JOURNAL OF AEROSPACE INFORMATION SYSTEMS
Vol. 22, No. 9, September 2025

Learning Policies for Autonomous Earth-Observing Satellite Scheduling over Semi-Markov Decision Processes

Mark Andrew Stephenson,* Lorenzzo Quevedo Mantovani,* and Hanspeter Schaub†
University of Colorado Boulder, Boulder, Colorado 80303

https://doi.org/10.2514/1.I011649

Advances in request scheduling algorithms are necessary to fully utilize the capabilities of agile Earth-observing satellites. While typical current approaches preplan sequences of image collections for a static set of requests, onboard autonomous methods that select requests to fulfill on-the-fly can adapt to opportunistic events and reduce the burden on operators. Prior approaches that utilized reinforcement learning for autonomy considered fixed-duration decision intervals, which limited performance over high-density requests and oversimplified mission operations. In this paper, reinforcement learning is used to derive policies that make decisions at variable intervals to improve performance by scheduling requests in a mission-like manner. The resulting policies perform competitively against optimal global schedules in a flight-like evaluation environment. This is achieved by training on a new semi-Markov decision process (sMDP) formulation of the problem, leveraging step-duration information when learning. Ablation studies over new sMDP-specific training algorithm modifications and observation spaces are performed to learn a highly performant policy. Benefits of the method, including the implicit ability to account for resource constraints, are demonstrated. Ultimately, variable-decision-interval reinforcement-learning-based policies are established as a viable method for autonomous Earth-observing satellite scheduling under mission-realistic conditions, providing a closed-loop and computationally inexpensive alternative to traditional preplanning methods.

Nomenclature

a	=	action
$\hat{\boldsymbol{c}}$	=	instrument boresight direction
$\mathcal{F}$	=	set of fulfilled requests
H	=	horizon length [orbits]
N	=	number of upcoming requests
O	=	request opportunities
o	=	request opportunity with interval $[t^o, t^c]$
R	=	set of all requests
$\boldsymbol{r}$	=	request location
r_i	=	request i priority and reward
s	=	state
t	=	time
Δt	=	step duration
$\mathcal{U}$	=	set of unfulfilled requests
V	=	Markov decision process value
$\boldsymbol{x}$	=	satellite position
z	=	battery charge
γ	=	discount rate
θ	=	angle between instrument and request directions
π	=	policy
ρ	=	request priority-location tuple
ϱ	=	reward density
$\boldsymbol{\sigma}$	=	satellite attitude
ϕ	=	elevation angle
$\boldsymbol{\omega}$	=	body angular velocity
$\boldsymbol{\Omega}$	=	reaction wheel speeds

Reference Frames

$\mathcal{E}$	=	Earth-fixed frame
$\mathcal{H}$	=	Hill's frame

I. Introduction

ON-DEMAND satellite imagery is beneficial for scientific observations, commercial operations, disaster response, or reconnaissance missions [1]. "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 in which requests are fulfilled and the time of fulfillment within the access window can be specified, leading to a greatly increased solution space for operations sequences [2]. Resource management for the satellite must also be considered when planning tasks.

Satellite observation scheduling problems are commonly represented as directed acyclic graphs (DAGs), in which vertices represent observation opportunities and edges represent feasible transitions between observations with edge weights corresponding to the incoming vertex's priority [3,4]. From this form, the problem can be transformed into an equivalent mixed-integer linear program (MILP) as it is reduced to a discrete routing problem with an optimization objective similar to the traveling salesman problem [5] or orienteering problem. In MILP formulations of the problem, edges of the DAG are assigned a binary decision variable representing if the edge is traveled, and constraints enforce that the solution must be a path of sequential edges, in addition to other constraints present in a particular problem formulation. A variety of high-performance commercial solvers exist to find the optimal solution to MILPs and quantify the suboptimality of intermediate solutions. Peng uses an MILP-based solution as a benchmark but encounters prohibitively long solution times in more complex, high-resolution cases [6]. A variety of authors use MILP solutions for target allocation problems in constellations, which demonstrate the utility for global optimization problems [7–10]. In Ref. [11], the authors utilize

Presented as Paper 24-012 at the AAS/AIAA GN&C Conference, Breckenridge, CO, February 2–7, 2024; received 20 February 2025; accepted for publication 17 April 2025; published online 23 May 2025.

*Ph.D. Student, Ann and H. J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive, Colorado Center for Astrodynamics Research; mark.a.stephenson@colorado.edu. Member AIAA (Corresponding Author).

†Distinguished Professor and Department Chair, Schaden Leadership Chair, Ann and H. J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive, Colorado Center for Astrodynamics Research. Fellow AIAA.

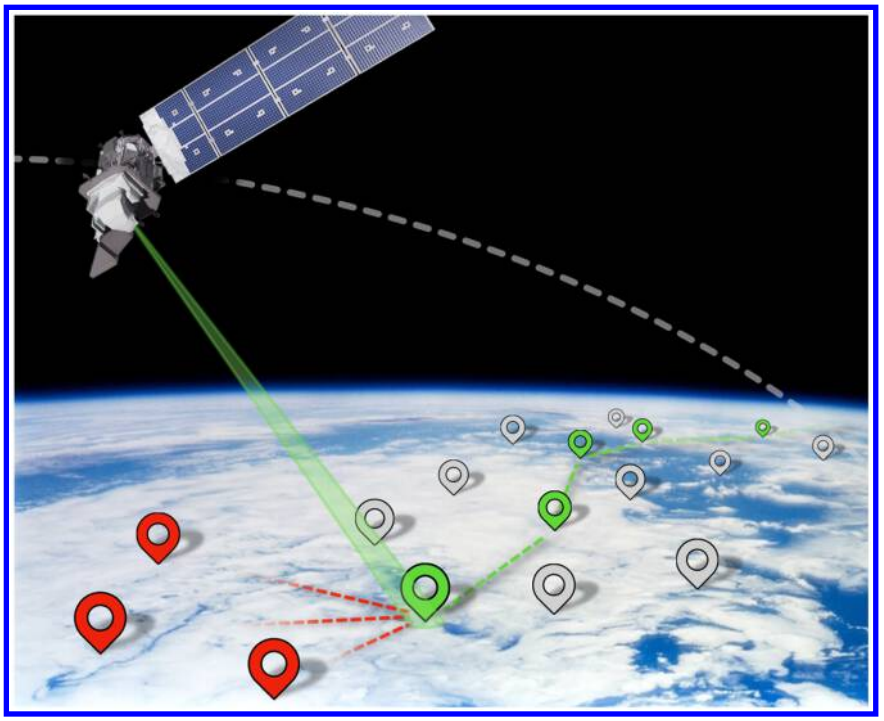


Fig. 1 Concept figure of a satellite selecting the next request to fulfill.

neural networks to construct graphs that accurately account for transition time dynamics and improve the efficiency of the MILP formulation. A popular alternative to MILP solvers is iterative local search (ILS), which leverages the problem structure to iteratively optimize short segments of the solution [12]. Peng develops an ILS algorithm for observation scheduling that can account for time-independent or time-dependent target values [6]. Tangpattanakul uses ILS to ensure fair resource allocation between users of a shared satellite [13].

Ultimately, preplanning approaches such as MILP 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 MILP, the linearization of resource dynamics 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 [12]. A variety of approaches have been proposed for reactive imaging (i.e., selecting tasks onboard based on the current environment state) [14,15]. In particular, reinforcement learning (RL) is a broadly applicable framework for closed-loop autonomy [16]. Nazari demonstrates that RL is effective at solving general waypoint routing problems [17]. Applied directly to spacecraft tasking, Harris [18], Hadj-Salah [19], and Eddy [20] formulate various Markov decision processes (MDPs) for the AEOSSP. In Refs. [20] and [19], the authors use simplified probabilistic models for the spacecraft dynamics. In [18] and later [21–23], Harris and Herrmann develop the problem with a full-fidelity simulation and utilize shields to ensure operational safety. Additional work proves the maintenance of these safety conditions using formal methods [24]. Across these formulations, fixed decision intervals are used, which can lead to suboptimal solutions when the satellite is capable of completing tasks in variable durations [18,19,22,23]. Eddy formulates the problem as a semi-Markov decision process (sMDP) that can be solved by a Monte Carlo tree search algorithm that accounts for transition time costs [20].

More recently, a variety of purpose-built RL-based pipelines for spacecraft scheduling have been proposed. Zhao considers target scheduling using two phases of RL, the first selecting observation windows and the second picking observation times within the windows [25]. Chun uses RL to aid in the solution of a graph representation of the problem space [26]. MDP representations of other spacecraft tasking problems, such as small-body imaging and communication network management, have also been formulated and solved with RL [22,27,28].

In this work, autonomous agents are demonstrated executing learned policies with variable-duration decision intervals for the AEOSSP that are competitive with MILP-based approaches when compared in a high-fidelity simulation. Figure 1 shows a satellite considering which upcoming request to fulfill. Prior literature only provides formulations of the MDP or demonstrates solutions over simplifications of the problem (such as fixed decision intervals or graph-based abstractions). To learn an efficient policy on a high-fidelity instance of the problem, the problem is formulated as an sMDP in which actions are taken only for a minimum necessary duration in order to avoid satellite downtime. Furthermore, cases that the MILP 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, generalization across request distributions, and robustness to mismodeling, as opposed to the brittle, high-cost solutions provided by other methods.

The quality of the learned policies is evaluated across a variety of ablation studies. In the power-free environment, two sMDP formulations are shown to be superior to the MDP formulation, the performance of different observation spaces is compared, and the resulting policies are evaluated against the MILP-based solutions, demonstrating closed-loop performance that is competitive with the optimal global solution. Finally, the ability to learn policies for power-constrained environments is demonstrated, showing that policies from RL are able to contend with realistic and challenging constraints in a sensible manner. Overall, the work demonstrates in a high-fidelity simulation environment that RL is a viable and competitive method for solving the AEOSSP with onboard autonomy.

II. Problem Formulation

A target-sequencing-only version of the AEOSSP is described and formalized as a partially observable semi-Markov decision process (POsMDP) so that it can be solved with deep reinforcement

learning (DRL). A power-constrained version of the problem is also developed that requires the agent to manage onboard resources.

A. Partially Observable Semi-Markov Decision Processes

MDPs are a framework for defining sequential decision-making problems [16]. They consist of a state space $\mathcal{S}$, action space $\mathcal{A}$, transition probability function $T(s'|s,a)$, reward function $R(s,a,s')$, and a discount factor γ. The agent observes the current state s_t and selects an action a_t based on a policy $\pi(a|s)$. The environment then transitions to a new state s_{t+1} according to the transition function, yielding a reward r_t. The goal of the agent is to learn the policy $\pi(a|s)$ that maximizes the expected value, or the γ-discounted sum of returns:

$$V^{\pi}_{\text{MDP}}(s) = r + \gamma \mathbb{E}[V^{\pi}_{\text{MDP}}(s')|T(s'|s,a),\pi(a|s)] \tag{1}$$

$$= r_0 + \gamma r_1 + \gamma^2 r_2 + \cdots = \sum_{i=0}^{\infty} \gamma^i r_i \tag{2}$$

where the discount factor $\gamma \in [0,1)$. The latter expression [Eq. (2)] gives the value in terms of a sequence of experience tuples (s_t, a_t, r_t).

For an sMDP, an additional notion of continuous time is included in the problem formulation. This can be notated as a probabilistic step-duration function $F(\Delta t|s,a,s')$ with a reward density ϱ instead of reward r [29], or as an MDP in which decisions are only made at certain supersteps, where each superstep consists of a variable number of substeps in the base MDP [30,31], among other conventions. In general, the one-step γ (called the discount rate in the sMDP context) discounted reward for a general $\varrho_t(t)$ is given by

$$r^{(\gamma)}_t = \int_{t_t}^{t_t+\Delta t_t} e^{\gamma(t-t_t)} \varrho_t(t)\, \mathrm{d}t \tag{3}$$

In this work, a step-duration function $F(\Delta t|s,a,s')$ is employed and rewards are distributed at the end of the step, i.e., $\varrho_t(t) = r_t \delta(t-(t_t+\Delta t_t))$, where δ is the Dirac delta function. As a result, experience tuples come in the form $(s_t, a_t, r_t, \Delta t_t)$, where Δt_t is the duration between s_t and s_{t+1}. The value function for this sMDP convention is given by

$$V_{\text{sMDP}}(s_0) = \gamma^{\Delta t_0} r_0 + \gamma^{\Delta t_0 + \Delta t_1} r_1 + \gamma^{\Delta t_0 + \Delta t_1 + \Delta t_2} r_2 + \cdots \tag{4}$$

$$= \sum_{t=0}^{\infty} \gamma^{\sum_{i=0}^{t} \Delta t_i} r_t \tag{5}$$

The final aspect of a POsMDP is partial observability. In a partially observable Markov decision process (POMDP), the agent receives an observation $Z(o|s)$ rather than the full state for decision-making. The concepts of the sMDP and POMDP can be combined to yield a POsMDP that incorporates both partial observability and variable step durations.

B. Request Sequencing Problem

1. Objective

The objective of the target sequencing problem is to maximize the sum of priorities of imaging requests fulfilled by an agile satellite equipped with a camera-like sensor in a fixed low-Earth orbit:

$$\begin{aligned} &\text{maximize} && \sum_{\rho_i \in \mathcal{F}} r_i \\ &\text{subject to} && \text{simulation feasible} \end{aligned} \tag{6}$$

subject to the constraints of the operation of the satellite, which are given by a high-fidelity simulation. No resource constraints are considered in this first problem formulation; that is, the satellite has unlimited power and data storage space.

2. Request Model

Requests are modeled as discrete, Earth-fixed points that are desired to be imaged, as opposed to continuous areas of land.[‡] Requests $\rho_i \in R$ consist of a tuple of a planet-fixed location and a priority, $\rho_i = (\boldsymbol{r}_i, r_i)$. All requests start in the unfulfilled set, $R = U$, and are moved to the fulfilled set $\mathcal{F}$ once imaged. Once a request has been fulfilled, it is assumed that there is no value in completing it again, though an operator could add a copy request in the same location if reimaging was desired.

Requests can only be fulfilled when associated constraints are met. In this paper, only elevation constraints are considered for access, but this method can generalize to other a priori constraints such as time-of-day requirements. The time intervals for which all constraints are met are called opportunities $[t^o, t^c] = w \in o \in O_i$, where O_i is the set of intervals for which request ρ_i can be fulfilled. If the satellite attempts to fulfill a request outside of an opportunity, the request is not fulfilled and no value is yielded.

For clarity of notation when indexing requests and opportunities, the subscript i refers to a global request index, while the subscript n refers to the nth upcoming request, ordered by opportunity close times $t < t^c_{n=1} \leq t^c_{n=2} \leq \ldots$ greater than the current time t.

3. Satellite Dynamics

The satellite has an imaging instrument pointing in the body-fixed $\hat{\boldsymbol{c}}$ direction. To fulfill a request, the satellite must point the instrument in the target direction:

$$\hat{\boldsymbol{c}}^{\text{ref}}(t,i) = \frac{\boldsymbol{r}_i - \boldsymbol{x}(t)}{|\boldsymbol{r}_i - \boldsymbol{x}(t)|} \tag{7}$$

within an angle threshold

$$\angle(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}^{\text{ref}}) < \delta\theta_{\max} \tag{8}$$

A reference body rate such that the instrument tracks the target

$$\boldsymbol{\omega}^{\text{ref}}_{\mathcal{B}/\mathcal{P}}(t;s,n) = \frac{({}^{\mathcal{P}}\mathrm{d}/\mathrm{d}t)\boldsymbol{x}(t) \times \hat{\boldsymbol{c}}^{\text{ref}}(t,n)}{|\boldsymbol{r}_i - \boldsymbol{x}(t)|} \tag{9}$$

must be met within some threshold

$$|\boldsymbol{\omega}_{\mathcal{B}/\mathcal{P}} - \boldsymbol{\omega}^{\text{ref}}_{\mathcal{B}/\mathcal{P}}| < \delta\omega_{\max} \tag{10}$$

when fulfilling a request. The pointing angle error $\delta\theta_{\max}$ and rate error $\delta\omega_{\max}$ thresholds must be met during an open opportunity $t^o \leq t \leq t^c$ to fulfill a request.

The satellite's attitude is controlled in order to navigate between requests. In this paper, a flight-proven modified Rodrigues parameter (MRP)-based attitude control algorithm [33] is used to slew the satellite between requests, requiring the controller to stabilize before an image is collected. As a result of needing to wait for satellite dynamics to slew and stabilize between requests, the transition times between requests must be determined by the dynamics simulator and can be challenging to predict. In the simulation used in this work, imaging is considered to be instantaneous once the pointing criteria are met; however, a nonzero imaging time could be added to the simulation. When training, this additional time would be accounted for in the implicit estimation of slew feasibility learned by the policy.

C. Markov Decision Process Formalization

The AEOSSP can be represented as a POsMDP. Actions correspond to high-level modes in the flight software, such as attempting to fulfill a certain request or—in the power-constrained variation—entering a charging mode; this mode-based approach to spacecraft RL was proposed by Harris et al., Eddy et al., and Herrmann and

[‡]Eddy and Kochenderfer explain how area-based request models can be decomposed into the point-based request paradigm in [32].

Table 1 Elements in the observation o and their normalization constants

Quantity	Norm.	Dim.	Observation variant S1	S2	S3	P	Description
${}^{\mathcal{E}}\boldsymbol{\omega}_{\mathcal{BE}}$	0.03 rad/s	3	✓	✓	✓	✓	Body angular rate
${}^{\mathcal{E}}\hat{\boldsymbol{c}}$	— —	3	✓	— —	— —	— —	Earth-frame instrument pointing direction
${}^{\mathcal{H}}\hat{\boldsymbol{c}}$	— —	3	— —	✓	✓	✓	Hill-frame instrument pointing direction
${}^{\mathcal{E}}\boldsymbol{r}_{\mathcal{BE}}$	r_E	3	✓	✓	✓	✓	Earth-fixed position, Earth radius-normalized
${}^{\mathcal{E}}\boldsymbol{v}_{\mathcal{BE}}$	v_{orb}	3	✓	✓	✓	✓	Earth-fixed velocity, orbital velocity-normalized
r_n	— —	N	✓	✓	✓	✓	Rewards $\in [0, 1]$ of next N requests
${}^{\mathcal{E}}\boldsymbol{r}_{n\in N}$	r_E	$3N$	✓	— —	— —	— —	Earth-frame positions of next N requests
${}^{\mathcal{H}}\boldsymbol{r}_{n\in N}$	r_E	$3N$	— —	✓	✓	✓	Hill-frame positions of next N requests
$\delta\theta_{n\in N}$	$\pi/2$ rad	N	— —	— —	✓	✓	Pointing error to next N requests
$\delta\omega_{n\in N}$	0.03 rad/s	N	— —	— —	✓	✓	Pointing rate error to next N requests
$t^o_{n\in N} - t, t^c_{n\in N} - t$	300 s	$2N$	— —	— —	✓	✓	Opportunity open and close of next N requests
t	$t_{\max}$	1		(Finite horizon MDP)			Time through episode (completion fraction)
Equation (13)	5.0	K	— —	— —	— —	✓	Upcoming request density
z	$z_{\max}$	1	— —	— —	— —	✓	Battery charge fraction
$t^o_{Ecl} - t, t^c_{Ecl} - t$	T	2	— —	— —	— —	✓	Next eclipse transitions, orbital period-normalized
${}^{\mathcal{H}}\hat{\boldsymbol{s}}$	— —	3	— —	— —	— —	✓	Hill-frame sun direction

Schaub [18,20,23]. The agent is rewarded for fulfilling requests based on an operator-assigned per-request priority.

Much of the POsMDP—states, observations, and transitions—implicitly emerges from an underlying simulation of the environment. The POsMDP and underlying simulator are implemented using BSK-RL,[§] a modular, open-source package for creating spacecraft tasking RL environments. BSK-RL uses the standard Gymnasium API for RL environments, making the package compatible with all major RL frameworks [34]. Internally, the spacecraft and environment dynamics are modeled using the Basilisk[¶] spacecraft simulation framework [35]. Basilisk is a high-fidelity, high-speed modular spacecraft simulator, which allows for component-level modeling of the spacecraft and its control algorithms. Ultimately, any effects that can be modeled in the simulation can be accounted for in the POsMDP and derived policies.

The POsMDP is defined by the tuple $(\mathcal{S}, \mathcal{A}, T, F, R, \mathcal{O}, Z)$. 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 state space is the complete space of simulator states required to maintain the Markov assumption. This includes directly observable variables such as the spacecraft's dynamic state and upcoming request information, unobserved variables including far-off request 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. Experiments are run for a variety of observations, given in Table 1. In general, these observations consist of some information about the spacecraft and some information about each of the N next upcoming unfulfilled requests. The request observations are ordered by their position along-track, making the assumption that nearby requests are most relevant for decision-making. The observation consists of information that the satellite would reasonably have available onboard with minimal uncertainty, allowing for closed-loop operation onboard, other than updates to the request list from the ground. Observation elements are normalized to fall approximately in $[-1, 1]$, which improves the performance of DRL algorithms. The number of upcoming unfulfilled requests in the observation N is tunable. Observation elements are taken directly from the state without noise, though other papers have shown that reasonable noise can be added to the observation without impacting performance as long as the noise was modeled in training and deployment [36]. With the observation expressed as a subset function of state $O(s) \subset s$, the observation probability function is deterministic.[**]

Action Space $\mathcal{A}$: The satellite has N imaging actions $a_{\text{im},n}$ that task the satellite with fulfilling the nth upcoming unfulfilled request. As in the observation space, the upcoming requests are ordered by their position along the track and are recomputed at each decision interval, making the assumption that the agent would never reasonably want to select a far-off request as the next task. As shown in the results section, N must be balanced to give the agent sufficient information and choice without facing performance challenges associated with a very large action and observation space. Formulating the observation and action spaces in this manner (parameterized by N) is also conducive to RL: it maintains a reasonably small dimensionality for the observations and actions and keeps those spaces a constant dimension, which is necessary for most DRL algorithms. The availability of an action does not imply feasibility: a request's opportunity window may close before the satellite can slew to and settle the instrument in the pointing direction in the high-fidelity simulation of the attitude control system. Since feasibility is not known a priori, an unfeasible action may be tasked, and the controller will attempt to complete it until one of the criteria in the transition probability function is met.

Transition Probability Function $T(s'|s, a)$ and Step Duration Function $F(\Delta t|s, a, s')$: 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 what conditions are met as the action is executed. For imaging actions, the simulator stops as soon as the satellite is successful or cannot be successful, thereby eliminating idle time. When generating the next state, the simulator runs until one of four conditions is met:

1) **Request Fulfilled:** If the request is successfully fulfilled (moved from $\mathcal{U}$ to $\mathcal{F}$), the step ends. No additional reward can be obtained from continuing to image the now-fulfilled request, so the satellite should retask immediately.

2) **Opportunity Close:** If the opportunity for the request being imaged closes before successful fulfillment, $t > t^c_n$, the step ends. There is no potential for the satellite to fulfill the request until a later orbit, so the satellite should retask immediately.

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

[¶]Data available online at https://avslab.github.io/basilisk/.

**Notated in the traditional manner, the observation probability function would be $Z(O(s)|s) = 1$.

3) **Maximum Step Duration Timeout:** If neither of the above conditions are met before $\Delta t_{\max} = 5$ min have elapsed, the step ends.

4) **Maximum Episode Duration Timeout:** If the maximum episode duration (five orbits in training, as specified case-by-case for evaluation) is met, the step ends. Actions for requests beyond the duration are interrupted.

The transition time Δt is deterministically the time elapsed between the previous state and the next state, which is determined by the simulator, $F(t' - t|s, a, s') = 1$.

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

$$R(s, a_{\text{im},n}, s') = \begin{cases} r_n & \text{if } \rho_n \in \mathcal{U} \text{ and } \rho_n \in \mathcal{F}' \\ 0 & \text{else} \end{cases} \tag{11}$$

D. Power-Constrained Problem Variation

1. Objective

The AEOSSP is commonly subject to resource constraints that need to be maintained by the scheduler in addition to the data-based objective. A variant of the previous problem is proposed in which the satellite has a power subsystem that must be managed:

$$\begin{aligned} \text{maximize} \quad & \sum_{\rho_i \in \mathcal{F}} r_i \\ \text{subject to} \quad & \text{simulation feasible} \\ & \text{satellite alive} \end{aligned} \tag{12}$$

Specifically, 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 $\dot{z}_{\text{base}}$, a reaction wheel power draw $\dot{z}_{\text{rw}}(\boldsymbol{\Omega})$, and, when the imaging instrument is turned on, an additional draw $\dot{z}_{\text{im}}$. The power generation of the solar panels is a function of the attitude over time and the eclipse state. As a result, the power state over time is highly dependent on the actions taken.

2. POMDP Modifications

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

1) **Simulation:** The underlying Basilisk simulation that gives state transitions is modified to include the power subsystem. While reformulating some planning algorithms to account for power is challenging, the POMDP formulation allows for the straightforward inclusion of power constraints directly from the power dynamics model.

2) **Observation Space:** As listed in Table 1, the observation in the power-constrained variation is augmented with relevant information about the charge state. This includes current charge z, the eclipse transition times t^o_{Ecl}, t^c_{Ecl}, and wheel speeds, among other elements. To encode long-term presence or absence of upcoming requests, the request density function

$$\left\{ \sum_{\rho_i \in \mathcal{P}_k} r_i \middle| k = 1, \ldots, K \right\}, \quad \mathcal{P}_k = \{\rho_i \in \mathcal{U} | t + (k-1)\Delta t \le t^o_i < t + k\Delta t\} \tag{13}$$

is added to the observation, computed from the satellite's onboard request list; it gives the cumulative priorities of unfulfilled requests in each of the next $K = 20$ intervals of $\Delta t = 5$ min. The intent of this observation is to reveal gaps in upcoming target availability without needing a very large N to expose a long horizon of requests to the agent. With this information, the agent should be able to identify if there are upcoming periods with fewer requests that could be used to perform resource management actions.

3) **Action Space:** The satellite is given an additional charging action, a_{charge}, which points the solar array toward the sun for 5 minutes. Charging is determined by the underlying simulation: Power is passively generated at any point when the solar panels are exposed to the sun (even if $a \neq a_{\text{charge}}$). Conversely, tasking a_{charge} does not guarantee charging, such as if the satellite is in eclipse. The charging action has no explicit reward or penalty associated with it; the agent must learn to use it to avoid failure states while accounting for the implicit opportunity cost of not imaging.

4) **Reward Function:** 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 $r_{\text{penalty}} \le 0$ is yielded for entering a failure state.

III. Methods

Reinforcement learning is used to find policies that approximate the solution to the POMDP; specifically, the widely used DRL algorithm proximal policy optimization (PPO) is leveraged. Two problem-specific considerations are discussed: the impacts of variable-interval decisions and the use of shielding to guarantee safety in the power-constrained problem. Finally, a pseudo-optimal solution to the request-sequencing problem from [11] is briefly described; this method is used as a comparison benchmark.

A. Reinforcement Learning

The objective of RL is to find the policy $a = \pi(s)$ that maximizes the sum of future rewards in the MDP; it is a function that defines the action an agent should take in response to an environment state or observation [16]. The learning agent does not have access to the underlying transition and reward 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 the ability of deep learning to generalize over continuous state and action spaces.

In the case of POMDPs, it is necessary to maintain a belief state to account for the unobserved variables in the state space. In this work, the belief state is not explicitly maintained; instead, the observation space is designed to include all relevant information for Markovian decision-making.

1. Proximal Policy Optimization

A popular and performant DRL algorithm is PPO [37]. PPO is a policy gradient method, meaning that it directly optimizes the policy π_θ by iteratively updating the policy network 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. A short summary is given in Algorithm 1.[††] In Ref. [23], the authors show that PPO performs well in a variety of similar spacecraft tasking problems.

Algorithm 1: Proximal policy iteration (PPO)

for $k = 1, 2, 3, \ldots$ **do**
  Collect trajectories by running policy π_{θ_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 and the value function
end for

In this work, policies are trained using RLlib's implementation of PPO [38] for 24 hours on 32 Intel Milan CPU cores, which corresponds to 7M–11M steps, 20k–25k three-orbit episodes, and 11–14 years of simulated mission time. Following a hyperparameter search, the following values are used: separate policy and value

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

networks, each with 2 layers of 2048 fully connected neurons, an input layer with dimension indicated by Table 1 (observation formulation dependent), and output layers of N (power-free) or $N+1$ (power-constrained) scalars for the value network and logits for the action network; discount rate γ is 0.997; training batch size is 3000 with 10 SGD iterations; gradient clipping is 0.5; PPO clip parameter is 0.2; learning rate is 3×10^{-5}; and generalized advantage estimation (GAE) λ is 0.95.

2. *Learning on an Infinite Horizon POsMDP*

The use of variable-duration decision intervals in the sMDP introduces challenges not typically encountered in RL, in which each step usually has an equal-time opportunity cost. Two methods are considered to encode this: restructuring the sMDP as an MDP and modifying PPO to discount according to the time spent at each step.

The first method is to consider the problem for finite episodes, at the end of which no further rewards can be obtained. The time-through-episode is added to the observation so that the agent can learn to treat time as an expendable resource. No modifications are needed to the learning algorithm, as the problem is no longer semi-Markovian.

Alternatively, the advantage estimator used by the PPO, known as the GAE, can be modified to account for the semi-Markovian nature of the problem. The standard form of the GAE

$$\hat{A}^{\mathrm{GAE}}_{t,\mathrm{MDP}} = \sum_{i=0}^{\infty} \lambda^i \gamma^i (r_i + \gamma V(s_{t+1}) - V(s_t)) \tag{14}$$

was introduced by Schulman et al. [39]. By rederiving the GAE using the sMDP form of value [Eq. (5)], a modified GAE can be expressed as

$$\hat{A}^{\mathrm{GAE}}_{t,\mathrm{sMDP}} = \sum_{i=0}^{\infty} \lambda^i \gamma^{\sum_{j=0}^{i} \Delta t_{t+j}} \left(r_{t+i} + \gamma^{\Delta t_{t+i+1}} V(s_{t+i+1}) - V(s_{t+i})\right) \tag{15}$$

Note that this convention matches the assumption that reward is received at the end of the episode. This formulation also differs from similar modified GAE in the RLlib examples [38] and in Menda et al. [31], as the exponentially decaying average is here taken over steps as opposed to duration.

With this modification, the agent can be trained in pseudo-infinite episodes, in which the episode is truncated after a fixed number of orbits and the value network is used to bootstrap estimates of additional value beyond that point. The inclusion of time-through-episode in the state is not theoretically necessary in this case.

To illustrate the impact of these modifications, an ablation study is performed to compare the performance of the standard GAE on finite episodes, the modified GAE on finite episodes, and the modified GAE on infinite truncated episodes. These experiments use the S3 observation on the sequencing-only problem formulation. Figure 2 shows that the modified GAE is more effective at learning efficient target sequencing than the standard GAE, which too strongly favors high-value requests. The time-inclusive, finite-horizon sMDP formulation is used throughout the rest of the paper (arbitrarily selected due to equivalent performance to the pseudo-infinite horizon formulation).

3. *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 [40]. In this work, the postprocessing 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 [21].

Safety is only a consideration for the power-constrained variation of the problem. A relatively simple, hand-crafted shield is

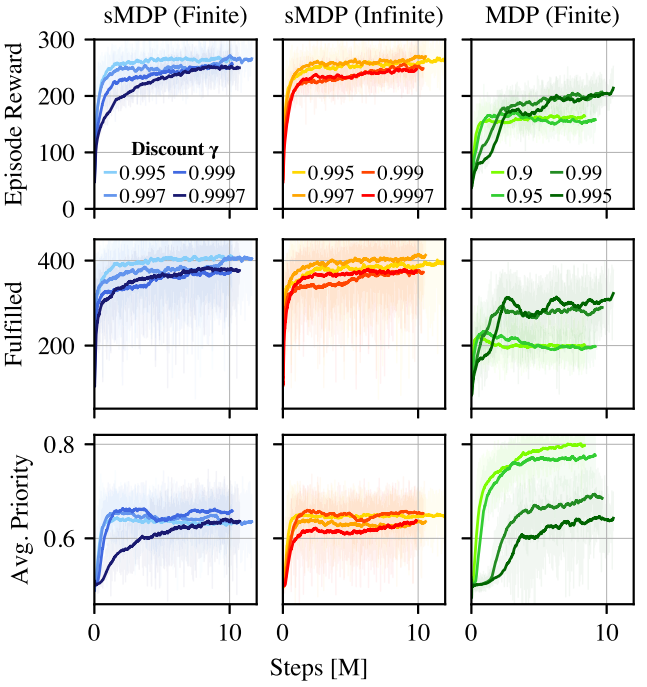


Fig. 2 Comparison of the step- and time-discounted methods over a range of γ.

employed. The shield selects the charging action if the battery level is below

$$z_{\mathrm{minsafe}} = \begin{cases} z_{\mathrm{floor}} - (t^c_{Ecl} - t^o_{Ecl})\dot{z}_{\mathrm{draw}} - t^o_{Ecl}\dot{z}_{\mathrm{gain}} & \text{if not in eclipse and} \\ & > z_{\mathrm{floor}} \\ z_{\mathrm{floor}} - t^c_{Ecl}\dot{z}_{\mathrm{draw}} & \text{if in eclipse} \\ z_{\mathrm{floor}} & \text{else} \end{cases} \tag{16}$$

which maintains the battery level at least at z_{floor}, with a higher requirement as the eclipse nears in order to survive the eclipse without losing power. Worst-case estimates of $\dot{z}_{\mathrm{gain}} = 100\%/\mathrm{orbit}$ and $\dot{z}_{\mathrm{draw}} = -100\%/\mathrm{orbit}$ are used, and z_{floor} is set to 25%. While this shield is arbitrary, it is effective at preventing the agent from entering failure states without being overconservative.

B. Pseudo-Optimal Benchmark

The method described in [11] is taken as a pseudo-optimal benchmark for the request sequencing problem. In short, opportunities for each request are discretized into access times. Supervised learning is used to train a neural network to predict slew durations between requests; these predictions estimate if the transition from one discrete request time to another is feasible. Feasible transitions are represented as edges on a directed acyclic graph of slews. An MILP is formulated to find the maximum reward path across the graph, which can be solved by commercial software to find a request sequence.

The MILP solution to the scheduling problem provides a good point of comparison for the RL-based methods. It is optimal up to the time discretization and the accuracy of the slew estimator, 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 requests 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 pre-planning algorithms. MILP planners and ILS planners, which

are more efficient than MILPs but lack optimality quantification, precompute a sequence of requests to be executed open loop. Additionally, they tend to be computationally expensive over larger request sets and longer horizons (e.g., 10,000 requests over a 10-orbit horizon can take hours to solve). As a result, they are brittle to a changing environment that necessitates frequent replanning and are typically limited to the ground-based planning paradigm. Furthermore, accurately accounting for nonlinear resources such as power generation and depletion is not possible with these methods; at best, such constraints can be approximated.

IV. Results: Sequencing Problem

A. Simulation Environment

The simulation is executed in the aforementioned BSK-RL environment. This environment is used for training and evaluation of the RL agents, as well as execution of the pseudo-optimal solutions used for comparison. The environment configuration code can be found in the BSK-RL repository and documentation at Examples → Environments → Agile Earth-Observing Satellite.[‡‡] This environment includes spacecraft dynamics with three reaction wheels controlled by flight software and SPICE-based orbital dynamics.

1. *Satellite Parameters*

Physical satellite properties and control gains used for sequencing environment are given in Table 2. Unlisted parameters are the defaults used by BSK-RL.

2. *Request Distributions*

Two distributions of requests are used in this work: uniform and cities; instances of these distributions at two different densities are shown in Fig. 3. "Uniform" distributes requests uniformly over the Earth's surface. "Cities" distributes requests at the locations of a random subset of the 42,000 most populous cities[§§]; this distribution is selected to be representative of realistic data sources that display tight clustering in some regions and nothing of interest in others. Each per-episode instance of each distribution and density has a unique set of requests, both in training and evaluation. As a result, the policy learns to generalize over randomized and not specific request lists.

The uniform distribution is considered for up to 10,000 globally distributed targets, while cities are considered for up to 3000 targets; these distributions lead to similar maximum local densities. For all distributions, the number of requests is capped at 1000 per orbit to prevent the environment from becoming too dense. For all spatial distributions, request priorities are uniformly distributed in $[0, 1]$. Request availability is determined by the minimum elevation angle $\phi_{\min}$ given in Table 2.

B. Ablation Studies

Two ablation studies are conducted to determine the impact of the observation space, the lookahead distance N, and the request distribution used in training on the performance of the RL agents.

1. *Request Lookahead and Distribution*

The first ablation study investigates how the request lookahead distance N and the training distribution impact performance. Six agents are trained for each combination of $N \in [16, 32, 64]$ and training distribution $\in$ [Uniform, Cities] using the S3 observation from Table 1. Once trained, the agents are evaluated on different densities of each distribution for a 10-orbit horizon. These results are compared to the MILP solution (found in $\lesssim$1 h) and the MILP upper bound in Fig. 4. The true optimality of the policy lies between the 1-hour solution and upper bound lines, but practically the MILP solver will not have the solution and upper bound converge in a reasonable solve time for large instances of the problem. The action distributions of each agent in the evaluation cases and the equivalent actions selected by the MILP are given in Fig. 5. Since lower index

Table 2 Parameters for the sequencing environment (control gains as in [33])

	Parameter	Value(s)
Physical	Inclination	45°
	Altitude	800 km
	$\delta\theta_{\max}$	0.01 MRP norm (2.29°)
	$\delta\omega_{\max}$	0.01 rad/s (0.57°/s)
	$u_{\max}$	0.4 N · m (per-axis, 3× RW)
	$m, \boldsymbol{I}$	330 kg, [82.1, 98.4, 121.0] kg · m^2
Control	K_1	0.25
	K_3	3.0
	$\omega_{\max}$	5.0 rad/s
Requests	$\phi_{\min}$	58°
	Field of regard	Circular, $r = 500$ km
	Distribution	{Uniform, Cities}
	r	Uniform(0,1)
	$\lvert R\rvert_{\text{uniform}}$	[100, 10,000]
	$\lvert R\rvert_{\text{cities}}$	[100, 3000]

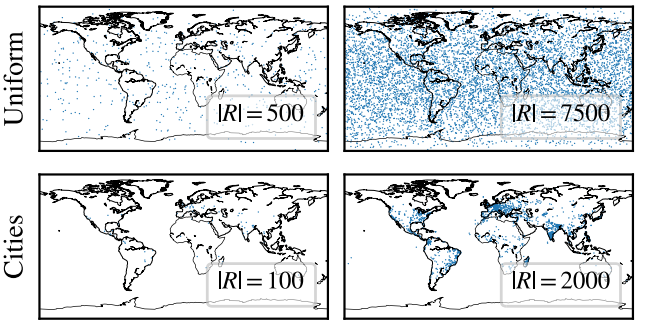


Fig. 3 Examples of each request distribution at different densities.

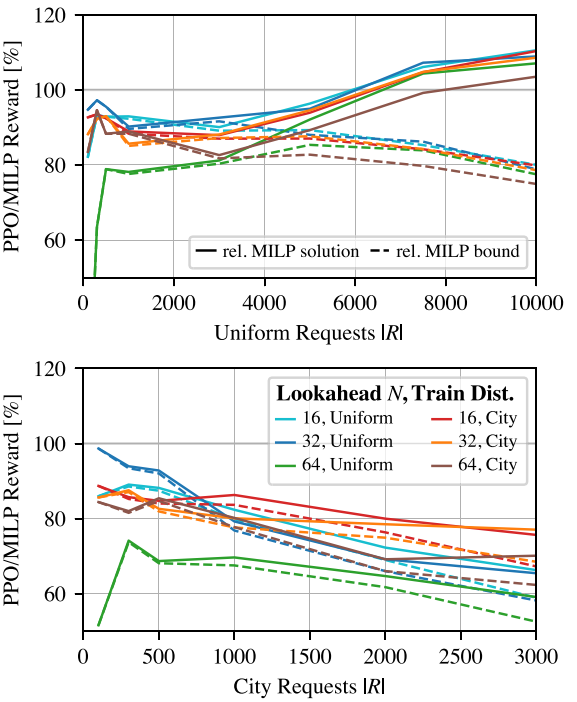
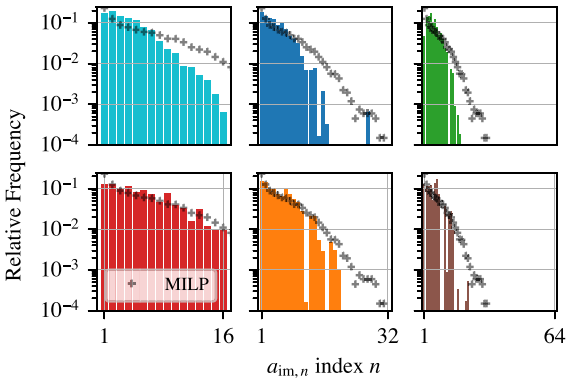


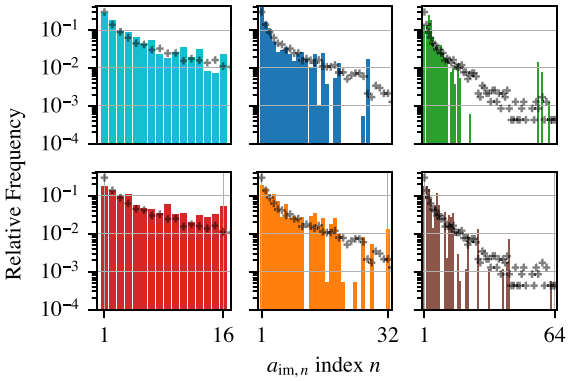
Fig. 4 Cumulative reward of RL agents compared to MILP solution (≤1 hour) and upper bound over request densities and distributions.

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

[§§]City location data from simplemaps.com [retrieved 13 September 2024].



a) Uniform evaluation environment



b) City evaluation environment

Fig. 5 Relative frequency of policy actions compared to equivalent MILP solution actions.

actions correspond to a request that is closer along-track, low-index actions tend to be selected more often. In the city request distribution, high action indices tend to be selected more often due to the clustering of requests, which sometimes leads to a large number of immediately upcoming requests that are infeasible to slew to that must be skipped.

When evaluated in new instances of the uniform request distribution environment, consisting of request sets not seen in training, all of the RL agents perform very well. At high request densities, every agent outperforms the MILP solver's 1-hour, open-loop solution using only about 10 ms of computation per decision in a closed-loop manner. Even when compared to the upper bound for cumulative reward found by the MILP solver, which is a conservative bound, the RL agents collected 80% of the bound. Across lower request densities, the best agents never collect less than 90% of the MILP solution or bound.

The city-distributed evaluation environment is more challenging for the RL agents due to some periods with a high number of conflicting requests. Performance is better on lower request densities than higher densities, with many agents collecting 80% or more of the optimal reward. At high densities, the best agents, which were trained in the city environment, collect 77% of the MILP solution and 68% of the upper bound.

Agent performance is affected by N, as seen in Fig. 4. If N is too small, necessary information and optimal actions may be truncated from their respective spaces; this is visible in Fig. 5, which gives the relative frequency of each action executed by the policy. The MILP solution is seen to select actions up to $a_{\mathrm{im},32}$ in the uniform environment and above $a_{\mathrm{im},64}$ in the city environment; the agents trained to consider only $N = 16$ upcoming requests are thus unable to mimic the selections of the MILP solution, particularly in the city case. However, too large of an N can lead to extraneous information in a dimensionally large observation space, which impacts the ability of the agent to learn, leading to depressed cumulative rewards in some $N = 64$ cases. In general, the $N = 32$ cases strike a balance between the competing effects.

The selection of training request distribution is a less significant factor in overall performance. For both the uniform and city request cases, the total number of requests is randomized each episode in training. As a result, the agent experiences a large range of local densities, allowing for generalization to cases where the request density is nonhomogeneous. Since the S3 observation parameterizes request information to be relative to the agent, the fact that the city requests do not cover the entire domain of Earth's surface does not significantly impact the ability of the agent to generalize to a globally distributed set of requests because the agent has learned throughout the entire relative domain. Still, there is a slight preference for the agents to be deployed in the same distribution that they were trained with.

2. *Observation Parameterization*

The second ablation study compares the performance of agents trained with different observation parameterizations. An agent is trained for 24 hours on a uniform request distribution with $N = 32$ for each S1, S2, and S3 observation given in Table 1. Observation S1 gives agent attitude and request locations in the planet-fixed frame, while S2 and S3 give those values in the agent-relative Hill frame. S3 also includes additional information about the angles to requests and access times. Notably, the observations are computable from each other's elements, so this study tests whether different representations of the same information can improve the agent's learning.

The training curve for each agent is shown in Fig. 6. The S3 observation outperforms the S1 and S2 observations in training, with the S1 observation performing about 20% worse than S3. Since the S3 observation more explicitly provides information needed to find a good policy, the network does not need to learn to extract that information from the state space. As a result, the expressivity of the policy network can be used more directly on tasking as opposed to implicitly learning the computations performed in S3.

3. *Analysis of Best Policies*

A closer analysis of the best policy for each request distribution is performed on a 10-orbit horizon, shown in Fig. 7. The best policies use $N = 32$ with the S3 observation trained on the uniform distribution for uniform evaluation and on the city distribution for city evaluation. These results are compared to the MILP-based solution.

For both request distributions, the fulfillment rate is lower, but the average fulfilled request priority is higher for the RL agents compared to the MILP solution. Since the overall performance is similar, this implies that a range of strategies for the problem can be successful. It follows that the RL agents converge on a strategy that requires less optimization of long sequences of lower priority requests if a more direct policy can achieve similar value.

Looking at the success rate of the policies—i.e., what fraction of the attempted actions are completed successfully—is also elucidating. Actions will fail if the selected request cannot be slewed to

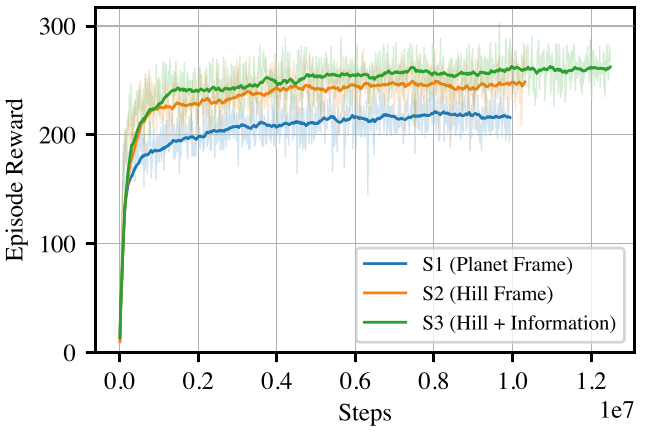
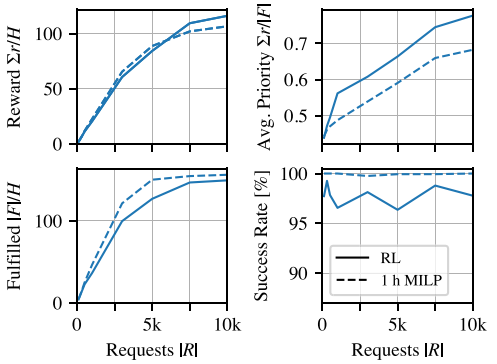
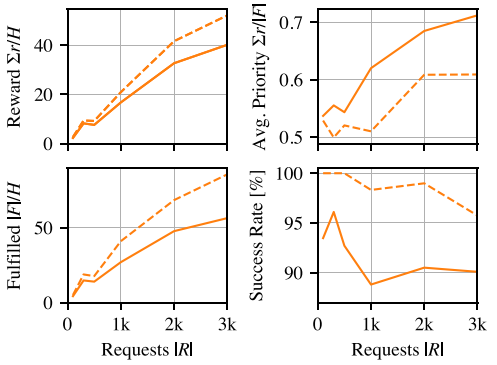


Fig. 6 Observation ablation study training curves.



a) Uniform policy and evaluation environment



b) City policy and evaluation environment

Fig. 7 Best policy for each distribution over a range of request counts.

before it is out of range. Compared to the MILP solution, which only misses requests in cases where the simulated dynamics differ too much from the model, the RL agents miss requests at a higher rate. This is not inherently bad because there is no explicit penalty for an unsuccessful action other than the opportunity cost, but it does imply room for improvement. This is particularly evident in the city distribution, where the tight clustering of requests causes even the MILP solver to have a lower success rate. In high-density instances of this case, the RL agent misses as many as 10% of all attempted requests.

V. Results: Power-Constrained Environment

A. Simulation Environment

The power-constrained environment uses the same simulation parameters as the sequencing environment (Table 2) with the addition of the power parameters in Table 3. Power generation using the solar panels is modeled based on solar incidence angle due to the spacecraft attitude and eclipse state. The power draw of the satellite is a function of the reaction wheel speeds (with η_{rw} representing the electrical-to-mechanical power conversion efficiency) and, when imaging, the instrument power draw. As with the power-free environment, the power-constrained environment can be found in the BSK-RL repository.

Table 3 Parameters for the power subsystem power-constrained environment

	Parameter	Value(s)
Power	$z_{\max}$	120 W · h
	z_0	$[0.4, 1.0] \times z_{\max}$
	$\dot{z}_{\text{base}}$	20 W
	$\dot{z}_{\text{im}}$	10 W
	η_{rw}	0.5

B. Experiments

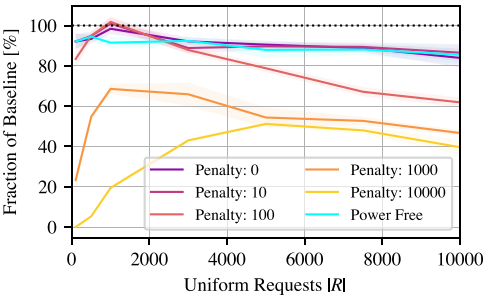
For each distribution of requests, five policies are trained with different failure penalties in the power-constrained environment. Failure penalties are a negative reward returned when the agent enters a failure state, i.e., when the battery level reaches zero. The value of this penalty must be tuned, like any other hyperparameter.

Following from prior results, it is assumed that the distribution of requests in the deployment environment is sufficiently well known to be able to train policies on the same distribution over a range of densities. This ensures that the power-constrained results are only considering the impact of the power subsystem on the agent's performance, not the impact of transferring between different request distributions.

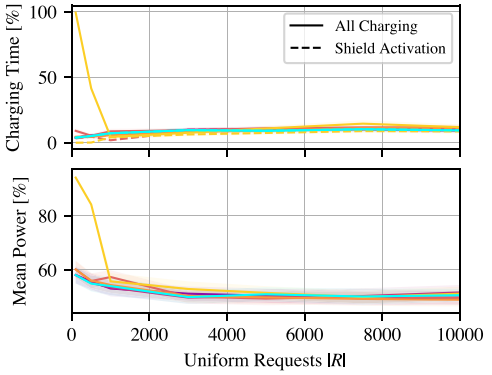
In testing, the policies trained with different failure penalties are evaluated with the shield given in Eq. (16) to ensure safety. An additional policy is trained in a power-free environment and deployed in the power-constrained environment with a shield; this provides a lower baseline for comparison, i.e., answering the question of how much better the other policies can perform if they are actively aware of power constraints. The results are compared to the performance of a power-free policy in a power-free environment with the same request list, such that statistics are given relative to an agent that has no concern about power. Results are given in Figs. 8 and 9.

Considering the uniform request distribution (Fig. 8), there is no intuitive way for the satellite to exploit the environment. Because requests are uniformly distributed, there is a low probability of a single 5-min period having significantly less reward available than any other period. As a result, the shielded power-free policy performs nearly as well as the shielded power-constrained policies. An exception to this trend is in the lower request density regime (e.g., 500–3000 global requests), in which there are periods of low reward availability that the power-constrained policies proactively choose to charge in, as reflected by higher rewards. While training curves show that the power-constrained policies did tend to stay alive, they have learned to do so in a less conservative manner than the shield dictates, as most of their charge actions are forced by the shield on the uniform distribution.

Compared to the uniform distribution of requests, the city distribution is more exploitable. The city distribution has periods

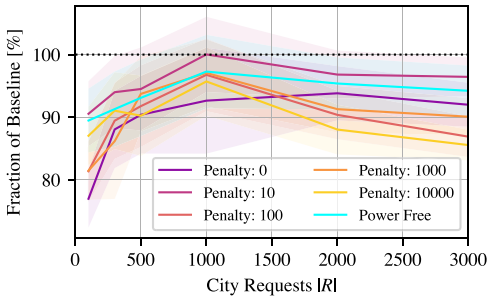


a) Total reward relative to power-free baseline

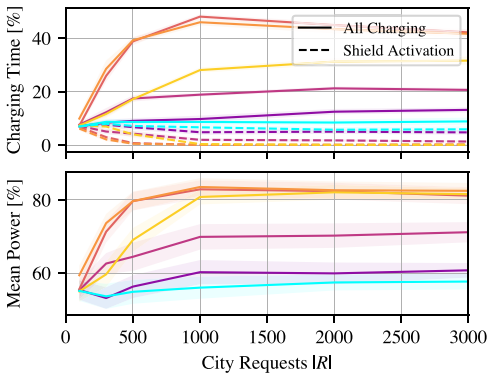


b) Power subsystem statistics

Fig. 8 Benchmark on uniform request distribution.



a) Total reward relative to power-free baseline



b) Power subsystem statistics

Fig. 9 Benchmark on city request distribution.

of high and low reward availability; while the shield has no awareness of rewards that may be neglected by charging, the power-constrained policies can learn to charge only in periods of low/no reward availability (e.g., over oceans and other sparsely inhabited areas). In Fig. 9, the best power-constrained policy is shown to always outperform the shielded power-free policy; it often achieves performance close to the power-free baseline, which indicates that it has learned to charge in periods that never cause it to lose potential rewards. The power-constrained policies also maintain the spacecraft in a safer state, as reflected by higher average charge levels over the course of the simulations. While the shield only tries to prevent failure, the RL-based solutions attempt to maximize success, which includes maintaining a higher charge level to remain farther from failure states.

VI. Conclusions

RL-based policies using sMDPs are competitive with optimal solutions to the AEOSSP; these policies have the benefits of being closed-loop and executable onboard as opposed to computationally expensive, open-loop schedules. When compared in a high-fidelity simulation environment across a range of request densities and distributions, policies trained with RL performed between optimal and 22% below the optimal solution bound found using a computationally expensive MILP solver. In some cases, the policy found superior solutions with real-time closed-loop planning than the preplanner could find in an hour of computation.

Practical considerations for training agents for the AEOSSP are also discussed. The AEOSSP is identified as a suitable candidate for the sMDP framework, and tests confirm that leveraging knowledge of the step duration improves the performance of the agent. Additionally, the choice of observation space and request lookahead distance are shown to have a significant impact on the performance of the agent.

An advantage of RL over other planning abstractions is demonstrated: implementing arbitrary resource constraints on the problem is relatively easy. While certain constraints can be challenging to fully implement in other optimization frameworks, RL can be trained with awareness of these constraints as long as they are present in the simulator. When combined with a shield to guarantee operational safety, the policy is shown to be able to leverage its understanding of safety considerations to minimize the impact of a power constraint on performance towards the objective by entering a charging state when fewer rewards are available.

This work demonstrates that RL-based policies can produce nearly optimal solutions to the AEOSSP onboard, providing a concrete advancement toward autonomous Earth-observing satellite operations. The operational costs and practical delays associated with ground-based planning are avoided, reducing the complexity of operations to uplinking new requests to the satellite as they become available. The optimization framework is more flexible than planners that require expert-informed abstractions of the system into a planning framework, instead only requiring a high-fidelity simulator of the spacecraft's capabilities and constraints to use as a learning environment. These conclusions suggest that RL should be considered a viable tool for autonomous EOS mission operations.

Acknowledgments

This work is supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) grant, 80NSSC23K1182. This work is also 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] Selva, D., and Krejci, D., "A Survey and Assessment of the Capabilities of Cubesats for Earth Observation," *Acta Astronautica*, Vol. 74, May 2012, pp. 50–68. https://doi.org/10.1016/j.actaastro.2011.12.014

[2] 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

[3] Gabrel, V., Moulet, A., Murat, C., and Paschos, V. T., "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. https://doi.org/10.1023/A:1018920709696

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

[5] Applegate, D. L., Bixby, R. E., Chvatál, V., and Cook, W. J., *The Traveling Salesman Problem: A Computational Study*, Princeton Univ. Press, Princeton, NJ, 2006, pp. 1–58.

[6] Peng, G., Dewil, R., Verbeeck, C., Gunawan, A., Xing, L., and Vansteenwegen, P., "Agile Earth Observation Satellite Scheduling: An Orienteering Problem with Time-Dependent Profits and Travel Times," *Computers & Operations Research*, Vol. 111, Nov. 2019, pp. 84–98. https://doi.org/10.1016/j.cor.2019.05.030

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

[8] Nag, S., Li, A. S., and Merrick, J. H., "Scheduling Algorithms for Rapid Imaging Using Agile Cubesat Constellations," *Advances in Space Research*, Vol. 61, No. 3, 2018, pp. 891–913. https://doi.org/10.1016/j.asr.2017.11.010

[9] Kim, J., Ahn, J., Choi, H.-L., and Cho, D.-H., "Task Scheduling of Agile Satellites with Transition Time and Stereoscopic Imaging Constraints," *Journal of Aerospace Information Systems*, Vol. 17, No. 6, 2020, pp. 285–293. https://doi.org/10.2514/1.I010775

[10] 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
[11] Stephenson, M., and Schaub, H., "Optimal Agile Satellite Target Scheduling with Learned Dynamics," *Journal of Spacecraft and Rockets*, Oct. 2024.
https://doi.org/10.2514/1.A36097
[12] Picard, G., Caron, C., Farges, J.-L., Guerra, J., Pralet, C., and Roussel, S., "Autonomous Agents and Multiagent Systems Challenges in Earth Observation Satellite Constellations," *International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2021), International Foundation for Autonomous Agents and Multiagent Systems*, International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2021, pp. 39–44.
https://doi.org/10.5555/3463952.3463961
[13] Tangpattanakul, P., Jozefowiez, 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
[14] Viros Martin, A., Cheng, K., Fang, A., Zheng, Z., Kress-Gazit, H., Mehta, A., Selva, D., and Sun, Y., "Decentralized Context-Based On-Board Planning for Earth Observation Missions," *AIAA Scitech 2021 Forum*, AIAA Paper 2021-1469, 2021.
https://doi.org/10.2514/6.2021-1469
[15] 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
[16] Sutton, R. S., and Barto, A., "Reinforcement Learning: An Introduction," *Adaptive Computation and Machine Learning*, 2nd ed., MIT Press, Cambridge, MA, 2018, pp. 47–72.
[17] Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M., "Reinforcement Learning for Solving the Vehicle Routing Problem," *NeurIPS*, Advances in Neural Information Processing, Curran Associates, Red Hook, NY, 2018.
https://doi.org/10.48550/arXiv.1802.04240
[18] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, AAS Paper 2019-0447, 2019.
[19] Hadj-Salah, A., Verdier, R., Caron, C., Picard, M., and Capelle, M., "Schedule Earth Observation Satellites with Deep Reinforcement Learning," *International Workshop on Planning & Scheduling for Space*, Nov. 2019.
https://doi.org/10.48550/arXiv.1911.05696
[20] Eddy, D., and Kochenderfer, M., "Markov Decision Processes For Multi-Objective Satellite Task Planning," *2020 IEEE Aerospace Conference*, Inst. of Electrical and Electronics Engineers, New York, 2020, pp. 1–12.
https://doi.org/10.1109/AERO47225.2020.9172258
[21] 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
[22] Herrmann, A., and Schaub, H., "Reinforcement Learning for the Agile Earth-Observing Satellite Scheduling Problem," *IEEE Transactions on Aerospace and Electronic Systems*, Vol. 59, No. 5, 2023, pp. 1–13.
https://doi.org/10.1109/TAES.2023.3251307
[23] 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, pp. 1–19.
https://doi.org/10.3389/frspt.2023.1263489
[24] Reed, R., Schaub, H., and Lahijanian, M., "Shielded Deep Reinforcement Learning for Complex Spacecraft Tasking," *American Control Conference*, Inst. of Electrical and Electronics Engineers, New York, 2024, pp. 2331–2337.
https://doi.org/10.23919/ACC60939.2024.10644855
[25] 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
[26] Chun, J., Yang, W., Liu, X., Wu, G., He, L., and Xing, L., "Deep Reinforcement Learning for the Agile Earth Observation Satellite Scheduling Problem," *Mathematics*, Vol. 11, No. 19, 2023, p. 4059.
https://doi.org/10.3390/math11194059
[27] Garcia Buzzi, P., Selva, D., and Sanchez Net, M., "Deep Q-lLearning for Delay Tolerant Network Management in a Reactive Imaging Earth Observing Constellation," *ASCEND*, AIAA Paper 2021-4041, 2021.
https://doi.org/10.2514/6.2021-4041
[28] Piccinin, M., Lunghi, P., and Lavagna, M., "Deep Reinforcement Learning-Based Policy for Autonomous Imaging Planning of Small Celestial Bodies Mapping," *Aerospace Science and Technology*, Vol. 120, Jan. 2022, Paper 107224.
https://doi.org/10.1016/j.ast.2021.107224
[29] Bradtke, S., and Duff, M., "Reinforcement Learning Methods for Continuous-Time Markov Decision Problems," *Advances in Neural Information Processing Systems*, Vol. 7, MIT Press, Cambridge, MA, 1994, pp. 393–400.
https://doi.org/10.5555/2998687.2998736
[30] Sutton, R. S., Precup, D., and Singh, S., "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning," *Artificial Intelligence*, Vol. 112, Nos. 1–2, 1999, pp. 181–211.
https://doi.org/10.1016/S0004-3702(99)00052-1
[31] Menda, K., Chen, Y.-C., Grana, J., Bono, J. W., Tracey, B. D., Kochenderfer, M. J., and Wolpert, D., "Deep Reinforcement Learning for Event-Driven Multi-Agent Decision Processes," *IEEE Transactions on Intelligent Transportation Systems*, Vol. 20, No. 4, 2019, pp. 1259–1268.
https://doi.org/10.1109/TITS.2018.2848264
[32] 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
[33] Schaub, H., and Piggott, S., "Speed-Constrained Three-Axes Attitude Control Using Kinematic Steering," *Acta Astronautica*, Vol. 147, June 2018, pp. 1–8.
https://doi.org/10.1016/j.actaastro.2018.03.022
[34] Towers, M., Terry, J. K., Kwiatkowski, A., Balis, J. U., de Cola, G., Deleu, T., Goulão, M., Kallinteris, A., A., K. G., Krimmel, M., et al., "Gymnasium," Oct. 2023.
[35] 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
[36] Brandonisio, A., Capra, L., and Lavagna, M., "Deep Reinforcement Learning Spacecraft Guidance with State Uncertainty for Autonomous Shape Reconstruction of Uncooperative Target," *Advances in Space Research*, Vol. 73, No. 11, 2023, p. S0273117723005276.
https://doi.org/10.1016/j.asr.2023.07.007
[37] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O., "Proximal Policy Optimization Algorithms," Aug. 2017.
https://doi.org/10.48550/arXiv.1707.06347
[38] 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," *Proceedings of the 35th International Conference on Machine Learning*, Vol. 80, Proceedings of Machine Learning Research, 2018, pp. 3053–3062.
https://doi.org/10.48550/arXiv.1712.09381
[39] Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P., "High-Dimensional Continuous Control Using Generalized Advantage Estimation," *International Conference on Learning Representations*, Curran Associates, Red Hook, NY, 2018.
https://doi.org/10.48550/arXiv.1506.02438
[40] 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, AAAI Press, Washington, D.C., 2018.
https://doi.org/10.1609/aaai.v32i1.11797

M. Kochenderfer
Associate Editor