Centralized and Distributed Strategies for Handover-Aware Task Allocation in Satellite Constellations

Joshua Holder,* Spencer Kraisler,† and Mehran Mesbahi‡
University of Washington, Seattle, WA 98195

https://doi.org/10.2514/1.G008363

As satellite constellations grow in size, there is an increasing need for autonomous, scalable, and real-time dynamic task assignment to address the unique operational challenges of such distributed systems. In particular, a time-varying task assignment (i.e., for observing various regions of Earth) often means that the corresponding satellite has to reorient itself or its sensors, costing time and energy. However, most assignment algorithms for area requests proposed in the literature do not account for the significant cost associated with task handover in satellite constellations. In this paper, we develop a framework for solving the seemingly non-deterministic polynomial-time (NP)-hard problem of optimal dynamic task allocation while minimizing task handover. In particular, we develop Handover-Aware Assignment with Lookahead (HAAL), an algorithm with centralized and distributed variants, and solutions that provably achieve 50% of the optimal value. We then proceed to show that HAAL significantly outperforms similar heuristic methods proposed in the literature for realistic constellation experiments with up to a thousand satellites. The algorithm scales polynomially in the number of satellites/tasks and offers a smooth tradeoff between computational efficiency and performance, allowing the designer to tune the algorithm based on available computing resources, communication bandwidth, and required performance.

I. Introduction and Prior Work

WITH the recent rise of large low-Earth-orbit (LEO) satellite constellations, there is an increasing need for automated allocation of tasks among the satellites. In this setting, a "task" is a subgoal necessary for achieving the overall objectives of the system that can be achieved independently of other such subgoals [1]. In the context of satellite constellations, tasks could include assigning satellites to Internet user terminals, aiming sensors at specific geographic regions, or observing objects in other orbits. While task allocation is a well-established research area in computer and information sciences [2], satellite constellations pose a new and distinctly challenging environment for optimization. In particular, unlike terrestrial communication networks or even traditional geostationary orbit systems, LEO satellite constellations are networks with dynamic nodes, time-varying links, and, in some cases, no centralized data fusion or coordinating entity.

The current literature on task allocation for satellite constellation has largely focused on networks with less than a hundred satellites [3–6]. In these settings, task allocation problems are often characterized by "point-based" tasks that need to be accomplished at a single epoch (i.e., take a single image of a location). This use-case is different from the "area requests" typical of LEO Internet mega-constellations or imaging sensor networks; in these aforementioned cases, there are often geographic regions or customers that the constellation aims to *consistently* serve, each of which could be served by one of many satellites in the area. For example, consider the task of connecting a specific user Internet terminal to a LEO mega-constellation. Connectivity needs to provided continuously, not just for one epoch, but the set of assignable satellites is time-varying, which means that greedily assigning the terminal to the closest satellite can be problematic: the task will often be reassigned at each time step due to the high speeds of the orbiting satellites. In the literature, each instance of a task being reassigned to a new satellite is called *handover* [7–9]. For every instance of task handover, the satellite must undergo a transient period of reorienting itself and its instruments and establishing a connection with the terminal, sometimes lasting up to five minutes [10].

For every task handover, the user terminal will experience degraded performance during the switch to the new satellite.

Despite a number of papers mentioning the importance of task handover [3,11–14], the time- and energy-consuming consequences are often overlooked in literature. In [14], the presented algorithm assigns the locations of thousands of users to the beams of satellites. However, satellites are then assigned to beams without examining performance requirements or minimizing the required number of handovers. In [15], beam assignments are chosen optimally by solving an integer program which does not account for handovers. Here, the authors note that this leads to a large amount of task switching in the solution.

Significant prior work has explored strategies that allow *users* to minimize handover between satellites in Internet constellations [16–18], but almost no other work addresses the reverse problem, ensuring that satellites handover tasks between themselves efficiently. Perhaps the most similar work in the literature is [19], where the authors apply reinforcement learning to maximize connection duration between satellites and user clusters. However, the algorithm proposed in the aforementioned work does not come with performance guarantees, cannot easily generalize across multiple realistic scenarios, and is entirely centralized.

Indeed, a majority of both theoretical [20] and constellation-specific [3,14,21] studies on task allocation focus on centralized algorithms. However, centralized control of satellites requires a costly ground network [22], is significantly less robust to satellite failures and other variations in the assignment problem, and faces scalability issues as the number of satellites increases [23]. These factors make distributed algorithms highly desirable in satellite constellation environments. However, existing works on distributed methods for satellite task allocation focus largely on small-scale constellations, often provide no guarantees of conflict-free assignments, and generally fail to handle the complexity of task handover in their formulations [24–26].

The main contribution of this paper is a fully distributed algorithm for addressing the handover-aware task allocation problem, opening the door for efficient and robust operations of next generation LEO mega-constellations to provide Internet service to or develop a dynamic sensor network for the entire Earth. In Sec. II, we

Received 20 March 2024; accepted for publication 17 March 2025; published online 11 April 2025.

*Graduate Research Assistant, W. E. Boeing Department of Aeronautics and Astronautics; josh.holder72@gmail.com.

†Graduate Research Assistant, W. E. Boeing Department of Aeronautics and Astronautics; kraisler@uw.edu.

‡J. Ray Bowen Professor, W. E. Boeing Department of Aeronautics and Astronautics; mesbahi@uw.edu. Fellow AIAA.

expand upon the overall problem formulation and discuss approaches that aim to solve analogous setups. In Sec. III, we outline the simplest version of the algorithm that we use to prove guarantees and build intuition. In Sec. IV we develop a more performant version of the algorithm, with both centralized and distributed variants, and test its performance with respect to the corresponding globally optimal assignments. In Sec. V, we test the proposed algorithms in operational scenarios spanning Earth observation and Internet constellations. Finally, in Sec. VI, we provide a summary of our contributions and discuss future research directions on dynamic task assignment for distributed space systems.

II. Problem Formulation

Consider a group of n satellites in a constellation with m tasks, over T time steps with $n \leq m$.[§] Let $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_T)$ be a sequence of T *benefit matrices* $\beta_k \in \mathbb{R}^{n\times m}$. Here, $[\beta_k]_{ij}$ represents the utility of satellite i performing task j at time k. Similarly, let $\boldsymbol{x} = (x_1, \ldots, x_T)$ be a sequence of T *assignment matrices* $x_k \in \{0,1\}^{n\times m}$, where $[x_k]_{ij} = 1$ if satellite i performs task j at time k and zero otherwise.[¶] Let $x_0 \in \{0,1\}^{n\times m}$ be the arbitrary initial assignment for the constellation. Now consider $\lambda \in \mathbb{R}$ as the *task handover penalty*, where λ is the operational cost of not assigning a satellite to the same task in two consecutive time steps (more on the interpretation of this parameter in our subsequent discussions). The problem of minimal-handover constellation task assignment can now be formalized as follows:

$$\max_{[x_k]_{ij}\in\{0,1\}} \mathcal{V}(\boldsymbol{x};\boldsymbol{\beta},\lambda,x_0) \coloneqq \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{T}[\beta_k]_{ij}[x_k]_{ij} - \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{T}\frac{\lambda}{2}\left([x_k]_{ij} - [x_{k-1}]_{ij}\right)^2 \quad (1a)$$

$$\text{s.t.} \sum_{j=1}^{m}[x_k]_{ij} = 1, \quad \text{for all } i,k \quad (1b)$$

$$\sum_{i=1}^{n}[x_k]_{ij} \leq 1, \quad \text{for all } j,k \quad (1c)$$

Here, the equality constraint (1b) implies that each satellite serves one task per time step, and (1c) implies that each task is assigned to at most one satellite per time step. The cost function (1a), referred to as the constellation operational *total value*, is split into two components: the *total benefit* from tasks assigned to the satellites and a *total handover penalty* penalizing instances of task handover.

The above constellation task assignment can be reformulated as a multi-assignment problem [1] or a Quadratic Assignment Problem (QAP) [27]. These problem classes however are non-deterministic polynomial-time (NP)-hard, where only heuristic solutions are achievable for realistic problem sizes. As such, our goal in this paper is developing an efficient and scalable algorithm that produces assignment matrices $\boldsymbol{x} = (x_1, \ldots, x_T)$; we subsequently show that this algorithm is theoretically within 50% of the optimal value and achieves significantly better empirical performance for large constellations. For general multitask assignment problems, there are xisting heuristic solutions achieving 50% of optimality [28]. Here, we will exploit the specific structure of the underlying operational cost to significantly improve empirical performance and efficiency on realistic problems.

Note that without a handover penalty and optimizing over a single time step, the problem reduces to the well-studied optimal assignment problem. That is, when $\beta \in \mathbb{R}^{n\times m}$ is the benefit matrix and

$$\mathcal{X} \coloneqq \left\{x \in \{0,1\}^{n\times m} : \sum_{j=1}^{m} x_{ij} = 1 \text{ (for all } i), \sum_{i=1}^{n} x_{ij} \leq 1 \text{ (for all } j)\right\}$$

is the set of all valid assignment matrices, the optimal assignment problem can be formalized as

$$\max_{x\in\mathcal{X}} \sum_{i=1}^{n}\sum_{j=1}^{m}\beta_{ij}\,x_{ij} \quad (2)$$

There is a rich research literature on the optimal assignment problem; in fact, it is known that this integer-constrained optimization admits a polynomial time solution in the centralized [20] setting. Optimal assignment has also been examined in the context of parallel [29] and distributed [30] settings. Although it might seem conceivable to approach minimal-handover task assignment via the more general QAP or multitask assignment formulations, we observe that our cost function consists of a series of assignment problems with quadratic coupling terms linking only adjacent time steps. As such, in this paper we proceed to develop an efficient and distributed solution strategy that exploits solutions to the simpler assignment problems (2).

III. Handover-Aware Assignment (HAA) Algorithm

In this section, we present a simple heuristic approach to problem (1); subsequently, we provide a 50% optimality guarantee of this proposed solution strategy for the minimal-handover task assignment problem. First, we define some relevant notation. The *assignment function* $\alpha: \mathbb{R}^{n\times m} \to \mathcal{X}$ will be the mapping from any benefit matrix to the assignment corresponding to the optimal solution of (2)

$$\alpha(\beta) \coloneqq \arg\max_{x\in\mathcal{X}} \sum_{i=1}^{n}\sum_{j=1}^{m}\beta_{ij}\,x_{ij} \quad (3)$$

Note that $\alpha(\beta)$ is efficiently computable for any β. Given benefit and assignment matrices, the *time step benefit* $B: \mathbb{R}^{n\times m} \times \mathcal{X} \to \mathbb{R}$ is the sum of the benefits over each assigned agent-task pair:

$$B(\beta, x) \coloneqq \sum_{i=1}^{n}\sum_{j=1}^{m}\beta_{ij}\,x_{ij}$$

The *handover penalty* $H: \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the penalty incurred by two consecutive assignments, x^- and x, and is proportional to the number of differences in the assigned agent-task pairs:

$$H(x^-, x) \coloneqq \frac{\lambda}{2}\sum_{i=1}^{n}\sum_{j=1}^{m}(x_{ij} - x_{ij}^-)^2 \quad (4)$$

Note that for any $x^-, x \in \mathcal{X}$, we have $(x_{ij} - x_{ij}^-)^2 = \text{XOR}(x_{ij}^-, x_{ij})$ and

$$0 \leq H(x^-, x) \leq n\lambda \quad (5)$$

Given a benefit matrix and two assignments, the *time step value* $V: \mathbb{R}^{n\times m} \times \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the total benefit subject to the handover penalty for the pair of assignments:

$$V(\beta, x^-, x) \coloneqq B(\beta, x) - H(x^-, x) \quad (6)$$

With this definition, we can reformulate the total value (1a) for the constellation as a sum of time step values:

$$\mathcal{V}(\boldsymbol{x}) = \sum_{k=1}^{T} V(\beta_k, x_{k-1}, x_k) \quad (7)$$

[§]If $n > m$, introduce $n - m$ "dummy" tasks with zero benefits.

[¶]As such, boldface letters are used to denote sequences over time.

Algorithm 1: Handover-Aware Assignment (HAA)

Given: $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_T), \lambda \in \mathbb{R}, x_0 \in \mathcal{X}$
1: **for** $1 \le k \le T$ **do**
2: $x_k \leftarrow \alpha(\beta_k + \lambda x_{k-1})$
3: **end for**
return $\boldsymbol{x} = (x_1, \ldots, x_T)$

Equation (7) suggests a natural approach for constellation task assignment. If, given a benefit matrix β and previous assignment x^-, we had a method of finding an assignment x optimal with respect to $V(\beta, x^-, x)$, we could construct a greedy sequence of assignments by using this x at each time step. This is the basic operating principle of Algorithm 1. In Lemma III.1, we will show that a simple method of finding such an x exists:

Lemma III.1: Fix $\lambda \in \mathbb{R}$ and let $\beta \in \mathbb{R}^{n\times m}, x^- \in \mathcal{X}$. Then

$$\alpha(\beta + \lambda x^-) = \arg\max_{x\in\mathcal{X}} V(\beta, x^-, x) \tag{8}$$

Proof: This follows by plugging in the definition of $V(\cdot)$ (6) and simplifying. See Appendix A. □

With Algorithm 1, we are essentially approximating the NP-hard problem of generating optimal solutions to (1) with the far easier problem of solving a sequence of standard assignment problems (2), which we do on Line 2. As such, Algorithm 1 does not necessary generate optimal solutions, as it makes greedy assignments with respect to the current assignment without considering subsequent time steps. However, we will see that Algorithm 1 performs well empirically and admits a suboptimality bound guarantee under certain conditions.

Before we proceed to show that Algorithm 1 generates solutions that are within 50% of the optimal, we first show that it yields assignments that contribute at least $n\lambda$ value at each time step and that those assignments achieve within $n\lambda$ as much time step value as the optimal solution:

Lemma III.2: Suppose $[\beta_k]_{ij} \ge \lambda$ and let $\boldsymbol{x}^* = (x_1^*, \ldots, x_T^*)$ be optimal assignments with respect to (1). Fix an arbitrary $x^- \in \mathcal{X}$ and set $x^\alpha \coloneqq \alpha(\beta_k + \lambda x^-)$. Then for all k we have,

$$V(\beta_k, x^-, x^\alpha) \ge n\lambda, \quad \text{and} \quad V(\beta_k, x^-, x^\alpha) + n\lambda \ge V(\beta_k, x_{k-1}^*, x_k^*)$$

Proof: Since $[\beta_k]_{ij} > \lambda$, we have

$$B(\beta_k, x) \ge n\lambda \quad \text{for all } x \in \mathcal{X} \tag{9}$$

In other words, all assignments provide at least $n\lambda$ benefit. Furthermore, when $x = x^-$, Equations 4 and (9) would imply,

$$V(\beta_k, x^-, x) = B(\beta_k, x) - H(x^-, x) \ge n\lambda$$

Thus, since x^α has been selected greedily with respect to value (Lemma III.1), we have

$$V(\beta_k, x^-, x^\alpha) \ge n\lambda, \quad \text{for all } k$$

Next, we will show that at each time step k, x^α yields within $n\lambda$ value of the optimal assignment (Intuitively, if $\alpha(\beta_k + \lambda x^-)$ ever yielded an assignment that was suboptimal by more than $n\lambda$, it could simply incur the maximum handover penalty of $n\lambda$ and switch to the better assignment.) First, rewrite the value gained at time step k by the optimal assignment x^* as

$$\begin{aligned} V(\beta_k, x_{k-1}^*, x_k^*) &= B(\beta_k, x_k^*) - H(x_{k-1}^*, x_k^*) \\ &= B(\beta_k, x_k^*) - H(x^-, x_k^*) \\ &\quad + \left[H(x^-, x_k^*) - H(x_{k-1}^*, x_k^*)\right] \end{aligned}$$

Using the bounds on H in Eq. (5), we can bound the difference between any two H evaluations as

$$\begin{aligned} V(\beta_k, x_{k-1}^*, x_k^*) &\le B(\beta_k, x_k^*) - H(x^-, x_k^*) + n\lambda \\ &= V(\beta_k, x^-, x_k^*) + n\lambda \end{aligned}$$

Lemma III.1 now implies that

$$V(\beta_k, x^-, x^\alpha) + n\lambda \ge V(\beta_k, x_{k-1}^*, x_k^*) \qquad \square$$

With Lemma III.2, we are ready to prove that Algorithm 1 computes assignments that are within 50% of the optimal.

Theorem III.3 (Optimality Bound on Algorithm 1): Suppose $[\beta_k]_{ij} \ge \lambda$. Let $x_0 = x_0^H = x_0^* \in \mathcal{X}$ be an arbitrary initial assignment. Let $\boldsymbol{x}^H = (x_1^H, \ldots, x_T^H)$ be the output assignments of Algorithm 1. Let $\boldsymbol{x}^* = (x_1^*, \ldots, x_n^*)$ be optimal assignments with respect to (1). Then

$$\mathcal{V}(\boldsymbol{x}^H) \ge \frac{1}{2}\mathcal{V}(\boldsymbol{x}^*)$$

Proof: Using Lemma III.2 and $x_k^H = \alpha(\beta_k + \lambda x_{k-1}^H)$, we have

$$V(\beta_k, x_{k-1}^H, x_k^H) \ge n\lambda \text{ for all } k \tag{10a}$$

$$V(\beta_k, x_{k-1}^H, x_k^H) + n\lambda \ge V(\beta_k, x_{k-1}^*, x_k^*) \text{ for all } k \tag{10b}$$

Next, we use (10b) to find a lower bound on the ratio of the values of the assignments of Algorithm 1 to the optimal assignments:

$$\begin{aligned} \frac{V(\beta_k, x_{k-1}^H, x_k^H)}{V(\beta_k, x_{k-1}^*, x_k^*)} &\ge \frac{V(\beta_k, x_{k-1}^H, x_k^H)}{V(\beta_k, x_{k-1}^H, x_k^H) + n\lambda} \\ &= \frac{V(\beta_k, x_{k-1}^H, x_k^H) + (n\lambda - n\lambda)}{V(\beta_k, x_{k-1}^H, x_k^H) + n\lambda} \\ &= 1 - \frac{n\lambda}{V(\beta_k, x_{k-1}^H, x_k^H) + n\lambda} \end{aligned} \tag{11}$$

Plugging in (10a) into (11), we now conclude that

$$\frac{V(\beta_k, x_{k-1}^H, x_k^H)}{V(\beta_k, x_{k-1}^*, x_k^*)} \ge \frac{1}{2}, \quad \text{for all } k \tag{12}$$

Hence, $\sum_{k=1}^T V(\beta_k, x_{k-1}^H, x_k^H) \ge (1/2)\sum_{k=1}^T V(\beta_k, x_{k-1}^*, x_k^*)$, completing the proof. □

IV. Handover-Aware Assignment with Lookahead (HAAL) Algorithm

Algorithm 1 only uses the benefit matrix at time k (β_k) in order to compute the assignment matrix x_k. In particular, the algorithm does not use the information contained in future benefit matrices, of which at least an estimate is often available. For example, in satellite applications, the well understood nature of orbital dynamics allows us to compute accurate satellite positions (and thus benefit matrices) hours or even days ahead of time. Such information can be used to select a sequence of assignments that perform better according to total value (7), rather than strictly maximizing the time step value (6). In this section, we will introduce the Handover-Aware Assignment with Lookahead (HAAL), an algorithm that leverages this future information in order to improve the total value of assignments at the expense of increased computational cost.

Consider an assignment x_k^H computed by Algorithm 1 maximizing the time step (net) value (6) at time step k. If x_k^H performs poorly in time step $k+1$ such that $H(x_k^H, x_{k+1}^H)$ is large, a better strategy might be to select some $\tilde{x} \in \mathcal{X}$ that satisfies the following inequalities:

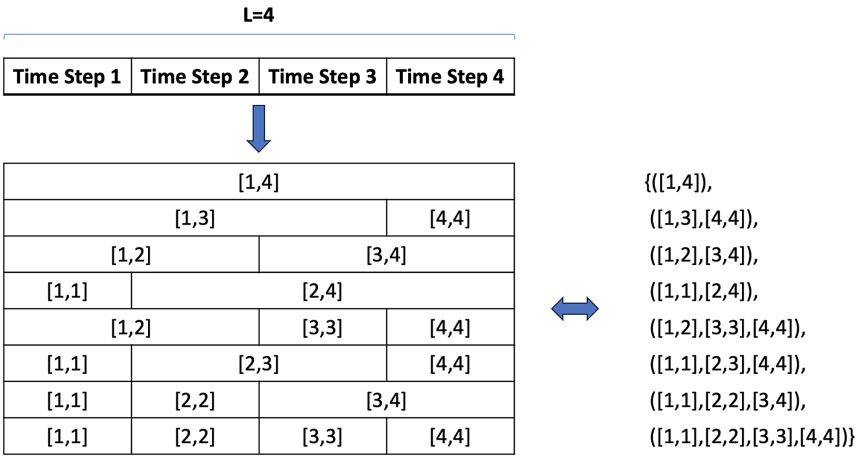


Fig. 1 Contents of the set $\mathcal{S}_4$.

$$V(\beta_k, x_{k-1}, \tilde{x}) \le V(\beta_k, x_{k-1}, x_k^H) \tag{13a}$$

$$V(\beta_k, x_{k-1}, x_k^H) + V(\beta_{k+1}, x_k^H, x_{k+1}^H) < V(\beta_k, x_{k-1}, \tilde{x}) + V(\beta_k, \tilde{x}, \tilde{x}) \tag{13b}$$

That is, $\tilde{x}$ yields slightly less value in time step k (13a) but performs significantly better in time step $k+1$ (13b). Since $\tilde{x}$ remains constant between k and $k+1$, we can rewrite the time step value as follows:

$$V(\beta_k, x_{k-1}, \tilde{x}) + V(\beta_{k+1}, \tilde{x}, \tilde{x}) = V\left(\sum_{t=k}^{k+1} \beta_t, x_{k-1}, \tilde{x}\right) \tag{14}$$

Now, we have a single value function that can be used to determine $\tilde{x}$ optimally using standard assignment problem methods (8), without having to resort to more expensive QAP or multi-assignment algorithms.

This is the operating principle behind HAAL: a "model-predictive-control-like" approach instead of a greedy one [31]. First, the algorithm partitions the time interval $[1,L]$, where L is a user-defined lookahead window, into a sequence of subintervals (Fig. 1). For example, $([1,2],[3,L])$. Then, a subroutine of Algorithm 1 computes a sequence of assignments by requiring the assignments on each subinterval to be constant. Next, the algorithm repeats this process over each possible partitioning of $[1,L]$. The first assignment of the most valuable assignment sequence is then executed. Last, the process repeats for each time step. We formalize this algorithm in the following sections.

First, a few preliminaries before we delve into the details of the proposed algorithm. In our discrete setting, *time intervals* are defined as integer tuples where $a \le b$. A *time interval sequence* is a sequence of time intervals $([a_1, b_1], [a_2, b_2], \ldots, [a_N, b_N])$ such that $a_{k+1} = b_k + 1$. For example, $([1,1],[2,3],[4,6])$ is a valid time interval sequence, but $([1,1],[2,3],[5,6])$ and $([1,1],[2,3],[3,6])$ are not. A time interval sequence *of length* L is a time interval sequence such that $a_1 = 1$ and $b_N = L$, where N is the number of time intervals in the time interval sequence. We denote the collection of all time interval sequences of length L as $\mathcal{S}_L$. For example, $\mathcal{S}_2 = \{([1,1],[2,2]), ([1,2])\}$, while Fig. 1 demonstrates $\mathcal{S}_4$. We can think of $\mathcal{S}_L$ as the collection of all partitionings of $[1,L]$ in which each partition is a time interval. Note that the cardinality of this set can be computed for a given L as $|\mathcal{S}_L| = 2^{L-1}$.

A. Algorithm Details

In this section, we will present our algorithm. We are given the same inputs as Algorithm 1 with the addition of a lookahead window L. This parameter indicates that at time k, the algorithm only uses information from at most the next L benefit matrices $\beta_k, \beta_{k+1}, \ldots, \beta_{\min(T,k+L)}$. To manage computational cost, we can set $L \le T$ because $|\mathcal{S}_L|$ (and thus the number of assignment problems we need to solve) scales exponentially in L. We will see later that the algorithm does not need to consider information from all T time steps at once in order to perform well.

At each time step k, the algorithm first determines the length of the effective lookahead window $L' = \min(L, T-k)$ in line 2. Next, the algorithm iterates over each time interval sequence in $\boldsymbol{s} \in \mathcal{S}_{L'}$, aiming to find the $\boldsymbol{s}^* \in \mathcal{S}$ which maximizes value using constant assignments in each of its constituent time intervals.

Thus, for each $\boldsymbol{s}$, the algorithm iterates over each time interval in $\boldsymbol{s}$ and computes the assignment x yielding the highest value when held constant over the time steps $[a_l, b_l]$. In order to find $x_l^{\boldsymbol{s}}$, we apply the same insights used in Eq. (14) to note that we can compute an optimal, constant assignment over the time step interval $[a_l, b_l]$ by maximizing value using the sum of all the benefit matrices in the time interval ($\beta_l^{\boldsymbol{s}}$, line 7). Thus, we solve a standard assignment problem using $\beta_l^{\boldsymbol{s}}$ (line 8) and append the benefit and assignment matrices to the sequences associated with $\boldsymbol{s}$ (line 9).

Once this process is complete for every time interval sequence $\boldsymbol{s} \in \mathcal{S}_{L'}$, we compute the value yielded by each sequence (line 11) and execute $x_1^{\boldsymbol{s}^*}$, the first assignment from the highest value time interval sequence (line 14). Note that assignments from other time intervals in $\boldsymbol{x}^{\boldsymbol{s}^*}$ are discarded, to be recomputed in the next time step when the algorithm has more information about the future, similar to strategies used in model predictive control [31]. This process continues until an assignment is generated at each time step k.

Note that when $L = 1$, Algorithms 1 and 2 are equivalent, and when $L = T$, Algorithm 2 is a strict improvement over Algorithm 1.

Algorithm 2: Handover-Aware Assignment w/Lookahead (HAAL)

Require: $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_T), \lambda \in \mathbb{R}, x_0 \in \mathcal{X}, L \ge 1$
1: **for** $1 \le k \le T$ **do**
2: $L' \leftarrow \min(L, T-k)$
3: **for** $\boldsymbol{s} \in \mathcal{S}_{L'}$ **do**
4: $x_0^{\boldsymbol{s}} \leftarrow x_{k-1}$
5: $\boldsymbol{\beta}^{\boldsymbol{s}} \leftarrow \emptyset$, $\boldsymbol{x}^{\boldsymbol{s}} \leftarrow \emptyset$
6: **for** $[a_l, b_l] \in \boldsymbol{s}$ **do**
7: $\beta_l^{\boldsymbol{s}} \leftarrow \sum_{t=a_l}^{b_l} \beta_{k+(t-1)}$
8: $x_l^{\boldsymbol{s}} \leftarrow \alpha(\beta_l^{\boldsymbol{s}} + \lambda x_{l-1}^{\boldsymbol{s}})$
9: Append $\beta_l^{\boldsymbol{s}}$ to $\boldsymbol{\beta}^{\boldsymbol{s}}$, append $x_l^{\boldsymbol{s}}$ to $\boldsymbol{x}^{\boldsymbol{s}}$
10: **end for**
11: $v^{\boldsymbol{s}} \leftarrow \mathcal{V}(\boldsymbol{x}^{\boldsymbol{s}}; \boldsymbol{\beta}^{\boldsymbol{s}}, \lambda, x_{k-1})$
12: **end for**
13: $\boldsymbol{s}^* \in \arg\max_{\boldsymbol{s} \in \mathcal{S}_{L'}} v^{\boldsymbol{s}}$
14: $x_k \leftarrow x_1^{\boldsymbol{s}^*}$
15: **end for**
return $\boldsymbol{x} = (x_1, \ldots, x_T)$

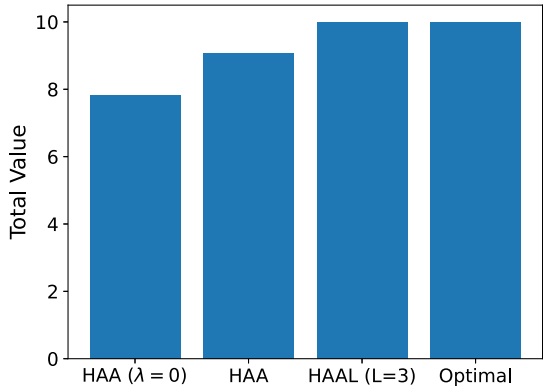


Fig. 2 Average performance across 50 runs of Algorithm 1 and Algorithm 2 versus naive and true optimal solution on small size problem instances.

B. Optimality of HAAL

We now show that Algorithm 2 produces assignments that are within 50% of the optimal.

Theorem IV.1 (Optimality Bound on Algorithm 2): Suppose $[\beta_k]_{ij} \geq \lambda$. Let $x_0 \in \mathcal{X}$ be an arbitrary initial assignment. Let $\boldsymbol{x}^H = (x_1^H, \ldots, x_T^H)$ be the assignments returned by Algorithm 2 using the lookahead window L. Let $\boldsymbol{x}^* = (x_1^*, \ldots, x_T^*)$ be optimal assignments with respect to (1). Then, $\mathcal{V}(\boldsymbol{x}^H) \geq (1/2)\mathcal{V}(\boldsymbol{x}^*)$.

Proof: The proof for this claim builds on Theorem III.3 to extend the result to the case where $L > 1$. See Appendix B for details. □

While the bounds introduced above are tight in the worst case, we can show empirically that Algorithm 2 performs much more favorably with respect to the true optimal solution even when $L < T$. Figure 2 shows the results of the algorithm on randomly generated $5 \times 5 \times 3$ dimensional benefit matrices for which a true optimal assignment can be computed via exhaustive search. Algorithm 1 and Algorithm 2 noticeably outperform solutions that do not account for handover (Algorithm 1 with $\lambda = 0$), and Algorithm 2 only performs imperceptibly worse than the true optimal solution when $L = 3$.

C. Distributed HAAL (HAAL-D)

One attractive feature of Algorithm 2 is that we can design a completely distributed version of the algorithm that can operate without central control and under realistic communication requirements. Algorithm 3 is the distributed version of Algorithm 2.

We model a constellation of satellites as an undirected graph $\mathcal{G} = ([n], \mathcal{E})$. The edge set $\mathcal{E}$ is constructed by $\{i, j\} \in \mathcal{E}$ if and only if satellites i and j can communicate via intersatellite links. We assume all satellites have knowledge of $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_T)$ and x_0.[**]

Define the set $\mathcal{T}_L \coloneqq \{[a, b] : a, b \in \mathbb{N}, 1 \leq a \leq b \leq L\}$, which can be interpreted as the set containing all unique time step intervals in $\mathcal{S}_L$; note that $|\mathcal{T}_L| = L(L+1)/2$.

Only a few key differences distinguish Algorithms 2 and 3. First, Algorithm 3 computes assignments for each time step interval $[a, b] \in \mathcal{T}_{L'}$ in parallel rather than in series. This is feasible since while in Algorithm 2 handover penalties are computed based on previous assignments in that time interval sequence (Algorithm 2 line 8), in Algorithm 3 we instead apply a handover penalty based solely on distance from the assignment at the beginning of the time interval sequence, x_{k-1} (line 6). This means that handover penalties in one time interval never depend on the results of the optimization for another time interval, and thus we can solve all $L(L+1)/2$ assignment problems at the same time. This is an approximation (which could also be used to parallelize Algorithm 2) and increases the size of satellite communications with $O(L^2)$, but importantly allows us to run multiple assignment auctions without meaningfully increasing the number of communications needed. By contrast, solving these assignment problems sequentially would result in the number of communications growing with $O(L\,2^{L-1})$.

Algorithm 3: Distributed Handover-Aware Assignment w/ Lookahead (HAAL-D)

Require: $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_T), \lambda \in \mathbb{R}, x_0 \in \mathcal{X}, L \geq 1$

1: **for** $1 \leq k \leq T$ **do**
2: $L' \leftarrow \min(L, T-k)$
3: **for** $j = 1 \ldots n$ *(run in parallel on each satellite)* **do**
4: **for** $[a, b] \in \mathcal{T}_{L'}$ **do** *(run in parallel)*
5: $\beta_{[a,b]} \leftarrow \sum_{t=a}^{b} \beta_{k+(t-1)}$
6: $x_{[a,b]} \leftarrow \alpha(\beta_{[a,b]} + \lambda x_{k-1}^j)$ *(solve by communicating with neighbors i.e. [30])*
7: **end for**
8: Wait for auctions for all $[a, b] \in \mathcal{T}_{L'}$ to converge
9: **for** $\boldsymbol{s} \in \mathcal{S}_{L'}$ **do**
10: $\boldsymbol{\beta}^{\boldsymbol{s}} \leftarrow \emptyset, \boldsymbol{x}^{\boldsymbol{s}} \leftarrow \emptyset$
11: **for** $[a, b] \in \boldsymbol{s}$ **do**
12: Append $\beta_{[a,b]}$ to $\boldsymbol{\beta}^{\boldsymbol{s}}$, append $x_{[a,b]}$ to $\boldsymbol{x}^{\boldsymbol{s}}$
13: **end for**
14: $v^{\boldsymbol{s}} \leftarrow \mathcal{V}(\boldsymbol{x}^{\boldsymbol{s}}; \boldsymbol{\beta}^{\boldsymbol{s}}, \lambda, x_{k-1}^j)$
15: **end for**
16: $\boldsymbol{s}^* \in \arg\max_{\boldsymbol{s} \in \mathcal{S}_{L'}} v^{\boldsymbol{s}}$
17: $x_k^j \leftarrow x_1^{\boldsymbol{s}}$
18: **end for**
19: **end for**

return $\boldsymbol{x}^i = (x_1^i, \ldots, x_T^i), \ \forall i = 1 \ldots n$

Additionally, rather than solving the assignment problems using a centralized method, we can use one of the distributed methods that exist in the literature [30]. These algorithms use an auction formulation to efficiently solve assignment problems of the form (2) without a central node, and while only allowing communication between agents and their neighbors in a connected graph. The output assignments are within $n\epsilon$ of optimality, where $\epsilon > 0$ is a small positive constant. Thus, Algorithm 3 initializes $L(L+1)/2$ auctions, one for each time interval in $\mathcal{T}_{L'}$, and solves them in a distributed fashion. When communicating with neighbors, the satellites share their prices and bidders for each of the $L(L+1)/2$ auctions simultaneously. Once all auctions have converged (line 8), all satellites will have computed identical near-optimal assignments $x_{[a,b]} \in \mathcal{X}$ for all time intervals $[a, b] \in \mathcal{T}_{L'}$.

With access to this information, each satellite independently computes the time interval sequence $\boldsymbol{s}^* \in \mathcal{S}_{L'}$ that provides the most value and executes the first assignment from this sequence (lines 9 to 17), as in Algorithm 2. This process continues until an assignment is generated at each time step k.

Despite distinct differences between Algorithms 2 and 3, we will subsequently show that their respective performances are remarkably similar.

V. Satellite Constellation Experiments

In this section, we compare Algorithm 2 and 3 to the following centralized and distributed task assignment procedures:[††]

1) *No handover (NHA)*: The output assignment sequence is computed as $\boldsymbol{x} = (\alpha(\beta_1), \ldots \alpha(\beta_T))$, without regard for the handover penalty.

2) *Greedy (GA)*: Each satellite chooses the task of maximal benefit and sequentially executes that same task until the task is out of view. Then, the satellite again chooses the assignable task of

[**]In a case where satellites only have information about their own benefits, an extra stage of the algorithm where satellites communicate the value they receive from each time interval sequence and select the sequence which provides the most benefit constellation-wide must be added after auctions converge; this more general version of the algorithm is used in the experiments in the following section.

[††]Python implementations of HAAL and comparison algorithms can be found at https://github.com/Rainlabuw/handover-aware-assignment.

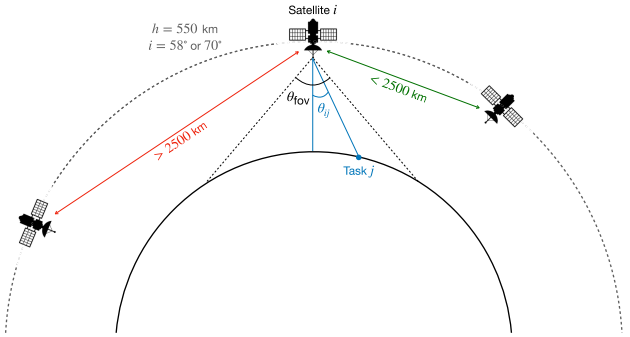


Fig. 3 Diagram of orbital configuration for satellite constellation experiments; the satellites in the constellations are at an altitude of 550 km with evenly distributed orbital planes at the inclination of 58 or 70 deg. Satellites can exchange messages over intersatellite links within the range of 2500 km.

maximal benefit, with conflicts resolved centrally by assigning tasks to the satellite of lower index.

3) *Consensus-Based Bundle Algorithm (CBBA)*: We apply CBBA (a fully distributed algorithm) to our problem setting, modifying it to only consider a lookahead window of length $L \leq T$ at any given time step. Each "bundle" is then size L, and tasks are scored according to handovers induced from previously selected tasks in the bundle. The reader is directed to [28] for further details on the CBBA algorithm.

Our experiments use satellites in a constellation as agents and points on the Earth surface as tasks. Simulated satellites are in circular orbits with an altitude of 550 km, can communicate with all satellites within 2500 km, and are able to complete tasks within a 120° field of view (FOV) assuming the satellite is nadir pointing (see Fig. 3). The benefit matrix is computed as

$$[\beta_k]_{ij} := \begin{cases} \beta_j^{\max} \exp\left(-\dfrac{[\theta_k]_{ij}^2}{2\sigma^2}\right) & \text{if } [\theta_k]_{ij} \leq \theta_{\text{FOV}} \\ 0 & \text{otherwise} \end{cases}$$

where $[\theta_k]_{ij} \in [0°, 360°)$ is the angle between satellite i and task j at time step k, $\beta_j^{\max}$ is the maximum possible benefit for task j, and $\sigma := [\theta_{\text{FOV}}/\sqrt{-2\log(0.05)}]$ is chosen so that when $[\theta_k]_{ij} = 120°$, we will have $([\beta_k]_{ij}/\beta_j^{\max}) = 0.05$. We chose a Gaussian function to mimic the deterioration of Gaussian beam intensity as pointing angle increases.

Using a high-fidelity orbital mechanics simulation, we compute $\mathcal{G}$ and β_{ij} for every satellite-task pair at 1-min intervals over the course of an entire orbit ($T = 93$), and use this information to build the benefit matrix with which we compute assignments.

We will run two experiments that demonstrate the performance, design tradeoffs, and scaling properties of the proposed algorithm.

A. Midsize LEO Internet Constellation with Randomly Placed Tasks

Our first experiment consists of 324 satellites (18 evenly spaced orbital planes at 58° inclination, 18 satellites each) which aim to complete 450 tasks randomly placed on the surface of the Earth. This simulates users of an Internet service, which might be inconsistently placed across the Earth's surface (see Fig. 4 for placement of tasks). We randomly set the benefits of each task according to $\beta_j^{\max} \sim U(1,2)$.

1. *Value Performance*

Figure 5a demonstrates that Algorithms 2 and 3 deliver far higher total value than other state-of-the-art task allocation algorithms. Both Algorithm 2 and Algorithm 3 perform significantly better than all other tested algorithms at all values of L, yielding ≈50% higher value than CBBA, the next best algorithm in this setting. Also note that despite the several slight differences between HAAL and HAAL-D, performance does not suffer when moving to the distributed setting.

2. *Communication Requirements*

In the distributed auction subroutine ([30]) we use on Algorithm 3 Line 6 (as well as in CBBA [28]), agents need to repeatedly communicate with neighbors in order to converge on a near-optimal, conflict-free assignment in finite time. Thus, in Fig. 5a, we measure the number of rounds of communications necessary for convergence as L increases.

Because we run our assignment problems in parallel (Algorithm 3 Line 4), we observe that communication requirements appear to scale at a rate far less than the rate of growth of the number of assignment problems being solved, $L(L+1)/2$. The increase in necessary communications that does exist can be attributed to the fact that as the lookahead window increases in satellite constellation scenarios, the satellites bidding on the same tasks are further away from each other in the communication graph. Note that even with 324 satellites and $L = 6$, the communication requirements are minimal. In order to assign tasks within the available 60 s, satellites need only to send approximately 80 messages to neighbors.

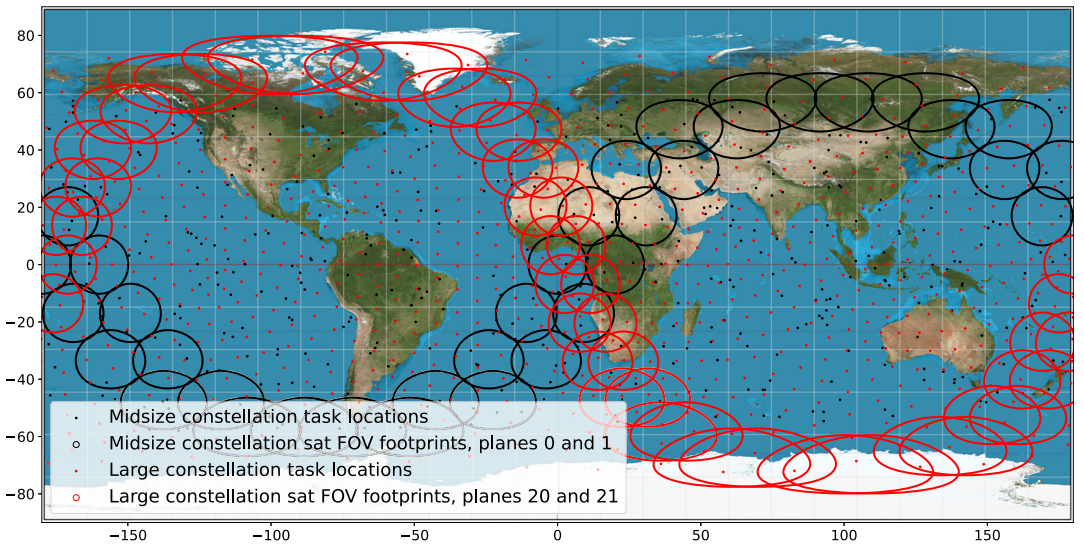
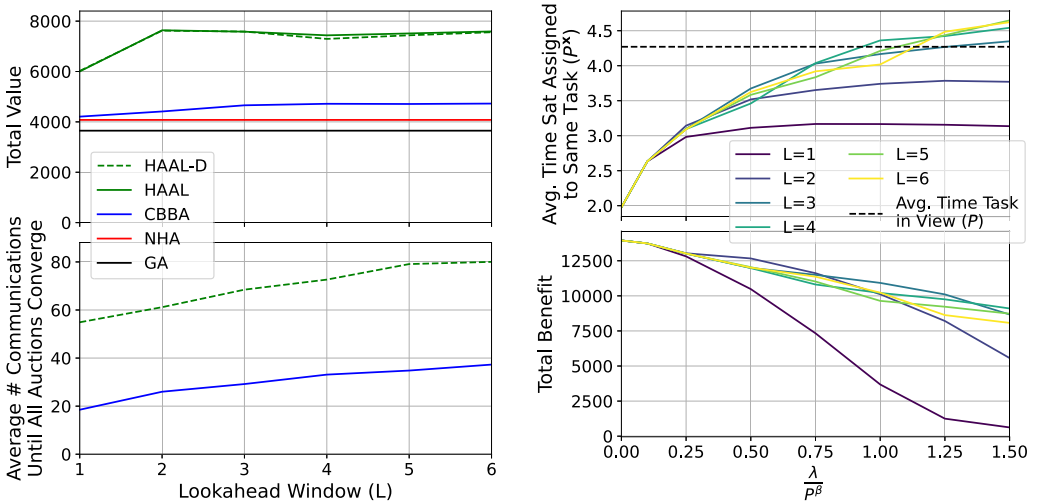


Fig. 4 Sample ground tracks and task locations for constellation experiments; 2/18 and 2/40 orbital plane field-of-views are shown for the midsize and large constellations, respectively.



a) Value captured (higher is better) and # of communications needed for convergence (lower is better), $\frac{\lambda}{P^\beta} = 0.45$

b) Algorithm 2 satellite tasking consistency and raw benefit captured as λ increases relative to P^β

Fig. 5 Results of midsize constellation experiments.

3. Selection of a Handover Penalty Multiplier λ

Figure 5b provides insights into the role of the handover multiplier λ in the constellation cost relative to task benefits. To this end, consider the *satellite-task pass* of a satellite-task pair (i, j) defined as the time interval $[a_{ij}, b_{ij}]$ during which task j is continuously in view of a satellite i. Note that based on our method of computing benefits, we can denote the set of all (i, j) that have an associated satellite-task pass in $\boldsymbol{\beta}$ as

$$\mathcal{P} \coloneqq \{(i,j) \,:\, \exists k \text{ s.t. } 1 \le k \le T,\ [\beta_k]_{ij} > 0\}$$

The corresponding satellite-task pass for (i, j) is then written as the unique $[a_{ij}, b_{ij}]$ such that the following condition holds:[‡‡]

$$[\beta_k]_{ij} > 0 \quad \text{for all } k \text{ s.t. } a_{ij} \le k \le b_{ij}; \text{otherwise}, [\beta_k]_{ij} = 0$$

Finally, we define the set of *assigned satellite-task passes* as $\mathcal{P}^{\boldsymbol{x}} \coloneqq \{(i,j) \in \mathcal{P} \,:\, \exists k \,\text{s.t.} a_{ij} \le k \le b_{ij},\ [x_k]_{ij} = 1\}$.

We can now define several useful quantities. The average length of a satellite pass can be written as $P = (1/|\mathcal{P}|) \times \sum_{(i,j)\in\mathcal{P}}(b_{ij} - a_{ij} + 1)$. Similarly, we can define the average length of time a satellite is actually assigned to a task during the appropriate satellite-task pass as $P^{\boldsymbol{x}} \coloneqq (1/|\mathcal{P}^{\boldsymbol{x}}|)\sum_{(i,j)\in\mathcal{P}^{\boldsymbol{x}}} \sum_{k=a_{ij}}^{b_{ij}} [x_k]_{ij}$. Finally, we define the average benefit yielded by assigning a satellite to a task for an entire pass as $P^{\boldsymbol{\beta}} \coloneqq (1/|\mathcal{P}|)\sum_{(i,j)\in\mathcal{P}} \sum_{k=a_{ij}}^{b_{ij}} [\beta_k]_{ij}$.

Empirically, we see that as $\lambda/P^{\boldsymbol{\beta}}$ increases, $P^{\boldsymbol{x}}$ increases and approaches P, the average length of time that task is in view. This implies that if the user wants satellites to be assigned to tasks for the entire time those tasks are assignable, $\lambda/P^{\boldsymbol{\beta}}$ should be set to $\approx$1. In such a scenario, switching tasks is only advantageous if the satellite can achieve a full pass worth of benefits before switching tasks again.

However, Fig. 5b also illustrates the tradeoff in using a high value of λ. Although task selections become more consistent as λ increases, total benefit captured decreases. This corresponds to satellites achieving tasks with nonzero benefit less frequently as they are more "hesitant" to switch from their existing assignments.

[‡‡]For notational clarity, we assume each (i, j) pair has at most one pass. However, this calculation can easily be extended to the case where each satellite-task pair has multiple passes.

4. Selection of a Lookahead Window Length L

Figure 5b also provides insight into a selection of L. Although a higher value of L corresponds to a higher computational load, longer lookahead windows also yield more consistent assignments and capture more benefit. This advantage becomes more pronounced as $\lambda/P^{\boldsymbol{\beta}}$ increases.

However, we also observe that there is generally a diminishing return as lookahead window L increases. Intuitively, this happens because the lookahead window extends to the point that satellites have full information about what benefits they can expect to receive from completing a given task. Thus, a reasonable upper limit on L would be the maximum possible number of time steps a task could be in view of a satellite. This is a geometry problem: If T is the period of the satellite, Δt is the simulation time step, and h is the altitude, then

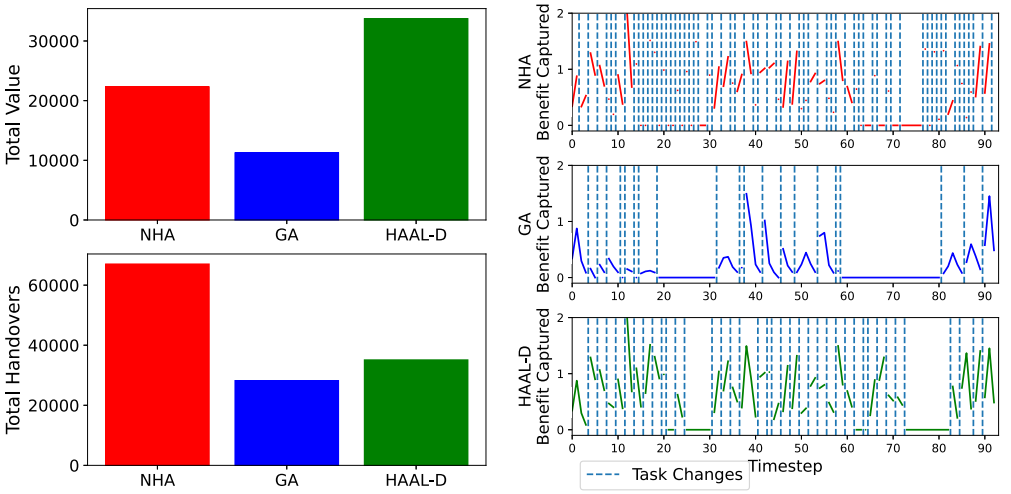
$$L^{\max} = \text{ceil}\left[\frac{T}{2\pi\Delta t}\left[\arcsin\left(\frac{R_{\text{Earth}} + h}{R_{\text{Earth}}}\sin\frac{\theta_{\text{FOV}}}{2}\right) - \frac{\theta_{\text{FOV}}}{2}\right]\right] \quad (15)$$

Using Eq. (15), we can compute that $L^{\max} = 6$ for the experiments in this paper.

B. Large-Scale Earth-Coverage Observation Constellation

Finally, to demonstrate the scalability and performance of the algorithm at realistic constellation sizes, we apply Algorithm 3 to a constellation of 1000 satellites (40 planes of 25 satellites each, $i = 70°$). We place 812 tasks evenly across Earth from latitude −70°–70°, with 188 "dummy" tasks providing zero benefit added to ensure that $n \le m$ (see Fig. 4). This mimics a constellation designed to continuously observe the entire surface of Earth. We randomly set the benefits of each nondummy task according to $\beta_j^{\max} \sim U(1,2)$ and use $\lambda = 0.5$. We then apply Algorithm 3 (with $L = 6$), GA, and NHA over a full orbit, tracking captured value and total number of task handovers (CBBA is not tested as it becomes computationally infeasible for large constellation sizes).

Figure 6 provides results of these experiments. As shown in Fig. 6a, although GA achieves slightly fewer handovers, Algorithm 3 again obtains far more value than other algorithms in the literature. Figure 6b provides a visual explanation of how Algorithm 3 achieves this. The y-axis of each plot tracks the raw benefit (not including handover penalty) of the task achieved by a representative satellite at each time step, with task handovers indicated by vertical lines. Although NHA consistently assigns satellites to tasks with high benefit, it requests a large number of task handovers (i.e. compare NHA and HAAL-D in time steps 19–28).



a) Value obtained (higher is better) and number of handovers induced (lower is better) for NHA, GA, and HAAL-D

b) History of benefits captured by a single satellite at each time step for NHA, GA, and HAAL-D. Each dotted blue line indicates a change in task assignment

Fig. 6 Results of large-scale constellation experiment.

Conversely, GA requests slightly fewer task handovers than HAAL-D, but consistently obtains less benefit in each time step and is characterized by long periods of satellite inactivity (i.e. compare GA and HAAL-D in time steps 0–40). Visually we can see that our algorithm strikes a balance, handing over tasks only when it can obtain several time steps of benefit.

VI. Conclusions and Limitations

In this paper, we develop Handover Aware Assignment with Lookahead (HAAL), a task allocation algorithm which generates feasible, conflict-free assignments while limiting task handover. The algorithm works by approximating the resulting seemingly NP-hard problem with a sequence of efficiently solvable assignment problems, and executing these assignments in a manner similar to model-predictive control. HAAL comes with performance guarantees and outperforms other solution methods found in the literature, even when scaling to realistic, large-scale constellation simulations with thousands of satellites and tasks.

Importantly, HAAL can be executed in a fully distributed setting, and by adjusting the amount of information the algorithm has access to (the lookahead window L), performance can be smoothly scaled up or down while only affecting the *size* of communication messages, not the *number* of communications. This has major implications for dynamic problem settings where tasks, satellites, and benefits can change over time (i.e., when cloud cover or network demand changes, or when satellites unexpectedly become unavailable). In these cases, the distributed nature of HAAL means that satellites could communicate with their respective neighboring satellites to change assignments and act on new information in real time without waiting for a centralized hub to react. Here, fast convergence to new assignments is critical; as such, it would be interesting to explore whether HAAL can be modified to leverage results of previous auctions and greatly decrease communication requirements.

One key limitation of this work is that we have assumed that each satellite can accomplish a single task at a time, and each task can only be completed by a single satellite. In many Internet constellation applications, satellites can serve multiple users at once, while in many imaging or sensing applications, multiple satellites can meaningfully and collaboratively complete the same task. Expanding the problem setting would allow HAAL to be applied to a wider range of applications.

Finally, it is notable that the minimal-handover task assignment problem can be quite naturally formulated as a sequential decision-making problem. With recent advances in deep reinforcement learning, dynamic-programming-based formulations of the problem have become more tractable and could be explored in this context to increase the overall operational performance in highly semantic and/or stochastic settings.

Appendix A: Proof of Lemma III.1

Proof: Note for $x^-, x \in \mathcal{X}$, we have

$$-\frac{\lambda}{2}(x_{ij} - x_{ij}^-)^2 = -\frac{\lambda}{2}x_{ij} + \lambda x_{ij}x_{ij}^- - \frac{\lambda}{2}x_{ij}^- \tag{A1}$$

Removing the constant x^- term and plugging Equation (A1) into the right side of Eq. (6), it follows that

$$\operatorname*{argmax}_{x\in\mathcal{X}} V(\beta, x^-, x) = \operatorname*{argmax}_{x\in\mathcal{X}} \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\beta_{ij}x_{ij} + \lambda x_{ij}x_{ij}^-\right) - \sum_{i=1}^{n}\sum_{j=1}^{m}\frac{\lambda}{2}x_{ij}$$

Finally, note that $\sum_{i=1}^{n}\sum_{j=1}^{m}(\lambda/2)x_{ij} = (\lambda/2)n$, for all $x \in \mathcal{X}$, and as such we can remove this term from the optimization, yielding

$$\operatorname*{argmax}_{x\in\mathcal{X}} V(\beta, x^-, x) = \operatorname*{argmax}_{x\in\mathcal{X}} \sum_{i=1}^{n}\sum_{j=1}^{m}\left(\beta_{ij} + \lambda x_{ij}^-\right)x_{ij}$$

Finally, we can apply Eq. (3), yielding

$$\operatorname*{argmax}_{x\in\mathcal{X}} V(\beta, x^-, x) = \alpha(\beta + \lambda x^-)$$

□

Appendix B: Proof of Theorem IV.1

Proof: Our general strategy will be to prove that any time Algorithm 2 achieves less than 50% of the optimal value (the difference from 50% being defined as Δ), it will earn back this extra value at some later time step, ensuring that by the end of T time steps it has reached 50% of the optimal value.

First, we define some notation used throughout the proof. For $1 \le k \le T - L + 1$ and $\boldsymbol{s} \in \mathcal{S}_L$, the assignment sequence $\boldsymbol{x}^{s,k} = (x_1^{s,k}, \ldots, x_L^{s,k})$ induced by s at time k is defined as

$$x_t^{s,k} \coloneqq \begin{cases} \alpha\left(\sum_{i=a}^{b} \beta_{k+(i-1)} + \lambda x_{t-1}^{s,k}\right) & \text{if } \exists [a,b] \in \boldsymbol{s} : t = a \\ x_{t-1}^{s,k} & \text{otherwise} \end{cases} \tag{B1}$$

where $x_0^{s,k} \coloneqq x_{k-1}^H$.

Next, we define several time interval sequences $\boldsymbol{s} \in \mathcal{S}_L$ of interest. Let $\boldsymbol{s}' = ([1,1],[2,2],\ldots,[L,L]) \in \mathcal{S}_L$ be the sequence of length 1 time intervals. Let $\boldsymbol{s}_k^* = \arg\max_{s \in \mathcal{S}_L} \sum_{t=1}^{L} V(\beta_{k+t-1}, x_{t-1}^{s,k}, x_t^{s,k})$ be the time interval sequence which induces assignments yielding the highest value over a lookahead window of length L, starting at time step k. Note that per the definition of Algorithm 2

$$x_k^H = x_1^{s_k^*,k} \quad \text{for all } k \tag{B2}$$

Thus

$$\sum_{t=2}^{L} V\left(\beta_{k+t-1}, x_{t-1}^{s_k^*,k}, x_t^{s_k^*,k}\right) + V\left(\beta_k, x_{k-1}^H, x_k^H\right) \geq \sum_{t=1}^{L} V\left(\beta_{k+t-1}, x_{t-1}^{s',k}, x_t^{s',k}\right) \tag{B3}$$

Set $v_k^* \coloneqq \max(n\lambda, V(\beta_k, x_{k-1}^*, x_k^*) - n\lambda)$, which implies that v_k^* is the minimum value such that $v_k^* + n\lambda \geq V(\beta_k, x_{k-1}^*, x_k^*)$ and $v_k^* \geq n\lambda$. Using the same reasoning as Theorem III.3, we can say that $\sum_{k=1}^{T} v_k^* \geq (1/2)\mathcal{V}(\boldsymbol{x}^*)$. It remains to show that $\mathcal{V}(\boldsymbol{x}^H) \geq \sum_{k=1}^{T} v_k^*$.

We can prove this via induction. First, note that when $k = 1$, we can rearrange Equation (B3) to obtain

$$\sum_{t=2}^{L} V\left(\beta_t, x_{t-1}^{s_1^*,1}, x_t^{s_1^*,1}\right) \geq \sum_{t=1}^{L-1} V\left(\beta_t, x_{t-1}^{s',1}, x_t^{s',1}\right) + V\left(\beta_L, x_{L-1}^{s',1}, x_L^{s',1}\right) - V\left(\beta_1, x_0^H, x_1^H\right)$$

Furthermore, note that Lemma III.2 applies to any assignment generated on a time step interval of length one, which includes $x_t^{s',k}$ for all $t = 1,\ldots,L$. Using this fact in conjunction with having v_k^* as the minimum value which satisfies the conditions in Lemma III.2, we conclude that

$$\sum_{t=2}^{L} V\left(\beta_t, x_{t-1}^{s_1^*,1}, x_t^{s_1^*,1}\right) \geq \sum_{t=1}^{L-1} v_t^* + \Delta_1 \tag{B4}$$

where $\Delta_1 = v_L^* - V(\beta_1, x_0^H, x_1^H)$. Inequality (B4) is the base case of our inductive argument.

In the inductive step, we aim to establish the following relationship when $k \leq T - L$:

$$\sum_{t=2}^{L} V\left(\beta_{k+t-1}, x_{t-1}^{s_k^*,k}, x_t^{s_k^*,k}\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k} \Delta_t$$
$$\Rightarrow \sum_{t=2}^{L} V\left(\beta_{k+t}, x_{t-1}^{s_{k+1}^*,k+1}, x_t^{s_{k+1}^*,k+1}\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k+1} \Delta_t \tag{B5}$$

where $\Delta_i = v_{L+i-1}^* - V(\beta_i, x_{i-1}^H, x_i^H)$.

To prove this, consider the time interval sequence $s_k^{*'} = \{[\max(1,a-1), b-1] : [a,b] \in \boldsymbol{s}_k^*,\ b \neq 1\} \cap \{[L,L]\} \in \mathcal{S}$, which corresponds to $\boldsymbol{s}_k^*$ shifted forwards by one time interval, and with a time interval of length one added to the end of the sequence to cover the last time interval in L.

One could directly induce a new assignment sequence $\boldsymbol{x}^{s_k^{*'},k+1}$ using $\boldsymbol{s}_k^{*'}$ at time $k+1$. Another way to generate an assignment sequence $\hat{\boldsymbol{x}} = (\hat{x}_1,\ldots,\hat{x}_L)$, which has constant assignments over all time intervals of $\boldsymbol{s}_k^{*'}$, is to reuse the segment of $\boldsymbol{x}^{s_k^*,k}$ that was not executed, $\hat{x}_{t-1} = x_t^{s_k^*,k}$, $t = 2\ldots L$, and let $\hat{x}_L = \alpha(\beta_{k+L} + \lambda x_L^{s_k^*,k})$. Then, as assignments induced by $s_k^{*'}$ at time $k+1$ are optimal under the condition that assignments are constant over time intervals in $s_k^{*'}$ (Lemma III.1 and Equation (B1)), it follows that

$$\sum_{t=1}^{L} V\left(\beta_{k+t}, x_{t-1}^{s_k^{*'},k+1}, x_t^{s_k^{*'},k+1}\right) \geq \sum_{t=1}^{L} V\left(\beta_{k+t}, \hat{x}_{t-1}, \hat{x}_t\right) = \sum_{t=2}^{L} V\left(\beta_{k+t-1}, x_{t-1}^{s_k^*,k}, x_t^{s_k^*,k}\right) + V\left(\beta_{k+L}, x_L^{s_k^*,k}, \alpha(\beta_{k+L} + \lambda x_L^{s_k^*,k})\right)$$

Then, Lemma III.2, Equation (B2), and the optimality of $\boldsymbol{s}_{k+1}^*$ when induced at time step $k+1$, lead to

$$V\left(\beta_{k+1}, x_k^H, x_{k+1}^H\right) + \sum_{t=2}^{L} V\left(\beta_{k+t}, x_{t-1}^{s_{k+1}^*,k+1}, x_t^{s_{k+1}^*,k+1}\right) \geq \sum_{t=2}^{L} V\left(\beta_{k+t-1}, x_{t-1}^{s_k^*,k}, x_t^{s_k^*,k}\right) + v_{k+L}^*$$

Rearranging and plugging in Δ_{k+1} along with the inductive step (Equation (B5)), it thus follows that

$$\sum_{t=2}^{L} V\left(\beta_{k+t}, x_{t-1}^{s_{k+1}^*,k+1}, x_t^{s_{k+1}^*,k+1}\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k} \Delta_t + \Delta_{k+1} = \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{k+1} \Delta_t$$

proving the inductive case. Note that this induction implies that at $k = T - L + 1$

$$\sum_{t=2}^{L} V\left(\beta_{T-L+t}, x_{t-1}^{s_{T-L+1}^*,T-L+1}, x_t^{s_{T-L+1}^*,T-L+1}\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L+1} \Delta_t$$

Adding $V(\beta_{T-L+1}, x_{T-L}^H, x_{T-L+1}^H)$ to both sides leads to

$$\sum_{t=1}^{L} V\left(\beta_{T-L+t}, x_{t-1}^{s_{T-L+1}^*,T-L+1}, x_t^{s_{T-L+1}^*,T-L+1}\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^*$$

From this point forward, $L \geq T - k$, so the algorithm has full information and the assignments induced by $\boldsymbol{s}_{T-L+1}^*$ are the exactly the ones chosen by Algorithm 2 (formally, $(x_{T-L+1}^H,\ldots,x_T^H) = \boldsymbol{x}^{\boldsymbol{s}_{T-L+1}^*,T-L+1}$). Thus

$$\sum_{k=T-L+1}^{T} V\left(\beta_k, x_{k-1}^H, x_k^H\right) \geq \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^* \tag{B6}$$

We now have enough information to bound $\sum_{k=1}^{T} V(\beta_k, x_{k-1}^H, x_k^H)$. Using the bound from Equation (B6) and the identity $V(\beta_i, x_{i-1}^H, x_i^H) = v_{L+i-1}^* - \Delta_i$, we find

$$\sum_{k=1}^{T} V\left(\beta_k, x_{k-1}^H, x_k^H\right) = \sum_{k=1}^{T-L} V\left(\beta_k, x_{k-1}^H, x_k^H\right) + \sum_{k=T-L+1}^{T} V\left(\beta_k, x_{k-1}^H, x_k^H\right) \geq \sum_{k=1}^{T-L} \left[v_{L+k-1}^* - \Delta_k\right] + \sum_{t=1}^{L-1} v_t^* + \sum_{t=1}^{T-L} \Delta_t + v_T^*$$

Cancelling Δ_k terms and bringing all v^* terms into one sum, we now conclude that

$$\sum_{t=1}^{T} V\left(\beta_t, x_{t-1}^H, x_t^H\right) \geq \sum_{t=1}^{T} v_t^* \geq \frac{1}{2}\sum_{t=1}^{T} V\left(\beta_t, x_{t-1}^*, x_t^*\right)$$

$$\text{implying that} \quad \mathcal{V}(\boldsymbol{x}^H) \geq \frac{1}{2}\mathcal{V}(\boldsymbol{x}^*) \qquad \square$$

Acknowledgment

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under grant DGE-2140004 and AFOSR grant FA9550-20-1-0053.

References

[1] Gerkey, B. P., and Matarić, M. J., "A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems," *International Journal of Robotics Research*, Vol. 23, No. 9, 2004, pp. 939–954. https://doi.org/10.1177/0278364904045564

[2] Skaltsis, G. M., Shin, H.-S., and Tsourdos, A., "A Survey of Task Allocation Techniques in MAS," *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, Inst. of Electrical and Electronics Engineers, New York, 2021, pp. 488–497. https://doi.org/10.1109/ICUAS51884.2021.9476736

[3] Monmousseau, P., "Scheduling of a Constellation of Satellites: Creating a Mixed-Integer Linear Model," *Journal of Optimization Theory and Applications*, Vol. 191, No. 2-3, 2021, pp. 846–873. https://doi.org/10.1007/s10957-021-01875-2

[4] Kennedy, A. K., and Cahoy, K. L., "Performance Analysis of Algorithms for Coordination of Earth Observation by Cubesat Constellations," *Journal of Aerospace Information Systems*, Vol. 14, No. 8, 2017, pp. 451–471. https://doi.org/10.2514/1.I010426

[5] Damiani, S., Verfaillie, G., and Charmeau, M.-C., "An Earth Watching Satellite Constellation: How to Manage a Team of Watching Agents with Limited Communications," *Fourth International Joint Conference on Autonomous Agents and Multiagent Systems*, Assoc. for Computing Machinery, 2005, pp. 455–462. https://doi.org/10.1145/1082473.1082543

[6] Stephenson, M. A., and Schaub, H., "Optimal Agile Satellite Target Scheduling with Learned Dynamics," *Journal of Spacecraft and Rockets*, 2024, pp. 1–12. https://doi.org/10.2514/1.A36097

[7] Juan, E., Lauridsen, M., Wigard, J., and Mogensen, P., "Handover Solutions for 5G Low-Earth Orbit Satellite Networks," *IEEE Access*, Vol. 10, 2022, pp. 93,309–93,325. https://doi.org/10.1109/ACCESS.2022.3203189

[8] Chowdhury, P. K., Atiquzzaman, M., and Ivancic, W., "Handover Schemes in Satellite Networks: State-of-the-Art and Future Research Directions," *IEEE Communications Surveys & Tutorials*, Vol. 8, No. 4, 2006, pp. 2–14. https://doi.org/10.1109/COMST.2006.283818

[9] Voicu, A. M., Bhattacharya, A., and Petrova, M., "Handover Strategies for Emerging LEO, MEO, and HEO Satellite Networks," *IEEE Access*, Vol. 12, 2024, pp. 31,523–31,537. https://doi.org/10.1109/ACCESS.2024.3368503

[10] Ritz, S., and Bracken, J., "GLAST Observatory Slew Rate and Pointing Knowledge Issues," TR NASA Goddard Space Flight Center, 2000, https://fermi.gsfc.nasa.gov/science/resources/swg/may00/SRitz.pdf.

[11] Park, S., and Kim, J., "Trends in LEO Satellite Handover Algorithms," *12th International Conference on Ubiquitous and Future Networks (ICUFN)*, 2021, Inst. of Electrical and Electronics Engineers, New York, 2021. https://doi.org/10.1109/ICUFN49451.2021.9528738

[12] Markovitz, O., and Segal, M., "Advanced Routing Algorithms for Low Orbit Satellite Constellations," *ICC. 2021—IEEE International Conference on Communications*, Inst. of Electrical and Electronics Engineers, New York, 2021, pp. 1–6. https://doi.org/10.1109/ICC42927.2021.9500740

[13] Roth, M. M. H., Brandt, H., and Bischl, H., "Distributed SDN-Based Load-Balanced Routing for Low Earth Orbit Satellite Constellation Networks," *11th Advanced Satellite Multimedia Systems Conference and the 17th Signal Processing for Space Communications Workshop (ASMS/SPSC)*, 2022, Inst. of Electrical and Electronics Engineers, New York, 2022, pp. 1–8. https://doi.org/10.1109/ASMS/SPSC55670.2022.9914690

[14] Pachler de la Osa, N., Guerster, M., del Portillo Barrios, I., Crawley, E., and Cameron, B., "Static Beam Placement and Frequency Plan Algorithms for LEO Constellations," *International Journal of Satellite Communications and Networking*, Vol. 39, No. 1, 2021, pp. 65–77. https://doi.org/10.1002/sat.1345

[15] Ha, V. N., Lagunas, E., Abdu, T. S., Chaker, H., Chatzinotas, S., and Grotz, J., "Large-Scale Beam Placement and Resource Allocation Design for MEO-Constellation SATCOM," arXiv preprint arXiv: 2303.06372, 2023. https://doi.org/10.1109/ICCWorkshops57953.2023.10283806

[16] Wu, Y., Hu, G., Jin, F., and Zu, J., "A Satellite Handover Strategy Based on the Potential Game in LEO Satellite Networks," *IEEE Access*, Vol. 7, 2019, pp. 133,641–133,652. https://doi.org/10.1109/ACCESS.2019.2941217

[17] He, S., Wang, T., and Wang, S., "Load-Aware Satellite Handover Strategy Based on Multi-Agent Reinforcement Learning," *GLOBECOM 2020—2020 IEEE Global Communications Conference*, Inst. of Electrical and Electronics Engineers, New York, 2020, pp. 1–6. https://doi.org/10.1109/GLOBECOM42002.2020.9322449

[18] Hozayen, M., Darwish, T., Kurt, G. K., and Yanikomeroglu, H., "A Graph-Based Customizable Handover Framework for LEO Satellite Networks," *2022 IEEE Globecom Workshops (GC Wkshps)*, Inst. of Electrical and Electronics Engineers, New York, 2022, pp. 868–873. https://doi.org/10.1109/GCWkshps56602.2022.10008514

[19] Pei, W., Zhu, R., Wei, J., Zhang, Y., Zhang, W., Xi, C., and Yang, B., "Stability and Satisfaction Index Optimization for Beam Allocation in Mega LEO Constellation," *2023 Opto-Electronics and Communications Conference (OECC)*, Inst. of Electrical and Electronics Engineers, New York, 2023, pp. 1–5. https://doi.org/10.1109/OECC56963.2023.10209707

[20] Kuhn, H. W., "The Hungarian Method for the Assignment Problem," *Naval Research Logistics Quarterly*, Vol. 2, No. 1-2, 1955, pp. 83–97. https://doi.org/10.1002/nav.3800020109

[21] 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

[22] Colton, K., and Klofas, B., "Supporting the Flock: Building a Ground Station Network for Autonomy and Reliability," *SmallSat Conference—Technical Session IX, Ground Systems, Planet Labs*, 2016, pp. 1–7, https://api.semanticscholar.org/CorpusID:55930756.

[23] van der Horst, J., and Noble, J., "Distributed and Centralized Task Allocation: When and Where to Use Them," *2010 Fourth IEEE International Conference on Self-Adaptive and Self-Organizing Systems Workshop*, Inst. of Electrical and Electronics Engineers, New York, 2010, pp. 1–8. https://doi.org/10.1109/SASOW.2010.8

[24] Picard, G., "Auction-Based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions," arXiv preprint arXiv: 2106.03548, 2021. https://doi.org/10.5555/3545946.3598677

[25] 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

[26] Phillips, S., and Parra, F., "A Case Study on Auction-Based Task Allocation Algorithms in Multi-Satellite Systems," *AIAA Scitech 2021 Forum*, AIAA Paper 2021-0185, 2021. https://doi.org/10.2514/6.2021-0185

[27] Burkard, R., Dell'Amico, M., and Martello, S., *Assignment Problems: Revised Reprint*, SIAM, 2012. https://doi.org/10.1137/1.9781611972238

[28] Choi, H.-L., Brunet, L., and How, J. P., "Consensus-Based Decentralized Auctions for Robust Task Allocation," *IEEE Transactions on Robotics*, Vol. 25, No. 4, 2009, pp. 912–926. https://doi.org/10.1109/TRO.2009.2022423

[29] Bertsekas, D., *Network Optimization: Continuous and Discrete Models*. Vol. 8, Athena Scientific, Belmont, MA, 1998.

[30] Zavlanos, M. M., Spesivtsev, L., and Pappas, G. J., "A Distributed Auction Algorithm for the Assignment Problem," *47th IEEE Conference on Decision and Control*, Inst. of Electrical and Electronics Engineers, New York, 2008, pp. 1212–1217.

[31] Rawlings, J., Mayne, D., and Diehl, M., *Model Predictive Control: Theory, Computation, and Design*, Nob Hill Publ., Madison, WI, 2017.

S. D'Amico
Associate Editor