Complex Spacecraft Specifications. In *American Control Conference*. Toronto, Canada.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. ArXiv:1707.06347 [cs].

Stephenson, M.; Mantovani, L. Q.; and Schaub, H. 2024. Intent Sharing For Emergent Collaboration In Autonomous Earth Observing Constellations. In *AAS Guidance and Control Conference*. Breckenridge, CO. Paper No. AAS 24-192.

Stephenson, M.; and Schaub, H. 2023. Optimal Target Sequencing In The Agile Earth-Observing Satellite Scheduling Problem Using Learned Dynamics. In *AAS/AIAA Astrodynamics Specialist Conference*. Big Sky, MO. Paper AAS 23–108.

Stephenson, M.; and Schaub, H. 2024a. BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking. In *International Astronautical Congress*. Milan, Italy.

Stephenson, M.; and Schaub, H. 2024b. Reinforcement Learning for Earth-Observing Satellite Autonomy with Event-Based Task Intervals. In *AAS Guidance and Control Conference*. Breckenridge, CO. Paper No. AAS 24-012.

Sutton, R. S.; and Barto, A. 2020. *Reinforcement learning: an introduction*. Adaptive computation and machine learning. Cambridge, Massachusetts London, England: The MIT Press, second edition edition. ISBN 978-0-262-03924-6.

Tang, C.-Y.; Liu, C.-H.; Chen, W.-K.; and You, S. D. 2020. Implementing action mask in proximal policy optimization (PPO) algorithm. *ICT Express*, 6(3): 200–203.

Tangpattanakul, P.; Jozefowiez, N.; and Lopez, P. 2015. A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite. *European Journal of Operational Research*, 245(2): 542–554.

University of Colorado Boulder Research Computing. 2023. Alpine. University of Colorado Boulder.

Wang, X.; Gu, Y.; Wu, G.; and Woodward, J. R. 2021a. Robust scheduling for multiple agile Earth observation satellites under cloud coverage uncertainty. *Computers & Industrial Engineering*, 156: 107292.

Wang, X.; Song, G.; Leus, R.; and Han, C. 2020. Robust Earth Observation Satellite Scheduling With Uncertainty of Cloud Coverage. *IEEE Transactions on Aerospace and Electronic Systems*, 56(3): 2450–2461.

Wang, X.; Wu, G.; Xing, L.; and Pedrycz, W. 2021b. Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions. *IEEE Systems Journal*, 15(3): 3881–3892.

Zhang, C.; Yuan, L.; Xie, M.; Zhang, S.; and Li, J. 2022. Autonomous mission planning of Earth observation satellite based on onboard cloud detection. *Advances in Space Research*, 70(8): 2178–2194.