



Safety and Performance Considerations in Shielded Autonomous Spacecraft Tasking

Lorenzo Quevedo Mantovani*[✉] and Hanspeter Schaub[†][✉]
University of Colorado Boulder, Boulder, Colorado 80303

<https://doi.org/10.2514/1.1011774>

This paper investigates the impact of shields on the performance of deep reinforcement learning (DRL)-based policies in the context of autonomous agile Earth-observing satellites. The scheduling phase during spacecraft operations determines the sequence of actions that the satellite should take to meet the mission requirements while adhering to the system's constraints. The increasing number of satellites, the growing demand for their services, and the need for fast replanning due to unexpected events or additional requests push the need for methods that can provide solutions in real time. DRL has shown promise for onboard scheduling, but deploying neural networks in critical systems, such as satellites, raises reliability concerns. Research has proposed using shielded neural networks (SNNs) to provide safety guarantees with DRL. This work provides a comprehensive analysis of the impact of different approaches to training and deploying SNNs on agent performance. Results show that training with action replacement while assigning penalties for shield intervention leads to higher cumulative reward and lower shield interference during testing. Further, shielded agents performed similarly to the best unshielded agents but with fewer safety violations. Although shields provided safety in 30-times-longer-than-training episodes, policies trained with safety concerns outperform those trained without them.

I. Introduction

EARTH-OBSERVING satellites (EOSs) 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's operation, there is a planning and scheduling phase responsible for providing the sequence of actions that 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-track and across-track, increasing the solution space. The growing demand for satellite imagery has led to oversubscribed systems, making the scheduling problem increasingly complex. Deep reinforcement learning (DRL) methods have shown promise in solving the AEOS scheduling problem [1,2], but deploying neural networks in critical space systems raises safety concerns. Shields have been proposed to provide safety guarantees [3–7], yet a systematic evaluation of how different shielding approaches can be incorporated during training—and how they affect overall performance—is still missing.

The AEOS scheduling problem is shown to be NP-hard, and different optimization techniques are investigated to solve the problem, such as greedy algorithms, dynamic programming, constraint programming, and local search [8]. Heuristics have been employed in the context of multisatellite systems [9], and a local search algorithm has been used to maximize the total reward obtained by an AEOS while minimizing the reward difference of users sharing the platform [10]. More recently, the use of an infeasibility-based graph has been proposed, where the best solution corresponds to the maximum independent set of the graph [11]. Other papers propose using budgeted uncertainty to deal with cloud coverage uncertainty and show the possibility of

solving the problem using column generation and simulated annealing methods [12,13]. Onboard replanning based on cloud coverage information is also explored [14]. Reference [15] provides 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 replanning due to unexpected events, such as cloud coverage or additional requests. This pushes the need for fast schedulers that can provide solutions in real time and in a closed-loop format. For example, increased onboard autonomy in the Earth Observing One satellite enabled more efficient use of spacecraft resources and improved mission return by detecting events of interest (e.g., volcanic eruptions) and adapting the observation plan accordingly [16]. Similarly, the IPEX CubeSat demonstrated onboard autonomy using machine learning algorithms for data processing and the CASPER planner [17,18]. Both missions employed layered safety checks to ensure safe operation during autonomous execution. More recently, the CogniSAT-6 spacecraft (launched in 2024) leverages dynamic targeting [19] to incorporate lookahead information and adapt its schedule onboard [20]. The use of reinforcement learning combined with deep neural networks (DNNs), known as DRL, has shown the ability to effectively solve the scheduling problem under complex dynamics and constraints while presenting high adaptability [2,5,21]. The fast evaluation of the DNNs after training shows the potential of DRL for onboard autonomy. Figure 1 illustrates high-level flight modes that can be tasked in AEOSs when running DRL-based policies in a closed-loop format.

For example, Monte Carlo tree search with DNNs is tested to solve the scheduling problem for EOSs and AEOSs, performing similarly to a genetic algorithm in [2,22]. An in-depth comparison of different DRL algorithms for EOS scheduling, such as advantage actor-critic, deep Q -network, and proximal policy optimization (PPO), is provided in [7]. Performance comparison of DRL with PPO and mixed-integer linear programming (MILP) under different target distributions is also investigated, showing promising results [23]. DRL is also shown to be effective in multisatellite deployment when communication is used to create collaboration among agents [24,25]. The uncertainty of clouds above targets is also accounted for in the AEOS scheduling problem, and the performance of agents with different observation capabilities is investigated [26].

However, using such policies obtained from DRL for autonomy with minimal human intervention in critical systems, such as satellites, raises concerns about their reliability. To mitigate this problem, it is possible to assign penalties during training when the agent

Received 16 September 2025; accepted for publication 16 June 2026; published online 17 August 2026. Copyright © 2026 by Lorenzo Quevedo Mantovani. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission. All requests for copying and permission to reprint should be submitted to CCC at www.copyright.com; employ the eISSN 2327-3097 to initiate your request. See also AIAA Rights and Permissions <https://aiaa.org/publications/publish-with-aiaa/rights-and-permissions/>.

*Ph.D. Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive; lorenzo.quevedomantovani@colorado.edu. Student Member AIAA (Corresponding Author).

[†]Distinguished Professor and Department Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive. Fellow AIAA.

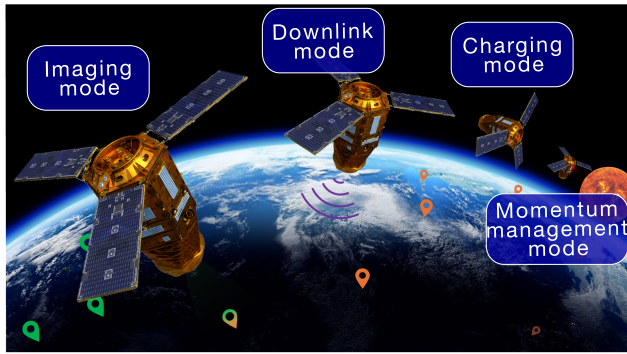


Fig. 1 Illustration of high-level flight modes that can be tasked in AEOSs running DRL-based policies for onboard closed-loop decision-making.

reaches a terminal state. Despite leading to a decrease in safety violations, these soft constraints require proper tuning of the penalty value, which can be difficult, and do not provide guarantees on safety [27]. To overcome this problem, recent research proposes using shielded neural networks (SNNs) to provide hard guarantees on safety while minimizing interferences [28]. Shields can be synthesized from linear temporal logic (LTL) [29] and formulated for single-agent [28] and multi-agent [30] systems. Different methods to incorporate safety guarantees, such as those provided by shields, during the training and deployment phases are investigated. They can be separated into three categories [27]: action replacement, action projection, and action masking.

Action replacement replaces the agent's action with a safe action if the original selected action is deemed unsafe (actions whose probability of leading to a terminal state exceeds a given threshold). Action projection projects the agent's action into a safe action using methods such as model predictive control (MPC), control barrier functions (CBFs), or optimization techniques while trying to minimize the difference between the original and safe actions. This method is mostly applicable to continuous actions, where a distance metric is more easily available. Action masking modifies the policy distribution over actions by changing the probabilities associated with each action, allowing the agent to sample only from the set of safe actions, with the set of safe actions being accounted for in the policy update [31]. Action masking can be particularly helpful in environments with large action spaces, focusing exploration on the subset of valid actions [32]. These three different methods are also combined with different learning tuples, where different information is used to update the policy during training, leading to different effects.

Reference [31] reports increased performance and convergence speed when action masking was implemented in a discrete snake game environment with relatively small state and action spaces. A similar result is seen in [32], which compares the use of action masking to assigning a small penalty when the agent selects an invalid action (but without replacement) and notices that the former scaled better with the number of invalid actions. Notably, the authors indicate that agents trained with masking but deployed without masking selected fewer invalid actions than when assigning penalties during training. Reference [33] also reports improved training performance when using a mask provided by a shield compared with an unshielded baseline in two different environments.

References [28,34] report improved learning performance when using shields during training with action replacement, both when assigning a penalty or not for being overruled by the shield. Using centralized and decentralized postposed shields during training in multi-agent scenarios is shown to preserve training performance while ensuring safety [30] and being superior to only assigning penalties for safety violations without corrections. Interestingly, in [34], an experiment shows better performance when training without and deploying with a shield. On the other hand, in [27], experiments on an inverted pendulum and a quadrotor stabilization task

show that action replacement combined with small penalties for safety interventions during training leads to the best performance in terms of reward and number of interventions when compared with masking or pure action replacement without penalties. Still, the authors state the need for further research on these different methods in more complex scenarios and how these methods can interact with policy training.

Shields have been applied to the satellite scheduling problem, where the spacecraft must maximize its rewards obtained by scanning nadir [4]. Two safety aspects are considered: battery levels and reaction wheel speeds (because speeds over the limit can drive the system unstable). The authors construct a safety Markov decision process (MDP) from the original partially observable MDP (POMDP), which includes unsafe states (states whose probability of leading to a terminal state exceeds a given threshold) 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. An action replacement approach (postposed shield) is used during training, where the original action (before correction) is used in the policy gradient update, while the reward is assigned based on the safe action passed to the environment. The training results indicate that the shielded agents achieved significantly better performance. When testing the policy with shields, the agents effectively avoid terminal states and return to safe regions of the state space. With a similar approach, Ref. [3] compares shielded training to an unshielded case with different failure penalties, showing that the shielded agents outperformed the unshielded ones during training.

SNNs have also been used in the scheduling of AEOSs with point targets, and, in addition to battery levels and reaction wheel speeds, the satellite's angular velocity is also considered a safety aspect [6]. Although the agent was trained to image a specific location on Earth, the results suggest that the SNN could still keep the agent safe when deployed in different environments, such as the moon, due to the use of canonical coordinates. The performance of DRL trained with a postposed shield and a rule-based policy are compared in a multisatellite system [35]. 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 1000 requests submitted to the satellites.

In a rendezvous and proximity operation context, Ref. [36] investigates the use of discrete control barrier functions as a shield and uses the action projection approach to minimize deviations from the original action. The authors compare training performance when enforcing different constraints and find that some constraints improved training while others hindered it. During training, the safe action returned by the shield was not used in the policy gradient. Their results indicate that the policy trained without the shield outperformed the policy trained with the shield when both were deployed with shields in terms of total fuel consumption. Similarly, Ref. [37] uses action projection based on an optimization process to keep the spacecraft in a passively safe trajectory while minimizing the deviation in the required velocity change. The policy is trained with a collision penalty and tested with and without the shield. In addition to providing safety, the shield allowed for an easy change of safety distance during deployment without retraining the policy.

Reference [38] expands the work on shields for complex autonomous AEOS operations, presenting a framework to generate shields from LTL specifications automatically and three distinct shield formulations. Their results indicate that policies trained without shields in an environment with safety constraints but deployed with shields lead to the best interaction between the policy and shield, as seen in fewer safety violations and lower shield interference. Reference [39] investigates the impact of the shields developed in [38] on the performance of autonomous AEOSs in long-term scenarios, where agents were tested in 30-times-longer-than-training episodes. The results provide insights into the minimal impact observed when shielding a policy with postposed shields (action replacement) during testing that was trained without a shield. Still, the effect of incorporating shields during training is not investigated.

Shields have shown promise in enhancing safety without hurting performance in autonomous systems. Although systematic studies have been conducted on how the different forms of incorporating shields during training impact the resulting policies, these studies are limited to simple environments or games. In the context of space systems, an extensive analysis of the impact of different shielded training and deployment formulations on performance is still missing. The main contribution of this paper is to help bridge this gap by providing a comprehensive analysis of the different methods of incorporating shields during training in AEOSs, their impact on the resulting policies, and the tradeoff between performance and safety provided by shields during testing. Agents are tested in episodes 30 times longer than training to analyze long-term survivability.

The remainder of this paper is organized as follows: Section II includes the problem formulation. Section III describes the different methods of incorporating shields during training. Section IV presents the simulation environment and training and testing configurations. Section V presents the training results, Sec. VI presents the testing results, and Sec. VII concludes the paper.

II. Methodology

A. Markov Decision Process Formalism

Sequential decision-making problems can be formulated as an MDP, which is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma \rangle$, where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $T(s'|s, a)$ is the transition probability function, $R(s, a, s')$ is the reward function, and γ is the discount factor. The agent interacts with the environment by taking actions $a \in \mathcal{A}$ in a given state $s \in \mathcal{S}$, which leads to a new state $s' \in \mathcal{S}$ with a probability $T(s'|s, a)$ and receives a reward r . The goal of the agent is to learn a policy $\pi: \mathcal{S} \rightarrow \mathcal{A}$ that maximizes the expected cumulative discounted reward [40]

$$\begin{aligned} V^\pi(s) &= \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^k r_k \right] \\ &= \mathbb{E} \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1}) \middle| a_t \sim \pi(s_t), s_{t+1} \sim T(s_t, a_t) \right], \end{aligned} \quad \forall s \in \mathcal{S} \quad (1)$$

where $V^\pi(s)$ is the value function of state s when following policy π .

Because different actions can have different durations—such as imaging targets closer to or farther from the spacecraft—the problem formulation is extended to a semi-MDP (sMDP), where the agent can take actions that have variable durations. Reference [41] shows superior performance when using an sMDP compared with an MDP in the context of AEOS scheduling, achieving competitive performance compared with an MILP solution. In sMDPs, the discounting factor γ becomes a factor of the elapsed time δt_k in a given action a_k instead of the number of steps, such that

$$\begin{aligned} V^\pi(s) &= \mathbb{E}[\gamma^{\Delta t_0} r_0 + \gamma^{\Delta t_0 + \Delta t_1} r_1 + \gamma^{\Delta t_0 + \Delta t_1 + \Delta t_2} r_2 + \dots] \\ &= \mathbb{E} \left[\sum_{k=0}^{\infty} \gamma^{\sum_{j=0}^k \Delta t_j} r_k \right], \quad \forall s \in \mathcal{S} \end{aligned} \quad (2)$$

The problem can be further extended to a partially observable sMDP (POsMDP) when the agent does not have full observability of the state space, which is defined by a tuple $\langle \mathcal{S}, \mathcal{A}, T, R, \gamma, \mathcal{O}, Z \rangle$ [42], where $\mathcal{O}$ is the observation space and $Z(o|s', a)$ is the observation function. The agent receives observations $o \in \mathcal{O}$ instead of the full state s . As the agent only has access to the observation instead of the state, the policy can be redefined as $\pi: \mathcal{O} \rightarrow \mathcal{A}$.

B. Problem Definition

The AEOS scheduling problem is formulated such that the agent can select from different flight modes, such as charge mode, downlink mode, momentum management mode, and imaging mode, as proposed in [3,43]. In the case of imaging targets, the agent can specify which target to image from a set of upcoming targets. The problem is defined as a POsMDP as follows:

1) **State Space $\mathcal{S}$:** It contains information about the spacecraft states, targets, and internal variables of the simulator. $\mathcal{S}$ consists of continuous and discrete states. The agent reaches a terminal state if the battery level reaches zero or if any of the reaction wheels exceed their maximum allowable speed.

2) **Action Space $\mathcal{A}$:** It contains $N + 3$ 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 N upcoming targets ($\mathcal{A}_{\text{image}} = \{a_{\text{im},j} | j \in [1, N]\}$) such that $\mathcal{A} = \{a_{\text{charge}}\} \cup \{a_{\text{downlink}}\} \cup \{a_{\text{desat}}\} \cup \mathcal{A}_{\text{image}}$.

3) **Transition Probability Function $T(s'|s, a)$:** Transitions are deterministic and generated by a generative model G provided by a high-fidelity simulator that propagates the spacecraft dynamics ($s' = G(s, a)$).

4) **Reward Function $R(s, a, s')$:** A target request τ_i is defined by its priority ρ_i and position $\mathbf{r}_i$ such that $\tau_i = (\rho_i, \mathbf{r}_i)$. Initially, all targets are in an unfulfilled list of requests $\mathcal{U}$; a target is moved to a fulfilled list $\mathcal{F}$ when it is successfully imaged. The agent receives a reward based on the target's priority, ρ_i , such that

$$R(s, a, s') = \begin{cases} \rho_i & \text{if } \tau_i \in \mathcal{U} \text{ and } a \in \mathcal{A}_{\text{image}} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

5) **Observation Space $\mathcal{O}$:** It is a subset of the dimensions in $\mathcal{S}$ and contains information that the agent can observe and that is useful for the decision-making problem. It includes data on the next N upcoming targets in $\mathcal{U}$, while internal simulator variables and targets beyond this set at a given time step remain unobserved. The observation space is based on [41], which conducted an ablation study to determine the most relevant variables for the AEOS scheduling problem. Time is not used in the observation space because infinite episodes (truncated at the time limit) are used instead of finite episodes (terminated). In addition to the target-specific information, the observation space includes a measure of the request density provided by Eq. (4), which indicates the target availability. As discussed in [41], the request density provides information about gaps in upcoming requests without the need for a large N , helping to plan resource management actions. The parameters were set as $K = 20$ and $\Delta t = 5$ min.

$$\left\{ \sum_{\rho_i \in \mathcal{P}_k} r_i \middle| k = 1, \dots, K \right\}, \quad \mathcal{P}_k = \{\rho_i \in \mathcal{U} | t + (k-1)\Delta t \leq t_i^o \leq t + k\Delta t\} \quad (4)$$

6) **Observation Function $Z(o|s', a)$:** The observation function is deterministic because 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 $\mathcal{U}$, ordered by their imaging window, making the system scalable to any number of targets in the environment. From the agent's perspective, targets are arriving dynamically. Reference [41] investigates the agent's performance with different N in the observation and action spaces, showing $N = 32$ to lead to good performance, and compares it to an MILP solution. For $N = 32$, the observation space has a size of 300 and the action space has a size of 35.

Despite rewards being awarded for imaging targets, they could also be assigned based on other criteria, such as downlinked information

(resembling mission operations as in [44]), a combination of imaging and downlink rewards [45], or other mission objectives.

III. Shielded Neural Networks

Although agents can learn to avoid unsafe states during training, there are no guarantees that the agent will remain in safe states during deployment [7]. In high-stakes applications, such as autonomous spacecraft operations, ensuring that the agent does not take actions that can lead to terminal states is crucial. Therefore, it is necessary to incorporate safety guarantees during deployment. Shields can be combined with the agent's policy to guarantee safety while minimizing interference with the agent's actions [28]. However, there are different ways to incorporate shields during training and deployment, which can lead to distinct performance levels [27].

Although different works have explored these different methods, simplified scenarios were used for evaluation, or environments significantly different from the AEOS scheduling problem were considered. Therefore, a few of the most promising methods were selected to be further investigated:

1) Action replacement: The agent's action a is replaced with a safe action a^ϕ if the shield detects that the agent's action would lead to an unsafe state. This method is implemented in [6,35,38]. This case uses the tuple $(o, a, o', R(s, a^\phi, s'))$ for policy updates, meaning that the policy gradient is unaware that the action was replaced by the shield—in line with the naive learning tuple presented in [27].

2) Action replacement with interference penalties: Similar to action replacement, but the agent receives an interference penalty r_{penalty} when the shield replaces its action, such that the tuple $(o, a, o', R(s, a^\phi, s') + r_{\text{penalty}})$ is used for policy updates. The penalty is used to encourage the agent to avoid unsafe actions, which can decrease safety violations.

3) Action masking: The agent's action is masked such that only safe actions are available to the agent. This method is implemented in [31,32,46]. In this case, the tuple $(o, a^\phi, o', R(s, a^\phi, s'))$ is used for policy updates. This approach is similar to the safe action case presented in Refs. [27,46].

Figure 2 illustrates the differences between action replacement and action masking.

A. Action Masking

Logits are the raw output of a DNN, which are the unnormalized log probabilities of each action. Logits are then passed to a softmax function to obtain the action probabilities, which are used to sample actions for the policy—or the highest probability is selected for deterministic policies. In action masking, the logits are modified to ensure that invalid actions have zero probability of being sampled.

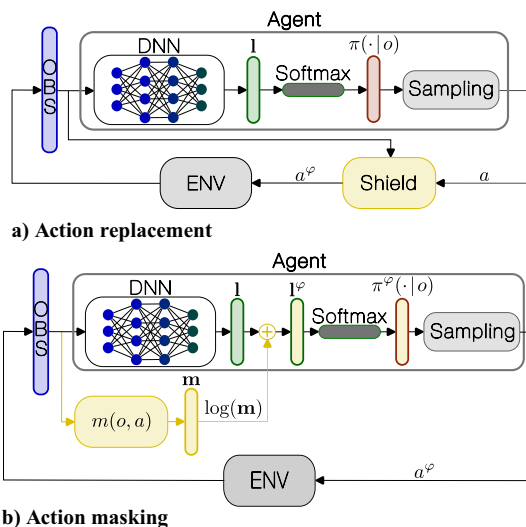


Fig. 2 Illustration of the differences between action replacement and action masking. The critic networks are not shown.

This can be done by modifying the logits directly or by modifying the action probabilities after the softmax function [46]; this work uses the former approach.

Action masking is implemented using a logit-level masking technique, where the log of the mask vector $\mathbf{m}$ of binary values—0 for invalid actions and 1 for valid actions—is added to the logits [32,46]. The vector $\mathbf{m}$ is obtained from an observation-dependent mask function $m(o, a)$ defined as

$$m(o, a) = \begin{cases} 1 & \text{if } a \in \mathcal{A}_{\text{safe}}(o) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

which is then added to the original logits $\mathbf{l}$, leading to the masked logits $\mathbf{l}^\phi$ such that

$$\mathbf{l}^\phi = \mathbf{l} + \log(\mathbf{m}) \quad (6)$$

where the log function is applied elementwise to the mask vector $\mathbf{m}$, leading to negative infinity values for invalid actions. The masked logits are then passed to the softmax function to obtain the masked action probabilities

$$\pi^\phi(a|o) = \frac{\exp(l^\phi(a|o))}{\sum_{a' \in \mathcal{A}} \exp(l^\phi(a'|o))} \quad (7)$$

leading to the masked policy π^ϕ , which has a zero probability of sampling the invalid (unsafe) actions.

Although the selected safe action is being passed to the policy update in the case of action masking, different policy gradient methods can be used to update the policy. The policy gradient can use the safe action a^ϕ combined with the original policy π —naive invalid action masking in [32] or composite objective invalid action masking (CO-PIAM) in [46]—or the masked policy π^ϕ —on-policy invalid action masking (On-PIAM) in [46].

Reference [32] shows that the gradient obtained using On-PIAM is a valid policy gradient. Further, Ref. [46] extends the analysis to demonstrate that On-PIAM directly optimizes the masked policy, with the downside of requiring the masking function $m(o, a)$ even during deployment. As such, the On-PIAM method is used in this work because deployment would occur with a shield to guarantee safety.

B. Shields

The shields' framework utilized in this paper is based on [38], which uses an SNN to provide safety guarantees. When shields are implemented with an action replacement method, the agent selects an action based on its policy, which is then verified by the shield. If the shield detects that the action would lead to an unsafe state, it replaces the agent's action with a safe action most likely to take the agent to a safe region. If action masking is used, the shield is used as the masking function [Eq. (5)].

In this paper, four different shields are considered. Three shields are from the work of [38], where a lower-dimensional safety MDP is abstracted from the original POMDP. The safety MDP state space $\bar{\mathcal{S}}$ is a finite set of states obtained by partitioning safety-relevant states (attitude rate norm, reaction wheel speeds, and battery fraction) from the original state space $\mathcal{S}$, keeping it trackable while preserving the Markov property. The safety MDP transition probabilities, $\bar{T}(s, a, s')$, are obtained through simulations, representing the probability of the agent transitioning to state $s' \in \bar{\mathcal{S}}$ given the current state $s \in \bar{\mathcal{S}}$ and selected action. The fourth shield is hand-made based on expert knowledge of the problem. The shields consider battery charge level and reaction wheel speeds to be safety aspects in all cases. The shields are described as follows:

1) One-Step (1S): Allows actions that have a probability $1 - p$ of leading to a safe state. Let $\bar{\mathcal{S}}_t$ be the set of terminal states (where the battery level is zero or any reaction wheels exceed their maximum allowable speed), $\bar{\mathcal{S}}_u$ be the set of unsafe states, and $\bar{\mathcal{S}}_s$ be the set of

safe states such that $\bar{S}_s = \bar{S} \setminus \bar{S}_u$ and $\bar{S}_u = \bar{S}_t$. Then, the set of safe actions at a state $s \in \bar{S}$ is defined as $\{a \in \mathcal{A} \mid \sum_{s' \in \bar{S}_u} \tilde{T}(s, a, s') < p\}$.

However, there is no guarantee that a safe action is available in the next state s' . If the agent reaches a state with no safe actions available, the action with the highest probability of leading to a safe state is selected.

2) Two-Step (2S): Allows actions that have a probability $1 - p$ of leading to a safe state in the next step and that have a safe action available in the second step. This shield is obtained by initially defining a set of unsafe states $\bar{S}_u$ (initialized as $\bar{S}_u = \bar{S}_t$) and recursively adding states to it when no safe action is available. Similar to 1S, the shield returns the action with the highest probability of leading to a safe state if the agent reaches an unsafe state.

3) Q-Optimal (Q): Constructed with dynamic programming, it obtains a policy π_S^* that minimizes the probability of reaching a terminal state. It allows only actions that have a probability $1 - p$ of leading to a safe state ($\bar{S}_u = \bar{S}_t$ and $\bar{S}_s = \bar{S} \setminus \bar{S}_u$); if none is available, it uses π_S^* to obtain the action more likely to lead to a safe state.

4) Handmade (H): Is based on expert knowledge of the problem, having heuristic rules to enforce safety.

Shields 1S, 2S, and Q also possess an additional safety layer, where the agent is forced to enter momentum management mode if any of the wheels' angular speed fraction exceeds 0.88 or into charge mode if the battery fraction falls below 0.12. If both conditions are met, charge mode takes precedence. The three shields use $p = 0.05$.

The H shield is based on the works of [7,23] and overrides the agent's selection to take the charge action if it is not in eclipse and the battery charge is below a certain threshold. The battery threshold depends on a minimum energy threshold, $E_{\min}$, and the time until the next eclipse. Therefore, the H shield obtains the eclipse duration, e_d , and estimates the energy consumption during the eclipse, E_e , in that period with

$$E_e = e_d P_d \quad (8)$$

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

$$E_t = E_e + E_{\min} - t_e P_c \quad (9)$$

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

IV. Training and Testing Environments

A. Simulation Environment

The simulation was implemented in BSK-RL,[‡] a Python package focused on environments for spacecraft planning and tasking [47]. 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 [48]. It is written in C and C++ and has a Python interface, which makes it easy to interact with 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—a modified Rodrigues

parameter (MRP) steering law ([49], pp. 387–518)—which then computes the reaction wheel torques, with the spacecraft's attitude being propagated by the high-fidelity simulator. Three reaction wheels are considered, positioned orthogonally to each other and aligned with the spacecraft principal axes. During the maneuver, the simulator checks 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, the data are added to the buffer, and the target is moved to the fulfilled list.

Similar to imaging actions, the downlink action is not 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 the 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 tasked with a charge action, the satellite maneuvers to position its solar panels to maximize energy generation. Still, charging is not guaranteed to be successful, as the satellite is required not to be in eclipse. Despite the charge action, the satellite passively charges 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 baseline power consumption to represent essential subsystems, 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 1 shows some spacecraft parameters, such as mass, and subsystem information, such as storage space. Values not mentioned here are set as the standard values for the modules in BSK-RL.

The simulation uses an event-based tasking method, where, after the agent takes an action, the simulator propagates the simulation until certain conditions are met [41]. For imaging actions, the satellite simulator propagates the dynamics until the target is successfully imaged, the opportunity to image it is missed, or a determined maximum time step is reached. Charge, downlink, and momentum management actions are propagated until the determined time step is reached. The maximum time step for imaging and downlink actions is 300 s; for other actions, the maximum time step is 60 s. As safety constraints, the satellite has to keep the battery charge level above zero, and the reaction wheels have to remain below a maximum angular speed threshold. The episode terminates

Table 1 Spacecraft parameters

Parameter	Value
Altitude	800 km
Inclination	45 deg
Mass	330 kg
Battery capacity	160 W · h
Base power consumption	20 W
Data storage capacity	50 MB
Relative angle limit for imaging	28 deg
Relative angular rate limit for imaging	0.01 rad/s
Minimum target image elevation	58 deg
Reaction wheels' maximum torque	0.4 N · m
Reaction wheels' maximum speed	1500 RPM
MRP steering gains (K_1, K_3)	0.25, 3.0
Servo gain	30
Maximum rate command	5 deg/s
Initial battery charge fraction	[0.4, 1.0]
Initial storage space fraction	[0.0, 1.0]
Initial reaction wheels' speed fraction	[−0.5, 0.5]
Number of requests in $\mathcal{U}$	[100, 3000]
Requests' priority (ρ_i)	[0, 1]

[‡]Data available online at https://avslab.github.io/bsk_rl/.

[§]Data available online at <https://avslab.github.io/basilisk/>.

if any of the conditions are not fulfilled. An example of the code used for training and testing is available on the BSK-RL website.[†]

To mimic more realistic request patterns, the simulation uses a city-based target distribution. The distribution is based on the world's most populated cities,** similar to [50].

B. Training Environment

A training environment was set up to train the agent and obtain a policy using DRL. The PPO algorithm [51] from RLlib [52] (version 2.35.0) was used. The training was performed in the University of Colorado Research Computing Alpine high-performance computing resource [53] using 32 AMD Epyc-7713 cores with 100 GB of RAM. Each policy was trained for up to 20 M steps or 72 h of wall-clock time.

Two fully connected layers of 2048 nodes each were used (with different networks for the actor and critic), with a learning rate of $3 \cdot 10^{-5}$, a training batch size of 3000 with 10 epochs, and a discount factor of 0.997. The gradient clipping was set to 0.5, PPO clipping was set to 0.2, and the generalized advantage estimation (GAE) was set to 0.95. Other parameters were set as the standard values in RLlib. The hyperparameters were selected based on [41], which shows that the following combination leads to good performance compared with an MILP solution.

Training episodes were limited to three orbits to allow for more frequent resets of the environment, leading to more randomization of the initial conditions and a more diverse training set. The episodes were truncated after three orbits, so the value function was used to estimate the value of the next state.

C. Trained Policies

Different policies were trained with the different methods discussed in Sec. III combined with the different shields presented in Sec. III.B. Additionally, three benchmark policies are provided. The original policy π_1 is trained without shields or penalties for reaching terminal states. Policy π_2 is identical to π_1 with the addition of a failure penalty of -10 (identified as the best penalty when deploying with the H shield in [41]). Lastly, π_∞ is trained with a very large battery and no reaction wheel speed limits, so it does not need to learn the safety aspects of the problem. Therefore, π_∞ is extremely greedy, taking only imaging and downlink actions. The cases with action replacement with penalties used shields H and Q because H tends to have the least interference and Q the highest, allowing for a better comparison of the effect of the penalty. Table 2 summarizes the trained policies. Each policy was trained three times with different seeds to reduce the bias toward a lucky or unlucky training run.

D. Testing Environment

The testing environment is similar to the training environment, with two main differences. First, the episode duration is set to 90 orbits (approximately 6.3 days) instead of three. Longer episodes allow for more realistic testing of the policies in long-term cases and better evaluation of the tradeoff between safety and performance provided by the shields. The test episode duration was selected based on [39] and represents a tradeoff between computational cost and episode length sufficient to evaluate the impact of shields on survivability. Because fewer than 7% of the baseline cases remained alive at the end of the episode, the 90-orbit duration allows this effect to be assessed while keeping the computational cost manageable. Second, requests were replenished during testing to ensure that the agent had a continuous stream of targets to image. The request replenishment is done by adding new targets to the unfulfilled list $\mathcal{U}$ every time a target is imaged, such that the number of requests in $\mathcal{U}$ remains constant over time. This change is relevant in the extended testing episodes, as otherwise the agent would run out of targets to image after a few orbits.

Table 2 Trained policies

Policy	Shield configuration		Penalties	
	Action replacement	Action masking	Interference	Failure
π_1	—	—	0	0
π_2	—	—	0	-10
π_∞	—	—	0	0
π_{H-R}	H	—	0	0
π_{H-M}	—	H	0	0
π_{1S-R}	1S	—	0	0
π_{1S-M}	—	1S	0	0
π_{2S-R}	2S	—	0	0
π_{2S-M}	—	2S	0	0
π_{Q-R}	Q	—	0	0
π_{Q-M}	—	Q	0	0
$\pi_{H-P_{0.01}}$	H	—	-0.01	0
$\pi_{H-P_{0.1}}$	H	—	-0.1	0
π_{H-P_1}	H	—	-1	0
$\pi_{H-P_{10}}$	H	—	-10	0
$\pi_{Q-P_{0.01}}$	Q	—	-0.01	0
$\pi_{Q-P_{0.1}}$	Q	—	-0.1	0
π_{Q-P_1}	Q	—	-1	0
$\pi_{Q-P_{10}}$	Q	—	-10	0

V. Training Results

Figure 3 shows the policy average reward over training as a function of training steps for policies π_1 , π_2 , and π_∞ . Training results are shown as an average of the three runs for each policy (average of the smoothed results). The shaded area represents the standard deviation. The results show that π_∞ achieved the highest average reward, as expected, because safety aspects did not constrain it (obtaining 100% aliveness). Interestingly, π_2 achieved a higher average reward than π_1 , mainly due to its increased survival rate. The bottom plot in Fig. 3 shows the percentage of episodes that completed the three orbits successfully without failures. Despite the failure penalty in case π_2 , it did not learn to avoid unsafe states entirely. Although the failure penalty applied to policy π_2 could be increased, a failure penalty of -10 led to the best performance during testing when shielded, as investigated in [41]. Higher penalties resulted in more conservative and lower-performing policies.

The training results for the shielded cases with action replacement and action masking are shown in Fig. 4. The top plots show the reward, and the bottom plots show the shield interference. For the action replacement cases (where the agent can select an unsafe action, which is then replaced by the shield), shield interference was measured as the percentage of actions replaced by the shield, given the total actions taken by the agent in a given episode. In the action masking cases (where unsafe actions are removed from the

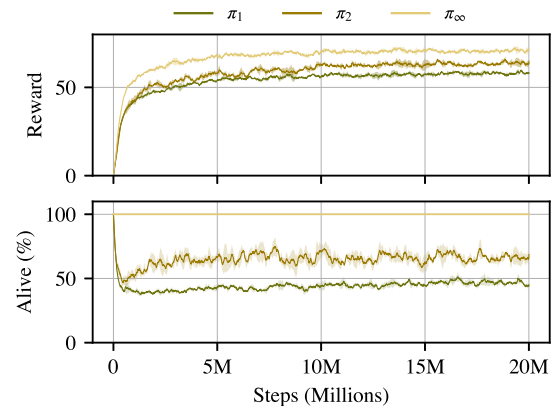


Fig. 3 Average reward during training for policies π_1 , π_2 , and π_∞ without shields.

[†]Data available online at https://avslab.github.io/bsk_rl/examples/training_without_shield.html.

**Obtained from <https://simplemaps.com/data/world-cities> v1.76.

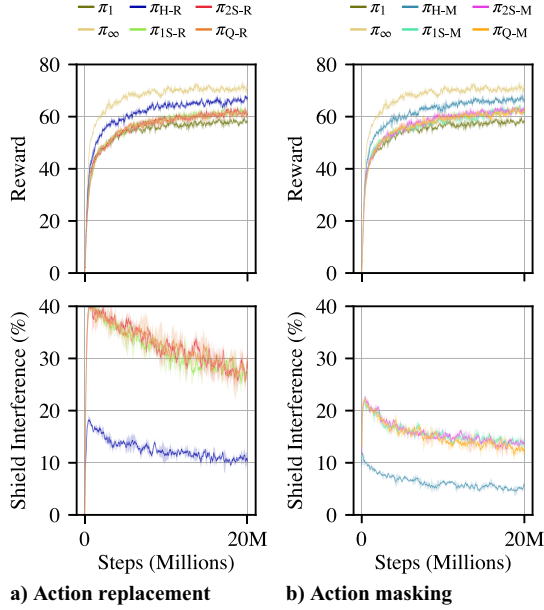


Fig. 4 Training results for policies trained with action replacement and action masking; π_1 and π_∞ are shown for reference.

policy's action set, preventing the agent from selecting them), shield interference was measured as the percentage of steps where not all actions were available to the agent, meaning that the agent could not select an action that would lead to an unsafe state. The different shield interference metrics are necessary because the two shielding mechanisms operate differently; nevertheless, both quantify the frequency with which the shield interferes with the agent's action choices.

The results in Fig. 4 indicate that all cases trained with shields achieved higher performance than π_1 . Interestingly, cases trained with action replacement showed a decrease in shield interference over training, which indicates that agents learned the safety aspects of the problem, even without explicitly using a^φ in the policy update. A hypothesis for this effect is the opportunity cost of having an action overridden by the shield. With city targets, there are opportunities to charge and desaturate the wheels in regions with few requests. If, instead, the agent is forced to take one of these actions over a highly dense region of requests, it would lead to a missed opportunity to acquire targets. The shield interference in Fig. 4 also matches the expected results. The H shield had the lowest interference, being the least conservative, whereas the 1S, 2S, and Q shields had higher interference. On the other hand, the action masking cases showed smaller interference than the action replacement cases, also decreasing over time. The cases trained with action masking are expected to show lower interference from shields, as a^φ was used in the policy gradient, which incentivized the agent to learn those actions.

The comparison between the different penalties for action replacement with the H and Q shields is shown in Fig. 5. Although the cases with a small penalty of -0.01 showed similar shield interference to those without penalties, the higher penalties led to decreased shield interference from the beginning of training. Although higher penalties led to a decrease in shield interference, they also hindered learning. This effect was more evident with the Q shield, which tended to have higher shield interference, with the $\pi_{Q-P_{10}}$ policy failing to learn in two training runs. Policies $\pi_{H-P_{0.1}}$ and π_{H-P_1} were among the best-performing policies overall.

VI. Testing Results

Each policy was tested in 90-orbit-long episodes with target replenishment, as aforementioned in Sec. IV.D. Each policy was tested with 20 different seeds in environments with 100, 500, 1000, 2000, and 3000 requests in the unfulfilled list $\mathcal{U}$ when combined

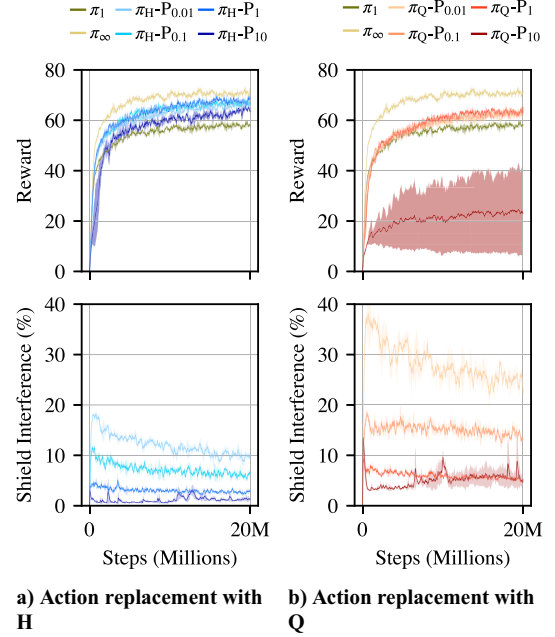


Fig. 5 Training results for policies trained with action replacement with different shield interference penalties; π_1 and π_∞ are shown for reference.

with a specific shield, resulting in 100 test cases for each unique combination of trained policy and shield. Policies π_1 , π_2 , and π_∞ were tested with all shield configurations to investigate the effect of the different shield formulations. Policies trained with action replacement with interference penalties were tested with the H and Q shields to investigate the effect of less and more restrictive shields on the results. In the remaining cases, policies were tested with the shields with which they were trained. Table 3 summarizes the testing cases with shields.

When policies were tested with shields, the action replacement approach was used, even with those trained with action masking, for two reasons. First, it is more likely that action replacement shields will be used in practice, where the shield returns the action with the highest probability of leading to a safe state. Second, when actions were masked out during training, one action was usually available to

Table 3 Testing case configurations

Policy	Shield used during testing				
	None	H	1S	2S	Q
π_1	✓	✓	✓	✓	✓
π_2	✓	✓	✓	✓	✓
π_∞	✓	✓	✓	✓	✓
π_{H-R}	✓	✓	---	---	---
π_{H-M}	✓	✓	---	---	---
π_{1S-R}	✓	---	✓	---	---
π_{1S-M}	✓	---	✓	---	---
π_{2S-R}	✓	---	---	✓	---
π_{2S-M}	✓	---	---	✓	---
π_{Q-R}	✓	---	---	---	✓
π_{Q-M}	✓	---	---	---	✓
$\pi_{H-P_{0.01}}$	✓	✓	---	---	✓
$\pi_{H-P_{0.1}}$	✓	✓	---	---	✓
π_{H-P_1}	✓	✓	---	---	✓
$\pi_{H-P_{10}}$	✓	✓	---	---	✓
$\pi_{Q-P_{0.01}}$	✓	✓	---	---	✓
$\pi_{Q-P_{0.1}}$	✓	✓	---	---	✓
π_{Q-P_1}	✓	✓	---	---	✓
$\pi_{Q-P_{10}}$	✓	✓	---	---	✓

the agent, leading to a case similar to action replacement (except for the difference in the policy update during training). This hypothesis was tested and confirmed by selecting one training run of π_{Q-R} with Q shield to be tested with both action replacement and action masking, showing identical results in terms of rewards and number of orbits alive across all test cases.

A. Policies Trained Without Shields

Policies π_1 , π_2 , and π_∞ were initially tested without any shields. Figure 6 shows the percentage of cases alive during testing as a function of the number of orbits completed in the episode across all test cases. As expected, π_2 had higher survivability (less than 10% of cases alive after 69.6 orbits) than π_1 (less than 10% of cases alive after 7.19 orbits) and π_∞ (less than 10% of cases alive after 5.1 orbits). Nevertheless, π_2 only had 6.33% of cases alive after 90 orbits, showing the necessity of shields to ensure safety during long-term operations. When analyzing the average number of completed orbits as a function of the number of requests, π_2 showed an increase as the number of targets increased, ranging from 7.01 orbits with 100 requests to 30.02 orbits with 3000 requests. The other two policies did not show a similar trend, with π_1 averaging 4.94 orbits and π_∞ averaging 3.33 orbits across all target counts.

When shielding policy π_∞ , all cases completed the 90-orbit episodes without failures. Figure 7 shows the reward, reward rate, and shield interference for π_∞ with the different shields as a function of the number of requests in the unfulfilled list $\mathcal{U}$. No significant differences are seen in terms of the obtained reward or as a function of the number of targets available to the agent. Although the unshielded policy showed a higher reward rate (reward divided by

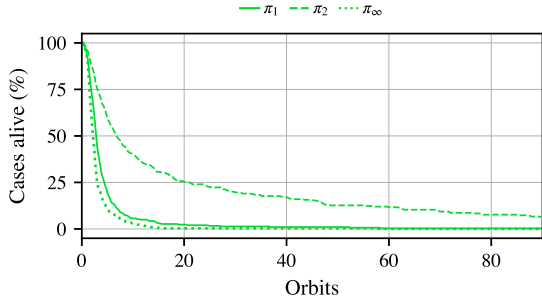


Fig. 6 Percentage of cases alive for policies π_1 , π_2 , and π_∞ tested without shields.

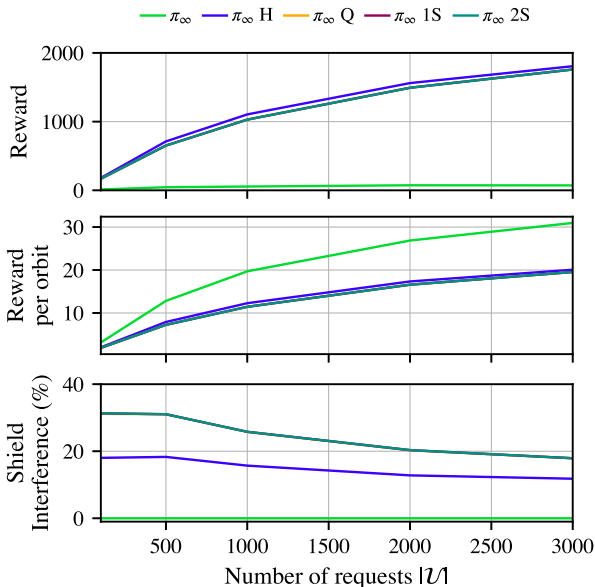


Fig. 7 Policy π_∞ tested unshielded and shielded as a function of the number of requests in $\mathcal{U}$.

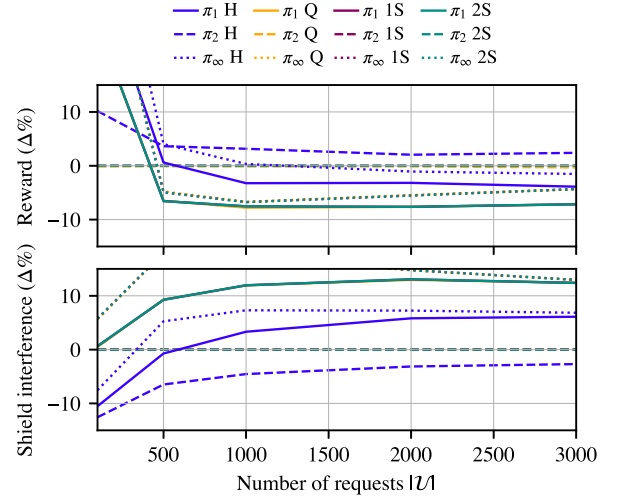


Fig. 8 Policies π_1 , π_2 , and π_∞ tested with different shields; π_2 2S used as reference.

the number of orbits alive), the total reward was low due to the low number of cases alive at the end of the episodes. Counterintuitively, shield interference tended to decrease with an increase in the number of requests. This behavior is linked to the agent taking longer actions in scenarios with fewer targets and having fewer opportunities to retask to charging or desaturation actions. On the other hand, the number of actions tends to increase in scenarios with more requests, with a smaller increase in the need to charge or desaturate the wheels, leading to lower shield interference.

A seed-by-seed comparison between π_1 , π_2 , and π_∞ with the different shields is shown in Fig. 8, with policy π_2 shielded with 2S as a reference. Each policy had three runs per seed (from the three distinct training runs), which were averaged to obtain the reward difference per seed. The average of 20 seed averages was then calculated for each number of targets. Overall, small differences are seen when comparing the different shields for the same policy, with indistinguishable results for cases with 1S and 2S. On the other hand, the differences between the policies are more evident. Shielded π_2 tended to show higher average rewards than shielded π_1 and significantly lower shield interference. The shielded π_∞ obtained comparable or better rewards than π_1 but higher shield interference, as expected, because it was not trained to avoid unsafe states. This result highlights the importance of properly training the agent with safety considerations—as done in π_2 —instead of relying on the shield to ensure safety—as in π_∞ —as it leads to less interference and higher performance.

B. Policies Trained with Shields

Section VI.A highlights the importance of training the agent with safety considerations when no shields are used during training. This section explores the performance of policies trained with shields using action replacement, action masking, and action replacement with penalties for shield interference. Initially, the policies trained with action replacement and action masking are compared, followed by those trained with action replacement with penalties.

1. Action Replacement vs Action Masking

To take a deeper look into the performance differences, Fig. 9 shows the average difference in reward obtained by the policies in a seed-by-seed comparison. The case π_{2S-M} 2S is selected as the reference. The results indicate that the action masking cases tended to outperform or show comparable rewards to the action replacement cases across the different target densities, with an average increase in rewards across all numbers of targets of +1.0% for H, +0.39% for 1S, +0.3% for 2S, and +1.8% for Q. Notably, differences tended to decrease with an increase in the number of requests. The results do not show a clear trend in terms of shield interference. Additionally, cases tested with the H shield tended to

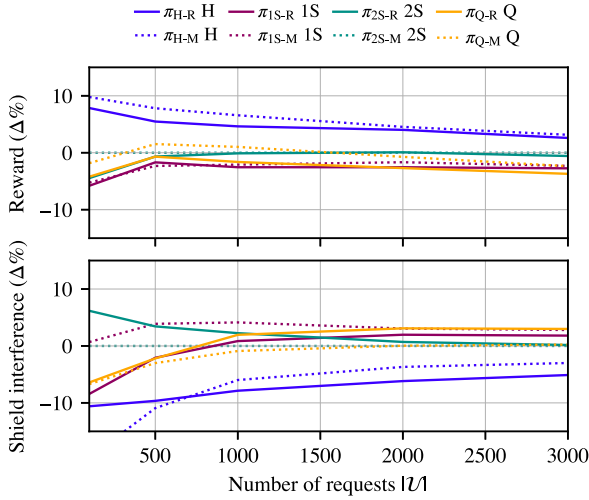


Fig. 9 Policies trained with action replacement and action masking tested with shields; π_{2S-M} 2S used as reference.

show the highest reward compared with other shields with the same policy because of its lower interference.

To investigate how much the policies trained with action replacement and action masking learned the safety aspects of the problem, they were tested without shields. Figure 10 reports the percentage of cases alive during testing as a function of the number of orbits. Surprisingly, π_{IS-R} and π_{H-R} presented the highest percentage of cases alive, with 10% of cases alive after 28.2 orbits. Still, no clear trend is seen in terms of survival rate, where 10% of cases were alive after 14.1 for π_{2S-M} , 10.5 for π_{2S-R} , 12.29 for π_{Q-M} , and 11.7 for π_{Q-R} . Nevertheless, the results are consistent with what was seen in [46], where On-PIAM masked training requires masking during deployment because the policy cannot avoid unsafe states entirely.

2. Action Replacement with Penalties

Figure 11 shows the percentage of cases alive during testing as a function of the number of orbits for policies trained with action replacement and interference penalties and tested without shields. Unlike pure action replacement and action masking, policies trained with penalties for shield interference better learned the problem's safety aspects. Although policies trained without interference penalties have the incentive to learn the safety aspects of the problem due to the opportunity cost of taking a different action than one that could award points, assigning negative penalties helps steer the agent in the right direction. Additionally, increasing the penalty for interference directly improved survivability. Both cases with 1.0 penalty showed the best performance when unshielded, with π_{H-P_1} obtaining 0.21% higher rewards on average than π_{Q-P_1} .

Figure 11 also shows the average percentage of the 90 orbits for which the agent was alive as a function of the number of requests. Cases π_{H-P_1} and π_{Q-P_1} showed a significant increase in the average number of completed orbits as the number of targets increased. For

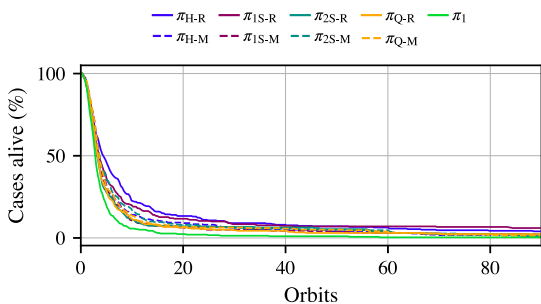


Fig. 10 Percentage of cases alive for policies trained with action replacement and action masking tested without shields across all numbers of requests.

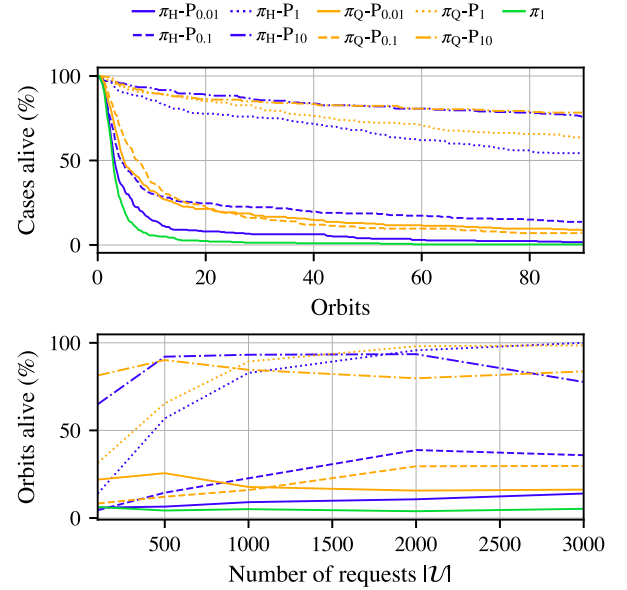


Fig. 11 Aliveness of policies trained with action replacement with interference penalty and tested without shields across all numbers of requests.

example, π_{H-P_1} achieved 100% survivability with 3000 requests, indicating that the agent was able to survive all 90-orbit-long episodes entirely. These results suggest a substantial improvement over the baseline in terms of survivability. The lower survivability observed in cases with fewer targets is likely due to the agent's actions having longer durations (as targets tend to be farther away) and having fewer opportunities to retask to charging or desaturation actions, which increases the likelihood of entering unsafe states. Furthermore, the agent has a stronger incentive to improve its performance in scenarios that offer more rewards, which occurs when more targets are available.

The seed-by-seed reward difference—using $\pi_{H-P_{0.1}}$ H as a reference—and the overall shield interference (averaged across all test cases) are shown in Fig. 12. Case π_2 2S and π_{2S-M} 2S are also included to provide a comparison with the cases shown in Figs. 8 and 9. Cases $\pi_{H-P_{0.01}}$ H and π_{H-P_1} H tended to outperform all other cases, with π_{Q-P_1} Q outperforming cases trained with the Q shield, indicating that penalties around 0.01 and 1.0 are the best

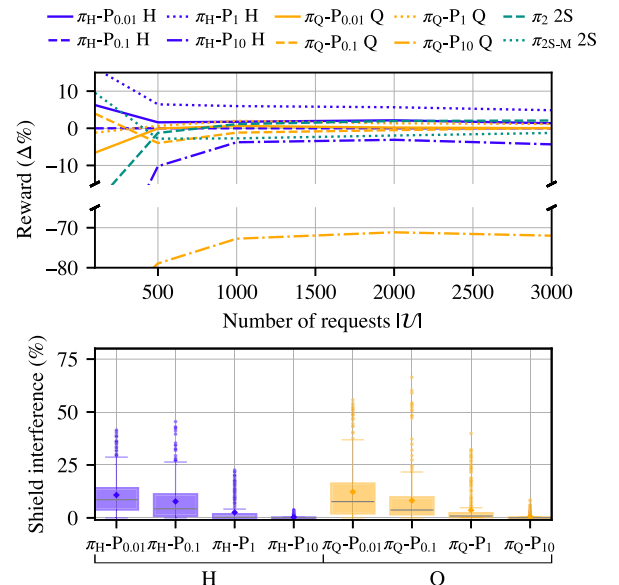


Fig. 12 Policies trained with action replacement with interference penalty and tested with shields; $\pi_{H-P_{0.1}}$ H used as reference.

options for interference penalties. Notably, most cases consistently outperformed π_{2S-M} 2S, which was seen as one of the best-performing cases among action replacement and action masking. Although smaller differences are seen when comparing against π_2 2S, $\pi_{H-P_{0.01}}$ H and π_{H-P_1} Q demonstrated superior performance when tested with fewer targets. The best-performing cases with Q and H also showed the lowest shield interference (average of 2.56% for π_{H-P_1} H and 3.61% for π_{H-P_1} Q), except for $\pi_{H-P_{10}}$ H and $\pi_{Q-P_{10}}$ Q (with -77.7% average reward difference), which ended as overly conservative policies.

When shielded with the respective shields used for training, π_{H-P_1} outperformed π_{Q-P_1} by 4.6% on average. Testing π_{H-P_1} with the Q shield led to an average decrease of 1.68%, whereas testing π_{Q-P_1} with the H shield led to an increase of 1.97% in performance. These results suggest that the more conservative Q shield can be used in policies trained with the H shield without significantly affecting performance while providing enhanced safety guarantees.

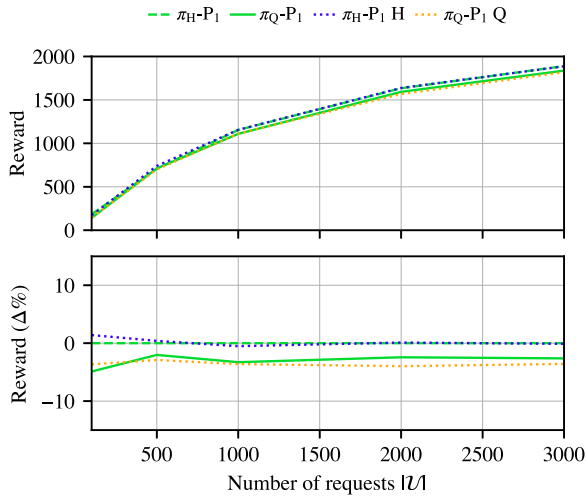


Fig. 13 Policies trained with action replacement with interference penalty and tested with and without shields. Only elite cases are considered; π_{H-P_1} used as reference.

C. Elite Cases

When testing policies without shields, the average performance of the policies was significantly affected due to the large number of failures, which prevented the agents from collecting rewards until the end of the episode. Consequently, shielded policies could have presented higher average rewards because the agents were alive. The elite cases were investigated to better compare the tradeoff between safety and performance provided by shields. Elite cases are defined as cases where the agent was alive at the end of the episode.

Figure 13 reports the average reward and reward difference for the elite cases of policies π_{H-P_1} and π_{Q-P_1} , both unshielded and shielded with their respective shields. Interestingly, only minor differences were observed between cases, indicating the small impact of the shields on the performance of the policy. On the other hand, the unshielded cases had 54.33% and 63.66% of agents alive at the end of the episode for π_{H-P_1} and π_{Q-P_1} , respectively, compared with 100% for the shielded cases.

The seed-by-seed differences in performance per number of targets are shown in the bottom plot in Fig. 13. Although the unshielded case tended to present a slight decrease in performance for the π_{Q-P_1} policy, shielded π_{H-P_1} showed comparable rewards to the elite unshielded cases in all target densities. These results show the minimal impact of shields on the performance of the policies while significantly improving agents' safety.

D. Overall Performance

The overall performance of the different policies was summarized by computing the average reward across all combinations of tested shields (excluding unshielded cases because of the high failure rates). Table 4 summarizes the average rewards per policy and the standard deviation in parentheses (the average standard deviations across seeds for the three training runs of each policy) and per $|U|$ with the four best cases highlighted in bold. Policy π_{H-P_1} showed the best performance in four out of the five request densities. Overall, policies trained with action masking tended to outperform those trained with action replacement (without interference penalties). Policy π_2 also showed strong performance in three out of five cases, outperforming π_1 and π_∞ . Policy π_∞ also obtained higher rewards than π_1 , highlighting the importance of properly training the agent with safety considerations. Notably, no policy failed when tested with the 1S, 2S, or Q shields, with one case failing when

Table 4 Average reward per policy averaged across all shields

Policy	Reward (standard deviation)					Alive, %
	$ U = 100$	$ U = 500$	$ U = 1000$	$ U = 2000$	$ U = 3000$	
π_1	148.97 (25.78)	652.56 (35.55)	1031.20 (66.45)	1478.94 (71.78)	1723.15 (77.39)	99.92
π_2	119.48 (45.22)	692.34 (32.56)	1112.38 (36.98)	1588.70 (46.28)	1848.78 (67.28)	100.00
π_∞	171.62 (10.45)	666.77 (42.35)	1048.19 (58.77)	1509.71 (75.41)	1770.17 (75.83)	100.00
π_{H-R}	173.05 (6.86)	709.95 (36.02)	1107.05 (53.89)	1577.31 (64.40)	1820.37 (77.98)	100.00
π_{H-M}	175.71 (5.13)	726.08 (15.33)	1128.11 (25.28)	1583.97 (38.73)	1829.50 (45.19)	100.00
π_{Q-R}	154.33 (8.77)	670.26 (29.28)	1044.63 (46.26)	1479.20 (73.93)	1712.75 (84.33)	100.00
π_{Q-M}	158.29 (13.86)	685.21 (28.85)	1071.53 (42.75)	1506.10 (58.21)	1734.62 (47.75)	100.00
π_{1S-R}	151.52 (23.82)	664.14 (36.29)	1036.01 (42.38)	1482.46 (67.20)	1731.97 (75.63)	100.00
π_{1S-M}	152.61 (16.82)	659.93 (29.82)	1039.86 (44.41)	1495.11 (54.69)	1738.32 (62.51)	100.00
π_{2S-R}	154.63 (12.04)	670.77 (19.02)	1060.71 (28.97)	1521.16 (43.53)	1767.60 (34.09)	100.00
π_{2S-M}	161.34 (11.11)	675.04 (24.80)	1060.24 (34.61)	1517.49 (52.33)	1776.41 (58.38)	100.00
$\pi_{H-P_{0.01}}$	152.02 (12.34)	681.05 (44.52)	1084.21 (55.69)	1556.88 (64.53)	1804.19 (69.68)	100.00
$\pi_{H-P_{0.1}}$	143.50 (58.87)	681.27 (57.28)	1076.72 (59.37)	1536.94 (57.35)	1785.94 (64.39)	100.00
π_{H-P_1}	165.76 (14.11)	736.70 (15.86)	1155.81 (21.94)	1631.53 (31.61)	1880.64 (36.04)	100.00
$\pi_{H-P_{10}}$	81.52 (81.29)	626.27 (101.28)	1049.73 (70.66)	1501.17 (51.38)	1711.72 (49.77)	100.00
$\pi_{Q-P_{0.01}}$	140.81 (28.17)	704.09 (28.01)	1110.48 (37.62)	1568.67 (56.49)	1816.94 (62.14)	100.00
$\pi_{Q-P_{0.1}}$	158.27 (13.36)	682.20 (32.15)	1088.60 (42.27)	1556.05 (53.99)	1815.84 (60.12)	100.00
π_{Q-P_1}	149.04 (13.28)	703.36 (27.55)	1115.96 (35.54)	1579.11 (48.20)	1830.46 (59.29)	100.00
$\pi_{Q-P_{10}}$	9.33 (5.60)	145.76 (207.89)	299.45 (444.74)	446.02 (672.19)	504.51 (762.83)	100.00

Note: The four highest reward values for each target count are highlighted in bold.

tested with the H shield, underscoring the importance of reliable automated shields, as presented in [38], which can ensure safety and avoid complex edge cases to which handmade shields are prone.

When analyzing the individual training runs and averaging by the different shields used during testing, the first and second training runs of $\pi_{\text{H-P}_1}$ obtained the highest average reward for 500, 1000, 2000, and 3000 targets, whereas the first training run of $\pi_{\text{H-P}_{10}}$ obtained the highest average reward for 100 targets. Table 5 shows the four best-performing policies. These results are consistent with what [27] presents, where policies trained with action replacement and interference penalties tend to outperform the other approaches.

When examining individual training runs and shields used for testing, different runs of policy $\pi_{\text{H-P}_1}$ shielded with H were among the best for all target densities. A combination of a safety-aware policy and a low-interference shield explains this outcome. Table 6 summarizes the four best individual combinations across all testing cases. Still, cases shielded with Q were also among the best-performing cases.

The latter tended to perform better when considering only cases trained with action replacement (without interference penalties) and action masking, as discussed in Sec. VI.B.1. Table 7 summarizes the four best-performing cases across all tested shields. Although these results show less significant improvements than in related work, this outcome is likely due to the small action space used in this work. For example, Refs. [32,46] report larger improvements when using action masking as the size of the action space increases. Further, when multiple safe actions are available, masking could help the agent explore more efficiently than simply enforcing the safest action with action replacement.

E. Discussion

The results shown in Fig. 12 suggest that applying penalties for shield interference during training with action replacement further improved performance. Overall, better-performing policies also tended to have lower shield interference, indicating better learning of the safety aspects of the problem. Although increasing penalties for shield interference leads to decreased interference during training and testing, high penalties can hinder learning and lead to overly conservative policies. Notably, shielding policy $\pi_{\text{H-P}_1}$ with Q resulted in a slight decrease in performance while providing increased safety guarantees.

These results point to an inherent safety-performance tradeoff. Whereas less conservative shields, such as H, may allow unsafe actions in edge cases, stronger shields like 1S, 2S, and Q, as noted in [38], can intervene more than necessary, reducing efficiency and highlighting the potential for developing less conservative automated shields. Nevertheless, shields substantially extended

Table 7 Best-performing policies trained with action replacement and masking averaged across all shields

No.	$ \mathcal{U} = 100$	$ \mathcal{U} = 500$	$ \mathcal{U} = 1000$	$ \mathcal{U} = 2000$	$ \mathcal{U} = 3000$
1	$\pi_{\text{H-M}}$	$\pi_{\text{H-M}}$	$\pi_{\text{H-M}}$	$\pi_{\text{H-M}}$	$\pi_{\text{H-M}}$
2	$\pi_{\text{H-R}}$	$\pi_{\text{H-R}}$	$\pi_{\text{H-R}}$	$\pi_{\text{H-R}}$	$\pi_{\text{H-R}}$
3	$\pi_{2\text{S-M}}$	$\pi_{\text{Q-M}}$	$\pi_{\text{Q-M}}$	$\pi_{2\text{S-R}}$	$\pi_{2\text{S-M}}$
4	$\pi_{\text{Q-M}}$	$\pi_{2\text{S-M}}$	$\pi_{2\text{S-R}}$	$\pi_{2\text{S-M}}$	$\pi_{2\text{S-R}}$

survivability during testing by keeping agents alive in testing episodes 30 times longer than training episodes. Although the average reward tended to decline with shielding, elite cases demonstrated that shields had a minimal impact on performance, as illustrated in Fig. 13. These findings suggest that shields can enhance safety without significantly compromising performance, making them a promising option for real-time decision-making in critical systems.

VII. Conclusions

This work explored the different methods to incorporate shields during training and testing of deep reinforcement learning (DRL)-based policies in the agile Earth-observing satellite (AEOS) problem. The tradeoff between safety and performance provided by shields is also explored. The problem was modeled as a partially observable semi-Markov decision process (POsMDP) and solved using the proximal policy optimization (PPO) algorithm. The results show that training the agent with safety considerations leads to better interactions between the agent and the shield during testing, as seen by higher rewards, lower shield interference, and higher survivability.

Although unshielded training coupled with penalties for failures leads to good policies, the results demonstrate that training with shields using action replacement and penalizing shield interventions leads to superior performance. The resulting agents also demonstrated significantly higher survivability when tested without shields and consistently lower shield interference when tested with shields. Smaller improvements were seen when comparing cases trained with action replacement (without shield interference penalties) and action masking, with the latter showing better performance in most cases. The results also indicate a small safety-performance tradeoff when policies are shielded.

The results indicate the viability of shielded DRL policies for real-time onboard decision-making in critical systems, such as satellites. They can also provide a starting point for designing and training policies for future missions, where safety and performance are critical.

Table 5 Best-performing policies per training run averaged per shield

No.	$ \mathcal{U} = 100$	$ \mathcal{U} = 500$	$ \mathcal{U} = 1000$	$ \mathcal{U} = 2000$	$ \mathcal{U} = 3000$
	Policy run	Policy run	Policy run	Policy run	Policy run
1	$\pi_{\text{H-P}_{10}}$ 1	$\pi_{\text{H-P}_1}$ 1	$\pi_{\text{H-P}_1}$ 1	$\pi_{\text{H-P}_1}$ 2	$\pi_{\text{H-P}_1}$ 2
2	$\pi_{\text{H-P}_{0.1}}$ 3	$\pi_{\text{H-R}}$ 3	$\pi_{\text{H-R}}$ 3	$\pi_{\text{H-R}}$ 3	$\pi_{\text{H-R}}$ 3
3	$\pi_{\text{H-P}_{0.1}}$ 1	$\pi_{\text{H-P}_1}$ 2	$\pi_{\text{H-P}_1}$ 2	$\pi_{\text{H-P}_1}$ 1	π_2 1
4	$\pi_{\text{H-M}}$ 3	$\pi_{\text{H-M}}$ 2	$\pi_{\text{H-P}_1}$ 3	$\pi_{\text{H-P}_1}$ 3	$\pi_{\text{H-P}_1}$ 3

Table 6 Best-performing policies per training run and shield

No.	$ \mathcal{U} = 100$	$ \mathcal{U} = 500$	$ \mathcal{U} = 1000$	$ \mathcal{U} = 2000$	$ \mathcal{U} = 3000$
	Policy run shield	Policy run shield	Policy run shield	Policy run shield	Policy run shield
1	$\pi_{\text{H-P}_{0.1}}$ 3 H	$\pi_{\text{H-P}_1}$ 1 H	$\pi_{\text{H-P}_1}$ 1 Q	$\pi_{\text{H-P}_1}$ 2 H	$\pi_{\text{H-P}_1}$ 2 H
2	$\pi_{\text{H-P}_{10}}$ 1 H	$\pi_{\text{H-R}}$ 3 H	$\pi_{\text{H-R}}$ 3 H	$\pi_{\text{H-P}_1}$ 1 H	π_2 1 H
3	π_{∞} 2 H	$\pi_{\text{H-P}_1}$ 1 Q	$\pi_{\text{H-P}_1}$ 1 H	π_2 1 H	$\pi_{\text{Q-P}_1}$ 1 H
4	$\pi_{\text{H-P}_{10}}$ 1 Q	$\pi_{\text{H-P}_1}$ 2 H	$\pi_{\text{H-P}_1}$ 2 H	$\pi_{\text{H-P}_1}$ 3 H	π_2 3 H

Acknowledgments

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). This work was partially supported by the Air Force Research Lab grant FA9453-22-2-0050. The authors also would like to acknowledge the support of Robert Reed and Morteza Lahijanian for their insights and help in developing and providing the different automated shields. Artificial intelligence tools were used to help improve the grammar and clarity of the manuscript. Final editing and approval were performed by the authors.

References

- [1] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, AAS Paper 19-447, Springfield, VA, 2019.
- [2] Herrmann, A., and Schaub, H., "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, Inst. of Electrical and Electronics Engineers, New York, 2023, pp. 1–13.
<https://doi.org/10.1109/TAES.2023.3251307>
- [3] Harris, A., and Schaub, H., "Deep On-Board Scheduling for Autonomous Attitude Guidance Operations," *AAS Guidance, Navigation and Control Conference*, AAS Paper 02-117, Springfield, VA, 2020.
- [4] Harris, A. T., and Schaub, H., "Spacecraft Command and Control with Safety Guarantees Using Shielded Deep Reinforcement Learning," *AIAA Paper 2020-0386*, 2020.
<https://doi.org/10.2514/6.2020-0386>
- [5] Harris, A., Valade, T., Teil, T., and Schaub, H., "Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning," *Journal of Spacecraft and Rockets*, Vol. 59, No. 2, 2022, pp. 611–626.
<https://doi.org/10.2514/1.A35169>
- [6] Nazmy, I., Harris, A., Lahijanian, M., and Schaub, H., "Shielded Deep Reinforcement Learning for Multi-Sensor Spacecraft Imaging," *2022 American Control Conference (ACC)*, IEEE Publ., Piscataway, NJ, 2022, pp. 1808–1813.
<https://doi.org/10.23919/ACC53348.2022.9867762>
- [7] Herrmann, A., and Schaub, H., "A Comparative Analysis of Reinforcement Learning Algorithms for Earth-Observing Satellite Scheduling," *Frontiers in Space Technologies*, Vol. 4, Nov. 2023, Paper 1263489.
<https://doi.org/10.3389/frspt.2023.1263489>
- [8] Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., and Bataille, N., "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](https://doi.org/10.1016/S1270-9638(02)01173-2)
- [9] Bianchessi, N., Cordeau, J.-F., Desrosiers, J., Laporte, G., and Raymond, V., "A Heuristic for the Multi-Satellite, Multi-Orbit and Multi-User Management of Earth Observation Satellites," *European Journal of Operational Research*, Vol. 177, No. 2, 2007, pp. 750–762.
<https://doi.org/10.1016/j.ejor.2005.12.026>
- [10] Tangpattanakul, P., Jozefowicz, N., and Lopez, P., "A Multi-Objective Local Search Heuristic for Scheduling Earth Observations Taken by an Agile Satellite," *European Journal of Operational Research*, Vol. 245, No. 2, 2015, pp. 542–554.
<https://doi.org/10.1016/j.ejor.2015.03.011>
- [11] Eddy, D., and Kochenderfer, M. J., "A Maximum Independent Set Method for Scheduling Earth-Observing Satellite Constellations," *Journal of Spacecraft and Rockets*, Vol. 58, No. 5, 2021, pp. 1416–1429.
<https://doi.org/10.2514/1.A34931>
- [12] Wang, X., Song, G., Leus, R., and Han, C., "Robust Earth Observation Satellite Scheduling with Uncertainty of Cloud Coverage," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 56, No. 3, 2020, pp. 2450–2461.
<https://doi.org/10.1109/TAES.2019.2947978>
- [13] Wang, X., Gu, Y., Wu, G., and Woodward, J. R., "Robust Scheduling for Multiple Agile Earth Observation Satellites Under Cloud Coverage Uncertainty," *Computers & Industrial Engineering*, Vol. 156, June 2021, Paper 107292.
<https://doi.org/10.1016/j.cie.2021.107292>
- [14] Zhang, C., Yuan, L., Xie, M., Zhang, S., and Li, J., "Autonomous Mission Planning of Earth Observation Satellite Based on Onboard Cloud Detection," *Advances in Space Research*, Vol. 70, No. 8, 2022, pp. 2178–2194.
<https://doi.org/10.1016/j.asr.2022.07.007>
- [15] Wang, X., Wu, G., Xing, L., and Pedrycz, W., "Agile Earth Observation Satellite Scheduling over 20 Years: Formulations, Methods, and Future Directions," *IEEE Systems Journal*, Vol. 15, No. 3, 2021, pp. 3881–3892.
<https://doi.org/10.1109/JSYST.2020.2997050>
- [16] Chien, S., Sherwood, R., Tran, D., Cichy, B., Rabideau, G., Castano, R., Davis, A., Mandl, D., Frye, S., Trout, B., et al., "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, Vol. 2, No. 4, 2005, pp. 196–216.
<https://doi.org/10.2514/1.12923>
- [17] Chien, S., Doubleday, J., Thompson, D. R., Wagstaff, K. L., Bellardo, J., Francis, C., Baumgarten, E., Williams, A., Yee, E., Stanton, E., et al., "Onboard Autonomy on the Intelligent Payload Experiment CubeSat Mission," *Journal of Aerospace Information Systems*, Vol. 14, No. 6, 2017, pp. 307–315.
<https://doi.org/10.2514/1.I010386>
- [18] Chien, S., Knight, R., Stechert, A., Sherwood, R., and Rabideau, G., "Using Iterative Repair to Improve the Responsiveness of Planning and Scheduling," *Proceedings of the 5th International Conference on Artificial Intelligence Planning Systems*, AAAI Press, 2000, pp. 300–307.
- [19] Candela, A., Swope, J., and Chien, S. A., "Dynamic Targeting to Improve Earth Science Missions," *Journal of Aerospace Information Systems*, Vol. 20, No. 11, 2023, pp. 679–689.
<https://doi.org/10.2514/1.I011233>
- [20] Chien, S., Zilberstein, I., Candela, A., Rijlaarsdam, D., Hendrix, T., Dunne, A., Oriol, A., and Puig, M. J., "Flight of Dynamic Targeting on the CogniSAT-6 Spacecraft," *Proceedings of the International Symposium on Artificial Intelligence, Robotics and Automation in Space (iSAIRAS)*, 2024, pp. 186–189.
- [21] Zhao, X., Wang, Z., and Zheng, G., "Two-Phase Neural Combinatorial Optimization with Reinforcement Learning for Agile Satellite Scheduling," *Journal of Aerospace Information Systems*, Vol. 17, No. 7, 2020, pp. 346–357.
<https://doi.org/10.2514/1.I010754>
- [22] Herrmann, A. P., and Schaub, H., "Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem," *Journal of Aerospace Information Systems*, Vol. 19, No. 1, 2022, pp. 70–82.
<https://doi.org/10.2514/1.I010992>
- [23] Stephenson, M., and Schaub, H., "Reinforcement Learning for Earth-Observing Satellite Autonomy with Event-Based Task Intervals," *AAS Guidance and Control Conference*, AAS Paper 24-012, Springfield, VA, 2024.
- [24] Herrmann, A., Stephenson, M. A., and Schaub, H., "Single-Agent Reinforcement Learning for Scalable Earth-Observing Satellite Constellation Operations," *Journal of Spacecraft and Rockets*, Vol. 61, No. 1, 2023, pp. 142–132.
<https://doi.org/10.2514/1.A35736>
- [25] Stephenson, M., Mantovani, L. Q., and Schaub, H., "Intent Sharing for Emergent Collaboration in Autonomous Earth Observing Constellations," *AAS Guidance and Control Conference*, AAS Paper 24-192, Springfield, VA, 2024.
- [26] Mantovani, L. Q., Nagano, Y., and Schaub, H., "Reinforcement Learning for Satellite Autonomy Under Different Cloud Coverage Probability Observations," *AAS Astrodynamics Specialist Conference*, AAS Paper 24-189, Springfield, VA, 2024.
- [27] Krasowski, H., Thumm, J., Müller, M., Schäfer, L., Wang, X., and Althoff, M., "Provably Safe Reinforcement Learning: Conceptual Analysis, Survey, and Benchmarking," *Transactions on Machine Learning Research*, 2023.
- [28] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S., and Topcu, U., "Safe Reinforcement Learning via Shielding," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 32, No. 1, 2018.
<https://doi.org/10.1609/aaai.v32i1.11797>
- [29] Pnueli, A., "The Temporal Logic of Programs," *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, Inst. of Electrical and Electronics Engineers, New York, 1977, pp. 46–57.
<https://doi.org/10.1109/sfcs.1977.32>
- [30] ElSayed-Aly, I., Bharadwaj, S., Amato, C., Ehlers, R., Topcu, U., and Feng, L., "Safe Multi-Agent Reinforcement Learning via Shielding," *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2021, pp. 483–491.
- [31] Tang, C.-Y., Liu, C.-H., Chen, W.-K., and You, S. D., "Implementing Action Mask in Proximal Policy Optimization (PPO) Algorithm," *ICT Express*, Vol. 6, No. 3, 2020, pp. 200–203.
<https://doi.org/10.1016/j.icte.2020.05.003>

- [32] Huang, S., and Ontañón, S., "A Closer Look at Invalid Action Masking in Policy Gradient Algorithms," *The International FLAIRS Conference Proceedings*, Vol. 35, 2022.
<https://doi.org/10.32473/flairs.v35i1.130584>
- [33] Varricchione, G., Alechina, N., Dastani, M., De Giacomo, G., Logan, B., and Perelli, G., "Pure-Past Action Masking," *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38, No. 19, 2024, pp. 21,646–21,655.
<https://doi.org/10.1609/aaai.v38i19.30163>
- [34] Könighofer, B., Lorber, F., Jansen, N., and Bloem, R., "Shield Synthesis for Reinforcement Learning," *Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles*, edited by T. Margaria, and B. Steffen, Vol. 12476, Lecture Notes in Computer Science, Springer International Publ., Cham, 2020, pp. 290–306.
https://doi.org/10.1007/978-3-030-61362-4_16
- [35] Bajenaru, V., Herrmann, A., Schaub, H., Ramirez, J., and Phillips, S., "Trustworthy Reinforcement Learning for Decentralized Control of Satellites," *AAS Guidance and Control Conference*, AAS Paper 23-041, Springfield, VA, 2023.
- [36] van Wijk, D., Dunlap, K., Majji, M., and Hobbs, K., "Safe Spacecraft Inspection via Deep Reinforcement Learning and Discrete Control Barrier Functions," *Journal of Aerospace Information Systems*, Vol. 21, No. 12, 2024, pp. 996–1013.
<https://doi.org/10.2514/1.I011391>
- [37] Stephenson, M., Prats, D. H., and Schaub, H., "Autonomous Satellite Inspection in Low Earth Orbit with Optimization-Based Safety Guarantees," *Proceedings of the 14th International Workshop on Planning and Scheduling for Space (IWPSS)*, edited by C. Artigues, J. Jaubert, C. Pralet, and S. Chien, International Workshop on Planning and Scheduling for Space, Toulouse, France, 2025, pp. 209–218.
- [38] Reed, R., Schaub, H., and Lahijanian, M., "Shielded Deep Reinforcement Learning for Complex Spacecraft Tasking," *2024 American Control Conference (ACC)*, Inst. of Electrical and Electronics Engineers, New York, 2024, pp. 2331–2337.
<https://doi.org/10.23919/ACC60939.2024.10644855>
- [39] Mantovani, L. Q., and Schaub, H., "Performance Evaluation of Shielded Neural Networks for Autonomous Agile Earth Observing Satellites in Long Term Scenarios," *Proceedings of the 14th International Workshop on Planning and Scheduling for Space (IWPSS)*, edited by C. Artigues, J. Jaubert, C. Pralet, and S. Chien, International Workshop on Planning and Scheduling for Space, Toulouse, France, 2025, pp. 112–121.
- [40] Sutton, R. S., and Barto, A., "Reinforcement Learning: An Introduction," *Adaptive Computation and Machine Learning*, 2nd ed., MIT Press, Cambridge, MA, 2020, pp. 47–72.
- [41] Stephenson, M. A., Mantovani, L. Q., and Schaub, H., "Learning Policies for Autonomous Earth-Observing Satellite Scheduling over Semi-Markov Decision Processes," *Journal of Aerospace Information Systems*, Vol. 22, No. 9, 2025, pp. 789–799.
<https://doi.org/10.2514/1.I011649>
- [42] Kaelbling, L. P., Littman, M. L., and Cassandra, A. R., "Planning and Acting in Partially Observable Stochastic Domains," *Artificial Intelligence*, Vol. 101, Nos. 1–2, 1998, pp. 99–134.
[https://doi.org/10.1016/S0004-3702\(98\)00023-X](https://doi.org/10.1016/S0004-3702(98)00023-X)
- [43] Herrmann, A., Vaz Carneiro, J., and Schaub, H., "Reinforcement Learning for the Multi-Satellite Earth-Observing Scheduling Problem," *Proceedings of the 44th Annual American Astronautical Society Guidance, Navigation, and Control Conference*, edited by M. Sandnas, and D. B. Spencer, Springer International Publ., Cham, 2024, pp. 1351–1368.
https://doi.org/10.1007/978-3-031-51928-4_74
- [44] Russino, J. A., Wang, D., Wagner, C., Rabideau, G., Mirza, F., Basich, C., Mauceri, C., Twu, P., Reeves, G., Tan-Wang, G., et al., "Utility-Driven Approach to Onboard Scheduling and Execution for an Autonomous Europa Lander Mission," *Journal of Aerospace Information Systems*, Vol. 22, No. 2, 2025, pp. 73–89.
<https://doi.org/10.2514/1.I011323>
- [45] Huterer Prats, D., and Schaub, H., "Autonomous Space-Based Imaging: Reinforcement Learning Scheduling with Imaging–Downlink Tradeoffs Across Orbital Regimes," *AAS Guidance, Navigation and Control Conference*, AAS Paper 26-052, Springfield, VA, 2026.
- [46] Hou, Y., Liang, X., Zhang, J., Yang, Q., Yang, A., and Wang, N., "Exploring the Use of Invalid Action Masking in Reinforcement Learning: A Comparative Study of On-Policy and Off-Policy Algorithms in Real-Time Strategy Games," *Applied Sciences*, Vol. 13, No. 14, 2023, Paper 8283.
<https://doi.org/10.3390/app13148283>
- [47] Stephenson, M., and Schaub, H., "BSK-RL: Modular, High-Fidelity Reinforcement Learning Environments for Spacecraft Tasking," *International Astronautical Congress*, International Astronautical Federation (IAF), Paris, France, 2024, pp. 1186–1197.
- [48] Kenneally, P. W., Piggott, S., and Schaub, H., "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *Journal of Aerospace Information Systems*, Vol. 17, No. 9, 2020, pp. 496–507.
<https://doi.org/10.2514/1.I010762>
- [49] Schaub, H., and Junkins, J. L., *Analytical Mechanics of Space Systems*, 4th ed., AIAA Education Series, AIAA, Reston, VA, 2018, pp. 387–518.
<https://doi.org/10.2514/4.105210>
- [50] Stephenson, M., and Schaub, H., "Optimal Target Sequencing in the Agile Earth-Observing Satellite Scheduling Problem Using Learned Dynamics," *AAS/AIAA Astrodynamics Specialist Conference*, AAS Paper 23-108, Springfield, VA, 2023.
- [51] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," Aug. 2017, <http://arxiv.org/abs/1707.06347>.
- [52] Liang, E., Liaw, R., Moritz, P., Nishihara, R., Fox, R., Goldberg, K., Gonzalez, J. E., Jordan, M. I., and Stoica, I., "RLlib: Abstractions for Distributed Reinforcement Learning," June 2018.
<https://doi.org/10.48550/arXiv.1712.09381>
- [53] University of Colorado Boulder Research Computing, "Alpine," Univ. of Colorado Boulder, Boulder, CO, 2023.
<https://doi.org/10.25811/K3W6-PK81>

D. Selva
Associate Editor