JOURNAL OF SPACECRAFT AND ROCKETS
Vol. 62, No. 3, May–June 2025

Optimal Agile Satellite Target Scheduling with Learned Dynamics

Mark A. Stephenson* and Hanspeter Schaub†
University of Colorado, Boulder, Boulder, Colorado 80303

https://doi.org/10.2514/1.A36097

Target sequencing is an important aspect of agile Earth-observing satellite scheduling. The objective of the problem is to find a feasible imaging sequence that maximizes the cumulative value of unique heterogeneously valued requests; no expendable resource constraints (power and data storage) are considered. Two challenges are encountered: transition times between targets are dictated by dynamics of an arbitrary controller, and the solution space is combinatorial with respect to request count. To find dynamically accurate time-dependent attitude maneuver transition times, a neural network is trained on simulated slew data; this allows for the generation of more physically accurate, and thus better performing, schedules than when using the traditional approach of lower-order models for slew feasibility. Next, slews between requests are represented by a sparsified graph, and a mixed-integer linear program is formulated to solve them; the sparse formulation keeps the problem tractable over longer planning horizons and larger numbers of requests when compared to the standard approach. The solutions are optimal up to the quality of the transition time estimator and a time discretization. Finally, the solutions are verified in a high-fidelity simulation, demonstrating validity when deployed on a lifelike system. Time-dependent request values and multiple satellites are also considered.

Nomenclature

$\hat{\boldsymbol{c}}$	=	instrument boresight direction
E	=	set of graph edges
e	=	graph edge
F	=	set of fulfilled requests
H	=	horizon length (orbits)
N	=	number of satellites
O	=	request opportunities
o	=	request opportunity with interval $[t^o, t^c]$
p	=	partition of vertices
R	=	set of all requests
$\boldsymbol{r}$	=	request location
r	=	request value
S	=	sequence of request visitations
t	=	time
U	=	set of unfulfilled requests
V	=	set of graph vertices
v	=	request visitation or graph vertex corresponding to a time-request pair
W	=	set of graph edge weights
w	=	graph edge weight
$\boldsymbol{x}$	=	satellite trajectory
x	=	vertex visitation decision variable
y	=	request-fulfillment-decision variable
z	=	transition-credit-decision variable
Δt	=	slew duration
δt	=	time discretization
$\boldsymbol{\theta}$	=	network parameters
θ	=	angle between instrument and request directions
ρ	=	request value-location tuple
$\boldsymbol{\sigma}$	=	satellite attitude
τ	=	loss skewness parameter
ϕ	=	elevation angle
$\boldsymbol{\omega}$	=	body rotation rate

Subscripts

$i(, j)$	=	vertex index(es)
n	=	request index dk

Superscripts

$\mathcal{B}$	=	body-fixed frame
$\mathcal{P}$	=	planet-fixed frame
s	=	satellite index

I. Introduction

THE agile Earth-observing satellite scheduling problem (AEOSSP) aims to maximize the quantity and value of requests fulfilled by an orbiting imaging satellite while managing resources [1]. As compared to a standard Earth-observing satellite that can only slew across-track to capture off-track targets, "agile" indicates that the satellite has three axes of control, giving it a larger field of regard by allowing requests to be imaged along-track. The transition time between two targets is thus both target- and time-dependent, as the satellite's initial and target attitude depends on what time what request is being imaged. The reward obtained from imaging can also be time dependent, if factors such as time of day or squint angle strongly impact image quality. Particularly in environments with a high density of requests, finding an optimal request sequence is both challenging and rewarding from a mission-objective perspective.

Earth-observing satellites are used to collect many types of data, for example, water and soil composition (NASA's Earth Observer 1 [2]), wildfire and flood monitoring (European Space Agency's Sentinel-2 [3]), and on-demand image requests from commercial, scientific, and defense customers (Centre National d'Études Spatiales's Pleiades [4] and Planet's Dove constellation [5]). In all of these cases, optimally scheduling observations increases the volume of data delivered by the satellite or constellation. In cases where certain observations are more critical, such as for a shared resource like Pleiades or when rare phenomena are observable, scheduling that optimizes for request priority is important.

Graph-based representations of the AEOSSP, in which imaging actions are represented by vertices, and feasible slews between targets are represented by edges, are common in the literature. Gabrel

Presented as Paper AAS 23-108 at the AAS/AIAA Astrodynamics Specialist Conference, Big Sky, Montana, August 13–17, 2023; received 16 April 2024; revision received 30 August 2024; accepted for publication 22 September 2024; published online 25 October 2024.

*Ph.D. Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, 3775 Discovery Drive, Colorado Center for Astrodynamics Research, Boulder, CO 80303; Mark.A.Stephenson@colorado.edu. Member AIAA (Corresponding Author).

†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, Boulder, CO 80303. Fellow AIAA.

et al. [6] offers an early graph-based treatment of the problem that assumes a simple transition-time model and is computationally limited to small problem instances; additionally, unlike more recent work, off-the-shelf optimization frameworks are not used. Augenstein [7] leverages the directed acyclic properties of graphs representing this problem. Augenstein's formulation treats each imaging event as a single vertex (as opposed to multiple vertices, each representing a different imaging time within an opportunity window). As a result, Augenstein is able to use a simple dynamic programming algorithm for optimization. Eddy and Kochenderfer's [8] exploration of efficient planning for constellations takes a similar approach to single-point request representation, focusing more on the deconfliction of requests between satellites than on individual agile imaging optimization. Peng et al. [9] briefly considers a graph-based method that uses multiple vertices per observation window, as in this work; this introduces challenges that prevent the use of simpler dynamic programming solutions for path maximization such as those used by Augenstein and necessitates the use of other combinatorial optimization methods.

Iterative local search and genetic algorithms are good at quickly finding near-optimal solutions to highly combinatorial problems such as the AEOSSP, although they lack the optimality guarantees of mixed-integer programs (MIPs). Lemaître et al. [10] offers one of the first local search approaches to an alternate formulation of the problem that considers strip imaging instead of point imaging, in comparison with other methods. Dilkina and Havens [11] compare a variety of early approaches, including iterative local search (ILS) and genetic methods, to the problem for a single satellite over small (6 h, ∼1000 request) test cases. Mao et al. [12] gives another genetic-algorithm-based method for multi-objective optimization but does not report on specific-scenario performance. More recently, Verbeeck et al.'s [13] advances in local search for time-dependent orienteering problems (a class that includes the AEOSSP) have led directly to the development of performant ILS-based AEOS solvers by Liu et al. [14] and Peng et al. [9]. These two papers consider both time-dependent transition times and time-dependent rewards, planning in up to medium-sized environments with hundreds of targets over a day-long period in tens to hundreds of seconds.

MIP and mixed-integer linear program (MILP) solutions to the AEOSSP offer an alternative method that quantifies solution suboptimality and, with enough time, finds and certifies optimality. As such, they can both be used practically as a planning algorithm or for benchmarking the performance of other methods relative to optimality. Peng [9] formulates the single satellite problem as a MIP to have an optimal comparison for their local search algorithm that considers time-dependent transitions and rewards but finds that the formulation is unable to find a solution in a reasonable amount of time or without running out of memory for anything but the smallest instances. Their MILP formulation is comparable to the MILP-D formulation described in this paper. Cho et al. [15], Chen et al. [16], Kim et al. [17], and Wang et al. [18] use MILP formulations for constellation-wide image and downlink scheduling under various constraints, each using heuristics to initialize the solver for improved speed, as multisatellite problems can quickly become intractably large. Cho et al. include continuous models for power and data, and models up to 12 satellites for up to 2 days' planning with up to 700 tasks; reasonable, but suboptimal, solutions are found within 10,000 s but are not evaluated in a secondary simulation environment. Chen et al. introduce continuous variables to more effectively schedule overlapping requests, considering up to 1000. Kim et al. only consider up to 100 tasks, although include complex task specifications such as stereoscopic imaging. Wang et al. introduces cloud uncertainty and optimizes over reward and riskiness for up to 300 requests. In general, the literature tends to consider at most a few hundred to a thousand targets over a few orbits to a day.

The treatment of transition times in existing literature tends to be simplified compared to the realities of spacecraft dynamics when transitioning from tracking one target to another. The most accurate (but most expensive) approach is to execute a dynamics model to test each transition, as proposed by [7]. Another approach is to use a low-order linear model that considers only the difference in roll angle [15,17] or the total angle change [9,14], or other unspecified abstract transition constraints [16,17]; although these methods are convenient for planning problems, they do not fully capture the nonlinear nature of attitude transitions. An ablation study in this paper demonstrates how lower-order models fall short. Nag et al. [19] use a discretized model of transitions among various attitudes; this captures more of the dimensions that impact transition dynamics but is still a lower-order and lower-dimensional model.

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 [20]. Parjan and Chien [21] address this with a decentralized approach that combines individual heuristic search with broadcasting to iteratively deconflict tasks. In the case of a sufficiently large constellation, Xu et al. [22] demonstrate that the optimization problem becomes one of achieving the best possible continuous coverage rather than considering individual target scheduling.

A newer approach is reinforcement learning (RL), which is a broadly applicable framework for autonomy that provides three primary benefits: low computational cost, closed-loop planning, and the ability to include any constraints that can be modeled [23]. Nazari et al. demonstrates that RL is effective at solving general waypoint routing problems [24]. Applied directly to spacecraft tasking, Harris et al. [25], Hadj-Salah et al. [26], Eddy and Kochenderfer [27], and other recent work [28] formulate various Markov decision processes (MDPs) for the AEOSSP. References [25,26] use simplified probabilistic models for the spacecraft dynamics. In [25] and later [29–31], Harris et al. and Herrmann and Schaub develop the problem with a full-fidelity simulation and utilize shields to ensure operational safety. Zhao et al. consider target scheduling using two phases of RL, the first selecting observation windows and the second picking observation times within the windows [32]. MDP representations of other spacecraft- tasking problems, such as small-body imaging, have also been formulated and solved with RL [33,34]. These methods represent an emerging alternative to preplanning methods but are still maturing and are less directly comparable to the other classes of approaches.

In this paper, novel developments are made toward accurately representing and efficiently solving the target sequencing portion of the AEOSSP (i.e., without resource management). First, time-dependent transition times between requests are modeled using a neural-network function approximator, a significant improvement in accurately expressing dynamics over the low-order models for transition times. This improvement in transition-time estimation translates to better-performing plans than when using common lower-order models because neither is time wasted due to an over-conservative estimator nor are requests missed due to an overly aggressive estimator. The structure of the problem is represented as a sparse graph that accounts for different imaging times for each request, and an efficient MILP formulation is developed for the sparsified graph that maintains optimality guarantees. Compared to the standard dense graph formulation, the sparse formulation is faster, more memory-efficient, and more scalable to larger problems, remaining tractable for thousands of requests. Formulations for time-independent and -dependent rewards are given, allowing for the best performance on a specific problem type. Solutions to the planning problem are evaluated in a full-fidelity spacecraft simulator, demonstrating end-to-end performance of the proposed method, as opposed to only analyzing the solution quality in the abstract planning space.

II. Problem Formulation

A. Optimization Objective

The objective of the agile multisatellite target sequencing problem is to determine a feasible sequence of request visitations $S^s = v_1^s, v_2^s, \ldots$ for each fixed-orbit satellite that maximizes the sum of imaging values of the requests satisfied. A request visitation $v_i^s = (t_i^s, \rho_i^s)$ for satellite s is a tuple of a visitation time and request, and a request $\rho_i^s = (r_i^s, \boldsymbol{r}_i^s(s,t))$ consists of a ground location and value function:

$$\underset{\{S^1,\ldots,S^N\}}{\arg\max} \quad \sum_{s=1}^{N}\sum_{i=1}^{|S^s|} r_i^s(s, t_i^s) \tag{1}$$

$$\text{subject to} \quad \Delta t^s(v_i^s, \boldsymbol{r}_{i+1}^s) \le t_{i+1}^s - t_i^s \quad \forall\ v_i^s, v_{i+1}^s \tag{2}$$

$$\rho_i^n \ne \rho_j^m \quad \forall \quad i \ne j \vee n \ne m \tag{3}$$

That is, find the sequences that maximize the sum of values at imaging time for each imaged target [Eq. (1)] while ensuring that there is sufficient time for each satellite to transition between requests [Eq. (2)] and while only imaging each target at most once [Eq. (3)].

B. Request-Value Model

Earth-observation models can be broadly classified into two categories: continuous imaging, in which the satellite scans a strip of the planet's surface; and point-based imaging, in which the satellite aims at a point target and collects a single, near-instantaneous image. Request models can be similarly classified: area requests, for which the operator specifies a region of interest; and point requests, for which individual point targets are identified. Eddy and Kochenderfer [8] describe that area requests can be decomposed into point targets, making the latter distinction less consequential. In this work, point-based observation and request models are considered.

Requests $\rho_n \in R$ take the form of a location-value tuple $\rho_n = (\boldsymbol{r}_n, r_n(s,t))$. For each satellite, each request has a set of opportunities $o \in O_n^s$, where o is an interval for which all imaging requirements are satisfied; these may include constraints on range, elevation, time of day, or other factors. Opportunities are known a priori due to the fixed nature of the satellite's orbit. For requests with time-independent value, the value is a constant r_n^{const} during opportunity and zero otherwise:

$$r_n(s,t) = \begin{cases} r_n^{\text{const}} & \text{if } t \in o \in O_n^s \\ 0 & \text{else} \end{cases} \tag{4}$$

If value is time-dependent,

$$r_n(s,t) \equiv \begin{cases} r_n^{t.d.}(s,t) & \text{if } t \in o \in O_n^s \\ 0 & \text{else} \end{cases} \tag{5}$$

where $r_n^{t.d.}$ may be a function of any time-dependent values, such as ϕ or illumination angle.

In many applications, it is undesirable to repeatedly fulfill the same request. Thus, requests are categorized as unfulfilled U or fulfilled F. All requests start in the unfulfilled set, $R = U$. If ρ_n is fulfilled, ρ_n is moved from U to F. The no-reimaging constraint [Eq. (3)] can be alternatively expressed using an alternative request-value function in the underlying model of the environment:

$$r_n^{\text{unique}}(s,t) = \begin{cases} r_n(s,t) & \text{if } \rho_n \in U \\ 0 & \text{if } \rho_n \in F \end{cases} \tag{6}$$

C. Fulfillment Dynamics

To fulfill ρ_n, the satellite must satisfy collect requirements for ρ_n at t. For the target to be collected, the sensor boresight direction $\hat{\boldsymbol{c}}$ must point in the target direction

$$\hat{\boldsymbol{c}}^{\text{ref}}(t; s, n) = \frac{\boldsymbol{r}_n - \boldsymbol{x}^s(t)}{|\boldsymbol{r}_n - \boldsymbol{x}^s(t)|} \tag{7}$$

within a threshold

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

This geometry is shown in Fig. 1. The satellite must be settled such that the angular rate of the boresight relative to the target is minimal; the reference angular velocity between the spacecraft frame $\mathcal{B}$ and planet frame $\mathcal{P}$ for tracking the target is

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

assuming zero about-boresight rotation. To collect the image, the body rate must be within a threshold of the reference

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

If Eqs. (8) and (10) are satisfied at t for some s and n, the request may be fulfilled and yield value r_n.

D. Transition Dynamics

To execute v_j, the satellite must be able to transition from its state at t_i to a state that satisfies the collect requirements for ρ_j at t_j. The slew transition time must be less than the time until the planned request visitation:

$$\Delta t^s(\boldsymbol{\sigma}_i, \boldsymbol{\omega}_{\mathcal{B}/\mathcal{P},i}, \boldsymbol{x}^s(t), \boldsymbol{r}_j) \le t_j - t_i \tag{11}$$

It is reasonably assumed that a satellite is always capable of tracking the target once it has settled on the target; if thresholds are met before the desired collect time, the satellite can track the target in a settled state until the chosen time is met.

The function Δt^s is challenging, if not impossible, to find analytically, due to the time-varying tracking reference and is highly dependent on the attitude control law implemented on the satellite. By imposing a few assumptions, the function gains properties that are useful when applied to the planning task:

1) The attitude controller is deterministic.

2) The controller performs identically, regardless of rotation about the boresight axis (e.g., an inertia-compensated controller that slews in an axis-agnostic manner) *or* the controller maintains a fixed about-boresight rotation relative to the trajectory (e.g., a controller that maintains zero yaw in the Hill frame).

3) As previously stated, the satellite's position trajectory is a known fixed orbit.

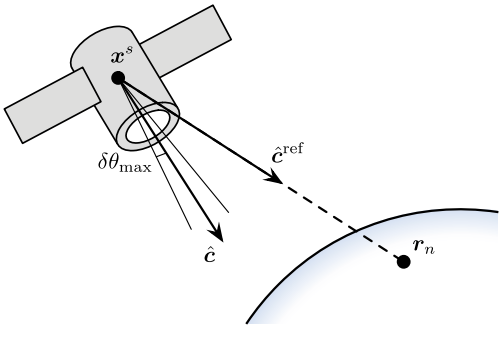

Fig. 1 Pointing directions, references, and thresholds.

With these assumptions, Eq. (11) reduces to

$$\Delta t^s(\hat{\boldsymbol{c}}_i, \boldsymbol{\omega}_{\mathcal{B}/\mathcal{P},i}, t_i, \boldsymbol{r}_j) \leq t_j - t_i \tag{12}$$

Additionally, there is a bijective mapping between an imaging action and upcoming target tuple $(v_i, \boldsymbol{r}_j)$ and the arguments to the transition time function in Eq. (12) via Eqs. (7) and (9). As a result, the transition feasibility constraint can be expressed directly between two request visitations:

$$\Delta t^s(v_i, v_j) \leq t_j - t_i \tag{13}$$

III. Slew Estimation

The introduction of Eq. (13) hints at methods employed in Sec. IV, namely, checking whether transitions among many v_i and v_j are feasible in order to generate a search space of feasible sequences to find the optimal sequence. However, even with intelligent choices for v_i and v_j, many evaluations of Δt^s are required. Whereas this is trivial for a toy problem that assumes constant or linear transition times, transitions for flightlike systems in general lack an analytical solution for Δt^s; as shown in the results, approximating transitions with a simplified model leads to a loss of optimality on the true system, as underestimates of Δt^s may incorrectly classify infeasible transitions as feasible, whereas overestimates of Δt^s may lead to idle time.

Running a dynamics simulation of every tested transition is prohibitively expensive. As an alternative, a neural network (NN) is trained to estimate Δt^s. This approach allows for a computationally low-cost evaluation of Δt^s for many slews.

A. Network Architecture

The slew estimation network is of the form

$$\Delta\hat{t}^s = \Delta t^s(\hat{\boldsymbol{c}}_0, \boldsymbol{\omega}_{\mathcal{B}/\mathcal{P},0}, \boldsymbol{x}^s(t_0), \frac{^{\mathcal{P}}\mathrm{d}}{\mathrm{d}t}\boldsymbol{x}^s(t_0), \boldsymbol{r}_n; \boldsymbol{\theta}) \tag{14}$$

with all quantities expressed in the $\mathcal{P}$ frame. The initial position and velocity of the satellite are given rather than t_0, allowing the estimator to generalize to any satellite with the same attitude control dynamics, assuming orbit evolves the same in the $\mathcal{P}$ frame over Δt^s for all t. This holds for two-body dynamics with J2 and altitude-based atmospheric drag perturbations; practically, other perturbations are negligible over the short timescales of Δt^s.

The estimator is a feedforward multilayer perceptron (MLP) shown in Fig. 2. To improve the performance of training, domain knowledge is leveraged: the attitude and rate errors [i.e., the left-hand side (LHS) of the inequalities in Eqs. (8) and (10)] are computed from and concatenated with the inputs passed to the MLP. This information is useful for the network because Δt tends to be highly correlated with the attitude and rate errors. The other inputs to the network are still necessary to resolve more subtle nonlinearities, such as those due to planetary rotation or controller saturation.

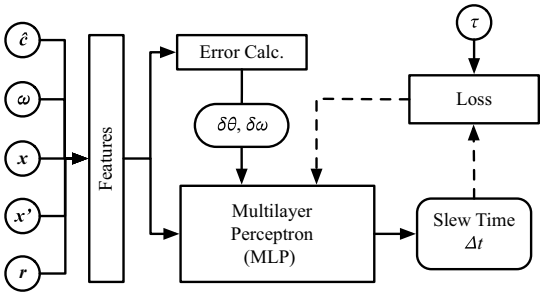

Fig. 2 Slew transition-time estimation network.

B. Training Procedure

To generate training data, slews to random targets nearby are executed in the simulation environment, ignoring opportunity-window limitations. Satellite states that are inputs to the NN are recorded at various points along each slew, including the initial and final states. The remaining duration of the slew from each point is calculated at the end of each slew for use in regression.

A feedforward MLP is trained over the state-time remaining tuples using the Adam optimizer [35]. An asymmetric loss function [36] combining the properties of root mean squared error and "pinball" losses is used:

$$L(t, \hat{t}; \tau) = \begin{cases} (1-\tau)(\hat{t}-t)^2 & \text{if } \hat{t} \geq t \\ \tau(\hat{t}-t)^2 & \text{if } \hat{t} < t \end{cases} \tag{15}$$

The parameter $\tau \in [0, 1]$ can be adjusted to tend toward overestimation of Δt^s (if $\tau > 0.5$), which is desirable for this application. An ablation study over τ is presented in Sec. V.B.

IV. Optimal Agile Satellite Scheduling

A two-step approach is taken to solve the AEOSSP. First, a graph is constructed for each satellite that represents the feasible transitions between requests. The solution space satisfying Eq. (2) is abstracted as an edge-weighted directed acyclic graph. Vertices $v \in V^s$ represent time-request visitation tuples. Visitations with a feasible transition between them are connected by edges $e^s_{i,j} = (v^s_i \to v^s_j) \in E^s$, where each edge has a scalar weight $w^s_{i,j}$ representing the value of fulfilling the request at v_j if it has not yet been fulfilled.

Then, a MILP is formulated to find the value-optimal sequence of requests [Eqs. (1) and (3)] inspired by formulations of the traveling salesman problem [37]. For each edge between vertices v^s_i and v^s_j, a binary variable $x^s_{i,j}$ represents the inclusion or exclusion of the transition in the sequence S^s. Vertices are partitioned by request

$$p_n = \{v^s_i | \rho^s_i = \rho_n\} \tag{16}$$

which allows for the formulation of nonrepetition constraints.

Three variations of the process are presented. A naïve approach, MILP-D(ense), for generating and solving a dense graph of feasible slews using the NN-based Δt^s estimator, is introduced. A modification to reduce the size of the graph, MILP-S(parse), by removing redundant edges, is presented. Finally, a method to account for time-dependent values, MILP-T(ime)D(ependent), is introduced. These methods are summarized in Table 1.

A. MILP-D: Dense

1. Graph Construction

Algorithm 1 describes the process of recursively constructing a graph of all feasible slews for a satellite under the assumption of time-independent request values, shown in Fig. 3. The algorithm begins with a single vertex v^s_0 representing the satellite's initial state as parameterized by the network inputs in Eq. (14). The transition time

Table 1 Comparison of MILP formulation complexities

	Graph $\lvert V\rvert$	Graph $\lvert E\rvert$	Binary variables	Time-dep. rewards
MILP-D	$\mathcal{O}(NH\lvert R\rvert/\delta t)$	$\mathcal{O}(H\lvert R\rvert\lvert V\rvert)$	$\lvert E\rvert \in \mathcal{O}(NH^2\lvert R\rvert^2/\delta t)$	— —
MILP-S	$\mathcal{O}(NH\lvert R\rvert/\delta t)$	$\mathcal{O}(\lvert R\rvert\lvert V\rvert)$	$\lvert R\rvert + \lvert E\rvert \in \mathcal{O}(NH\lvert R\rvert^2/\delta t)$	— —
MILP-TD	$\mathcal{O}(NH\lvert R\rvert/\delta t)$	$\mathcal{O}(\lvert R\rvert\lvert V\rvert/\delta t)$	$2\lvert E\rvert \in \mathcal{O}(NH\lvert R\rvert^2/\delta t^2)$	✓

Algorithm 1 Dense Slew Graph Construction for Satellite s

$V^s, A^s \leftarrow \{v_0^s\}$
$E^s, W^s \leftarrow \{\}$
for $v_i \in A^s$ **do**
  $J = \{\}$
  for $\rho_n \in R$ **where** $\exists o \in O_n^s$ s.t. $o_{\text{close}} > t_i$ **and** $\Delta t^s(v_i, \boldsymbol{r}_n) \leq o_{\text{close}} - t_i$ **do**
    $t_j \leftarrow \text{clamp}(\text{ceil}(t_i + \Delta t^s(v_i, \boldsymbol{r}_n), \delta t)), o_{\text{open}}, o_{\text{close}})$
    $J \leftarrow J \cup \{v_j = (t_j, \rho_n)\}$
  end for
  if sparse graph **then** execute Algorithm 2 **end if**
  for $v_j \in J$ **do**
    if $v_j \notin V$ **then**
      $V^s \leftarrow V^s \cup \{v_j\}$; $A^s \leftarrow A^s \cup \{v_j\}$
    end if
    $E^s \leftarrow E^s \cup \{(v_i \rightarrow v_j)\}$; $W^s \leftarrow W^s \cup \{r_j\}$
  end for
  if time-dependent value **then** execute Algorithm 3 **end if**
  $A^s \leftarrow A^s \setminus \{v_i\}$
end for

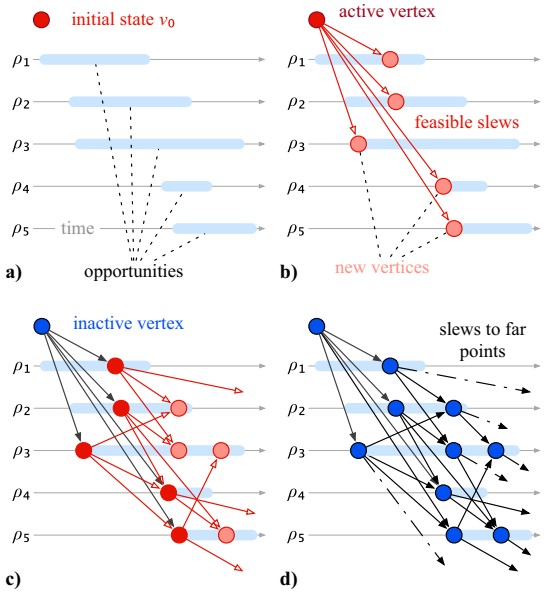

Fig. 3 Dense slew feasibility graph construction using Algorithm 1. a) setup, b) 1 iteration, c) 2 iterations, and d) many iterations.

$\Delta t^s(v_0^s, \cdot)$ is evaluated for all requests with opportunities in the upcoming planning horizon (if possible, leveraging efficient batch-wise evaluation offered by most NN frameworks). If the estimated transition time indicates that the target is reachable before or during an opportunity window, a new vertex is added to the graph at the arrival time, clamped within the opportunity window. Note that only the earliest arrival from some v to some o must be included while preserving the optimality of the solution space for fixed-value targets, because opportunities reachable from ρ_n at t_i are a superset of opportunities reachable from ρ_n at $t_j > t_i$, with earlier departure times yielding the same or earlier arrival times. An edge with weight equal to the incoming target's value is added between the new vertex and the initial vertex. Any new vertices added this way are marked active A. The process is repeated for each active vertex until no new vertices are added. The maximum-weight path starting at v_0^s that avoids target duplication is the optimal sequence of requests to fulfill.

The design of the NN (i.e., such that there exists a bijection between the network inputs and time-request tuples recorded at each vertex) keeps the dimensionality of the graph low while retaining as much information about the dynamics as possible. If one were to include about-boresight rotation or other time-evolving quantities in the vertex information and network inputs, the dimensionality of the graph would increase exponentially with the planning horizon, making the problem intractable. Furthermore, visitation times are discretized to some δt so that edges are formed between existing vertices, allowing the graph to "reconnect" to itself. Without this discretization, $|V|$ would grow exponentially with the planning horizon.

2. *Mixed-Integer Linear Program*

The MILP used to solve the dense graph is straightforward. To enforce nonrepetition, each request partition may only be visited by one satellite at most once. Formally,

$$\text{maximize} \quad \sum_{s=1}^{N} \sum_{i,j} w_{i,j}^s x_{i,j}^s \tag{17}$$

$$\text{subject to} \quad \sum_{h} x_{h,i}^s + b_i^s \geq \sum_{j} x_{i,j}^s \quad \forall\, i \tag{18}$$

$$\sum_{s=1}^{N} \sum_{v_j^s \in p_n} \sum_{i} x_{i,j}^s \leq 1 \quad \forall\, n \tag{19}$$

where

$$b_i^s = \begin{cases} 1 & \text{if } i = i_{\text{start}} \\ 0 & \text{else} \end{cases} \tag{20}$$

The objective function, Eq. (17), aims to maximize the sum of values for traveled edges, recalling that the weight $w_{i,j}^s$ of edge $(v_i^s \rightarrow v_j^s)$ is equal to the reward for visiting v_j^s. The first constraint, Eq. (18), is the in–out constraint. For each vertex, the number of outgoing connections must be less than (at the end of the sequence) or equal to (at all other points) the number of incoming connections. The one exception is at each satellite's initial vertex, where b_i^s is set to 1 to seed the graph. Because the graph is acyclic, the summation on the LHS must be 0, reducing the constraint to

$$1 = \sum_{j} x_{i_{\text{start}},j}^s \tag{21}$$

As a consequence, all other vertices are restricted to one incoming and one outgoing edge selected, producing a feasible sequence. The second constraint, Eq. (19), enforces at most one visitation of each partition, preventing duplicate credit for a single request by ensuring that each request is in U when visited.

Although it would be possible to account time dependence with this formulation by applying the aforementioned method to the graph, this would lead to optimization problems of an unacceptable size. Peng et al. [9] uses an even denser method as an optimal benchmark for time-dependent instances with small target sets, but notes that it becomes too large for the computer's memory in larger instances.

B. MILP-S: Sparse

1. *Graph Construction*

Many of the edges in Algorithm 1 are redundant, as the feasibility of $v_i \rightarrow v_k$ can be inferred from the existence of $v_i \rightarrow v_j \rightarrow v_k$ through the composition of edges. Figure 4 demonstrates the removability of edges from the standpoint of transition feasibility.

Practically, generating the dense graph and deleting removable edges is computationally expensive. The generation of the sparse graph can be approximated if an upper bound on maximum transition time Δt_{max}^s is known by modifying Algorithm 1 with Algorithm 2. For potential vertices J extending from v_i, only the vertices with t_j between the minimum $t_{j,\text{min}}$ and one Δt_{max}^s later need to be added to

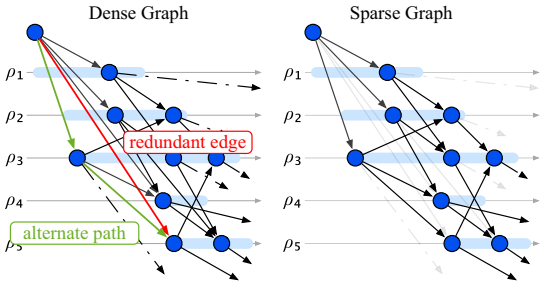

Fig. 4 Comparison between the dense graph with one removable edge highlighted and the sparse graph.

Algorithm 2 Sparse Graph Modification for Algorithm 1.

$t_{j,\min} \leftarrow \min(t_j$ **for** $(t_j, \rho) \in J)$
for $v_j \in J$ **do**
  if $t_j > t_{j,\min} + \Delta t^s_{\max}$ **then**
    $J \leftarrow J \setminus \{v_j\}$
  end if
end for

the graph; any subsequent vertices are guaranteed to be reachable from those in the interval. Although this does not produce the sparsest possible graph, it reduces $|E|$ by a factor of $\mathcal{O}(|R|)$ at low computational cost.

2. *Mixed-Integer Linear Program*

Unlike the preceding formulation, the optimization problem over the sparse graph is not a constrained path maximization problem, as solutions may pass through an already-visited partition without receiving to access other regions of the graph. To handle this, an additional binary optimization "slack" variable y_n is added for each partition p_n, representing fulfillness of r_n across all satellites:

$$\text{maximize} \quad \sum_n r_n y_n \tag{22}$$

$$\text{subject to} \quad \sum_h x^s_{h,i} + b^s_i \geq \sum_j x^s_{i,j} \quad \forall i \tag{23}$$

$$y_n \leq \sum_{s=1}^{N} \sum_{v^s_j \in p_n} \sum_i x_{i,j} \quad \forall n \tag{24}$$

The objective function, Eq. (22), maximizes the sum of request values for each partition visited. The first constraint, Eq. (23), is the same in–out constraint as in the preceding formulation. Equation (24) enforces that the fulfillment variable y_n can only be 1 if the partition p_n is visited at least once by any satellite. Solutions produced by this method are equivalent to those from the naïve method but result from a smaller MILP formulation, as listed in Table 1.

C. MILP-TD: Time-Dependent Value

1. *Graph Construction*

In cases where request values are time-dependent, only including the earliest arrival time for each request in the graph is insufficient, as vertices may not be added or reachable for high-value request times. For each vertex, an additional vertex is created if a higher-value fulfillment time follows in the opportunity window (up to discretization δt), as illustrated in Fig. 5; candidate vertices that do not increase reward are ignored. Algorithm 3 describes the modification to Algorithm 1 to account for time-dependent rewards. This method is compatible with both the dense and sparse graph construction methods, but only the latter combination is considered so that the problem remains tractable.

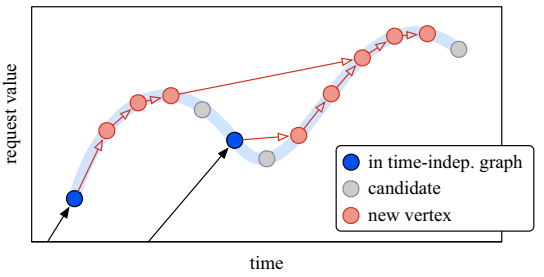

Fig. 5 Additional vertices added for time-dependent rewards.

Algorithm 3 Time-Dependent Request-Value Modification for Algorithm 1.

for $t \in [t_i : \delta t : o_{\text{close}}]$ **do**
  if $r_i(t) > r_i(t_i)$ **then**
    $v_j \leftarrow (\rho_i, t)$
    if $v_j \notin V$ **then**
      $V \leftarrow V \cup \{v_j\}$
      $A \leftarrow A \cup \{v_j\}$
    end if
    $E \leftarrow E \cup \{(v_i \rightarrow v_j)\}$
    $W \leftarrow W \cup \{r_i(t)\}$
    break from iteration
  end if
end for

2. *Mixed-Integer Linear Program*

In addition to the sequence decision variable $\boldsymbol{x}$ (and instead of $\boldsymbol{y}$ from the preceding section), a binary optimization variable $z^s_{i,j}$ is added for each $x^s_{i,j}$, representing receiving credit for visiting v^s_j. The problem is constrained to only allow credit to be assigned for a single visitation of each partition:

$$\text{maximize} \quad \sum_{s=1}^{N} \sum_{i,j} w^s_{i,j} z^s_{i,j} \tag{25}$$

$$\text{subject to} \quad \sum_h x^s_{h,i} + b^s_i \geq \sum_j x^s_{i,j} \quad \forall\ i \tag{26}$$

$$z^s_{i,j} \leq x^s_{i,j} \quad \forall\ i, j \tag{27}$$

$$\sum_{s=1}^{N} \sum_{v^s_j \in p_n} \sum_i y^s_{i,j} \leq 1 \quad \forall\ n \tag{28}$$

The objective function, Eq. (25), maximizes the sum of credited request visitations. Equation (26) is the familiar in–out constraint. Equation (27) restricts credit assignment to edges that are visited. Finally, the constraint in Eq. (28) only allows for credit to be assigned once per partition for any satellite.

V. Results

A. Simulation Environment

The scenario described in the problem formulation is modeled using Basilisk[‡] [38], a high-performance modular spacecraft simulation framework written in C++ and Python. The low Earth orbit environment and satellite flight software and dynamics are integrated at 2 Hz. The environment uses a SPICE-based model of planetary motion and gravitation for orbital dynamics propagation. Of particular relevance to this work, attitude dynamics are modeled to a high fidelity: rigid-body dynamics models of the spacecraft bus and four reaction wheels are computed using the back-substitution method [39]. The wheels are driven by flight-proven control software that feeds torque commands to the reaction wheels [40]. The complete

[‡]hanspeterschaub.info/basilisk

simulation environment is available in the BSK-RL (Basilisk Reinforcement Learning) repository.[§]

1. Satellite Configuration

In this paper, the satellite is modeled as having an instrument with a boresight pointed in a body-fixed $\hat{\boldsymbol{c}}$ direction. The satellite must be pointing the boresight at the target within some angle threshold $\delta\theta$ and have a body rate relative to the target less than some $\delta\omega$ before imaging. Four reaction wheels are used for attitude control, producing relatively high $u_{\max}$ to meet the agility requirements of the mission. An exponentially stable controller as defined in [40] that generates and tracks an attitude trajectory in modified Rodrigues parameter space is executed by the satellites in this paper. Additionally, the control torque is clipped by the maximum torque of reaction wheels when performing maneuvers in the simulation. Even with this combination of factors, the nonlinear controller satisfies the two requirements of the method: maneuvers are deterministic and are kinematically identical for any about-boresight rotation; thus, the resulting slew transition times can be fully learned by the network. Request fulfillment is considered to be instantaneous once the satellite is settled; however, if a non-zero image processing time was included in the simulation used to estimate slew times, the network could learn to account for it.

The satellite's orbit is circular, set with a fixed inclination and altitude and randomized true anomaly and ascending node. At most, 1 day of planning (15 orbits) is considered, as it is typically considered to be the practical limit for nonadaptive preplanning. This work could be applied to all orbits about a given body as long as the slew estimator is trained over the entire domain of orbits. In practice, most satellites will only operate over a subset of orbits, as implemented here. Important satellite parameters are given in Table 2. Other satellite parameters are the defaults used by SteeringImagerSat in BSK-RL; the specific Basilisk modules used by SteeringImagerSat can also be found in the repository.

2. Request Distributions

Representative sets of target requests are generated for experiments. Two distributions of target locations across various total request counts are considered, as shown in Fig. 6: a uniform distribution around the globe, and a clustered distribution in which city locations[¶] serve as a proxy for image request frequency, inspired by Eddy and Kochenderfer [8]. In the latter case, the satellite can travel for half an orbit without encountering any request, then have hundreds available in a few minutes over very populous areas. A lower maximum target count is considered for cities to keep the maximum local request density similar. In both cases, each initialization of the environment at a given density randomizes the locations of uniform points or the subset of cities selected.

For this study, imaging requests are created with relatively simple constraints for visibility. Only a minimum elevation angle constraint $\phi_{\min}$ between the target and satellite must be satisfied, meaning that imaging windows are determined by a view cone about the satellite nadir. Because request limitations only impact the preprocessing step of window generation, more exotic a priori constraints could be applied without impacting the performance of this work.

Request rewards are randomized over a uniform distribution $r \in [0, 1]$. When time-dependent rewards are considered, the value is penalized as a function of elevation angle:

$$r_n(\phi) = r_n \frac{\phi}{\pi/2} \tag{29}$$

A complete listing of request parameters is given in Table 3.

Table 2 Top: Satellite and orbit parameters. Bottom: Control gains [40]

Parameter	Value, s
Horizon H	≤ 1 day (15 orbits)
Inclination	45 deg
Altitude	800 km
$\delta\theta_{\max}$	0.01 MRP norm (2.29 deg)
$\delta\omega_{\max}$	0.01 rad/s (0.57 deg/s)
$u_{\max}$	0.4 N · m (per axis)
$m, \boldsymbol{I}$	330 kg, [82.1, 98.4, 121.0] kg · m^2
K_1, K_3	0.25, 3.0
$\omega_{\max}$	5 rad/s

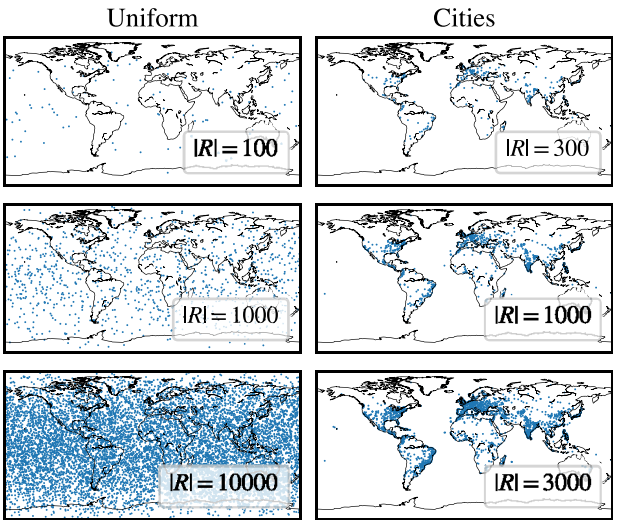

Fig. 6 (Left) Uniformly distributed requests. (Right) Requests distributed over the world's 43,000 most populous cities.

B. Slew Estimation

A data set consisting of state information at 10-s intervals along 1.8×10^5 slews to randomly selected nearby targets is generated, with a subset shown in a lower-dimensional space in Fig. 7. The network is trained on this data set with varying-loss skewness τ. Figure 8 shows the percent of predictions for the validation data set that falls within a certain error threshold as a function of τ. As expected, higher values of τ lead to a tendency to overestimate slew durations.

To evaluate the proper tuning of τ, a challenging (i.e., high request density, $|R| = 10{,}000$) simulation environment is instantiated and a "greedy" policy is implemented. In this policy, the satellite is always tasked to image the soonest request that is predicted to be accessible by the estimator. This is a worst-case stress test for the estimator, as the MILP solutions will often favor higher-value requests over sooner ones. The number of successful slews and the total number of requests attempted are recorded over 800 orbits per τ.

Figure 9 shows expected behavior: as the skewness parameter τ increases, the predictor becomes more conservative, attempting fewer requests but succeeding at a higher rate. The $\tau = 0.75$ is selected for the remainder of the experiments, as it does not show a drop in requests satisfied compared to 0.5 while having a success rate >99%.

Table 3 Target parameters

Parameter	Value, s	
$\phi_{\min}$	58 deg	Minimum elevation angle
r_{FOR}	500 km	Field of regard radius (from $\phi_{\min}$)
r	[0, 1]	Request value
Distribution	{uniform, cities}	— —
$\lvert R\rvert$	[100, 10000]	Number of requests (uniform)
$\lvert R\rvert$	[300, 3000]	Number of requests (cities)

[§]github.com/AVSLab/bsk_rl

[¶]City location data from simplemaps.com, CC BY 4.0.

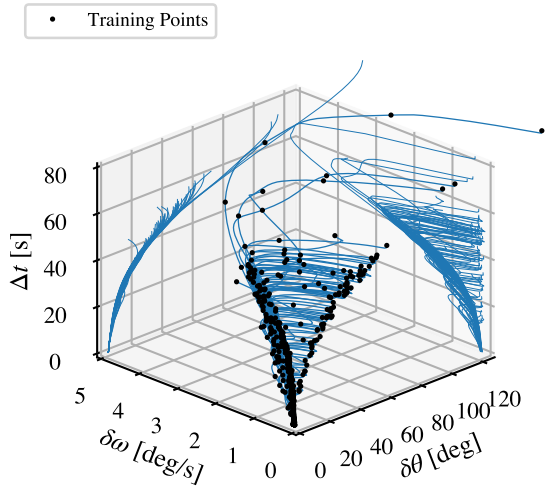

Fig. 7 A subset of 88 of the 1.8×10^5 slews used in training, plotted in lower-dimensional space.

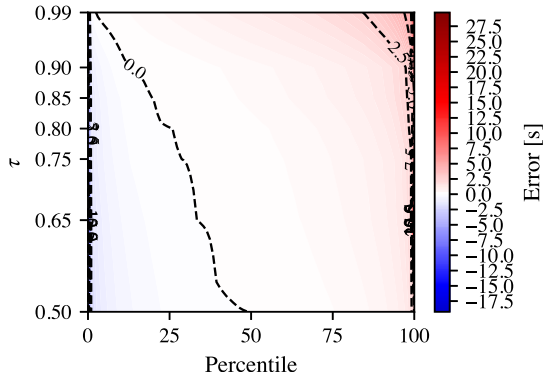

Fig. 8 Prediction errors across 198k validation points as a function of loss skewness τ by percentile of points within error threshold.

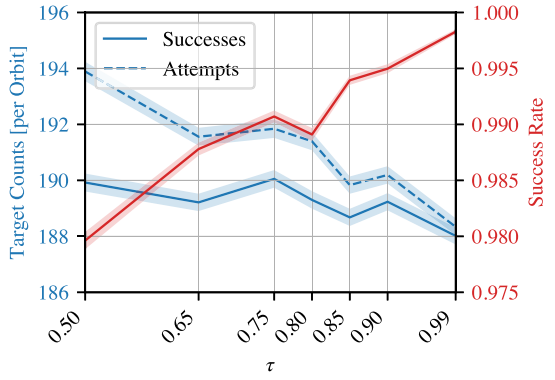

Fig. 9 Mean success rates of a greedy policy using the prediction network with varied τ; $|R| = 10{,}000$; 95% confidence intervals are shown.

C. Optimal Target Sequencing

Having demonstrated the performance of the slew estimator, the MILP formulations can be evaluated. An ablation study over δt is performed and the performance of MILP-S over the naïve MILP-D and a linear transition model are compared (Sec. V.C.1). Finally, benchmarks are performed over a range of target densities, distributions, and horizons for single-satellite, time-dependent value, and multisatellite cases. All MILPs are solved using the Gurobi solver [41].

1. Ablation Studies

a. Ablation Study on Time Discretization. Figure 10 shows the impact of the time discretization δt on the cumulative value of the optimal

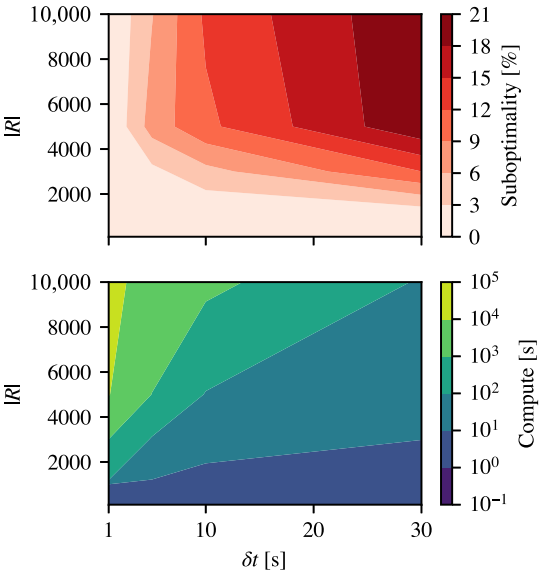

Fig. 10 Ablation study over δt using MILP-S, showing a decrease in cumulative value as δt increases.

sequence of requests over one-orbit horizons, uniform request distribution. The value of the sequence decreases as δt increases, especially in high request-density scenarios. The discretization $\delta t = 10$ s is selected for further studies, balancing the tradeoff between compute time and solution quality. In the densest cases ($|R| = 10{,}000$), this leads to a discretization error of <10%, whereas there is effectively no discretization error in sparse cases ($|R| \leq 2000$).

b. MILP Formulation Comparison. The relative sizes and compute times of the naïve MILP-D formulation and the condensed MILP-S formulation are compared in Fig. 11 over a small range of uniform request densities and horizons. The order of the number of edges in the graph is reduced by a factor of H (see Table 1) and some large constant in the sparse formulation, corresponding to a reduction in compute times. Extrapolating the trends indicates that MILP-D becomes intractable significantly more quickly than MILP-S. Beyond the time benefits of MILP-S, the relatively low memory requirement also allows for greater scalability.

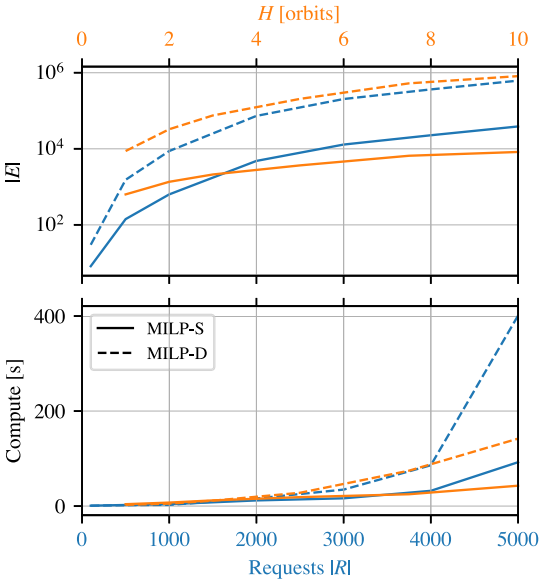

Fig. 11 Comparison between graph sizes and compute time between MILP-D and MILP-S. When varying H, $|R| = 1000$ requests; when varying $|R|$, $H = 1$ orbit.

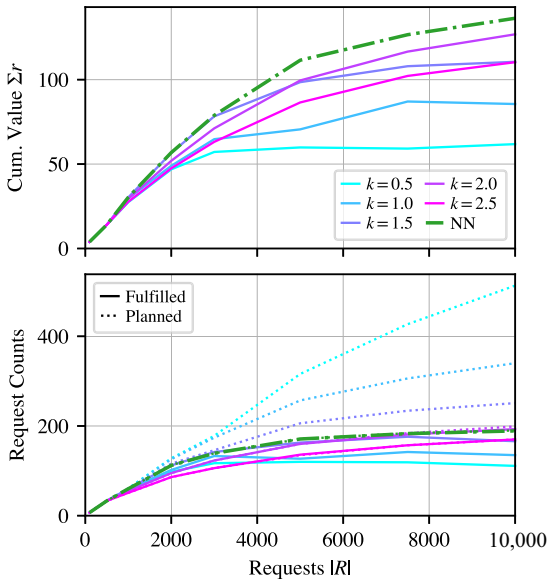

Fig. 12 Performance with a linear transition model with varying k vs an NN transition estimator.

c. Comparison to Linear Transition Model. A primary claim of this work is that the use of an NN to estimate transition times is more effective than a lower-order model. To evaluate this, the NN is replaced with a linear model of the form

$$\Delta t^s = k\delta\theta \tag{30}$$

This model is evaluated over a range of k and $|R|$ and compared to the NN-based solution for one-orbit horizons. Figure 12 gives the results of this study. Varying k shows expected behavior: too low of a k leads to many failed slews (i.e., a large gap between planned and fulfilled requests), whereas too high of a k leads to significant idle time; in both cases, the overall reward is depressed. Because the transitions are nonlinear with respect to $\delta\theta$ and other variables, even the best value of k achieves a lower cumulative value than the NN-based solution, as never are all slews accurately predicted; also, different values of k are best for different request densities. This effect is especially prominent as the request density grows and correct planning becomes more important. NN evaluation only accounts for a small portion of the graph construction time, so using it instead of a linear model does not significantly impact the overall compute time.

2. *Single-Satellite Performance*

MILP-S is benchmarked over uniform and city-distributed requests, varying $|R|$ and H with multiple trials at each point, for a total of 15 orbits worth of trials completed at each density. The results are collected in Figs. 13 and 14. The horizon-normalized cumulative value of the sequence $\Sigma r/H$ gives the value of the best MILP solution found and, in cases where the solver did not converge to optimality within 1000 s of solve time, an upper bound on the true solution value. Cases where the solver could not find any solution within 1000 s are excluded; practically, however, these cases tend to be those with the longest horizons and highest request densities so additional solve time could be allocated. The cumulative value tends to be lower for longer horizons, as the solutions are effectively less greedy; the satellite is competing with its future self for a limited number of requests. The average request value shows that as the number of requests increases, the satellite is able to be more selective about picking higher-value requests. The overall size of the problem is given in the requests and opportunities plot, showing the number that the counts of each along-track over the horizon, giving insight into the size of the problem. The fulfillment rate $F_{\%}$ gives the ratio of requests fulfilled to the number of possible requests along-track; longer horizons and fewer requests give the satellite more

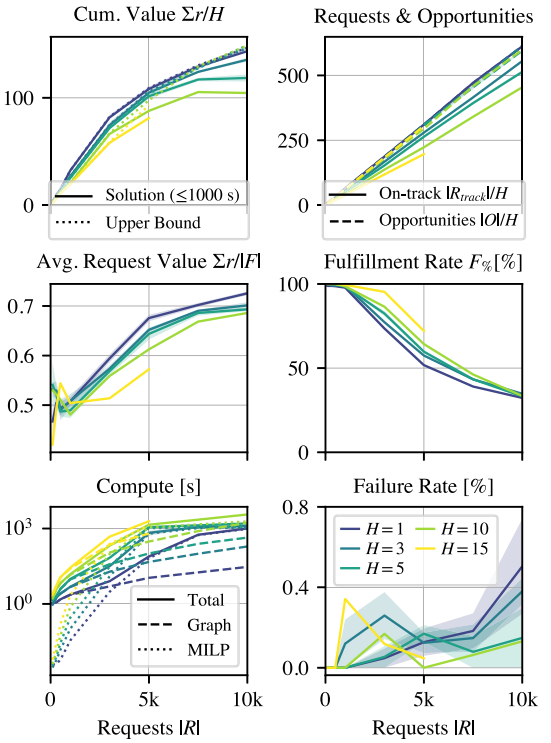

Fig. 13 Benchmark of MILP-S on uniformly distributed, constant-valued requests. Shaded region corresponds to one standard error of the mean over trials.

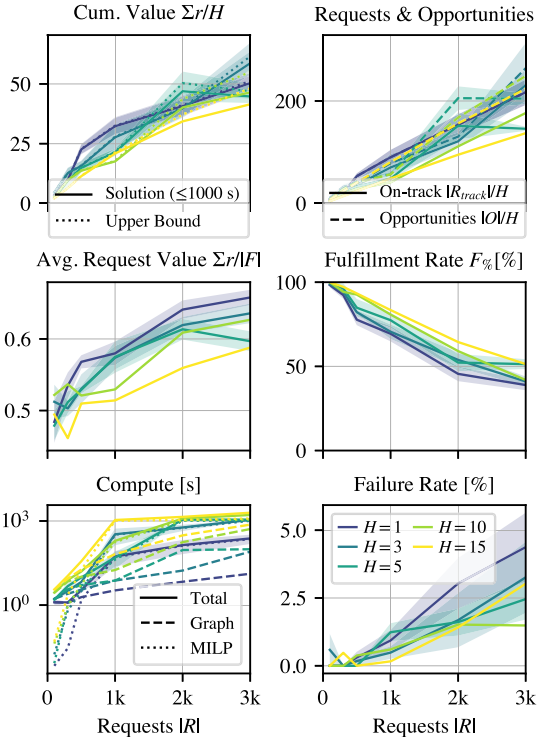

Fig. 14 Benchmark of MILP-S on city-distributed, constant-valued requests. Shaded region as in Fig. 13.

opportunities to fulfill all requests. Finally, the success rate (plotted as its additive compliment, the failure rate) measures how effectively the MILP solutions can be executed in the high-fidelity simulation environment. It is given as the percent of requests successfully fulfilled in simulation out of the total planned requests by the MILP solver. The generally high success rates indicate that the solution pipeline is effective. Some depression in success rate is observed in

high-density city-distributed cases, as certain regions have extremely high local request densities, leading to an especially difficult problem instance.

3. *Time-Dependent Reward Performance*

Solutions for a single satellite with requests with time-dependent rewards are benchmarked in Fig. 15. Unsurprisingly, compute times tend to be worse than the fixed-value counterpart experiments due to the larger size of the graph; however, they are still tractable over a wide range of problem sizes. Despite many solutions having a sizeable gap between the timed-out MILP solution and time upper bound, comparison to the constant-valued requests implies that the computed solution is likely close to the true optimum. The average request value is likewise lower because the satellite can only fulfill a request for maximum value if it is perfectly on-track. Still, the depression is minimal, indicating that the best possible imaging times are being chosen.

4. *Multisatellite*

The performance of multisatellite cases is examined in Fig. 16. Two constellations are compared: "String," which has $N = 12$ satellites separated by 5-deg true anomaly, and "Walker," which has $N = 12$ satellites in four planes of three satellites with 0 phase factor. For both, a 45-deg inclination is maintained. The former is a case where satellites are frequently sharing opportunities (see the high number of opportunities and low number of on-track requests), whereas the latter has less overlap between satellites. As expected, both constellations are undersubscribed, given the high number of overlapping requests and relatively high request-fulfillment rate of a single satellite. Examining the compute times, the graph construction time scales by N; however, each satellite's graph construction is independent so it could be parallelized. A more interesting trend is observed in the MILP solution time (which is still capped at 1000 s for these experiments, hence the differing number of requests evaluated for each constellation): although the "Walker" constellation takes longer than the single satellite to solve as $|R|$ increases, the "String" cases become easier to solve than the single satellite as $|R|$ increases.

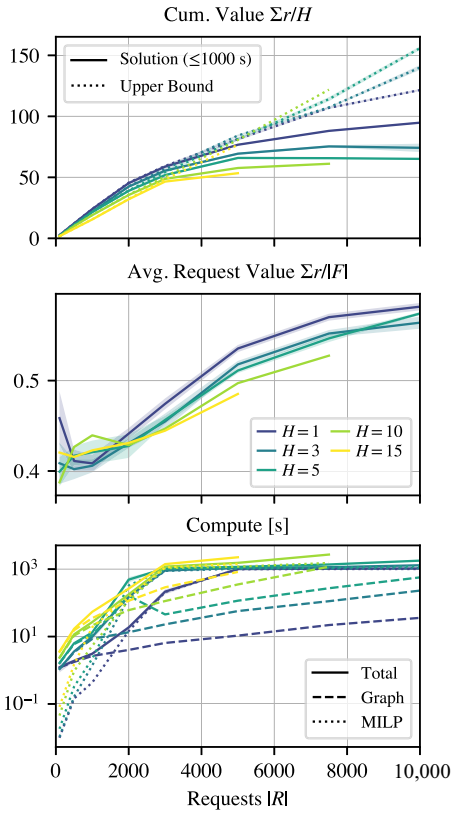

Fig. 15 Benchmark of MILP-TD on uniformly distributed, time-dependent requests. Shaded region as in Fig. 13.

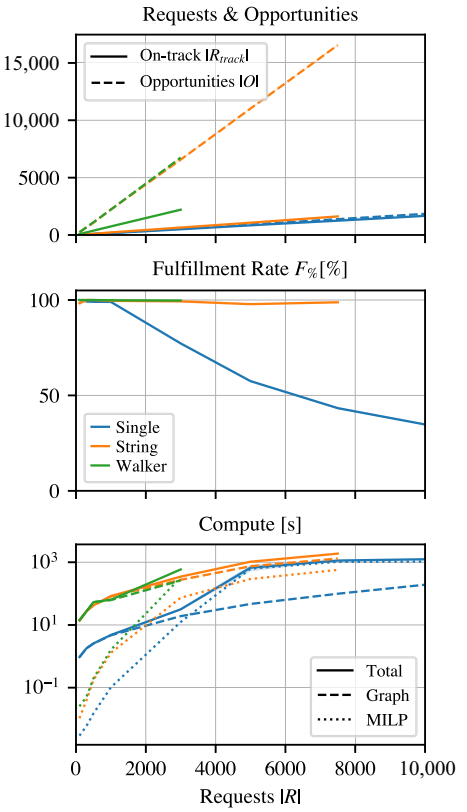

Fig. 16 Benchmark of MILP-S on uniformly distributed, constant-value requests compared to a "String" constellation and a "Walker" constellation; $H = 3$ orbits.

VI. Conclusions

Two contributions are made toward solving the Earth-observing satellite scheduling problem (AEOSSP) under realistic conditions: a framework for estimating slew durations based on the true performance of the satellite is introduced, and an improved formulation of the mixed-integer linear program (MILP) is designed to handle larger instances of the problem. The solutions are executed in a full-fidelity simulation of a spacecraft, validating the effectiveness of the solution pipeline; in doing so, optimality can be claimed with respect to the true system and not a simplified abstraction of the system; this is a unique evaluation of the "sim-to-real" gap not found elsewhere in the literature. Learning the transition time function, as opposed to using a lower-order heuristic, ensures that this gap is minimized when the plan is executed. Benchmarks performed over a variety of request densities, planning horizons, satellite counts, and request-value types demonstrate the broad applicability of the methods; the upper request densities considered are the highest found in the literature, and still demonstrate acceptable performance. Demonstrating success on high global (10,000 uniform targets) and local (3000 city-distributed targets) density cases using the sparse formulation (MILP-S) when the more typical formulation MILP-D is intractable shows that this method is able to handle large problem instances that no other MILP formulation in the literature can. Furthermore, the method performs competitively in time with iterative local search (ILS) and other iterative methods over small and medium problem instances while providing the additional benefit of optimality quantification and certification; although the scalability of the alternatives to the largest instances is not known, the given method conclusively does scale.

Insights into the difficulty of different scenarios are gained from the benchmarks, providing trends that are generally applicable to the problem. Distributions matter: locally high densities are the primary driver of difficulty in the problem, hence the 10,000 uniform targets

being similarly challenging to 3000 city-distributed targets. However, the planning horizon's impact on planning times can be minimized with a good solver formulation. When considering multi-agent cases, the interaction between satellite density and request density becomes relevant, with a cluster of satellites being able to trivialize the problem through an exhaustion of possible conflicting requests.

As with other efficiently designed ground-based planners, this method is practical and performant for offline planning but too computationally expensive for onboard planning. The high quality of the plans produced because of accurate dynamics learned from a model of the system means that replanning on-the-fly necessitated by plan infeasibility is close to zero, even when generating very aggressive plans. The learning of transition times could be leveraged by other types of solvers to grant them the same benefit. This work is also useful as an optimal benchmark for other methods, such as MILPs that make simplifying assumptions to satellite dynamics or approaches that lack optimality guarantees like ILS and reinforcement learning, because the optimality of this method's solutions are with respect to a high-fidelity simulation of the system.

Acknowledgments

This work is supported by a NASA Space Technology Graduate Research Opportunity (NSTGRO) Grant 80NSSC23 K1182. This work is also supported by the Air Force Research Lab Grant FA9453-22-2-0050.

Appendix: Compute Resources

Networks were trained on an M2 Pro Macintosh. Benchmarks were performed on an Intel i9 13900 KF CPU (24 cores) with 64 GB of RAM running Ubuntu 20.04.

References

[1] 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

[2] Ungar, S., Pearlman, J., Mendenhall, J., and Reuter, D., "Overview of the Earth Observing One (EO-1) Mission," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 41, No. 6, 2003, pp. 1149–1159. https://doi.org/10.1109/TGRS.2003.815999

[3] Spoto, F., Sy, O., Laberinti, P., Martimort, P., Fernandez, V., Colin, O., Hoersch, B., and Meygret, A., "Overview of Sentinel-2," *2012 IEEE International Geoscience and Remote Sensing Symposium*, Inst. of Electrical and Electronics Engineers, New York, 2012, pp. 1707–1710. https://doi.org/10.1109/IGARSS.2012.6351195

[4] Gleyzes, M. A., Perret, L., and Kubik, P., "Pleiades System Architecture and Main Performances," *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Vol. XXXIX-B1, July 2012, pp. 537–542. https://doi.org/10.5194/isprsarchives-XXXIX-B1-537-2012

[5] Safyan, M., "Planet's Dove Satellite Constellation," *Handbook of Small Satellites*, edited by J. N. Pelton, and S. Madry, Springer International Publ., Cham, 2020, pp. 1057–1073. https://doi.org/10.1007/978-3-030-36308-6_64

[6] 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

[7] 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 Digital Library, 2014, pp. 11–16, https://aaai.org/conference/spring-symposia/sss14/.

[8] 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

[9] 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

[10] Lemaître, M., Verfaillie, G., Jouhaud, F., Lachiver, J.-M., and Bataille, N., "Selecting and Scheduling Observations of Agile Satellites," *Aerospace Science and Technology*, Vol. 6, No. 5, 2002, pp. 367–381. https://doi.org/10.1016/S1270-9638(02)01173-2

[11] Dilkina, B., and Havens, B., *Agile Satellite Scheduling via Permutation Search with Constraint Propagation*, Actenum Corporation, Vancouver, British Columbia, 2005.

[12] Mao, T., Xu, Z., Hou, R., and Peng, M., "Efficient Satellite Scheduling Based on Improved Vector Evaluated Genetic Algorithm," *Journal of Networks*, Vol. 7, March 2012. https://doi.org/10.4304/jnw.7.3.517-523

[13] Verbeeck, C., Sörensen, K., Aghezzaf, E.-H., and Vansteenwegen, P., "A Fast Solution Method for the Time-Dependent Orienteering Problem," *European Journal of Operational Research*, Vol. 236, No. 2, 2014, pp. 419–432. https://doi.org/10.1016/j.ejor.2013.11.038

[14] Liu, X., Laporte, G., Chen, Y., and He, R., "An Adaptive Large Neighborhood Search Metaheuristic for Agile Satellite Scheduling with Time-Dependent Transition Time," *Computers & Operations Research*, Vol. 86, Oct. 2017, pp. 41–53. https://doi.org/10.1016/j.cor.2017.04.006

[15] 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

[16] Chen, X., Reinelt, G., Dai, G., and Spitz, A., "A Mixed Integer Linear Programming Model for Multi-Satellite Scheduling," *European Journal of Operational Research*, Vol. 275, No. 2, 2019, pp. 694–707. https://doi.org/10.1016/j.ejor.2018.11.058

[17] 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

[18] 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

[19] 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

[20] 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, Richland, SC, Beaverton, Oregon, 2021, pp. 39–44. https://doi.org/10.5555/3463952.3463961

[21] Parjan, S., and Chien, S. A., "Decentralized Observation Allocation for a Large-Scale Constellation," *Journal of Aerospace Information Systems*, Vol. 20, No. 8, 2023, pp. 447–461. https://doi.org/10.2514/1.I011215

[22] Xu, Y., Zhang, Y., Wang, Z., He, Y., and Fan, L., "Self-Organizing Control of Mega Constellations for Continuous Earth Observation," *Remote Sensing*, Vol. 14, No. 22, 2022, p. 5896. https://doi.org/10.3390/rs14225896

[23] Sutton, R. S., and Barto, A., *Reinforcement Learning: An Introduction*, 2nd ed., Adaptive Computation and Machine Learning, MIT Press, Cambridge, MA, London, England, 2018, pp. 197–255.

[24] Nazari, M., Oroojlooy, A., Snyder, L., and Takac, M., "Reinforcement Learning for Solving the Vehicle Routing Problem," *NeurIPS*, MIT Press, Cambridge, MA, 2018. https://doi.org/10.48550/arXiv.1802.04240

[25] Harris, A., Teil, T., and Schaub, H., "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, American Astronautical Soc., AAS Paper 19-447, 2019.

[26] Hadj-Salah, A., Verdier, R., Caron, C., Picard, M., and Capelle, M., "Schedule Earth Observation Satellites with Deep Reinforcement Learning," IWPSS Paper 2019, Nov. 2019. https://doi.org/10.48550/arXiv.1911.05696

[27] 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

[28] Stephenson, M., and Schaub, H., "Reinforcement Learning for Earth-Observing Satellite Autonomy with Event-Based Task Intervals," *AAS Rocky Mountain GN&C Conference*, American Astronautical Soc., AAS Paper 24-012, 2024.

[29] 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

[30] 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, Oct. 2023, pp. 1–13. https://doi.org/10.1109/TAES.2023.3251307

[31] Herrmann, A., and Schaub, H., "A Comparison of Deep Reinforcement Learning Algorithms for Earth-Observing Satellite Scheduling," *AAS/AIAA Spaceflight Mechanics Meeting*, American Astronautical Soc., AAS Paper 23-116, Jan. 2023.

[32] 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

[33] Herrmann, A., and Schaub, H., "Reinforcement Learning for Small Body Science Operations," *AAS Astrodynamics Specialist Conference*, American Astronautical Soc., AAS Paper 22-563, 2022.

[34] 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

[35] Kingma, D. P., and Ba, J., "Adam: A Method for Stochastic Optimization," *International Conference on Learning Representations*, Jan. 2017. https://doi.org/10.48550/arXiv.1412.6980

[36] Britney, R. R., and Winkler, R. L., "Bayesian Point Estimation and Prediction," *Annals of the Institute of Statistical Mathematics*, Vol. 26, No. 1, 1974, pp. 15–34. https://doi.org/10.1007/BF02479801

[37] 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.

[38] 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

[39] Allard, C., Diaz Ramos, M., and Schaub, H., "Computational Performance of Complex Spacecraft Simulations Using Back-Substitution," *Journal of Aerospace Information Systems*, Vol. 16, No. 10, 2019, pp. 427–436. https://doi.org/10.2514/1.I010713

[40] 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

[41] *Gurobi Optimizer Reference Manual*, Gurobi Optimization LLC, 2023.

C. N. McGrath
Associate Editor