(Preprint) AAS 24-012

REINFORCEMENT LEARNING FOR EARTH-OBSERVING SATELLITE AUTONOMY WITH EVENT-BASED TASK INTERVALS

Mark Stephenson[*] and Hanspeter Schaub[†]

Advances in target scheduling algorithms are necessary to fully utilize the capabilities of agile Earth-observing satellites. While most current approaches preplan sequences of target collections, on-board autonomous methods that select targets on the fly can adapt to opportunistic science events and rapid changes to target requests, as well as reduce the burden on operators. In this paper, reinforcement learning (RL) is used to train satellites to select upcoming targets in a high-fidelity simulation environment, learning to account for target value and the feasibility of slewing to nearby targets over a range of target densities. In the Markov decision process formulation, imaging actions are tasked for the duration it takes to execute the action as opposed to using fixed decision intervals, yielding a system that more efficiently represents the problem. Novel contributions of this work are the comparison of the performance of RL against a pseudo-optimal mixed-integer program solution in a high-fidelity spacecraft simulation environment and the demonstration of intelligent resource management by the RL agent in a power-constrained case.

INTRODUCTION

On-demand satellite imagery is hugely beneficial for scientific observations, commercial operations, disaster response, or reconnaissance missions. "Agile" satellites are well suited to these tasks as they are capable of performing along-track slews to image targets in front and behind them, as opposed to traditional Earth-observing satellites (EOSs) that can only slew across-track. The objective of the agile Earth-observing satellite scheduling problems (AEOSSPs) is to sequence operations such that the value of images taken by an agile Earth-observing satellite (AEOS) is maximized. While agility can greatly increase the imaging throughput of a satellite, it complicates planning operations because both the order and timing of each target can be modified, leading to a greatly increased solution space for operations sequences [1]. Resource management for the satellite must also be considered when planning tasks.

Satellite observation scheduling problems are commonly represented as directed acyclic graphs, in which vertices represent observation opportunities and edges represent feasible transitions between observations [2, 3]. In this form, the problem can be expressed as a mixed-integer program (MIP) as it is reduced to a discrete routing problem similar to the traveling salesman problem [4]; a variety of high-performance commercial solvers exist to find the optimal solution to MIPs and quantify the suboptimality of intermediate solutions. Peng uses a MIP-based solution as a benchmark but

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

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

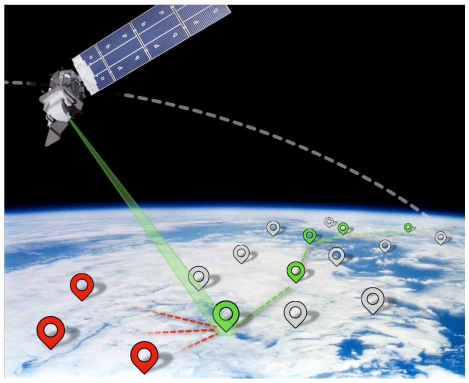

Figure 1: The decision agent selects which observation to schedule next among N multiple upcoming targets.

encounters prohibitively long solution times in more complex, high-resolution cases [5]. A variety of authors use MIP solutions for target allocation problems in constellations, which demonstrate the utility for global optimization problems [6, 7, 8, 9]. Reference [10] utilizes neural networks to construct graphs that accurately account for transition time dynamics and improves the efficiency of the MIP formulation. A popular alternative to MIP solvers is iterative local search (ILS), which leverages the problem structure to iteratively optimize short segments of the solution [11]. Peng develops an ILS algorithm for observation scheduling that can account for time-independent or -dependent target values [5]. Tangpattanakul uses ILS to ensure fair resource allocation between users of a shared satellite [12].

Ultimately, preplanning approaches such as MIP and ILS solvers suffer from two major limitations: 1) high computational requirements and — as a result — slow solution times, especially as the problem size increases; and 2) being open loop, brittleness to a changing, uncertain, or mismodeled environment (in the case of MIP, the linearization of resources guarantees some degree of mismodeling if included). If the plan is unsuccessfully executed or more desirable objectives are added, either a new plan must be computed on the ground and reuploaded to the satellite or — computation capacity permitting — the satellite may locally repair a short horizon of the upcoming plan [11]. Reinforcement learning (RL) is a broadly applicable framework for closed loop autonomy [13]. Nazari demonstrates that RL is effective at solving general waypoint routing problems [14]. Applied directly to spacecraft tasking, Harris [15], Hadj-Salah [16], and Eddy [17] formulate various Markov decision processes (MDPs) for the AEOSSP. References [15] and [16] use simplified probabilistic models for the spacecraft dynamics. In [15] and later [18, 19, 20], Harris and Hermann develop the problem with a full-fidelity simulation and utilize shields to ensure operational safety. Zhao considers target scheduling using two phases of RL, the first selecting observation windows and the second picking observation times within the windows [21]. MDP representations of other spacecraft tasking problems, such as small-body imaging, have also been formulated and solved with RL [22, 23].

The goal of this work is to demonstrate autonomous agents executing learned policies (Figure 1) for the AEOSSP that are competitive with MIP-based approaches when compared in a high-fidelity simulation. This requires formulating the MDP to only propagate actions for the minimal necessary duration to avoid satellite downtime, as opposed to using fixed-duration actions as in other work. Furthermore, cases that the MIP planner cannot handle, such as nonlinear spacecraft power resource constraints, are solved with RL. These RL-based solutions enjoy low onboard computation costs, closed-loop opportunism, and robustness to mismodeling, as opposed to the brittle, high-cost solutions provided by other methods.

PROBLEM STATEMENT

The basic, target-scheduling-only AEOSSP is described and formalized as a partially-observable Markov decision process (POMDP) so that it can be solved with RL. A modified, power-constrained version of the problem is also developed.

Agile Earth-Observing Satellite Scheduling Problem

The objective of the AEOSSP is to maximize the value of imaging requests fulfilled by a satellite equipped with a camera-like sensor. The satellite is agile, meaning that it can not only slew about-track to image targets not directly nadir of the satellite but also slew along-track, allowing complex scheduling of imaging tasks. The satellite is on a fixed low-Earth orbit, meaning that the rotation of the Earth changes the accessibility of targets over time.

Target-Based Request Model In this work, requests are modeled as individual points (as opposed to continuous areas of land); Eddy and Kochenderfer explain how area-based request models can be decomposed into the point-based request paradigm in [24].

Imaging requests $\rho \in R$ consist of a tuple of a target's planet-fixed location and value, $\rho_i = (\boldsymbol{r}_i, r_i)$. All imaging requests start as unfulfilled, $R = U$, and are fulfilled by imaging the target, moving it to the fulfilled set F. Once a target has been fulfilled, it is assumed that there is no value in reimaging it (though an operator could add a new request in the same location, if reimaging is desired). Targets can only be imaged when associated constraints are met, such as view angle or time-of-day restrictions. In this paper, only the view angle constraints are considered, but the methods are general enough to account for others. The intervals for which the constraints are met are called opportunity windows $[\tau^o, \tau^c] = w \in W_i$, which are the set of intervals for which target i can be imaged.

For clarity of notation, when indexing targets and opportunity windows in this work, the subscript i refers to the unfulfilled request with the i^{th} upcoming still-open window, ordered by window closing time. For example, r_2 is the value of the second upcoming target and τ_2^c is the closing time of that opportunity.

Three distributions of targets given in Figure 2 are used in this work: uniform, cities, and Swiss-cheese. "Uniform" distributes targets equally over Earth's surface. "Cities" distributes targets at the locations of a random subset of the 42,000 most populous cities[*]. Finally, "Swiss-cheese" uses a uniform distribution of targets, then removes targets in random circular regions up to radius $r_E/4$. The Swiss-cheese distribution is only used in training, as a means of exposing the satellite to target-free periods. Uniform and Swiss-cheese distributions are considered for up to 10000 globally distributed targets, while cities are considered for up to 3000 targets; these distributions lead to similar maximum local densities.

Imaging Dynamics The satellite has a camera pointing in the body-fixed $\hat{\boldsymbol{c}}$ direction. To image a target, the satellite must point the camera within some $\delta\theta$ and track the target within some $\delta\dot{\theta}$, accounting for the relative motion of the satellite and the Earth. These constraints must be met

[*]City location data from simplemaps.com.

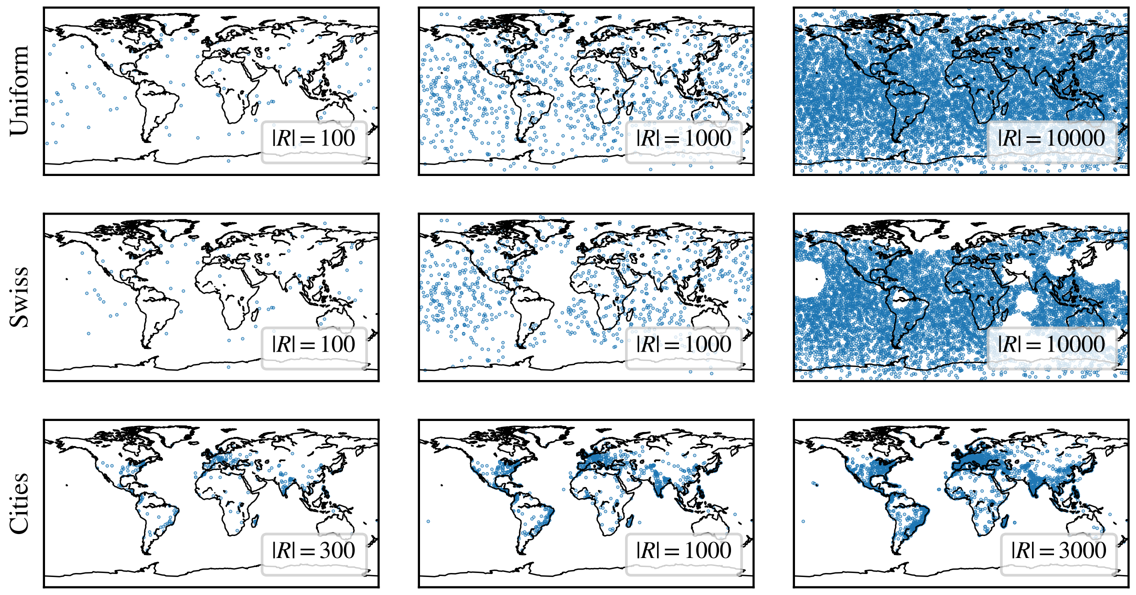

Figure 2: Examples of each target distribution over different densities.

while an opportunity window for the target is open (i.e. other constraints are also satisfied). As a result, transitions between requests are dynamically complex. The satellite has a nonzero angular velocity when tracking a target and must slew to the next target and track it with a different nonzero angular velocity. It may not be dynamically feasible to steer to and settle on some upcoming requests before their opportunity window closes.

Simulation The satellite is modeled using the Basilisk[†] spacecraft simulation framework [25]. Basilisk is a high-fidelity, high-speed modular simulation system, making it a strong tool for this work. The spacecraft is modeled to the component level in a realistic space environment: reaction wheels, power system, and data collection system are all modeled in an environment that includes orbital perturbations and SPICE-based solar system dynamics. In this work, the satellite is designed to be a typical 300 kg small-sat.

The satellite's flight software uses flight-heritage control algorithms. The satellite is guided to targets using a modified Rodrigues parameter (MRP)-based attitude controller and rate servos for reaction wheels [26]. The complete simulation setup is similar to that shown in [20]. Flight software, spacecraft dynamics, and the environment are propagated at 2 Hz.

Markov Decision Process Formalization

The AEOSSP can be represented as a POMDP. Actions correspond to high-level modes in the flight software, such as imaging a certain target or entering a charging mode; this mode-based approach to spacecraft RL was proposed by Harris et al., Eddy et al., and Herrmann and Schaub [15, 17, 20]. The agent is rewarded for imaging targets based on an operator-assigned per-target value. Much of the POMDP — states, observations, and transitions — implicitly emerges from the previously defined simulation of the environment.

[†]http://hanspeterschaub.info/basilisk/

Quantity	Normalization	Description
${}^{\mathcal{E}}\boldsymbol{\omega}_{\mathcal{B}\mathcal{E}}$	0.03 rad/s	Body angular rate
${}^{\mathcal{E}}\hat{\boldsymbol{c}}$	-	Instrument pointing direction
${}^{\mathcal{E}}\boldsymbol{r}_{\mathcal{B}\mathcal{E}}$	r_E	Earth-fixed position, Earth radius-normalized
${}^{\mathcal{E}}\boldsymbol{v}_{\mathcal{B}\mathcal{E}}$	v_{orb}	Earth-fixed velocity, orbital velocity-normalized
t	t_{max}	Time through episode (completion fraction)
$\{r_m \mid m \in 1, \ldots, M\}$	-	Rewards $\in [0,1]$ of next M targets
$\{{}^{\mathcal{E}}\boldsymbol{r}_m \mid m \in 1, \ldots, M\}$	r_E	Positions of next M targets
Equation 1	5.0	Upcoming reward density
z	z_{max}	Charge fraction
$\tau^o_{Ecl}, \tau^c_{Ecl}$	T	Next eclipse transitions, orbital period-normalized

Table 1: Elements in the observation o and their normalization constants. The lower portion of the table is only included in the power-constrained problem.

The POMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, R, \mathcal{O}, Z)$. The simulator provides a deterministic generative model $G(s, a) = s'$. The elements of the POMDP are defined as follows:

- **State Space $\mathcal{S}$:** The space of complete simulator states required to maintain the Markov assumption. This includes directly observable variables such as the spacecraft's dynamic state and upcoming target information, hidden variables including far-off target information and request statuses, and variables required for simulation such as controller integrator states. Practically, only a subset of the state is relevant to the scheduling problem, which is exposed through the observation function.

- **Observation Space $\mathcal{O}$ and Observation Probability Function Z:** The observation space $\mathcal{O}$ consists of a selection of dimensions from the state space and transformations thereof. The selected elements are those presumed to be relevant to decision-making for the problem, based on expert knowledge and experimentation. In this POMDP, the observation consists of the elements given in Table 1. Observation elements are normalized to fall approximately in $[-1, 1]$, which improves the performance of deep reinforcement learning (DRL) algorithms. The number of upcoming unfulfilled target requests in the observation M is tuneable. To encode long-term reward availability, the reward density function

$$\left\{ \sum_{i \in U} \begin{cases} r_i & \text{if } t + (k-1)\Delta t \leq \tau^o_i < t + k\Delta t \\ 0 & \text{else} \end{cases} \mid k = 1, \ldots, K \right\} \tag{1}$$

is added to the observation; it gives the cumulative reward available in each of the next $K = 20$ intervals of $\Delta t = 5$ minutes. The intent of this portion of the state is to reveal gaps in upcoming target availability, which is relevant for long-term planning once resource constraints are introduced. A large value of M would expose the same information, but the potential for dense targets would require it to be very high; the horizon exposed by Equation 1 is not dependent on density and contributes fewer elements to the observation that individual target information would.

Observation elements are taken directly from the state without noise, though other work has shown that noise can be added to the observation without impacting performance as long as the noise was modeled in training and deployment [27]. With the observation expressed as a subset function of state $O(s) \subset s$, the observation probability function is deterministic: $Z(O(s)|s) = 1$.

- **Action Space $\mathcal{A}$:** The satellite has N imaging actions $a_{\text{im},n}$ that task imaging of the target with the n^{th} upcoming imaging opportunity in the set of unfulfilled requests. The availability of an action does not imply feasibility: a target's opportunity window may close before the satellite can slew and settle to image it.

- **Transition Probability Function $T(s'|s,a)$:** Transitions are deterministic and generated by the simulator, resulting in $T(G(s,a)|s,a) = 1$. The simulator propagates for a variable duration at each step, depending on conditions met as the action is taken. In particular, the simulator stops as soon as an action is successful or cannot be successful. When generating the next state, the simulator runs until one of three conditions is met (hence the description of the problem as "event-based"):

  1. **Imaging Successful:** If the target is successfully imaged (moved from unfulfilled U to fulfilled F), the step ends. No more reward can be obtained from continuing to image a fulfilled request, so the satellite should retask immediately.
  2. **Opportunity Window Close:** If the opportunity window for the target being imaged closes before the target is imaged, $t > \tau_i^c$, the step ends. There is no potential for the satellite to image the target until a later orbit, so it is best for the satellite to retask immediately.
  3. **Maximum Step Duration Timeout:** If neither of the above conditions are met before $\Delta t = 5$ minutes has elapsed, the step ends.

- **Reward Function $R(s,a,s')$:** The agent is rewarded for successfully fulfilling target requests. The reward function yields the request value if the request is fulfilled during the step, and zero otherwise:

$$R(s, a_{\text{im},n}, s') = \begin{cases} r_i & \text{if } \rho_n \in U \text{ and } \rho_n \in F' \\ 0 & \text{else} \end{cases} \tag{2}$$

The POMDP is implemented using `bsk-rl`[‡], an open-source package for creating modular RL environments for spacecraft tasking. The package is designed to be easily configurable, customizable, and reproducible. `bsk-rl` uses the standard Gymnasium API for RL environments, making the package compatible with all major RL frameworks [28].

Power-Constrained Problem Variation

The AEOSSP is commonly subject to resource management constraints that need to be maintained by the scheduler. A variant of the previously described problem is proposed in which the satellite has a power subsystem that must be managed. The satellite is equipped with a battery with capacity z_{max} and solar panels with efficiency η and area A with their normal antiparallel to the instrument pointing direction. The satellite has a baseline power draw which is a function of other subsystem states and an additional $\dot{z}_{\text{im}}$ which is added when tasked with imaging a target. Since

[‡]https://github.com/AVSLab/bsk_rl/

the power generation of the solar panel is a function of the attitude over time, it is not possible to accurately account for it with other planning methods without running complete simulations of proposed plans, a prohibitively expensive process.

The POMDP is modified in three places to allow the agent to monitor and respond to the constraints of the power system:

1. As listed in Table 1, the observation is augmented with the current charge fraction z and the eclipse transition times $\tau^{o}_{Ecl}, \tau^{c}_{Ecl}$, giving the agent sufficient information to make decisions about power management. The charge level indicates if the satellite is at risk of dying, and the eclipse times show when the satellite can charge.

2. The satellite is given an additional charging action, a_{charge}, which points the solar array towards the sun for 1 minute. Charging is determined by the underlying simulation, so if the action is taken during eclipse no power will be generated.

3. If the satellite enters a discharged state where $z = 0$, the episode ends. This is a hard constraint that the agent must learn to avoid. A reward $r_{\text{fail}} \leq 0$ is rewarded for entering the failure states.

Other aspects of the base POMDP were formulated to help learning in the power-constrained variation. The upcoming reward density observation allows the satellite to identify upcoming low-reward periods where it can charge with minimal loss of reward. The maximum step duration limit helps prevent failures during learning by preventing discharge when selecting a far-off target by allowing the agent to periodically reconsider its choice of action (ultimately allowing for the creation of a shield), which is further discussed in a later section.

METHODS

Reinforcement learning, a class of methods for finding policies that approximate the solution to a MDP, is reviewed and, proximal policy optimization (PPO), a widely-used DRL algorithm, is described. Two problem-specific considerations for RL are discussed: the impacts of variable-interval decisions and the use of shielding for safety in the power-constrained problem. Finally, a pseudo-optimal solution to the target-sequencing-only problem from [10] is briefly described. This method is used as a benchmark for comparison in the experiments.

Reinforcement Learning

The objective of RL is to find the policy $a = \pi(s)$, a function that defines the action an agent should take in response to an environment state, that maximizes the sum of future rewards for a MDP [13]. The agent does not have access to the underlying model of the MDP; instead, it must learn through interaction with the environment. DRL algorithms use neural networks to represent the policy, taking advantage of generalization properties inherent to deep learning and allowing them to be applied to problems with continuous state and action spaces.

Proximal Policy Optimization A popular and performant DRL algorithm is PPO [29]. PPO is a policy gradient method, meaning that it directly optimizes the policy π_θ by iteratively updating the policy parameters θ as the agent gains experience in the environment. To prevent unlearning, PPO limits the policy update to a small region around the current policy using the loss function

$$L\left(s,a,\theta_k,\theta\right)=\min\left(\frac{\pi_\theta(a\mid s)}{\pi_{\theta_k}(a\mid s)}A^{\pi_{\theta_k}}(s,a),\;\; g\left(\epsilon,A^{\pi_{\theta_k}}(s,a)\right)\right) \tag{3}$$

where the clipping function g is

$$g(\epsilon,A)=\begin{cases}(1+\epsilon)A & A\geq 0\\ (1-\epsilon)A & A<0\end{cases} \tag{4}$$

A short summary is given in Algorithm 1[§]. Reference [20] shows that PPO performs well in a variety of similar spacecraft tasking problems.

Algorithm 1 Proximal Policy Iteration

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

Throughout this work, all policies are trained using RLlib's implementation of PPO [30] for 5M steps, approximately 20k to 25k episodes or 11 to 14 years of simulated mission time. Training takes 10 to 15 hours of wall-clock time on 32 Intel Milan CPU cores. The separate policy and value networks are each 2×2048 node, the discount factor γ is set to 0.0003, the training batch size is 10,000, and the minibatch size is 250. Other hyperparameters use Stable Baseline 3 defaults for PPO (notably *not* the RLlib defaults, which use VTrace instead of GAE and do not normalize advantages).

Considerations for Variable-Interval, Infinite-Horizon MDPs The use of event-based decision intervals in the POMDP introduces challenges not typically encountered in RL, in which each step usually has an equal time-cost. The most theoretically sound way of handling this is to discount rewards by elapsed time $\gamma^{\Delta t}$ rather than elapsed steps $\gamma^{\Delta n}$. In practice, this is not an effective strategy in a reward-dense environment: The continuous accumulation of rewards means that with γ close to 1, the impact of the time discount between immediate next actions is minimal even if they are relatively different in duration, while with a small γ, future actions are quickly disregarded in favor of a greedy strategy.

Instead of duration-based discounts, a hard time limit of three-orbit-long episodes is used in training and the episode completion fraction is included in the state. This allows the policy to correlate higher completion fractions with less remaining potential and thus lower value. As a result, the policy will learn to prefer shorter actions because they have less of an impact on future value.

When using the policy in a deployment-type setting (i.e. one that is effectively infinite horizon), the completion fraction element of the observation is set to a constant value instead of being incremented with time. In this work, it is set to zero to avoid potentially using reckless or greedy behavior learned at the end of episodes. In practice, this is not the case: any completion fraction in

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

$[0, 1]$ results in the same action to be selected by the policy with high probability. This result makes sense given the use of separate policy and value networks in PPO[¶]: the value network makes use of the completion fraction element which in turn allows for a better policy to be learned, but there is no reason to suspect the policy itself would strongly depend on the completion fraction.

The variable-interval nature of the problem also leads to practical challenges. Typical implementations of PPO run copies of the environment in parallel and step the environments simultaneously once all environments have completed the previous step. In this problem, the duration of a step can vary from a few seconds to several minutes of simulation time; this leads to many works idling while waiting for the longest-interval steps to complete. Also, in the power-constrained problem, agent deaths lead the environments to reset at different steps, which is also a time-intensive task. To address these issues, the environments are stepped asynchronously using RLlib's asynchronous proximal policy optimization (APPO) variation.

Shielded Reinforcement Learning Shielded RL was introduced by Alshiekh et al. as a way of making safety guarantees about the safety of an RL agent [31]. In this work, the post-processing approach is taken, in which the shield evaluates the policy's chosen action against the observation; the shield either accepts the action as safe or overrides the potentially unsafe action with a safe action. Harris et al. apply this to the spacecraft tasking to meet the safety requirements of space systems [18].

Safety is only a consideration for the power-constrained variation of the problem. A relatively simple hand-crafted shield is employed. The shield selects the charging action if the battery level is below

$$z_{\text{minsafe}} = \begin{cases} z_{\text{floor}} - (\tau^c_{Ecl} - \tau^o_{Ecl})\dot{z}_{\text{draw}} - \tau^o_{Ecl}\dot{z}_{\text{gain}} & \text{if not in eclipse and } > z_{\text{floor}} \\ z_{\text{floor}} - \tau^c_{Ecl}\dot{z}_{\text{draw}} & \text{if in eclipse} \\ z_{\text{floor}} & \text{else} \end{cases} \tag{5}$$

which maintains the battery level at least at z_{floor}, with a higher requirement as eclipse nears. Worst-case estimates of $\dot{z}_{\text{gain}} = 100\%/\text{orbit}$ and $\dot{z}_{\text{draw}} = -100\%/\text{orbit}$ are used, and z_{floor} is set to 25%.

A novel strategy using the shield is employed in this work to increase the speed of training. While training with a shield usually leads to suboptimal policies that exclusively rely on the shield for safety operations (as demonstrated in later results), it eliminates the need to learn to stay alive. Practically, this can greatly improve the time-efficiency of learning as frequent resets due to agent deaths are completely avoided; especially in complex environments, resetting can be time-expensive. To partially leverage the benefits of shielded training without limiting the quality of the final policy, agents in this paper are trained with a parameterized version of the shield that is ramped down from fully engaged to completely disengaged over the first 1M steps of training. In particular, $z_{\text{floor}} : 20\% \rightarrow 0\%$, $\dot{z}_{\text{gain}} : 100\%/\text{orbit} \rightarrow 150\%/\text{orbit}$, and $\dot{z}_{\text{draw}} : -100\%/\text{orbit} \rightarrow 0\%/\text{orbit}$. As shown in Figure 3, this avoids a slow, failure-dense initial period while achieving a strong final unsheilded performance.

Pseudo-Optimal Benchmark

The hybrid analytical/simulation-based approach to the target sequencing problem described in [10] is taken as a pseudo-optimal benchmark. In short, opportunity windows for each target are

[¶]While not an inherent feature of PPO, the use of separate policy and value networks is known to generally improve the performance of the algorithm.

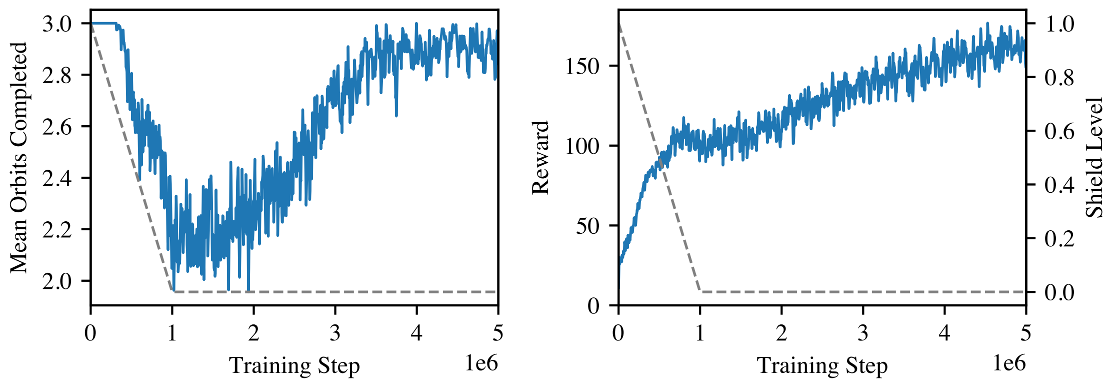

Figure 3: Orbits completed before failure and rewards for each episode during training in the zero-penalty power-limited case. The shield is ramped down (dashed line) over the first 1M steps of training.

time-discretized into individual opportunity times. Supervised learning is used to train a neural network to predict slew durations between targets; these predictions estimate if the transition from one discrete target-time to another is feasible. Feasible transitions are represented as edges on a directed acyclic graph of slews. A MIP is formulated to find the maximum reward path across the graph, which can be solved by commercial software to find a pseudo-optimal target sequence.

This MIP solution to the scheduling problem is a good benchmark. It is optimal up to the accuracy of the neural network and the time discretization, both of which are shown to be minimally impactful even on larger problems. In particular, the use of the neural network for slew duration predictions makes this method more performant than heuristic-based approaches because it suffers neither from idle time due to a conservative heuristic nor from missed targets due to an overly aggressive heuristic; as a result, solutions from this method can be directly and successfully executed in the full-fidelity simulation environment.

However, this work is motivated by limitations inherent to that class of planners. MIP planners and ILS planners, which are more efficient than MIP but lack optimality quantification, are both preplanning approaches. They both plan a sequence over an upcoming horizon and tend to be computationally expensive, meaning that they are brittle to a changing environment and typically limited to ground-based planning. For denser target distributions and longer horizons considered in this work, the MIP planner may take over an hour to find the solution. Furthermore, accurately accounting for nonlinear resources such as power generation and depletion is not possible with these methods; at best, an inaccurate heuristic can be used.

TARGET SEQUENCING-ONLY PROBLEM

The generation of agents for the target sequencing-only problem (i.e. sans power management) is considered first. The performance can be directly compared to the pseudo-optimal results given by the MIP solver which only solves the power-free problem.

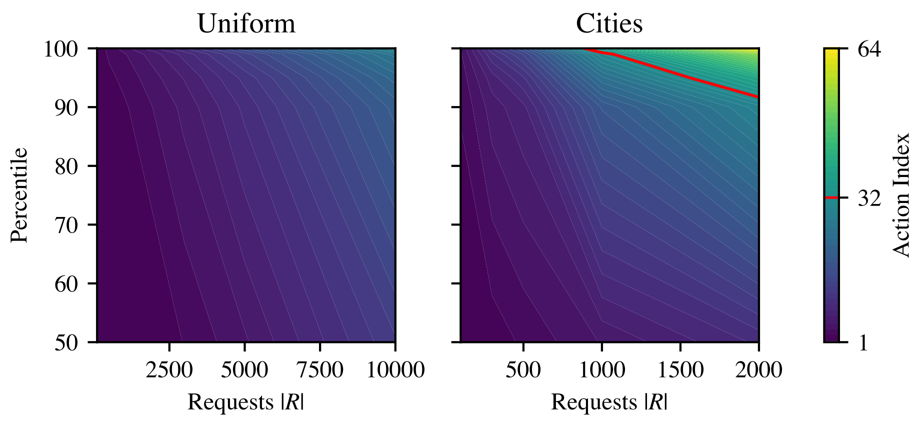

Figure 4: Distribution of upcoming target index tasked in MIP solutions.

Parameter Variation

Three factors are of particular interest: the state space target lookahead M, the action space target lookahead N, and the target distribution used in training. The impact of these parameters on the performance of the policy across the two test distributions (uniform and cities) and a range of densities is desired. Ideally, some training parameters will produce a policy that generalizes well across all test conditions.

Action Space Target Lookahead The number of targets N considered in the actions space must be sufficiently large to allow the policy to select the best response. If N is too small relative to the density of targets, the satellite may never succeed at taking any images since action a_N would not have sufficient time to settle on the target before going out of range. However, large N adds complexity to the network and to training. Since MIP solutions are known, the index of actions selected in the pseudo-optimal solution can be used as a reference, shown in Figure 4. Due to the very tight clustering of cities in some areas, the cities distribution of targets requires higher action indices in the worst cases. All of the uniform actions and most of the city actions are captured by $N = 32$, which is primarily used in Figure 5 and beyond. This heuristic is not entirely rigorous as there could be a policy equally performant to the MIP solution that selects further-spaced, higher-valued targets, thus requiring a larger action space; however, Figure 5 shows that $N = 64$ does not perform better than $N = 32$.

State Space Target Lookahead The number of targets M in the state space must be at least equal to those in the action space N, as the policy intuitively should be provided with information on any target it may select. It may be beneficial to have $M > N$ so that the policy may consider more future targets when selecting what to image; this is akin to how ILS-based solvers optimize over short intervals of the sequencing task, though the network only estimates the first target of the interval as opposed to explicitly optimizing an entire sequence. Three values of M are considered: $N = 32$, $2N = 64$, and $3N = 96$.

Figure 5 shows that $M > N$ does not improve performance over $M = N$, with variations in M producing similar results. Discussed in more depth later, it is shown that the policies tend to favor higher value targets over more targets, unlike the MIP solutions.

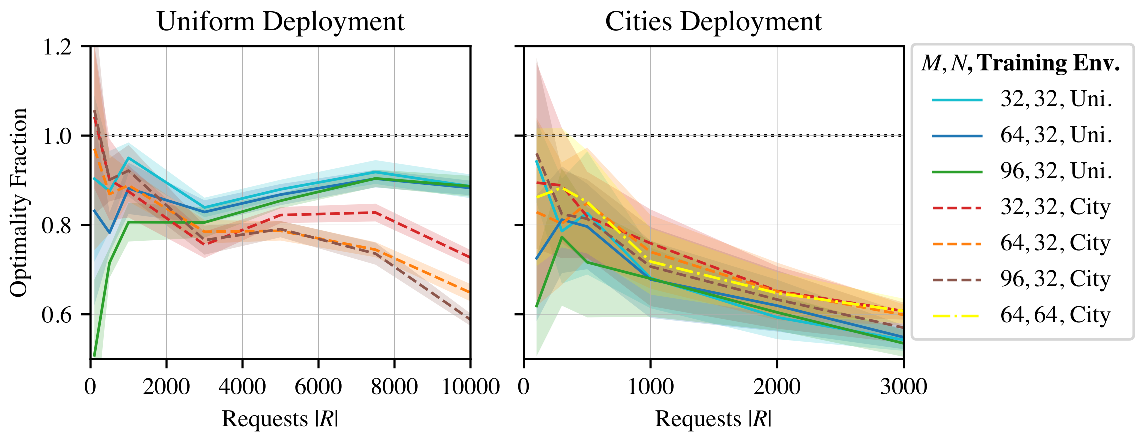

Figure 5: Performance of each policy relative to MIP solution. Tested for 15 orbits over a range of target distributions and densities. 1σ error bars.

Target Distributions Policies are trained with either the uniform or cities target distribution. For each episode, a new target set is generated within the densities used in tests, $\in [500, 10000]$ uniform targets or $\in [300, 3000]$ city targets. It is expected that uniform target training should generalize to city target testing; the local segment of targets that the agent considers appears to be a uniform distribution of the local city density. However, city-based training is expected to generalize poorly to uniform targets, as there is a lack of training data in regions lacking cities.

These intuitions are confirmed by Figure 5: Uniform training performs nearly as well as city-based training when tested on the cities distribution, showing its ability to generalize to non-uniform distributions. In the other direction, city-based training does not generalize well when tested in the uniform environment, performing significantly worse than the uniformly-trained policies. Disappointingly, neither performs particularly well on cities when compared to the MIP solution.

Results

The $M = 32$, $N = 32$, uniform-target trained policy is selected for further analysis due to the satisfactory performance exhibited in Figure 5. The policy found by PPO is compared to global MIP solutions when tested over uniform (Figure 6) and city-based (Figure 7) target distributions, target densities, and planning horizons.

When deployed in a uniform target environment, Figure 6 shows that PPO is able to achieve comparable cumulative rewards to the MIP solutions, shown in the lower right subplot. Since the MIP solver is global, it is able to achieve higher rewards over shorter planning horizons through greedy behavior; this is opposed to the PPO policy, which always acts for the pseudo-infinite horizon and thus shows no difference in reward per orbit over different horizons. As a result MIP and PPO rewards are most similar over the longest horizon (15 orbits), showing near-optimality from the PPO strategy.

Despite similar overall rewards in the uniform case, the strategies displayed by PPO versus MIP are quite different. PPO tends to select fewer, higher value targets than the MIP solutions. It is theorized that this type of solution is easier to find with PPO since imaging many low-value targets

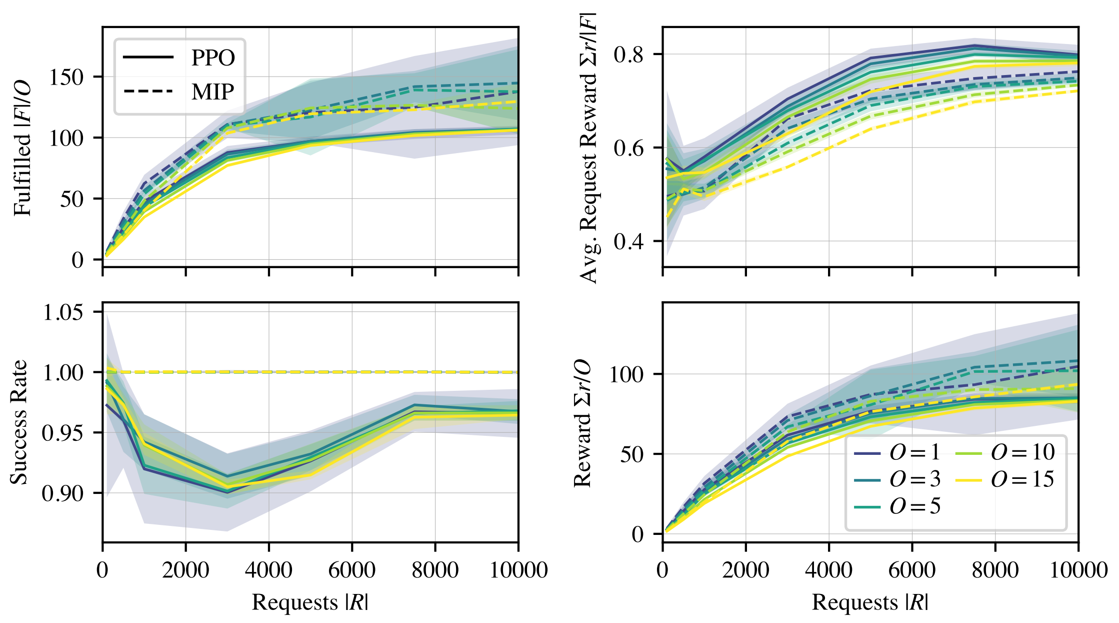

Figure 6: Over uniform targets, comparison of the policy found by PPO versus global MIP solutions for different orbit O horizons. 1σ error bars. *Fulfilled:* Number of targets imaged per orbit by the satellite. *Avg. Request Reward:* Average value of imaged targets. *Success Rate:* Fraction of targets imaged before out of range. *Reward:* Cumulative per-orbit reward.

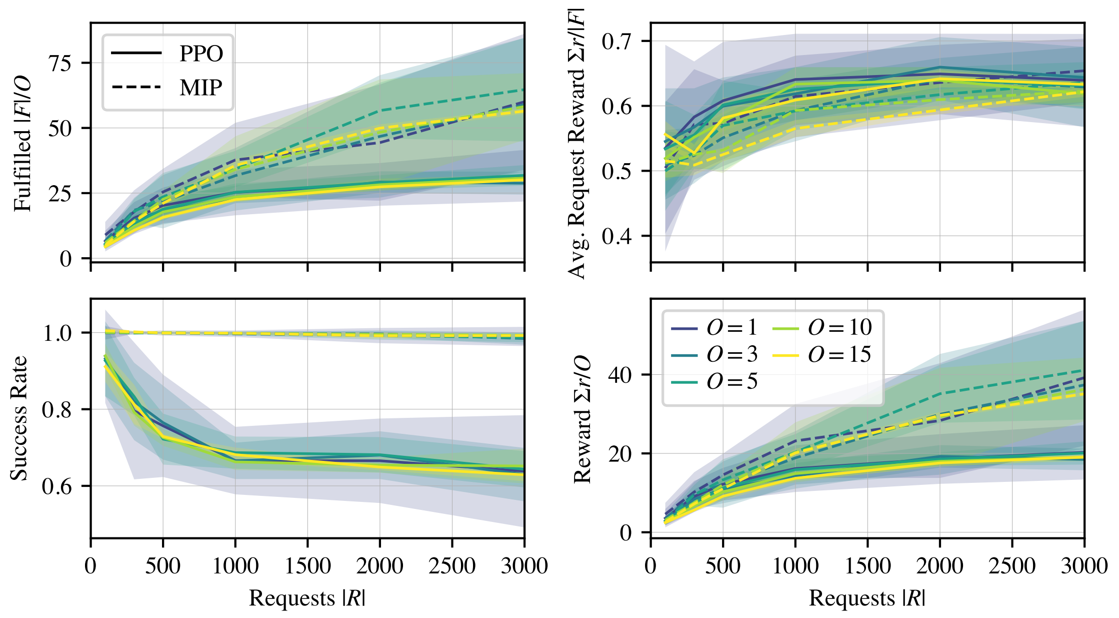

Figure 7: Over city targets, comparison of the policy found by PPO versus global MIP solutions.

quickly requires understanding longer value traces. As Figure 5 shows, the PPO policy does not learn to utilize additional information on the future target space, even though the existence of a better global solution from the MIP implies that it could be beneficial.

In the cities target environment, Figure 7 displays many of the same trends as in the uniform environment, but with notably lower rewards with the PPO policy versus the MIP solutions. This underperformance can be explained by a large amount of wasted time (i.e. a low per-action success rate) with PPO; this could explain the PPO-MIP gap in the uniform case as well. In general, high-density clusters in the cities distribution make it an especially difficult planning environment, challenging even the MIP solver with high solving times. A better understanding of the challenges of the cities environment is needed to improve performance in future work.

Despite the RL-based approach not achieving the pseudo-optimal reward found by the MIP solution, it shows very good performance (within 15% of optimal on uniform and 20-40% on cities) while having key advantages with respect to computation requirements and robustness. Evaluating the policy takes approximately 10 milliseconds at each decision interval regardless of the environment, while the MIP solutions take hundreds to thousands of seconds to find a solution for the long-horizon, high-density cases where the policy's performance is slightly depressed (see [10] for deeper analysis of MIP solution times). Furthermore, the policy produced by RL is closed loop: Actions are selected in response to a current and local view of the environment, even if the environment is non-stationary. Compare this to preplanning methods such as the MIP approach which are brittle to changes in the target set or other unexpected events, requiring resolving of part or all of the plan in such situations.

POWER-CONSTRAINED SCHEDULING PROBLEM

Training for the power-constrained scheduling problem, in which the satellite has a finite battery capacity and the ability to enter a sun-pointing charging mode, is considered next. The primary goal is to find the training method that minimizes loss in reward compared to the power-free problem due to time spent charging.

Power-Constrained Policies

Six policies and deployment schemes are tested in the power-constrained environment. The same hyperparameters are used as with the power-free agents, other than an addition of an action space logit for the charge action.

- **Unshielded Deployment (Unshielded Training):** The agent is trained and deployed without a shield in the power-constrained environment. This method may find good policies that an extra conservatism in the shield prevents, but does not guarantee safety in deployment. The policy is trained twice, once with zero failure penalty and once with a -300 failure penalty. To speed training, the training uses the previously described method of initially ramping down the shield to prevent frequent resets due to failures.
- **Shielded Deployment (Unshielded Training):** The same agents trained without a shield (zero and -300 failure penalty) are deployed with a shield (Equation 5). This guarantees safety but potentially comes at the expense of performance.
- **Shielded Training:** The agent is trained with a shield preventing unsafe actions during training and is deployed with the shield. This guarantees safety but is known to lead to suboptimal

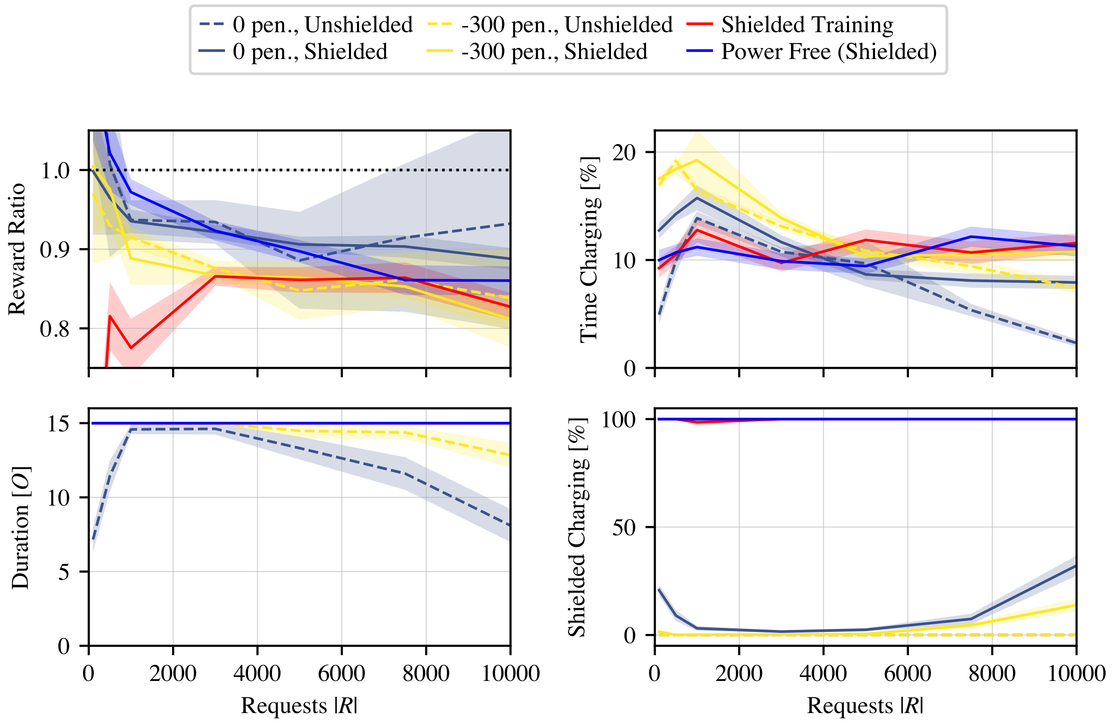

Figure 8: Deployment of policies in the power-limited environment with uniform targets for 15 orbits. $1\sigma_{\bar{x}}$ error bars. *Reward Ratio:* Reward while alive compared to the best RL agent in the power-free environment. *Time Charging:* Percent of episode spent in charge task. *Duration:* Number of orbits completed before failing. *Shielded Charging:* Percent of charging actions tasked by the shield as opposed to the base policy.

agents that "hang" on the shield.

- **Shielded Power-Free Policy:** The agent trained in the power-free environment is deployed with the shield.

Agents newly trained for the power-constrained environment are trained using the Swiss-cheese distribution of targets. This generalizes to any local target density in the same way as uniform training but also ensures that the upcoming target density state (Equation 1) has a training domain that includes periods without any targets. This is important for generalization to cases like the cities distribution, where the satellite should intuitively charge over oceans and other areas without targets.

Results

Figure 8[‖] and Figure 9 (uniform and city targets, respectively) compare the performance of each policy deployed in the power-constrained environment for 15 orbits, using the power-free environment and policy as a baseline. Disqualifying the unshielded policies due to their high failure rates, the 0-failure-penalty shielded policy performs the best in both environments.

[‖]n.b. Some reward ratios are greater than one due to favorable target generation in low-request cases.

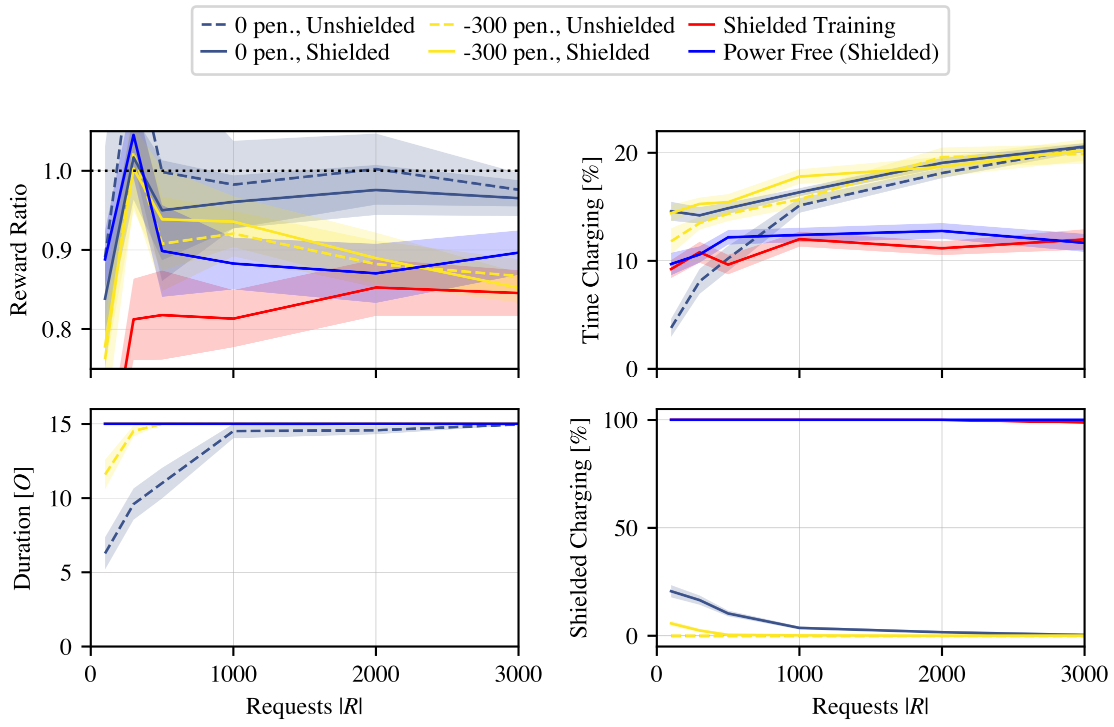

Figure 9: Deployment of policies in the power-limited environment with city targets.

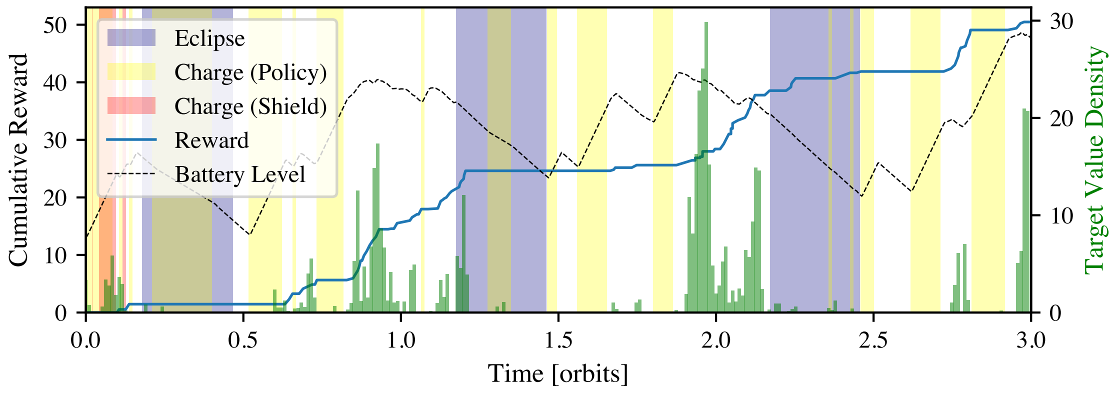

Figure 10: Charging and reward behavior of the policy over a $|R| = 3000$ city target distribution. The policy chooses to charge when there are not targets present.

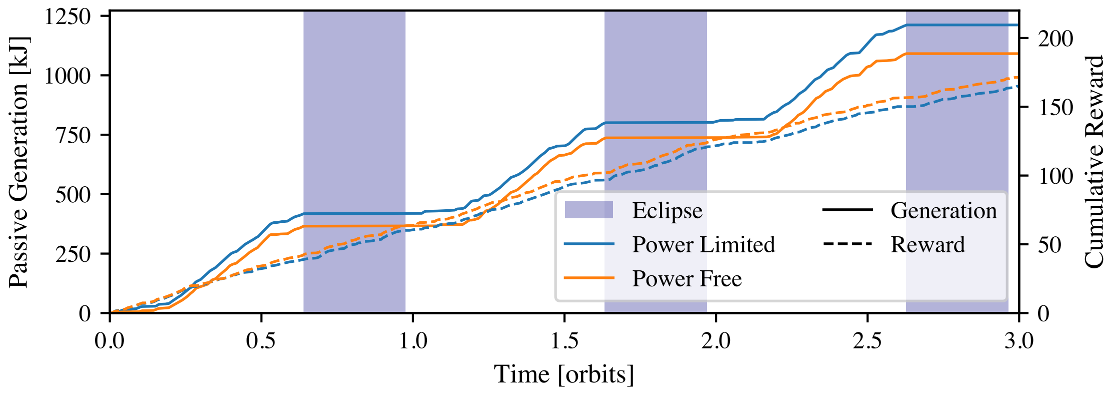

Figure 11: Passive (i.e. non-charge mode) power generation of a power-limited vs. a power-free satellite in the same $|R| = 3000$ uniform target environment. The power-constrained policy learns to generate additional power passively while completing science objectives with negligible loss in reward.

Most notably, the shielded policy performs as well or better than the power-free policy wrapped with a shield in all cases. This implies that the policy has learned to charge in an anticipatory manner, such as during reward-sparse periods as opposed to charging just when the battery is low enough to require maintenance, which may otherwise occur during high-reward segments of the episode; this positive behavior of charging when few or no targets are present is displayed in Figure 10. In fact, Figure 8 shows that the shielded policy tends to spend more time charging than the shield-wrapped power-free policy while still achieving higher overall rewards, due to this anticipatory behavior. The majority of the times the shielded policy charges are on its own accord and not due to the shield, further demonstrating "smart" behavior. Shielded training is found to be poor across all cases, with the agent almost never learning to charge other than when forced to by the shield.

In the example time-series data given in Figure 11, another aspect of the intelligent behavior of the best policy in the power-constrained environment is demonstrated. Comparing the passive power generation (i.e. power generation when not in the charging mode) to that of a power-free agent, it becomes apparent that the power-constrained agent has learned to image targets that simultaneously increase power generation as a way of reducing dedicated charging time; the power generation is greater than that of the power-free satellite which has no incentive to increase power generation. The selected targets increase power generation by creating a favorable geometry with the solar panels and instrument being antiparallel.

CONCLUSION

This paper presents an MDP for the AEOSSP and uses RL to find a performant policy for it. In some cases, performance similar to a pseudo-optimal MIP-based global solution is achieved; additionally, the policy can be executed closed loop and with significantly less computation power than the global solution. Furthermore, satisfactory performance is demonstrated in a power-constrained version of the environment, which is a scenario that cannot be adequately represented by the MIP.

Additional work is required to reach pseudo-optimal performance across all cases with the RL-based solution. In particular, the challenging case of city-distributed targets must be solved. Additionally, reformulating the MDP to be independent of state and action space lookaheads M and N is desirable, as the current formulation is not robust to tight clusters of $> N$ targets.

ACKNOWLEDGEMENT

This work is partially supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) grant, 80NSSC23K1182. This work is also partially supported by the Air Force Research Lab grant FA9453-22-2-0050.

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

REFERENCES

[1] X. Wang, G. Wu, L. Xing, and W. Pedrycz, "Agile Earth Observation Satellite Scheduling Over 20 Years: Formulations, Methods, and Future Directions," *IEEE Systems Journal*, Vol. 15, Sept. 2021, pp. 3881–3892, 10.1109/JSYST.2020.2997050.

[2] V. Gabrel, A. Moulet, C. Murat, and V. T. Paschos, "A new single model and derived algorithms for the satellite shot planning problem using graph theory concepts," *Annals of Operations Research*, Vol. 69, 1997, pp. 115–134.

[3] S. Augenstein, "Optimal Scheduling of Earth-Imaging Satellites with Human Collaboration via Directed Acyclic Graphs," *The Intersection of Robust Intelligence and Trust in Autonomous Systems: Papers from the AAAI Spring Symposium*, 2014, pp. 11–16.

[4] D. L. Applegate, R. E. Bixby, V. Chvatál, and W. J. Cook, *The Traveling Salesman Problem: A Computational Study*. Princeton University Press, 2006.

[5] G. Peng, R. Dewil, C. Verbeeck, A. Gunawan, L. Xing, and P. Vansteenwegen, "Agile earth observation satellite scheduling: An orienteering problem with time-dependent profits and travel times," *Computers & Operations Research*, Vol. 111, Nov. 2019, pp. 84–98, 10.1016/j.cor.2019.05.030.

[6] D.-H. Cho, J.-H. Kim, H.-L. Choi, and J. Ahn, "Optimization-Based Scheduling Method for Agile Earth-Observing Satellite Constellation," *Journal of Aerospace Information Systems*, Vol. 15, Nov. 2018, pp. 611–626, 10.2514/1.I010620.

[7] S. Nag, A. S. Li, and J. H. Merrick, "Scheduling algorithms for rapid imaging using agile Cubesat constellations," *Advances in Space Research*, Vol. 61, Feb. 2018, pp. 891–913, 10.1016/j.asr.2017.11.010.

[8] J. Kim, J. Ahn, H.-L. Choi, and D.-H. Cho, "Task Scheduling of Agile Satellites with Transition Time and Stereoscopic Imaging Constraints," *Journal of Aerospace Information Systems*, Vol. 17, June 2020, pp. 285–293, 10.2514/1.I010775.

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

[10] M. Stephenson and H. Schaub, "Optimal Target Sequencing In The Agile Earth-Observing Satellite Scheduling Problem Using Learned Dynamics," *AAS/AIAA Astrodynamics Specialist Conference*, Big Sky, MT, Aug. 2023.

[11] G. Picard, C. Caron, J.-L. Farges, J. Guerra, C. Pralet, and S. Roussel, "Autonomous Agents and Multiagent Systems Challenges in Earth Observation Satellite Constellations," *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021)*, May 2021, pp. 39–44, 10.5555/3463952.3463961.

[12] P. Tangpattanakul, N. Jozefowiez, and P. Lopez, "A multi-objective local search heuristic for scheduling Earth observations taken by an agile satellite," *European Journal of Operational Research*, Vol. 245, Sept. 2015, pp. 542–554, 10.1016/j.ejor.2015.03.011.

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

[14] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, "Reinforcement Learning for Solving the Vehicle Routing Problem," *NeurIPS*, 2018.

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

[16] A. Hadj-Salah, R. Verdier, C. Caron, M. Picard, and M. Capelle, “Schedule Earth Observation satellites with Deep Reinforcement Learning,” Nov. 2019. arXiv:1911.05696 [cs].

[17] D. Eddy and M. Kochenderfer, “Markov Decision Processes For Multi-Objective Satellite Task Planning,” *2020 IEEE Aerospace Conference*, Big Sky, MT, USA, IEEE, Mar. 2020, pp. 1–12, 10.1109/AERO47225.2020.9172258.

[18] A. Harris, T. Valade, T. Teil, and H. Schaub, “Generation of Spacecraft Operations Procedures Using Deep Reinforcement Learning,” *Journal of Spacecraft and Rockets*, Vol. 59, Mar. 2022, pp. 611–626, 10.2514/1.A35169.

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

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

[21] X. Zhao, Z. Wang, and G. Zheng, “Two-Phase Neural Combinatorial Optimization with Reinforcement Learning for Agile Satellite Scheduling,” *Journal of Aerospace Information Systems*, Vol. 17, July 2020, pp. 346–357, 10.2514/1.I010754.

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

[23] M. Piccinin, P. Lunghi, and M. Lavagna, “Deep Reinforcement Learning-based policy for autonomous imaging planning of small celestial bodies mapping,” *Aerospace Science and Technology*, Vol. 120, Jan. 2022, p. 107224, 10.1016/j.ast.2021.107224.

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

[25] P. W. Kenneally, S. Piggott, and H. Schaub, “Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework,” *Journal of Aerospace Information Systems*, Vol. 17, Sept. 2020, pp. 496–507, 10.2514/1.I010762.

[26] H. Schaub and S. Piggott, “Speed-constrained three-axes attitude control using kinematic steering,” *Acta Astronautica*, Vol. 147, June 2018, pp. 1–8, 10.1016/j.actaastro.2018.03.022.

[27] A. Brandonisio, L. Capra, and M. Lavagna, “Deep reinforcement learning spacecraft guidance with state uncertainty for autonomous shape reconstruction of uncooperative target,” *Advances in Space Research*, July 2023, p. S0273117723005276, 10.1016/j.asr.2023.07.007.

[28] M. Towers, J. K. Terry, A. Kwiatkowski, J. U. Balis, G. d. Cola, T. Deleu, M. Goulão, A. Kallinteris, A. KG, M. Krimmel, R. Perez-Vicente, A. Pierré, S. Schulhoff, J. J. Tai, A. J. S. Tan, and O. G. Younis, “Gymnasium,” Oct. 2023. original-date: 2022-09-08T01:58:05Z.

[29] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, “Proximal Policy Optimization Algorithms,” Aug. 2017. arXiv:1707.06347 [cs].

[30] E. Liang, R. Liaw, P. Moritz, R. Nishihara, R. Fox, K. Goldberg, J. E. Gonzalez, M. I. Jordan, and I. Stoica, “RLlib: Abstractions for Distributed Reinforcement Learning,” June 2018. arXiv:1712.09381 [cs].

[31] M. Alshiekh, R. Bloem, R. Ehlers, B. Könighofer, S. Niekum, and U. Topcu, “Safe Reinforcement Learning via Shielding,” Sept. 2017. arXiv:1708.08611 [cs].