(Preprint) AAS 23-108

OPTIMAL TARGET SEQUENCING IN THE AGILE EARTH-OBSERVING SATELLITE SCHEDULING PROBLEM USING LEARNED DYNAMICS

Mark Stephenson[*] and Hanspeter Schaub[†]

The target sequencing problem 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 summed value of heterogeneously-valued targets. Two main challenges are encountered: 1) transitions between targets are time dependent and are expressed by complex dynamics, and 2) the size of the solution space is combinatorial with respect to the number of targets. To find time-dependent transition times that accurately reflect the system dynamics, a neural network is trained on simulated slew data. Next, feasible slews between targets are expressed in a graph. Mixed-integer programs are formulated to traverse the graphs that are two orders of magnitude faster than a naive solution, making this approach competitive with other methods that account for time-dependent transition times. The solutions are guaranteed to be optimal up to the quality of the transition time estimator and a time discretization, with one formulation supporting time-dependent imaging rewards. Finally, the solutions are verified in a full-fidelity satellite simulation, showing that the solutions are valid when deployed to a lifelike system.

INTRODUCTION

The agile Earth-observing satellite (AEOS) scheduling problem aims to maximize the quantity and value of ground targets imaged by an orbiting satellite while managing resources [1]. As compared to a standard EOS which can only slew side-to-side to capture off-track targets, “Agile” indicates that the satellite has three axes of control, giving the satellite a larger observation area by allowing targets forward and backwards along-track to be imaged. The transition time between two targets is thus both target and time dependent, as the satellite’s initial and target attitude depends on when and where it is imaging. The reward obtained from imaging can also be time dependent, if factors such as time of day and squint angle strongly impact image quality. Particularly in environments with a high density of targets, finding an optimal target sequence is both challenging and rewarding.

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 (ESA’s Sentinel-2 [3]), and on-demand image requests from commercial, scientific, and defense customers (CNES’s Pleiades [4] and Planet’s Dove constellation [5]). In all of these cases, optimally scheduling observations improves the data throughput of the satellite. In cases where some observations may be more critical,

[*]PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80303. AIAA Member.
[†]Professor and Department Chair, Schaden Leadership Chair, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AAS Fellow, AIAA Fellow.

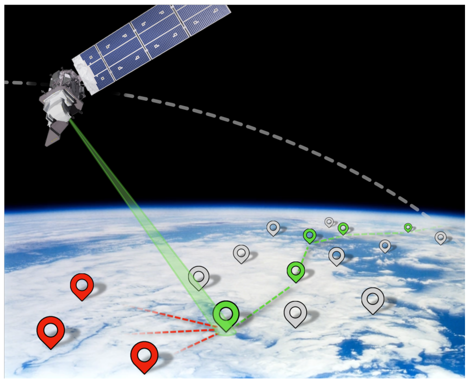


Figure 1: An agile satellite imaging a sequence of ground targets.

such as for a resource like Pleiades shared between multiple stakeholders or when rare or important phenomena are observable, scheduling that optimizes for request priority is important.

PRIOR WORK

Graph-based representations of the AEOS scheduling problem, in which imaging actions are represented by vertices and feasible slews between targets are represented by edges, are common in the literature. Gabrel [6] offers an early graph-based treatment of the problem. 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); Eddy's [8] exploration of efficient planning for constellations takes a similar approach to request representation. Peng [9] briefly considers a graph-based method that uses multiple vertices per observation window; this introduces challenges that prevent the use of simpler dynamic programming solutions for path maximization such as those used by Augenstein.

Iterative local search algorithms are good at solving highly combinatorial problems such as the AEOS scheduling problem. Lemaître [10] offers one of the first local search approaches to the problem. More recently, Verbeeck's [11] advances in local search for time-dependent orienteering problems (a class which includes the AEOS scheduling problem) have led to the development of performant AEOS solvers by Liu [12] and Peng [9]. These papers consider both time-dependent transition times and time dependent rewards, planning in environments with hundreds of targets over a day-long period in tens to hundreds of seconds.

Mixed-integer program (MIP) solutions to the AEOS scheduling problem offer an alternative method that quantifies solution 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 small instances. Cho [13], Kim [14], and Wang [15] use MIP 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.

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]. One approach is to use a low-order linear model that considers only the difference in roll angle [13, 14] or the total angle change [12, 9]; while these methods are convenient for planning problems, they do not fully capture the nonlinear nature of attitude transitions. Nag [16] uses a discretized model of transitions between various attitudes; this captures more of the dimensions that impact transition dynamics, but is still a lower-order model.

In this paper, novel developments are made in accurately representing and efficiently solving the AEOS target sequencing problem. First, time-dependent transition times between imaging actions are modeled using a neural network function approximator, a significant improvement in accurately expressing dynamics over the low-order models for transition times used in other literature. The structure of the problem is represented as a graph that accounts for different imaging times for each request. Finally, more efficient MIP formulations are developed for solving the problem with optimality guarantees — up to a time discretization — that remain tractable even for relatively large instances of the problem. Formulations for time-independent and a time-dependent rewards are given, allowing for the best performance for a specific problem's restrictions. Because transition times used in planning are meant to be representative of a highly nonlinear physical system, solutions to the planning problem are validated in a full-fidelity spacecraft simulator.

PROBLEM FORMULATION

The goal of the AEOS target sequencing problem is to determine a feasible sequence of imaging actions for an orbiting satellite on a fixed orbit $\boldsymbol{r}_s(t)$ that maximizes the sum of imaging rewards of the requests satisfied. Because of the agile nature of the satellite, each target has opportunity windows where imaging is possible. The times and durations of these windows are determined by the satisfaction of multiple constraints, including but not limited to: satellite elevation angle relative to the target ϕ, target-satellite range, time of day, or other predetermined restrictions depending on the sensor type (e.g. radar, optical, etc.). In some formulations, the value of the target is constant regardless of imaging time; in others, the target has a time-dependent reward due image quality being affected by those factors. A target may only be imaged during one of its respective windows, and transitions between image-taking actions for different targets must be feasible; that is, the satellite must be able to reach and image a target before the target's window has closed. These transition dynamics are highly nonlinear, as the satellite must control the pointing direction of its instrument in two dimensions to capture a target that has a position that is moving relative to the satellite, thus exhibiting time-dependent transition times.

Optimization Objective

The objective of the problem can be stated as follows: Find the sequence of imaging actions $S = v_1, ...v_i$ that maximizes the sum of imaging rewards

$$\sum_{v_i=(\rho_i,t_i)\in S} r_i(t_i) \tag{1}$$

and satisfies a slew feasibility constraint

$$t_i + \Delta t_s(v_i, \rho_{i+1}) \leq t_{i+1} \quad \forall \quad v_i, v_{i+1} \in S \tag{2}$$

That is, maximize the sum of rewards for imaged targets while ensuring that there is enough time to transition between each imaging action.

Target Request Model

Models for imaging actions can be broadly classified into two categories: continuous imaging, where the satellite continuously scans a strip of land in the direction the sensor is pointed; and point-based imaging, where the satellite aims at a target and takes a single, instantaneous picture. Imaging request models can be likewise classified: area requests, where the operator desires for images of a region of land to be collected; and point requests, where individual targets are identified. The latter distinction is less consequential as Eddy [8] describes how area requests can be decomposed into point targets. Thus for a point-based imaging model, only point requests must be considered. .

In this paper, the problem is considered for point-based requesting and imaging model. Targets requests $\rho \in$ the set of requests R take the form of a tuple $\rho_i = (\boldsymbol{r}_n, r_n(t))$, expressing the target's planet-fixed location $\boldsymbol{r}_n$ and the target reward as a function of time $r_n(t)$. Each target has imaging windows $[\tau^o, \tau^c] = w \in W_n$ expressed as time intervals from τ^o to τ^c during which imaging constraints are met. For targets with time-independent rewards, reward is constant r_n during windows and zero otherwise:

$$r_n(t) = \begin{cases} r_n & \text{if } t \in w \in W_n \\ 0 & \text{else} \end{cases} \tag{3}$$

For time-dependent rewards,

$$r_n(t) = \begin{cases} r_n(f(t)) & \text{if } t \in w \in W_n \\ 0 & \text{else} \end{cases} \tag{4}$$

where r_n is a function of time-dependent parameters $f(t)$ such as time of day or elevation angle ϕ.

Imaging actions can be uniquely identified as a combination of a target and a time at which it is imaged $v = (\rho_n, t)$, which yields reward $r_n(t)$ if successful. However, 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 unfulfilled, $R = U$. Once an imaging action v has been successfully completed, ρ_n is moved from U to F. The reward function is then wrapped with logic to reward uniquely satisfied requests:

$$r_n(t) = \begin{cases} r_n(t) & \text{if } \rho_n \in U \\ 0 & \text{if } \rho_n \in F \end{cases} \tag{5}$$

Imaging and Transition Dynamics

To successfully complete an imaging action $v = (r_n, t)$, the satellite must transition from its current dynamic state at time t_0 to one that satisfies imaging requirements for r_n at t. For imaging to be successful, the sensor boresight direction $\hat{\boldsymbol{c}}$ must point at the target

$$\hat{\boldsymbol{c}}^{\text{ref}}(t) = \frac{\boldsymbol{r}_n - \boldsymbol{r}_s(t)}{|\boldsymbol{r}_n - \boldsymbol{r}_s(t)|} \tag{6}$$

typically within some threshold $\delta\theta$, and the sensor must be settled such that the angular rate of the boresight relative to the target is minimal, typically expressed as a rate threshold $\delta\omega$. The target boresight frame $\mathcal{C}$ rate relative to the planet-fixed frame $\mathcal{P}$ is

$$\boldsymbol{\omega}^{\text{ref}}_{\mathcal{C}/\mathcal{P}}(t) = \frac{{}^{\mathcal{P}}\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}_s(t) \times \hat{\boldsymbol{c}}^{\text{ref}}(t)}{|\boldsymbol{r}_n - \boldsymbol{r}_s(t)|} \tag{7}$$

assuming zero about-boresight rotation, $\boldsymbol{\omega}_{\mathcal{C}/\mathcal{P}} \cdot \hat{\boldsymbol{c}} = 0$. The two thresholds must be met after a slew of duration $\Delta t_s \leq t - t_0$ to successfully image at the desired imaging time. It is reasonably assumed that satellite is always capable of tracking the target once it has settled on the target, so arriving before the imaging time means that the satellite can simply track the target in a settled state until it is the chosen imaging time.

A pertinent question for find optimal target sequences then follows: How long will it take a satellite in a given state to transition to imaging another target? Mathematically, it is desired to find the slew transition time Δt_s to settle pointing at $\boldsymbol{r}_n$ given the current pointing direction $\hat{\boldsymbol{c}}_0$ and upcoming orbital trajectory $\boldsymbol{r}_s(t \in [t_0, t_0 + \Delta t])$: Symbolically, an expression for $\Delta t_s(\hat{\boldsymbol{c}}_0, \boldsymbol{\omega}_{\mathcal{C}/\mathcal{P},0}, \boldsymbol{r}_s(t), \boldsymbol{r}_n)$ is desired. This function is challenging — if not impossible — to analytically construct due to the time-varying reference state and is highly dependent on the control law used. However, this function does exists if the controller satisfies two assumptions:

1. The 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).

This parameterization of the slew duration function is purposeful: there is an bijective mapping between the combination of an imaging action v_0 and upcoming target ρ_n and the slew transition time function inputs

$$(v_0, \rho_n) \to (\boldsymbol{r}_0, t_0, \boldsymbol{r}_n) \leftrightarrow (\hat{\boldsymbol{c}}_0, \boldsymbol{\omega}_{\mathcal{C}/\mathcal{P},0}, \boldsymbol{r}_s(t), \boldsymbol{r}_n) \tag{8}$$

using Equation 6 and 7. Restated, the slew duration function with the chosen parameterization can be uniquely evaluated from any imaging action to any target while utilizing all information about the dynamic state. This property will prove useful when constructing graphs of feasible slews in the next sections. This form for transition times allows for the use of high-fidelity dynamics models when planning, compared to current literature that use simplified linear models to describe the highly nonlinear system.

METHODS

Supervised Learning of Transition Dynamics

It is desired to express the slew transition time function accurately and computationally inexpensively. A neural network function approximator is selected for this task with a multi-layer perceptron (MLP) architecture, leading to a slew duration function with network parameterization $\boldsymbol{\theta}$:

$$\Delta \hat{t}_s = \Delta t_s(\hat{\boldsymbol{c}}_0, \boldsymbol{\omega}_{\mathcal{C}/\mathcal{P},0}, \boldsymbol{r}_s(t), \boldsymbol{r}_n; \boldsymbol{\theta}) \tag{9}$$

With access to a simulation of the spacecraft's dynamics, many slews from arbitrary initial states to random targets can be simulated and timed. This data can then be regressed over to learn the complex, nonlinear dynamics of dynamic-to-dynamic slews. The complete architecture is diagrammed in Figure 2.

Notice that function being approximated requires the upcoming trajectory as an input. Since it nontrivial to pass a function or a dataseries to a neural network, knowledge of orbital mechanics is

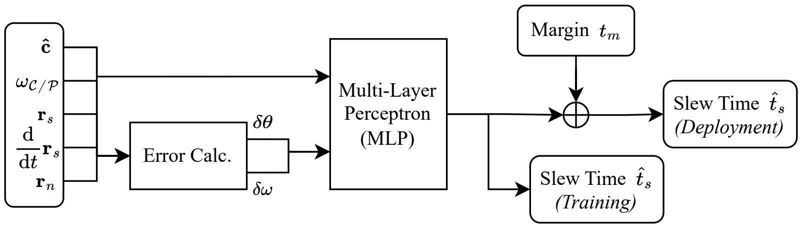

$\hat{\mathbf{c}}$
$\omega_{\mathcal{C}/\mathcal{P}}$
$\mathbf{r}_s$
$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{r}_s$
$\mathbf{r}_n$
Error Calc.
$\delta\theta$
$\delta\omega$
Multi-Layer Perceptron (MLP)
Margin t_m
Slew Time $\hat{t}_s$ (Deployment)
Slew Time $\hat{t}_s$ (Training)

Figure 2: Slew transition time estimation network.

used to simplify the input parameterization. Under the assumption that no orbital maneuvers will be made, the satellite's current state and future motion under the influence of Earth's gravity can be uniquely identified by its current position $\boldsymbol{r}_s$ and velocity $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}_s$. The input vector for the MLP is thus $\{\hat{\boldsymbol{c}}, \boldsymbol{\omega}_{\mathcal{C}/\mathcal{P}}, \boldsymbol{r}_s, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}_s, \boldsymbol{r}_n\}$.

Knowledge of the system's operation can be leveraged in the network design. For a control system with typical angle and rate thresholds for imaging, those errors can be explicitly computed from the input state using

$$\delta\theta = \angle(\hat{\boldsymbol{c}}, \hat{\boldsymbol{c}}^{\text{ref}}) \tag{10}$$

$$\delta\omega = |\boldsymbol{\omega}_{\mathcal{C}/\mathcal{P}} - \boldsymbol{\omega}^{\text{ref}}_{\mathcal{C}/\mathcal{P}}| \tag{11}$$

and then be added as inputs to the MLP, as shown in the "Error Calc." block of Figure 2. While the network would eventually infer these thresholds with enough training, explicitly providing them improves training by exposing the direct cause of the challenging discontinuity in the training data (i.e. $t_s = 0$ if thresholds met, $t_s > 0$ otherwise). This performance improvement is confirmed by a hyperparameter search.

Once trained, the network is almost ready to be used in a planning context. The final limitation stems from the fact that the network's predictions have errors equally distributed on either side of zero (for most loss functions). This can be an issue in the planning context, as a slight duration underprediction could result in the extra time needed to actually complete the slew exceeding the available time. To overcome this, a margin t_m is added to predictions, increasing the percentage of slews are completable within their predicted duration.

Planning Over Feasibility Graphs

With a method of evaluating the feasibility of transitioning between two imaging actions, planning across all possible imaging actions can be considered abstractly. Methods of constructing graphs representing all possible transitions are considered. These are designed to capture the time-dependence of transition times and rewards without generating graphs that are prohibitively large to search over. With tractably-sized graphs, mixed-integer programs can be formulated to find an optimal path across the graph, thus computing an optimal sequence of imaging actions.

Graph Abstraction The problem is formulated using a directed weighted graph. Directed weighted graphs are defined by a set of vertices $v \in V$ connected by edges $e_{i,j} = (v_i \to v_j) \in E$, where each edge $e_{i,j}$ has a scalar weight $w_{i,j}$ associated with it. In this problem, each vertex represents an imaging action which has a target and a time associated with it. Edges represent a feasible transition between two imaging actions in which the transition time is less than the difference in imaging

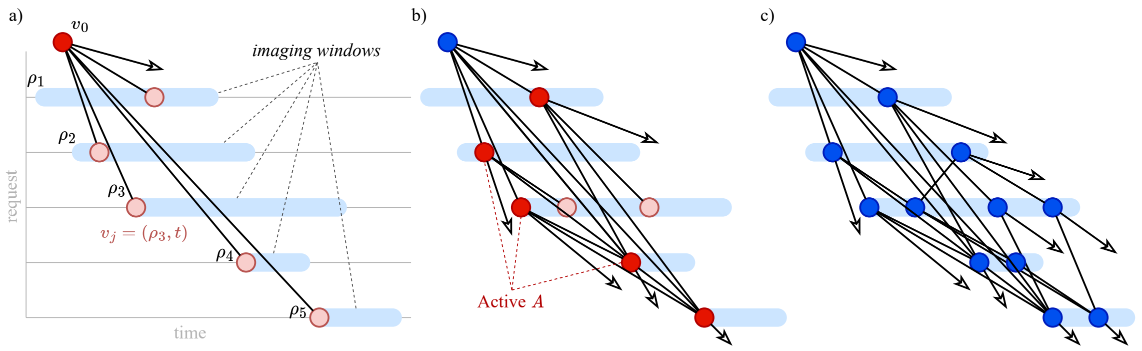


Figure 3: Slew feasibility graph construction using Algorithm 1. a) One iteration. b) Two iterations. c) Many iterations.

times; for an edge to exist between $v_i = (\rho_i, t_i)$ and v_j,

$$\Delta t_s(v_i, \boldsymbol{r}_j) \leq t_j - t_i \tag{12}$$

must be satisfied. The weight $w_{i,j}$ of an edge is the reward of the imaging action at the incoming vertex v_j. The resulting graph is acyclic because transition times are always positive, so no cycles are ever constructed. Evaluating the slew transition time function from an arbitrary vertex of the graph is only possible because of the carefully chosen parameterization. This allows the graph to be constructed to an arbitrary depth from an arbitrary vertex without the need to record and propagate states that evolve across many vertices; if a quantity such as about-boresight roll was added to the state at each vertex, the dimensionality of the graph would increase and the problem would become intractably large.

The most basic approach to slew feasibility graph construction with time dependence is to create vertices at some discretization Δt of each observation window of every target. Then, for each pair of vertices (v_i, v_j), check if Equation 12 is satisfied; if so, add $(v_i \to v_j)$ to E with weight $w_{i,j} = r_j(t_j)$. This method produces intractably large graphs, as noted by [9].

In the case of time-independent rewards, a reduction can be made to the size of the graph by leveraging that there is no advantage in arriving at a target later if it is possible to arrive earlier. Algorithm 1 describes the reduced process, which is shown in Figure 3. Starting with a vertex v_0 representing the current state of the satellite, the feasibility of slews to all targets with upcoming windows are evaluated. If feasible, a vertex is added at the arrival time, which is ceilinged to the next nearest time discretization Δt and clamped to be restricted to the valid imaging window w. This process is repeated until expansion has been performed from all vertices. The discretization is important for maintaining a compact graph by forcing arrival times that are approximately the same to be condensed to a single vertex. Since each vertex connects to up to the total number of windows, the number of edges is $\mathcal{O}(|R|^3)$, making this process produce large graphs for long planning horizons.

In implementation, two details can greatly improve performance. If a function approximator is being used for slew estimation, Δt_s is relatively expensive to evaluate one-at-a-time. One can leverage batchwise performance improvements offered by most neural network frameworks by evaluating Δt_s for all elements in $A \times R$ only when there are no elements in A for which Δt_s has not been computed. To further reduce evaluations of Δt_s, a maximum slew duration $\Delta t_{s,\text{max}}$ can be

Algorithm 1 Dense slew graph construction.

```
1: $V \leftarrow \{v_0\}$
2: $A \leftarrow \{v_0\}$
3: $E \leftarrow \{\}$
4: $W \leftarrow \{\}$
5: for $v_i \in A$ do
6:     for $\rho_n \in R$ where $\exists\, w \in W_n$ s.t. $\tau^c > t_i$ and $\Delta t_s(v_i, \boldsymbol{r}_n) < \tau^c - t_i$ do
7:         $t_j \leftarrow \text{clamp}(\text{ceil}(t_i + \Delta t_s(v_i, \boldsymbol{r}_n), \Delta t), w)$
8:         $v_j \leftarrow (\rho_n, t_j)$
9:         if $v_j \notin V$ then
10:            $V \leftarrow V \cup \{v_j\}$
11:            $A \leftarrow A \cup \{v_j\}$
12:        end if
13:        $E \leftarrow E \cup \{(v_i \rightarrow v_j)\}$
14:        $W \leftarrow W \cup \{r_j\}$
15:    end for
16:    $A \leftarrow A \setminus \{v\}$
17: end for
```

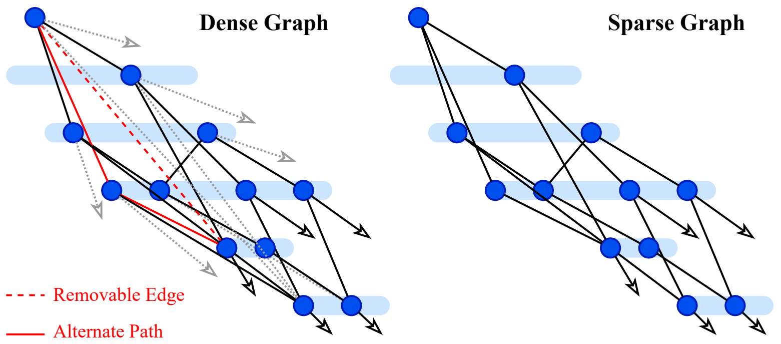


Figure 4: Comparison between the dense graph with one removable edge highlighted and the sparse graph.

identified from the function approximator's training data. If the soonest upcoming window opening $\tau^o > t_i + \Delta t_{s,\text{max}}$, the slew is known to be possible with an arrival time of τ^o without evaluating Δt_s.

Minimal Graph Construction Many of the edges in the graph produced by Algorithm 1 are unnecessary as they can be expressed as the composition of other edges. Consider a graph with the edges $v_i \rightarrow v_j \rightarrow v_k$ and $v_i \rightarrow v_k$. In the context of this problem, the latter can be deleted from the graph: At best, the three-vertex path is more valuable with the inclusion of an extra request and at worst, the middle request has already been completed and is worth nothing. By removing any edge that is expressible as the composition of other edges, a majority of edges can be eliminated thus greatly decreasing the size of the planning space; the only consideration this introduces is that the optimal path through the graph may encounter one request multiple times, but should only be rewarded for it once. A diagram showing removable edges is given in Figure 4.

Algorithm 2 Time-dependent reward graph construction, inserted after line 5 of Algorithm 1.

```
1:  for $t \in t_i : \Delta t : \tau^c$ do
2:      if $r_i(t) > r_i(t_i)$ then
3:          $v_j \leftarrow (\rho_i, t)$
4:          if $v_j \notin V$ then
5:              $V \leftarrow V \cup \{v_j\}$
6:              $A \leftarrow A \cup \{v_j\}$
7:          end if
8:          $E \leftarrow E \cup \{(v_i \to v_j)\}$
9:          $W \leftarrow W \cup \{r_i(t)\}$
10:         break
11:     end if
12: end for
```

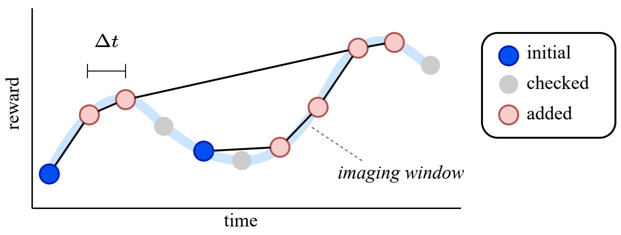


Figure 5: Additional vertices added for time-dependent rewards.

In practice, there is a faster way to construct a sparse graph than constructing the dense graph and removing unnecessary edges. It is known that there is some $\Delta t_{s,\max}$, the longest duration to slew from any reasonable dynamic state to settled target pointing. When adding edges and vertices from some vertex v_i, slews need only to be added from the arrival time of the shortest feasible slew $t_{j,\min}$ to $t_{j,\min} + \Delta t_{s,\max}$; any requests with windows beyond the latter time are know to be reachable by v_i via $v_{j,\min}$. This restriction can be added in line 6 of Algorithm 1. The result may contain some edges that are the composition of other edges if multiple slews in a row are shorter than $\Delta t_{s,\max}$. Despite this, the goal of a sparser graph is achieved since at every vertex slews are only added for requests within a fixed duration, so the total number of edges is quadratic complexity $\mathcal{O}(|R|^2)$ instead of cubic complexity $\mathcal{O}(|R|^3)$ as is the case for the fully-connected graph.

Time-Dependent Reward Construction In cases with time-dependent rewards, a denser graph must be constructed that includes additional vertices at points subsequent to existing vertices that yield a greater reward. Algorithm 2 describes the process: In short, for each active vertex it is checked if there is a time (discretized by Δt) in the current window that yields a greater reward than the current vertex's time. If so, a new vertex is added and connected to the graph and marked as active. Figure 5 illustrates how an imaging window with time-dependent rewards would have additional vertices added to the graph to make higher-valued times accessible.

Mixed-Integer Program-Based Path Maximization

The MIP formulation of graph traversal is based on formulations of the traveling salesman problem [17]. For each edge between vertices v_i and v_j, a binary variable $x_{i,j}$ is created, where $x_{i,j} = 1$ represents including the transition between the two vertices in the final sequence and $x_{i,j} = 0$ repre-

Table 1: Comparison of MIP formulation complexities.

	Graph $\|V\|$	**Graph $\|E\|$**	**Binary Variables**	**Optimal**	**Time-Dep. Rewards**
MIP-1	$\mathcal{O}(\|R\|)$	$\mathcal{O}(\|R\|^3)$	$\|E\| \in \mathcal{O}(\|R\|^3)$	✓	expensive
MIP-2	$\mathcal{O}(\|R\|)$	$\mathcal{O}(\|R\|^2)$	$\|R\| + \|E\| \in \mathcal{O}(\|R\|^2)$	✓	
MIP-3	$\mathcal{O}(\|R\|)$	$\mathcal{O}(\|R\|^2)$	$2\|E\| \in \mathcal{O}(\|R\|^2)$	✓	✓

sents exclusion. If the graph does not contain the edge $(i, j) \notin E$, $x_{i,j}$ can be set to a constant zero, reducing the number of variables in the optimization space. Vertices are partitioned into $p_n \in P$ by target request, where

$$p_n = \{v_i \mid v_i = (\rho_i, t_i) \in V, \rho_i = \rho_n\} \tag{13}$$

These partitions allow for non-repetition constraints to be added, allowing for fulfilledness status (F vs. U) to be respected when computing rewards.

Three formulations of the MIP are proposed. Table 1 summarizes and compares the different methods in terms of complexity and capabilities.

MIP-1: Fully Connected with Partitions The first formulation is a naïve approach that searches over all possible transitions (i.e. the dense graph). To enforce non-repetition, a valid sequence may only visit each partition once. Formally,

$$\text{maximize} \quad \sum_{i,j} w_{i,j} x_{i,j} \tag{14}$$

$$\text{subject to} \quad \sum_{h} x_{h,i} + b_i \geq \sum_{j} x_{i,j} \quad \forall\, i \tag{15}$$

$$\sum_{j \in p_n} \sum_{i} x_{i,j} \leq 1 \quad \forall\, n \tag{16}$$

where

$$b_i = \begin{cases} 1 & \text{if } i = i_{\text{start}} \\ 0 & \text{else} \end{cases} \tag{17}$$

The objective function, Equation 14, aims to maximize the sum of rewards for traveled edges, recalling that the weight $w_{i,j}$ of edge $(v_i \to v_j)$ is equal to the reward for visiting v_j. The first constraint, Equation 15, 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 (at all other points) to the number of incoming connections. The one exception is at the initial vertex, where the b_i term is set to one to seed the graph. Since the graph is acyclic, the summation on the LHS must be zero, reducing the constraint to

$$1 \geq \sum_{j} x_{i_{\text{start}},j} \tag{18}$$

As a consequence, all other vertices are restricted to one incoming and one outgoing edge selected, producing a feasible sequence. The second constraint, Equation 16, enforces at most single visitation of each partition, meaning each request is fulfilled at most once.

While it would be possible to account time dependence with this formulation by applying the previously described method to the graph, this would lead to optimization problems of an unacceptable size. Peng [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.

MIP-2: Minimally Connected with Partitions The second formulation makes use of the minimal form of the graph, where edges are not included if an alternate path exists. While this significantly reduces the number of variables, it introduces a problem in which the optimal sequence may have to pass through an already-visited partition to reach further areas of the graph. To handle this, an additional binary optimization variable y_n is added for each partition p_n, which represents receiving reward for visiting the partition at least once. In this formulation, only time-independent rewards are possible.

$$\begin{aligned} \text{maximize} \quad & \sum_n r_n y_n & (19) \\ \text{subject to} \quad & \sum_h x_{h,i} + b_i \geq \sum_j x_{i,j} \quad \forall\, i & (20) \\ & y_n \leq \sum_{j \in p_n} \sum_i x_{i,j} \quad \forall\, n & (21) \end{aligned}$$

The objective function, Equation 19, maximizes the sum of rewards for each partition visited. The first constraint, Equation 20, is the same in-out constraint as in the previous formulation. Instead of limiting the sequence to visiting each partition once, Equation 21 enforces that the rewarding variable y_n is allowed to be 1 only if the partition p_n is visited at least once. Solutions produced by this method are equivalent to those from the naïve method.

As listed in Table 1, MIP-2 is in theory significantly better than MIP-1 because the size of the graph and thus the number of binary optimization variables varies with the number of requests squares instead of cubed. Because of this, it is expected that MIP-2 should be more scalable to large target sets than MIP-1.

MIP-3: Minimally Connected with Time-Dependent Rewards The third formulation allows for optimizing over the minimal graph, and if the graph is modified as previously described, allows for optimization with time-dependent rewards. An additional binary optimization variable $y_{i,j}$ is added for each $x_{i,j}$. While $x_{i,j}$ represents passing through a vertex, $y_{i,j}$ represents being rewarded for the imaging action at v_j.

$$\begin{aligned} \text{maximize} \quad & \sum_{i,j} w_{i,j} y_{i,j} & (22) \\ \text{subject to} \quad & \sum_h x_{h,i} + b_i \geq \sum_j x_{i,j} \quad \forall\, i & (23) \\ & y_{i,j} \leq x_{i,j} \quad \forall\, i, j & (24) \\ & \sum_{j \in p_n} \sum_i y_{i,j} \leq 1 \quad \forall\, n & (25) \end{aligned}$$

The objective function, Equation 22, maximizes the sum of rewards for visited imaging actions. Time-dependent rewards are supported because rewards are computed on a per-action basis, so the best imaging time can be selected. Equation 23 is the familiar in-out constraint. Equation 24 restricts imaging actions to edges that are being visited. Finally, the constraint in Equation 25 only allows for one imaging action per partition.

While this formulation is more capable than MIP-2, it uses nearly double the binary variables. As per Table 1, the number of variables vary with $|R|^2$ like MIP-2 as opposed to $|R|^3$ like MIP-1, so MIP-3 is still expected to be scalable to large target sets.

RESULTS

First, the configurations of the satellite, environment, and simulation used for results in this paper are reviewed, noting where greater generality could be applied. Then, the accuracy of the slew transition time estimator is reviewed for three control laws to show general applicability. Next, the performance of the three formulations of the MIP solver are compared, showing competitiveness with other approaches to the problem. Finally, the optimal solutions are verified in the high-fidelity simulation environment, proving that this method is performant under realistic conditions.

Scenario Simulation

The scenario described in the problem formulation is modeled using Basilisk[*] [18], a high performance spacecraft simulation framework written in C++ and Python. The LEO environment and satellite flight software and dynamics are integrated at 2 Hz. Of particular relevance to this work, attitude dynamics are modeled to full fidelity, with flight-proven control software driving reaction wheel models that control the spacecraft attitude. The complete simulation environment is available in the Basilisk RL repository[†].

Satellite Configuration In this paper, the satellite is modeled as having an instrument with a boresight pointed in a body-fixed ${}^{\mathcal{B}}\hat{\boldsymbol{c}}$ 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. Three control schemes are considered:

- **MRP Steering:** An exponentially stable controller that generates and tracks a trajectory in modified Rodrigues parameter (MRP) space [19] is primarily used for this paper. Since the trajectory planner prescribes body rates, maneuvers about any axis are identical, thus satisfying the requirement of being kinematically the same for any rotation about the boresight.

- **MRP Feedback:** A PD feedback controller in MRP space [20] that is compensated for moments of inertia (thus satisfying the about-boresight identicality requirement) is implemented to show the methods in this paper working for multiple controller types.

- **MRP Steering (Saturated):** The steering controller is again implemented, but with reaction wheels that torque-saturate at $u_{\text{max}} = 0.3$ Nm, or about 5% of the time during typical maneuvers. With this controller, the assumption that control is identical for any about-boresight rotation is broken when saturation occurs on one axis. This case allows the performance of assumption-breaking systems to be considered.

The satellite's orbit is circular, set with a fixed inclination and altitude and randomized true anomaly and ascending node. At most one day of planning (15 orbits) is considered, as is typically considered to be the practical limit for non-adaptive 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.

[*]hanspeterschaub.info/basilisk
[†]github.com/AVSLab/bsk_rl

Table 2: Satellite and orbit parameters.

Parameter	**Value(s)**
Horizon O	$\leq$ 1 day (15 orbits)
Inclination	45°
Altitude	500 km
Controller	{steering, feedback, steering (saturated)}
$\delta\theta$	0.01 MRP norm (2.29°)
$\delta\omega$	0.01 rad/s (0.57°/s)
u_{max}	$1.0\ \text{N} \cdot \text{m}$
$m, \boldsymbol{I}$	330 kg, $[82.1, 98.4, 121.0]$ kg $\cdot$ m^2

Table 3: Target parameters.

Parameter	**Value(s)**	
ϕ_{min}	45°	minimum elevation angle
r	$[0, 1]$	request value
$\lvert R\rvert$	$[0, 10000]$	number of requests
Distribution	{uniform, cities}	

Target Distributions Representative sets of target requests are generated for experiments. Two distributions of target locations across various total target counts are considered, as show in Figure 6: a uniform distribution around the globe, and a clustered distribution in which city locations[‡] In the latter case, the satellite can travel for half an orbit without encountering any targets, then have hundreds available in a few minutes over very populous areas.

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

Target 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:

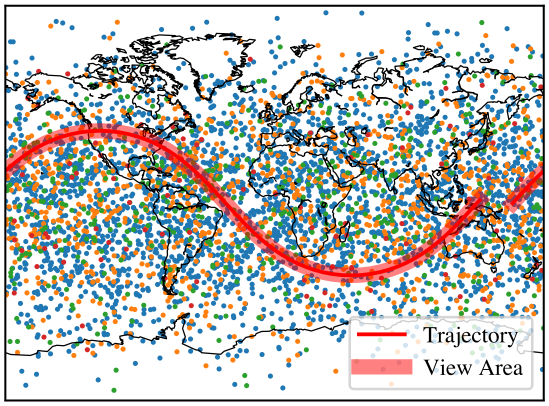
$$r_n(\phi) = r_n \frac{\phi}{\pi/2} \tag{26}$$

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

Slew Transition Time Estimation

Network Creation For each controller, N training points are generated. To produce them, an instance of the simulation environment is instantiated. Random upcoming requests are tasked. The satellite's state $\{\hat{\boldsymbol{c}}, \boldsymbol{\omega}_{\mathcal{C}/\mathcal{P}}, \boldsymbol{r}_s, \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{r}_s\}$ is recorded at 10 second intervals; once a request is satisfied (considering only pointing and rate requirements and neglecting opportunity window validity), the time-to-imaging from each previous state is computed, and each tuple of state and remaining slew time is added as a training data point.

[‡]City location data from simplemaps.com serve as a proxy for image request frequency, inspired by Eddy [8]., CC BY 4.0.



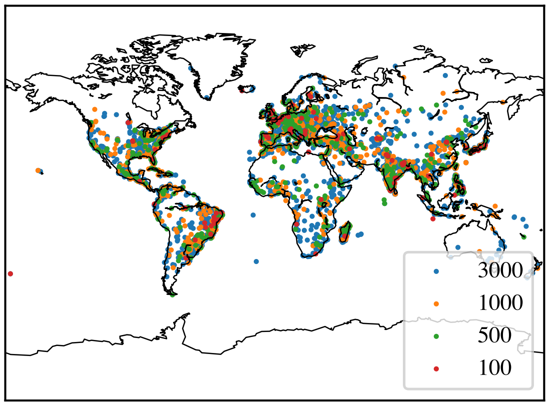


Figure 6: (Left) Examples of uniformly distributed imaging requests with the satellite field of view highlighted for one orbit. (Right) Imaging requests distributed across the world's 43,000 most populous cities.

Table 4: Slew estimation network hyperparameters.

Parameter	**Search Range**	**Value**
width	$[20, 120]$	120
depth	$[1, 15]$	7
batch size	$[1 \times 10^4, 1 \times 10^5]$	1×10^5
learning rate	$[0.0001, 0.01]$	0.01
loss	{MSE, MSLE, % error}	MSE
activation	{relu, gelu, tanh}	relu
use errors	{true, false}	true

A hyperparameter search is performed over the values given in Table 4 to determine a suitable network architecture for the slew transition time estimator. The hyperparameter space is searched using a Gaussian process, generating an estimation of each parameter's impact on the model quality. To increase the speed of the search, training in the hyperparameters search was performed on a randomly selected 10% segment of the training data for the unsaturated steering controller. The search confirmed that the precomputation of angle and rate errors within the network is necessary for good performance, but the network is otherwise robust to a range of hyperparameters as long as it is sufficiently large. When training networks, the learning rate is decreased by $0.9\times$ if validation loss improves by less than ΔL over 50 epochs. Training is terminated if an improvement of ΔL is not seen over 500 epochs, and the best network weights are returned. Since each loss function scales differently, ΔL is 0.1 for MSE, 0.005 for MSLE, and 1.0 for percent error.

The training data for each controller is regressed over using the network architecture described in Table 4, resulting in the performance given in Table 5. The two controllers that adhere to the necessary assumptions on about-boresight rotation agnosticism – steering and feedback – successfully learn to predict slew times with 99th percentile errors on the order of seconds. The saturated steering controller largely performs well because the assumptions are only occasionally violated; however, the model is unable to properly estimate the minority of slews where torque saturation makes the maneuver take significantly longer than without saturation. Presumably, giving the estimator full attitude instead of just pointing direction would alleviate this issue as it could learn which axes lead

Table 5: Slew estimation network performance for each controller. Percentile errors are computed for the absolute value of the difference between the predicted and actual slew duration over validation data (20% of N).

	Training Data		Percentile Error [s]				
Contoller	$\Delta t_{s,\max}$	N	50%	90%	99%	99.9%	Max
Steering	77.5	1.18×10^5	0.93	2.72	4.87	7.44	14.76
Feedback	91.0	1.15×10^5	0.89	4.57	10.60	20.46	26.46
Steering (saturated)	142.0	1.17×10^5	0.84	3.03	14.42	46.76	107.84

to highly-saturated maneuvers, but the addition of complete attitude information is incompatable with the graph construction algorithm.

Network Deployment The performance of the slew estimator is validated in the simulation environment. To do so, a greedy policy is defined that selects the next action to be the closest upcoming request which the slew estimator predicts to be reachable before the observation window closes:

$$a = \arg\min_{\rho \in U} \tau^c \text{ s.t. } t + \Delta t_s(\boldsymbol{x}, \rho) < \tau^c \tag{27}$$

The satellite attempts to fulfill the selected request until it is successful or the observation window closes. This greedy policy is selected as a stress test for performance because it encourages the satellite to aggressively attempt short slews that are beneficial in a planning context but can be risky without a good estimator.

This experiment also serves to find the margin t_m (i.e. the amount of time to add on top of the estimator's predictions) that is best for performance. Intuitively, too small of a margin will lead to some unsuccessful imaging attempts where in which the actual transition time is slightly underestimated. On the other hand, too large of a margin will be overconservative and cause the satellite to not attempt some shorter feasible slews, reducing performance. For this experiment, margins are tested at the percentile error values in Table 5.

The policy is deployed in a uniformly distributed 10000 request environment for one day of simulation. Figure 7 displays the results as a function of percentile error used for the margin and the actual margin duration. The upper plots show the percent of imaging attempts that succeed; the lower plots show the relative count of attempted and successful actions, normalized by the count of zero-margin attempts for each controlled. The data follows the predicted trends: lower margins lead to many attempts with reduced success rates, while higher margins result in perfect success rates but fewer targets imaged. For the steering controller, the 90th percentile error (2.72 seconds) maximizes the number of successes while minimizing failures. In general, some margin is necessary to optimize performance, but it varies from case to case.

Optimal Sequencing

The graph construction algorithm is implemented in Julia [21], and the MIP formulations are solved using Gurobi [22]. First, the sensitivity of the solutions to the time discretization is investigated. Then, the relative performance of the MIP formulations are compared, showing that MIP-2 is the superior formulation for fixed-value targets. MIP-2 is then benchmarked over a range of parameters, with solutions validated in the simulation environment to demonstrate the performance of the entire toolchain. Finally, time-dependent reward cases are solved with MIP-3.

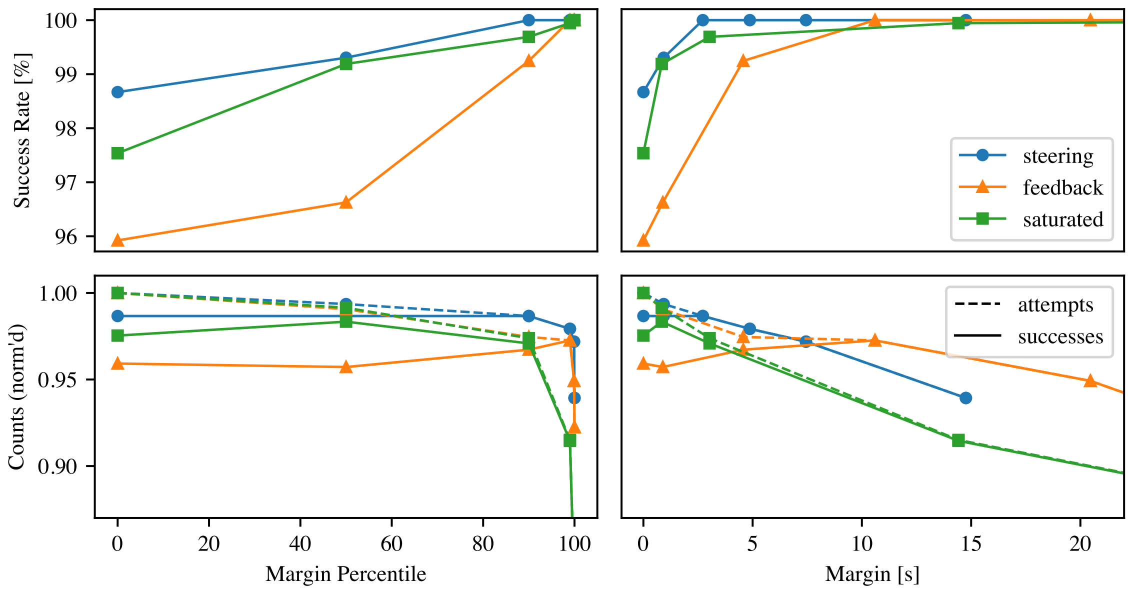


Figure 7: Greedy policy performance using the slew duration estimator for each controller. For purposes of comparison, counts are normalized against each controller's number of attempts with zero margin.

Table 6: Total execution time and reward as a function of target density and time discretization Δt, with marginal reward loss for increased Δt reported as a percentage. Three orbits, uniform distribution.

	Time [s]				**Reward $\Sigma\rho$**						
$\lvert R\rvert$	$\Delta t = 1$	5	10	30	1	$\Delta\%$	5	$\Delta\%$	10	$\Delta\%$	30
100	1.2	1.1	1.1	1.1	10.0	*0*	10.0	*0*	10.0	*0*	10.0
500	2.8	2.6	2.5	2.5	45.9	*-0.7*	45.6	*0*	45.6	*-1.5*	44.9
1000	6.4	5.0	4.7	4.3	86.3	*0*	86.3	*-0.3*	86.0	*-0.4*	85.6
3000	972.5	39.7	19.8	9.8	170.2	*-1.6*	167.4	*-2.3*	163.5	*-7.5*	151.3
10000	$> 10^4$	21422.0	502.0	81.2	-	-	262.8	*-3.3*	254.0	*-7.2*	235.6

Table 7: Graph construction and MIP solver times for each MIP formulation, with associated data structure sizes. Three orbits, uniform distribution.

	Graph Time [s]			**Vertices $\|V\|$**			**Edges $\|E\|$**			**MIP Time [s]**			**Binary Variables**		
$\|R\|$	MIP-1	MIP-2	MIP-3	MIP-1	MIP-2	MIP-3	MIP-1	MIP-2	MIP-3	MIP-1	MIP-2	MIP-3	MIP-1	MIP-2	MIP-3
100	0.9	1.2	6.3	42	17	17	279	16	16	0.019	0.003	0.014	279	57	32
500	1.5	2.6	3.6	315	293	293	16279	567	657	0.63	0.019	0.033	16279	675	1134
1000	1.8	4.2	4.1	690	662	662	69395	1379	1379	4.2	0.059	0.088	69395	1563	2758
3000	24.1	14.4	16.3	6218	6196	6196	1.8×10^6	39697	39697	322.5	5.7	9.5	1.8×10^6	40232	79394

Discretization Sensitivity The impact of the granularity of the time discretization Δt on execution time and solution quality is explored in Table 6. Over a range of uniform request counts $|R|$, the execution time and total reward for each parameter set are reported, as well as $\Delta\%$, the percent loss of reward due to changing the timestep to a courser value. Smaller discretizations are found to produce marginally better solutions at the expense of exponentially longer solution times.

Ultimately, $\Delta t = 10$ seconds is selected for all further experiments; this yields solution times on the order of seconds that are less than a percent worse than finer discretizations for $|R| \leq 1000$, and solution times on the order of hundreds of seconds for dense uniform distributions for $|R| = 10000$ with only a few percent loss of reward, noting that the finer discretizations become prohibitively expensive at these densities.

Formulation Comparison The performance of the three MIP formulations are compared in Table 7, with each cases planning for three orbits over $|R|$ uniformly distributed targets. The time to construct the graph for each method, as well as the number of edges and vertices in the resulting graph are listed. The time to solve the MIP is given alongside the number of variables in the MIP. It is seen that MIP-2 is one to two orders of magnitude faster than the naïve approach of MIP 1 in all cases, which is directly correlated with smaller graphs and sets of decision variables. MIP-2 is also faster than MIP-3, indicating that MIP-3 should only be used when it is necessary in cases with time-dependent rewards.

Time-Independent Reward Benchmarks With MIP-2, established as the most performant method for time-independent rewards, it is benchmarked across a range of target densities, distributions, and planning horizons to demonstrate its performance and limitations. The time discretization is set to $\Delta t = 10$ seconds, and the target optimality gap for the solver is set to 1% with an alternative 1000 second timeout. Each solution is then validated in the simulation environment, reporting the percentage of requests in the planned solution that the system was able to successfully complete in the full-fidelity simulation.

Figure 8 gives results for uniformly distributed targets, averaged over multiple trials. The size of the problem is determined by the number of requests $|R|$ and planning horizon in orbits O; a subset of these requests $|R_{\text{track}}|$ are accessible from the orbital track and yielding some number of imaging opportunity windows $|W|$. These problem size parameters are given in the upper left subplot. Since the results produced by this method are optimal up to the time discretization, the solutions statistics follow intuitive trends. In the lower left subplot, the fulfillment rate $F_{\%} = |F|/|R_{\text{track}}|$ is given,

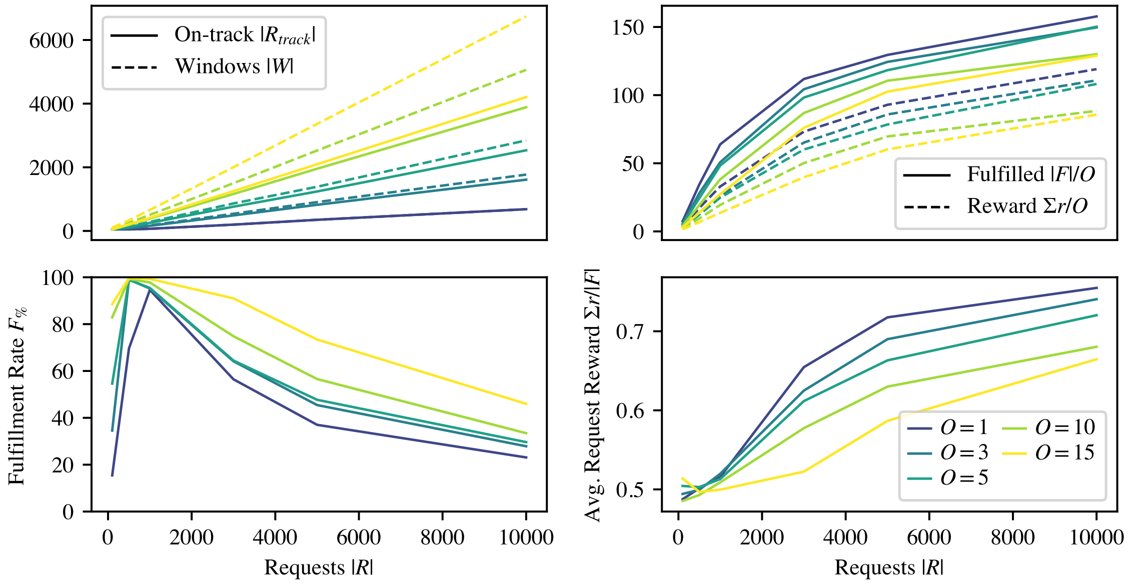


Figure 8: Planning results using MIP-2 over uniformly distributed targets.

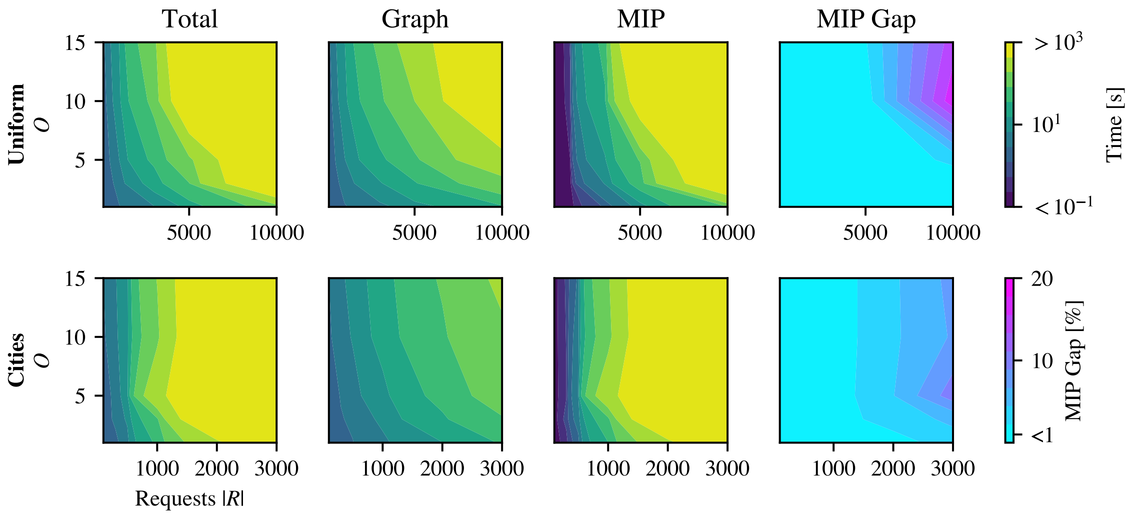


Figure 9: Execution time for each request distribution.

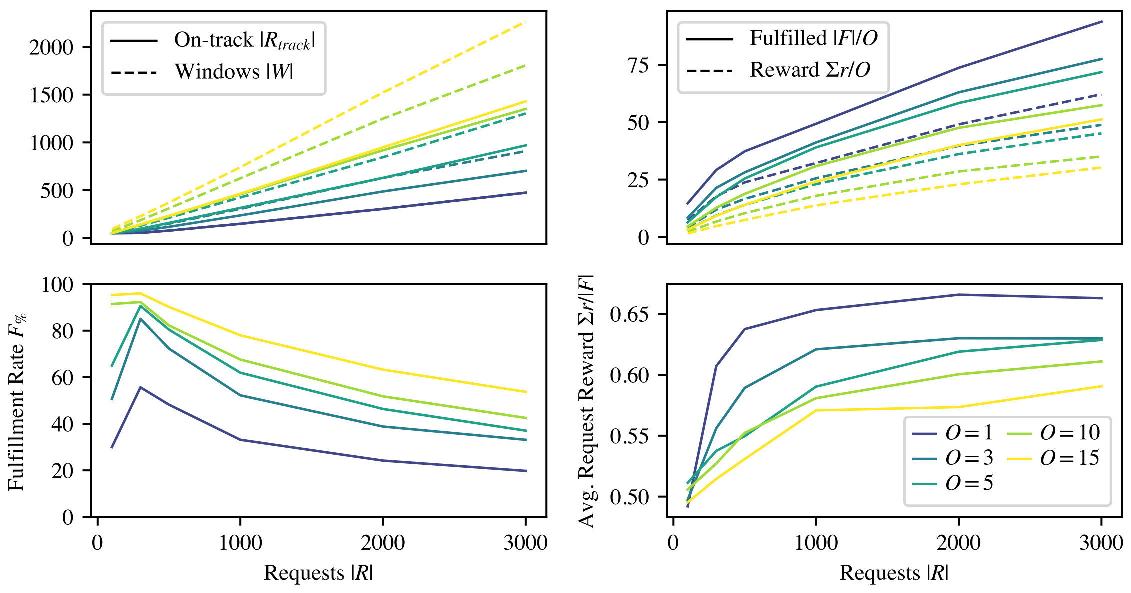


Figure 10: Planning results using MIP-2 over the city-based target distribution.

which expresses the percent of possible requests fulfilled. Fulfillment rates are highest for sparse targets over a long planning horizon, as there are many opportunity windows to collect relatively few targets. The upper right subplot gives the orbit-normalized number of requests fulfilled $|F|/O$ and the orbit-normalized reward $\Sigma r/O$. These values intuitively increase with denser target sets; however, longer planning horizons lead to lower fulfillment numbers per orbit as the options for targets become exhausted. The same effect is observed in average reward per fulfilled request $\Sigma r/|F|$: at low densities or long planning horizons, the planner is less able to be selective about choosing only high-value targets, so the average target reward is closer to the mean value of 0.5.

In the upper row of Figure 9, the total execution time for each plan is decomposed into the time taken to construct the graph and the time used by the MIP solver. For uniform distributions, the solver can compute day-long plans for $\leq$ 3000 targets and up to a 3 orbit long plans for denser target sets within the 1000 second timeout. Because the solver terminates if the 1% optimality gap is not reached before the timeout, the gap at solver timeout is also given. For the largest instances, the solution is up to 20% suboptimal at timeout (though the actual suboptimality could be lower). Alternative to accepting a wider optimality gap, long-duration planning for high target densities could instead be decomposed into a suboptimal sequential planning problem where every few orbits are solved as an independent problem without attempting to optimize across segments.

A similar experiment is conducted for requests with a city-based distribution, with results given in Figure 10. Due to the highly non-uniform target distribution, the same initial orbit conditions are selected for every simulation so that it is guaranteed to cover high density areas. The trajectory passes over Columbia, Europe, India, and New South Wales over the course of the first orbit. Request locations are still randomized for each trial. The city-based distribution makes for a very challenging planning environment. Requests occur in only 10-20% of satellite view area, resulting in an effective $5-10\times$ multiplier of target density for the same $|R|$ between uniform and city-based distributions in areas with targets. Thus in the minority of the time when targets are available, there tend to be many choices. This means that the the average reward per target $\Sigma r/|F|$ is higher for the city-based distribution than for the uniform distribution when considering the same number of requests. While satisfying higher-quality requests is easier due to the greater selection when targets are present, this also leads to a lower fulfillment rate $F_{\%}$. Thus, the fulfillment rate acts as a proxy for the amount of decisions the solver must make between conflicting targets. This proxy behavior is evident when comparing the fulfillment rates of the uniform and city distributions: the city rates decay $5-10\times$ faster with respect to $|R|$, similar to the effective target density factor.

The lower row of Figure 9 shows that the execution times for the city distribution depend on $|R|$ and the planning horizon more strongly than for the uniform case, with the higher effective density leading to longer planning times. The practical limit for request count and planning horizon are reduced relative to uniform requests by a factor similar to the difference in effective target density and fulfillment rate: It is reasonable to optimally plan for $|R| = 1000$ for a one-day horizon for cities, while for a uniform distribution $|R| = 5000$ takes a comparable time. The large region of problem configurations with a MIP optimality gap $> 1\%$ (i.e. where the solver timed out) explains the decrease in average request reward $\Sigma r/|F|$ for higher request counts in the lower right subplot Figure 10; if the solver was allowed more time to approach optimality, the expected increasing trend of average target value with respect to request count would be continued. Since graph construction remains relatively cheap even for the city-based distribution, other methods of solving the graph could be considered for larger instances.

To verify solutions, plans produced by the MIP solver are tasked in the full-fidelity spacecraft simulator. For uniformly distributed targets, 99.9% of planned requests are completed successfully; this is in line with expectations from the greedy policy success rate evaluations. For the city-based distribution, a similarly high success rate of 98.7% is observed; this rate is slightly lower because more challenging slews are attempted in dense areas. Thus, it is shown the abstract planning framework is applicable to a lifelike scenario.

Time-Dependent Reward Benchmarks Planning is demonstrated using MIP-3 for requests with elevation angle-based time dependent rewards as defined by Equation 26. Results are given in Figure 11 for a uniform target distribution. The number of requests fulfilled is comparable to that for uniformly distributed targets with time-independent rewards. However, the reward achieved is significantly lower, which is expected as the maximum request reward is only possible when imaging a target that is directly nadir. For longer planning horizons with few requests this impact is minimized, as each target is likely to have multiple opportunity windows so the one with the most advantageous imaging angle may be selected. However, as the number of requests increases the typical trend of shorter horizons capturing — on average — more valuable targets returns as the potential for greedier behavior on a short horizon again dominates. Planning with time-dependent rewards is slower, by virtue of both larger graphs and more decision variables in the MIP formulation.

CONCLUSION

This paper presents two novel developments for the AEOS scheduling problem: 1) a framework for representing the problem with more accurate transition dynamics, which uses a neural network function approximator to accurately represent transition times; and 2) more compact and efficient MIP formulations to make the problem tractable even for relatively large instances. These developments address the challenges of complex transition dynamics and combinatorially large solution spaces present in the problem.

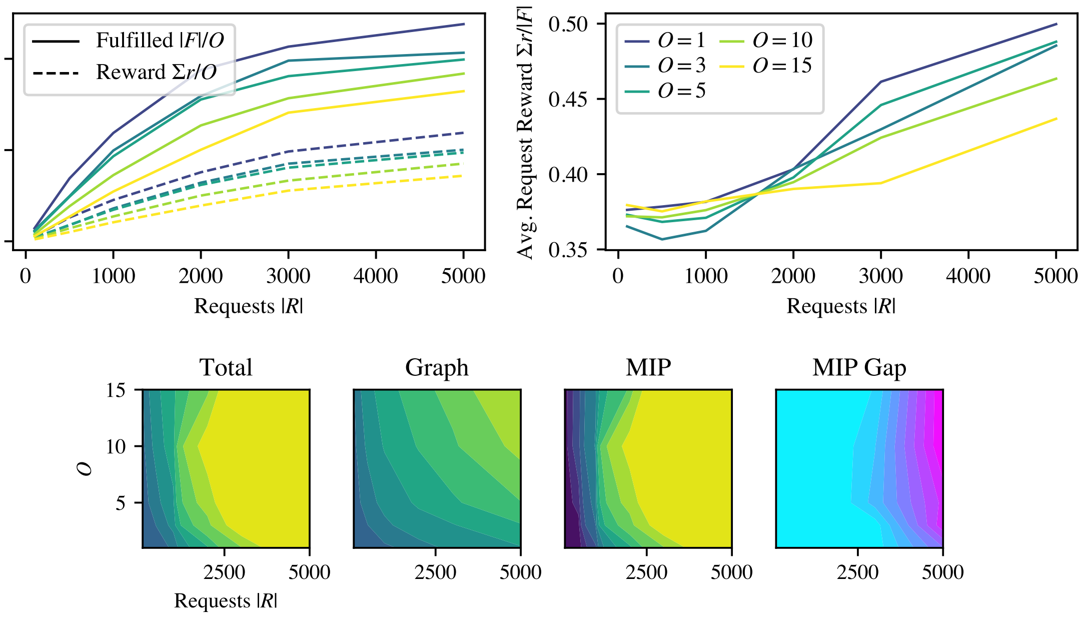


Figure 11: Planning results for MIP-3 over uniform targets with time-dependent rewards. Contour scaling same as previous plots.

These methods are implemented, the sensitivity to parameters is explored, and benchmarks are performed across a range of problem sizes. Primarily, the method is tested for up to day-long planning horizons, up to 10000 imaging requests, and for uniform and highly clustered distributions of requests. The size at which the problem becomes intractable for the methods presented is found to dependent on target distribution and other factors, and the limiting sizes are clearly given. Finally, the optimal sequences are validated by successfully executing them in a high-fidelity spacecraft simulation, showing that methods are applicable for real-world use. Because of the optimality guarantee (up to a time discretization, transition estimator quality, and optimality tolerance) of the MIP solver, the methods presented can serve as a baseline for comparison for other algorithms, such as iterative local search or various reinforcement learning (RL) approaches. In particular, this work can be used to benchmark RL-based schedulers for the AEOS problem similar to those demonstrated by Herrmann in [23] and [24].

Beyond being used as an optimality benchmark for other methods, areas for future work include finding heuristic solutions to "warm start" the MIP solver and developing methods to adaptively set the time discretization points to minimize optimality loss.

ACKNOWLEDGEMENT

This work is supported by the Air Force Research Lab grant FA9453-22-2-0050.

REFERENCES

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

[2] S. Ungar, J. Pearlman, J. Mendenhall, and D. Reuter, "Overview of the Earth Observing One (EO-1) Mission," *IEEE Transactions on Geoscience and Remote Sensing*, Vol. 41, June 2003, pp. 1149–1159, 10.1109/TGRS.2003.815999.

[3] F. Spoto, O. Sy, P. Laberinti, P. Martimort, V. Fernandez, O. Colin, B. Hoersch, and A. Meygret, "Overview Of Sentinel-2," *2012 IEEE International Geoscience and Remote Sensing Symposium*, Munich, Germany, IEEE, July 2012, pp. 1707–1710, 10.1109/IGARSS.2012.6351195.

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

[5] M. Safyan, "Planet's Dove Satellite Constellation," *Handbook of Small Satellites* (J. N. Pelton and S. Madry, eds.), pp. 1057–1073, Cham: Springer International Publishing, 2020, 10.1007/978-3-030-36308-6.

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

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

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

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

[10] M. Lemaître, G. Verfaillie, F. Jouhaud, J.-M. Lachiver, and N. Bataille, "Selecting and scheduling observations of agile satellites," *Aerospace Science and Technology*, Vol. 6, Sept. 2002, pp. 367–381, 10.1016/S1270-9638(02)01173-2.

[11] C. Verbeeck, K. Sörensen, E.-H. Aghezzaf, and P. Vansteenwegen, "A fast solution method for the time-dependent orienteering problem," *European Journal of Operational Research*, Vol. 236, July 2014, pp. 419–432, 10.1016/j.ejor.2013.11.038.

[12] X. Liu, G. Laporte, Y. Chen, and R. He, "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, 10.1016/j.cor.2017.04.006.

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

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

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

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

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

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

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

[20] H. Schaub, R. D. Robinett, and J. L. Junkins, "Globally Stable Feedback Laws for Near-Minimum-Fuel and Near-Minimum-Time Pointing Maneuvers for a Landmark-Tracking Spacecraft," *Journal of the Astronautical Sciences*, Vol. 44, Oct. 1996, pp. 443–466.

[21] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A Fresh Approach to Numerical Computing," *SIAM Review*, Vol. 59, Jan. 2017, pp. 65–98, 10.1137/141000671.

[22] Gurobi Optimization, LLC, "Gurobi Optimizer Reference Manual," 2023.

[23] A. Herrmann, M. Stephenson, and H. Schaub, "Reinforcement Learning For Multi-Satellite Agile Earth Observing Scheduling Under Various Communication Assumptions," *AAS Rocky Mountain GN&C Conference*, Breckenridge, CO, Feb. 2023.

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