Distributed Observation Allocation for a Large-Scale Constellation

Shreya Parjan
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, USA
shreya.parjan@jpl.nasa.gov

Steve Chien
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, USA
steve.a.chien@jpl.nasa.gov

Ryan Harrod
Jet Propulsion Laboratory
California Institute of Technology
Pasadena, USA
ryan.w.harrod@jpl.nasa.gov

ABSTRACT

Increased space and ground sensing are enabling new measurements of a wide range of Earth Science phenomena, including volcanism, flooding, wildfires, and weather. Large-scale observation constellations of hundreds of assets already exist (e.g. Planet) with several constellations of tens of thousands of assets planned. New challenges exist to rapidly assimilate available data and to optimize measurements (e.g. direct assets) to best observe these complex and dynamic spatiotemporal phenomena. We describe automated centralized and distributed Artificial Intelligence/Multi Agent methods to allocate observations in a constellation and compare their performance using realistic problem and orbit distributions.

KEYWORDS

Sensorweb, Artificial Intelligence, Autonomous Systems, Internet of Things, Distributed Constraint Optimization, Multi-Agent Planning

ACM Reference Format:
Shreya Parjan, Steve Chien, and Ryan Harrod. 2022. Distributed Observation Allocation for a Large-Scale Constellation. In *The 13th Workshop on Optimization and Learning in Multiagent Systems (OptLearnMAS-22), at AAMAS 2022 (virtual), Auckland, New Zealand, May 10, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

Worldwide there is an explosion of information sources relevant to environmental monitoring. Ground based sensors are being deployed at an incredible rate, and their data is more easily accessible via the Internet of Things (IoT). This explosion of networked sensors even extends to space, where traditional and New Space actors have deployed worldwide monitoring assets such as Terra and Aqua, Suomi-NPP, Sentinel, and Planet's Dove, Skysat, and other constellations (with over 100 spacecraft).

With many potential information sources and space assets come two distinct problems:

(1) combining the many information sources to track complex spatiotemporal science phenomena; and
(2) tasking the large set of space assets with varying orbits, costs, and capabilities.

Elsewhere the end-to-end sensorweb concept has been described [4] including deployments to track flooding [7, 11], volcanic activity [5], and wildfires [6]. However, those pilots did not study control of constellations as they utilized only Earth Observing-1 under direct sensorweb control, although they did submit requests to commercial providers in a federated approach [3]. Prior work in automated scheduling of constellations includes operational control of Dove [15] and Skybox (later Skysat) [1] as well as recent studies on coordination in large-scale constellations like the work described in this paper [2, 12, 14]. Most of these describe centralized resource allocation approaches, though notable exceptions exist [13, 14]. Centralized approaches are vulnerable to: 1. loss or compromise of the master node and 2. unreliable communication among spacecraft. Even if a new master node can be elected, this may be a time-consuming process. Distributed resource allocation avoids these issues.

Motivated by work on Maximum Gain Messaging (MGM) and Distributed Stochastic Algorithm (DSA), two incomplete Distributed Constraint Optimization Problem (DCOP) algorithms [10, 16], we present two types of Broadcast Decentralized (BD) algorithms, BD Request Satisfaction and BD Contention, for a less computationally intensive, heuristic search-based approach to the problem of observation request allocation for large-scale satellite constellations. We aim to maximize the number of requests satisfied by agent observations and minimize the number of future requests agents cannot observe due to data volume or slew constraints.

Existing work [13, 14] describes Parallel Single-Item Auctions and Sequential Single-Item Auctions that rely on a single auctioneer agent. In our BD algorithms, like the Consensus-Based Bundle Algorithm (CBBA) [13], coordination instead occurs exclusively between scheduling agents that may vary in their orbits, costs, and capabilities. Unlike the CBBA approach, our BD algorithms consider requests iteratively, eliminating the overhead of constructing and evaluating bundles of requests. Existing work also applies the complete Distributed Pseudo-tree Optimization Procedure (DPOP) algorithm to solving the observation request allocation problem [14]. In contrast, our MGM and DSA inspired methods use a heuristic, semi-stochastic approach for satellite agents to update whether they are self-assigned to attempt to schedule an overflight for a request, based on shared information about request satisfaction alone (BD Request Satisfaction) or request satisfaction along with reward and number of excess overflights for the current request (BD Contention).

In this paper, Section 2 describes the request allocation problem for large-scale constellations, Section 3 compares our Broadcast Decentralized algorithms to our Centralized and Highly Decentralized algorithms, and Section 4 evaluates all algorithms on a sample allocation problem involving thousands of observation requests distributed among hundreds of satellites.

The 13th Workshop on Optimization and Learning in Multiagent Systems (OptLearnMAS-22), at AAMAS 2022 (virtual), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 10, 2022, Auckland, New Zealand.

2 PROBLEM FORMULATION

We study the problem of allocating observation requests to satellites. Problem inputs are:

(1) $[H_s, H_e]$: the scheduling horizon, starting at time H_s and ending at time H_e;
(2) A: a set of agents $\{a_1 \dots a_m\}$, where individual spacecraft are the primary agents in our implementation[1];
(3) K: a set of orbits $\{k_1 \dots k_m\}$, one for each spacecraft agent in A;
(4) T: a set of point targets $\{t_1 \dots t_o\}$, each defined with a name and single pair of coordinates; and
(5) R: a set of requests $\{r_1 \dots r_p\}$, where a request r_c is defined by which target t_b to observe and when in $[H_s, H_e]$ the user would like t_b to be observed.

From the above, we generate a set of overflights where each overflight is an opportunity for a spacecraft to view a specific target to satisfy a request. Associated with each such overflight is a status indicating if (in the current schedule) the spacecraft elects to satisfy that request at that opportunity, and an associated reward for that satisfied overflight, request pair.

(6) O: a set of overflights $\{O_{111} \dots O_{hij}\}$. O_{hij} specifies a time (in Unix seconds) agent a_h could potentially schedule request r_i. To distinguish between multiple overflights by the same agent for the same request, we use an additional overflight index j.
(7) S: a set of statuses $\{s_{11} \dots s_{mp}\}$. s_{mp} has value 1 if agent a_m schedules an observation to satisfy request r_p and 0 otherwise.
(8) W: a set of computed rewards (Table 1) $\{w_{11} \dots w_{mp}\}$. w_{mp} specifies the reward agent a_m reaps for satisfying request r_p.

We enforce the following physical constraints:

- *Slew*: in our experiments we use a simple model requiring $|O_{amj} - O_{ank}| > offset$, where O_{amj} and O_{ank} are overflights by the same agent for different requests and *offset* may be specified by the user. We use *offset* = 30 seconds, but in a more detailed model it would depend upon the pointings required for the targets and spacecraft agility.
- *Data Volume*: an agent cannot schedule more than n_{cap} requests during the scheduling horizon. In our experiments, $n_{cap} = 1.5 \times |R|/|A|$, so no single spacecraft can get more than 150% of an evenly divided share of requests.

Given the above formulation, the scheduler's objective is to select a subset of the total overflights to maximize some function of satisfied requests. In our preliminary formulation, the objective is maximization of the number of requests satisfied but more realistic formulations incorporating request priority, fairness, and other metrics are clear areas for future work. A solution has the following form:

$$\max \sum_{i=1}^{m} \sum_{j=1}^{p} w_{ij} \text{ such that (2) and (3) hold:} \tag{1}$$

$$s_{id} + s_{ie} \leq 1 \text{ if } |O_{idj} - O_{iek}| \leq offset \tag{2}$$

where O_{idj} and O_{iek} are overflights selected by agent a_i to observe each pair of requests r_d and r_e.

$$\sum_{j=1}^{p} s_{ij} \leq n_{\text{cap}} \tag{3}$$

for each agent a_i.

3 A RANGE OF APPROACHES

We present and analyze several families of algorithms for the request-overflight allocation problem. For each target, a request may be either for a single (e.g. observe on day 3) or repeat (e.g. observe every hour) observation during $[H_s, H_e]$. We consider three algorithms: Centralized, Broadcast Decentralized, and Highly Decentralized. We further subdivide Broadcast Decentralized into Contention-based and Request Satisfaction-based approaches. These methods vary in their levels of centralization and the amount and types of information shared among agents (Figure 1).

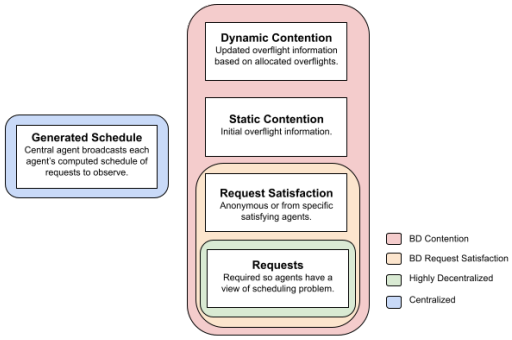

Figure 1: Information shared among agents in the various algorithm types.

In the iterative Centralized algorithm, the central node only shares each agent's schedule of requests to observe after each iteration. In all decentralized algorithms, the initial set of requests must be broadcasted to each agent so it can establish a local view of the scheduling problem. Highly Decentralized is a single-pass algorithm in which all agents receive the set of requests and attempt to schedule them without any inter-agent communication.

The Broadcast Decentralized algorithms consist of two phases: initialization and iteration. In initialization, agents determine how many overflights they have during the scheduling horizon for each request r_i. In the BD Contention algorithm, this is broadcasted out for all other agents to record for the initial iteration of the algorithm. Agents with overflights for r_i self-assign to find an overflight to observe it with some probability, $P_{initialize}$ (Table 1).

[1]In our current formulation, 1 agent = 1 spacecraft, but in alternative formulations such as a federated system, an agent might represent multiple spacecraft or there could be multiple levels of hierarchy. For complex spacecraft platforms with many instruments, multiple agents per spacecraft would also be an option.

In the iteration phase, if a request was satisfied by multiple agents' overflights in the previous iteration, excess agents stochastically unassign from it. If it was unsatisfied, additional unassigned agents probabilistically self-assign to attempt to schedule the request in the current iteration. Agents update whether they are self-assigned to observe a request in the current iteration based on state information from the previous iteration of the algorithm. This allows agents to avoid race conditions both in our current single-threaded simulation of parallel execution and in a future parallelized implementation. In BD Request Satisfaction, agents broadcast whether they satisfied a request and update whether they are self-assigned to attempt to schedule it in the current iteration based on how many other agents satisfied it. In BD Contention, agents broadcast whether they satisfied a request, along with the number of free overflights they have remaining for that request and their reward for satisfying it (a function of the number of free overflights and number of conflicts the selected overflight has with the agent's overflights for other requests).

In the following paragraphs, we will use these functions:

(1) $ASSIGN(r_i, a_j)$: If agent a_j is not at capacity, select the overflight by a_j for request r_i that conflicts with the fewest overflights a_j has for other requests. If successful, mark r_i as satisfied, rule out a_j's overflights that conflict with the identified overflight, add the identified overflight to a_j's schedule of observations, and increment the count of requests a_j is assigned to observe in the scheduling horizon.
(2) $SELF\_ASSIGN(r_i, P_{initialize})$: the current agent checks its set of daytime overflights (solar zenith angle $> 90°$) to see if it has overflights for request r_i. If so, the agent sets its self-assignment status for request r_i from 0 to 1 with probability $P_{initialize}$ (Table 1). Returns a status of 0 or 1.
(3) $BROADCAST(r_i, a_j, items)$: the current agent a_j instructs all agents to record its values for the variables in *items* for request r_i in the current iteration of the algorithm.
(4) $SCHEDULE(r_i)$: the current agent searches for the overflight time it has for request r_i that conflicts with the fewest overflights it has for other requests. If successful, the agent marks r_i as satisfied, rules out its overflights that conflict with the identified overflight, adds the identified overflight to its schedule of observations, and increments the number of requests it is assigned to observe in the scheduling horizon. Returns 1) the assignment as a tuple with form (spacecraft ID, request index, overflight time) and 2) the number of conflicts the selected overflight has with the current agent's other overflights.
(5) $ELIMINATE\_CONFLICTS(assignment_{a_j,r_i})$: given the overflight time selected for agent a_j to observe request r_i in the tuple $assignment_{a_j,r_i}$, agent a_j records which of its overflights are no longer feasible if request r_i is scheduled.
(6) $UPDATE\_STATUS(r_i, P_{assign}, P_{unassign})$: based on whether request r_i was satisfied, the current agent either unassigns itself from request r_i with probability $P_{unassign}$ or assigns itself to request r_i with probability P_{assign} (Figures 2, 3).
(7) $SHUFFLE(R)$: randomize order of the list of requests, R.
(8) $SORT(R)$: sort the list of requests, R, using one of the sorting methods named in Table 1.

3.1 Centralized Algorithm

Our baseline, Algorithm 1, solves the request allocation problem at a master node and uses a Squeaky Wheel Optimization (SWO) [9] to try to find the optimal schedule. The Centralized algorithm considers requests in ascending order by an assigned ranking. The initial ranking for a request is determined by the total number of overflights that all agents have for it in the scheduling horizon. From R, it creates a min-heap R_{min_heap} such that the top of R_{min_heap} is initially the request with the fewest overflights in the scheduling horizon. Each request r_i in R also maintains a max-heap A_{max_heap,r_i} of agents with overflights for r_i. The agent at the top of A_{max_heap,r_i} has the most overflights for r_i.

Algorithm 1 pops the request with the lowest ranking, r_0, from R_{min_heap} and assigns it to the agent with the most free overflights for it in the scheduling horizon. If that agent is not at capacity, the algorithm goes on to consider the request with the next lowest ranking. Otherwise, it assigns r_0 to the agent with the next most overflights for it in the scheduling horizon. It repeats this procedure before moving onto the next request until either r_0 is satisfied or there are no remaining agents with overflights for r_0. A key drawback of the Centralized algorithm is that the master node manages state information for all agents, preventing parallelization. If compromised, the master node can be re-elected, though this process may be time consuming.

Once all requests have been considered, the SWO algorithm rewards and increases the ranking assigned to a request based on whether it was satisfied or not. If the request was satisfied, its ranking is increased. If the request was not satisfied, its ranking is assigned to 0, to move it to the front of the priority queue in the following iteration. Once all requests have been rewarded or penalized, R_{min_heap} is created once more, in ascending order based on ranking and the whole algorithm iterates again. The algorithm continues iterating until a predefined number of iterations is reached, or an early exit condition is detected, such as all satellites being subscribed at capacity.

3.2 Highly Decentralized Algorithm

The Highly Decentralized algorithm sits on the other end of the centralization spectrum, with no feedback shared among agents. For this single-pass algorithm, only the initial set of requests is broadcasted to all agents. Subsequently, agents independently schedule requests based on their local view of the problem alone. Because selecting an overflight to observe a request may eliminate other conflicting overflights, the order in which requests are considered matters. Thus, each agent randomizes the order in which it considers and attempts to schedule requests. In our current implementation of Algorithm 2, requests do not have associated priorities. If an overflight is successfully found, conflicting overflights for other requests are eliminated from the agent's set of available overflights.

3.3 Broadcast Decentralized Algorithms

3.3.1 Motivation: Distributed Constraint Optimization. Our Broadcast Decentralized (BD) algorithms are adapted from the Maximum Gain Messaging algorithm (MGM) and Distributed Stochastic Algorithm (DSA) for solving Distributed Constraint Optimization Problems (DCOPs) [8].

Algorithm 1: Centralized

function Centralized (R, O);
Input : A set of unsatisfied requests R, a set of overflights O.
Output : S, a set of schedules for each agent $a_j \in A$.
while $iteration < max\_iterations$ **do**
  while $|R_{min\_heap}| > 0$ **do**
    pop r_0, request with the next lowest ranking, from R_{min_heap}
    if r_0 *has non-zero overflights in the scheduling horizon* **then**
      while $|A_{max\_heap,r_0}| > 0$ *and* r_0 *unsatisfied* **do**
        pop a_{max,r_0}, agent with the next most overflights for r_0, from A_{max_heap,r_0}
        $ASSIGN(r_0, a_{max,r_0})$
      end
    end
  end
  if *all requests are satisfied* **or** *all satellites are at max capacity* **or** *the priority ordering in the next iteration will be the same as the current iteration* **then**
    exit
  end
  $iteration++$
end
return S;

Algorithm 2: Highly Decentralized

function Highly Decentralized (R, O);
Input : A set of unsatisfied requests R, a set of overflights O.
Output : S, a set of schedules for each agent $a_j \in A$.
for *each agent* $a_j \in A$ **do**
  $SHUFFLE(R)$
  for *each request* $r_i \in R$ **do**
    if a_j *not at capacity* **then**
      $assignment_{a_j,r_i}, num\_conflicts_{a_j,r_i} = SCHEDULE(r_i, a_j)$
      if $assignment_{a_j,r_i}$ *valid* **then**
        $ELIMINATE\_CONFLICTS(assignment_{a_j,r_i})$
      end
    end
  end
end
return S;

DCOPs are defined as tuples $P = (A, X, D, C, \alpha)$:

(1) $A = \{a_1 \dots a_p\}$, a set of agents.
(2) $X = \{x_1 \dots x_n\}$, a set of variables such that $n \geq m$.
(3) $D = \{D_1 \dots D_n\}$, a set of domains for the corresponding $x_i \in X$.
(4) $C = \{c_1 \dots c_k\}$, a finite set of constraint functions involving the variables in X.
(5) α: $X \rightarrow A$, a mapping of variables to agents.

Complete solutions to DCOPs assign a value to each $x_i \in X$ while satisfying all constraint functions in C.

For our problem:

(1) $A = \{a_1 \dots a_m\}$, the set of spacecraft agents.
(2) $X = \{o_{11} \dots o_{mp}, s_{11} \dots s_{mp}, w_{11} \dots w_{mp}\}$ includes variables where, for example: agent a_m has o_{mp} free overflights for request r_p and if $s_{mp} = 1$, a_m schedules request r_p with reward w_{mp}.
(3) $D = \{D_1 \dots D_{3\times m\times p}\}$, a set of domains for the variables in X: the free overflight counts and rewards take on non-negative integer values and the scheduling variables are binary decision variables with domain $\{0, 1\}$.
(4) $C = \{c_1, c_2\}$, where for some agent a_i:
  - c_1: $s_{id} + s_{ie} \leq 1$ if $|O_{idj} - O_{iek}| \leq offset$ (where O_{idj} and O_{iek} are the overflights selected by agent a_i to observe each pair of requests r_d and r_e)
  - c_2: $s_{i1} + \dots + s_{ip} \leq n_{cap}$

As before, our objective (Eq. 1) is to maximize the reward across all agents for satisfying each request in R while adhering to the constraints in C.

Both MGM and DSA are incomplete, synchronous, search-based algorithms. In MGM, an agent communicates exclusively with its neighbors: agents whose variables appear in the same cost function(s) as that agent's own variables. Under our problem formulation, for each request and at each iteration, the set of neighbors would be all agents with free overflights for the request. Each agent initializes its variable values randomly and repeats the following procedure until some termination condition is reached:

(1) Broadcast variable values out to all neighbors.
(2) Receive values of neighbors' variables and compute the maximum gain obtained by changing own variable values.
(3) Broadcast that gain out to all neighbors.
(4) Receive neighbors' gains and update value if own gain is the largest.

DSA is similar, but instead of steps 3 and 4, agents stochastically decide whether to take on the variable values associated with maximum gain.

Combining the stochastic updating procedure from DSA and the more informed updating procedure of MGM, agents in the Broadcast Decentralized algorithms update their self-assignment statuses for a particular request in the semi-stochastic manner outlined in Figures 2, 3 for the Contention and Request Satisfaction algorithms respectively. This helps avoid local maxima when optimizing for the number of successfully scheduled requests over iterations of the algorithms.

The generic DCOP algorithms and our BD algorithms differ in the order in which agents receive broadcasts. To maintain a consistent view of the problem across all agents for each iteration, our agents broadcast state information out to all others. Broadcasting information to all agents increases the overall number of messages exchanged but eliminates the need to continuously recompute or reverify agents' sets of neighbors. An excellent optic for future work is the investigation of methods for computing neighborhoods to

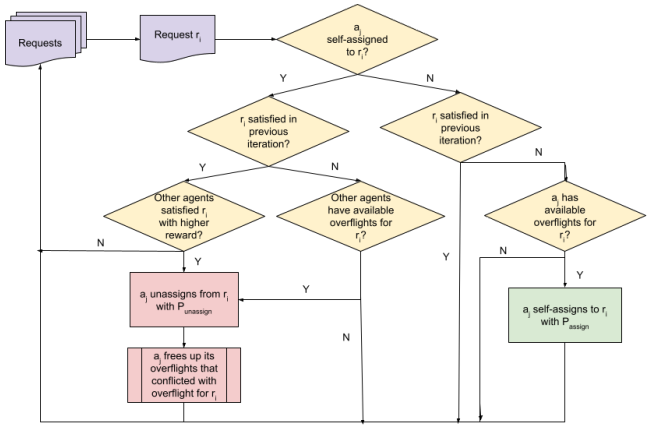

Figure 2: $UPDATE\_STATUS(r_i, P_{assign}, P_{unassign})$: Agent self-assignment update procedure for BD Contention.

limit inter-agent communication to an "as needed" basis (in contrast to our "broadcast to all" approach).

3.3.2 Broadcast Decentralized Algorithmic Variations. We introduce two novel Broadcast Decentralized algorithms with twenty variations along the following axes: required shared information, agent allocation initialization, agent request sort procedure, and agent reward for request satisfaction (Table 1). First, our approaches vary in the amount of information shared between agents. The BD Contention algorithm requires agents to share whether they satisfied a request, their reward for doing so, and the number of free overflights they have remaining for that request every iteration (Algorithm 3). BD Request Satisfaction only requires agents to broadcast whether or not they satisfied a request (Algorithm 4).

In both Broadcast Decentralized algorithms, agents iterate through the set of requests broadcasted to them by the central node. Since a selected overflight to schedule one request may eliminate conflicting overflights that an agent has for another request, the order in which requests are considered matters. For each of the two Broadcast Decentralized algorithms, the variations differ in their sort functions but require agents to share the same information. The three sort variations to determine the order in which agents consider requests are:

(1) *Iterative Global Free Overflight Sort*: Agents sort requests at the start of each iteration in ascending order by the cumulative number of available overflights all agents have for each request.
(2) *Iterative Local Free Overflight Sort*: Agents sort requests at the start of each iteration in ascending order by the number of available overflights they alone have for each request.
(3) *Iterative Random Sort*: Agents randomize the order of the requests at the start of each iteration.

Because Iterative Global Overflight Sort requires contention information (the number of free overflights for a request) to be shared among agents, the only two Broadcast Decentralized Request Satisfaction sort variations are Iterative Local Free Overflight Sort and Iterative Random Sort.

Table 1: Broadcast Decentralized algorithmic variations.

Feature	*BD Contention*	*BD Request Satisfaction*
Required Shared Information for Iteration Phase Update Procedure	• Reward for observing request • Number of remaining, free overflights for request • Request satisfaction	• Request satisfaction
Allocation Initialization ($P_{initialize}$) Options	• P_{random} = user-specified probability • P_{ratio} = # of agent overflights for request / # of all agent overflights for request • P_{total_ofs} = 1 / # of all agent overflights for request)	• P_{random} = user-specified probability
Local Request Sort Procedure Variations	• Iterative Global Free Overflight Sort (GFO) • Iterative Local Free Overflight Sort (LFO) • Iterative Random Sort (RD)	• Iterative Local Free Overflight Sort (LFO) • Iterative Random Sort (RD)
Reward Function Variations for Iteration-Phase Update Procedure	• $W_{difference}$ = # of agent overflights for request - # of agent conflicts with selected overflight • W_{ratio} = # of agent overflights for request / 1 + # of agent conflicts with selected overflight	• N/A (reward not used)

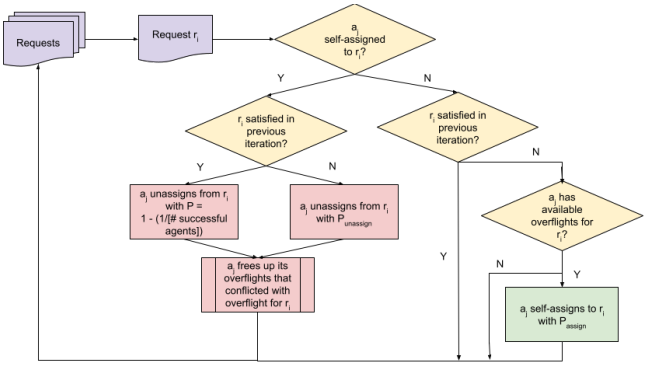

Figure 3: $UPDATE\_STATUS(r_i, P_{assign}, P_{unassign})$**: Agent self-assignment update procedure for BD Request Satisfaction.**

During the initialization phase of BD Contention, $P_{initialize}$, the probability with which an agent initially self-assigns to attempt to schedule a request, can take on one of three forms: P_{random} is a user-specified value between 0 and 1. P_{ratio} is the ratio of the number of initial overflights the agent has for the request to the total number of initial overflights all agents have for the request. Finally, P_{total_ofs} is a ratio of 1 to the total number of initial overflights all agents have for the request. Since overflight information isn't shared in BD Request Satisfaction, $P_{initialize}$ must be supplied by the user as P_{random}.

During the iteration phase, the procedures by which agents update whether they are self-assigned to a request for BD Contention and BD Request Satisfaction are outlined in Figures 2, 3, respectively. In general, agents update whether they are self-assigned to a request in the current iteration based on broadcasted information from the previous iteration. BD Contention requires agents to consider their reward for satisfying a request in their update procedure. The reward may either be a simple difference between the number of free overflights the agent has for the request and the number of conflicts its selected overflight has with the agent's other overflights or a ratio of the number of free overflights to 1 + the number of conflicts.

The first reward function, $W_{difference}$, rewards agents that select an overflight with fewer conflicts with their overflights for other requests to schedule the request and have more free overflights (more excess capacity) for the request. Agents with more capacity are also more likely to have more conflicts and $W_{difference}$ can also take on negative values when the number of conflicts exceeds the number of free overflights. The second reward function, W_{ratio}, offers users a non-negative alternative by computing a ratio instead.

When an agent unassigns itself from a request r_u, it must add back the overflights for other requests that conflicted with the overflight it may have selected to observe r_u. In BD Contention, the agent must also broadcast the updated number of overflights for each of these other requests to maintain consistency across all agents' views of the problem.

Algorithm 3: Broadcast Decentralized Request Satisfaction

function BDRequestSatisfaction (R, O);
Input : A set of requests R, a set of overflights O.
Output : S, a set of schedules for each agent $a_j \in A$.
for *each agent* $a_j \in A$ **do**
  $SORT(R)$
  for *each request* $r_i \in R$ *that* a_j *has overflights for* **do**
    $assignment\_status_{a_j,r_i} = SELF\_ASSIGN(r_i, a_j, P_{initialize})$
    $num\_ofs_{a_j,r_i} = |[o_{ji1}...o_{jin}]|$
    $of\_found_{a_j,r_i} = False$
    $BROADCAST(r_i, a_j, [of\_found_{a_j,r_i}])$
  end
end
for *i iterations* **do**
  if *all requests are satisfied* **or** *all satellites are at max capacity* **or** *the requests satisfied in the previous two iterations are the same* **then**
    exit
  end
  for *each agent* $a_j \in A$ **do**
    for *each request* $r_i \in R$ **do**
      if *not on the first iteration* **then**
        $UPDATE\_STATUS(r_i, P_{assign}, P_{unassign})$
      end
      if $assignment\_status_{a_j,r_i} == 1$ **then**
        $assignment_{a_j,r_i}, num\_conflicts_{a_j,r_i} = SCHEDULE(r_i, a_j)$
        if $assignment_{a_j,r_i}$ *valid* **then**
          $ELIMINATE\_CONFLICTS(assignment_{a_j,r_i})$
          $of\_found_{a_j,r_i} = True$
          $BROADCAST(r_i, a_j, [of\_found_{a_j,r_i}])$
        end
      end
    end
  end
end
return S;

Algorithm 4: Broadcast Decentralized Contention

function BDContention (R, O);
Input : A set of requests R, a set of overflights O.
Output: S, a set of schedules for each agent $a_j \in A$.
for *each agent* $a_j \in A$ **do**
  for *each request* $r_i \in R$ *that* a_j *has overflights for* **do**
    $num\_ofs_{a_j,r_i} = |[o_{ji1}...o_{jin}]|$
    $BROADCAST(r_i, a_j, [num\_ofs_{a_j,r_i}])$
  end
end
for *each agent* $a_j \in A$ **do**
  $SORT(R)$
  for *each request* $r_i \in R$ *that* a_j *has overflights for* **do**
    $assignment\_status_{a_j,r_i} = SELF\_ASSIGN(r_i, a_j, P_{initialize})$
  end
end
for *i iterations* **do**
  if *all requests are satisfied* **or** *all satellites are at max capacity* **or** *the requests satisfied in the previous two iterations are the same* **then**
    exit
  end
  for *each agent* $a_j \in A$ **do**
    for *each request* $r_i \in R$ **do**
      if *not on the first iteration* **then**
        $UPDATE\_STATUS(r_i, P_{assign}, P_{unassign})$
      end
      if $assignment\_status_{a_j,r_i} == 1$ **then**
        $assignment_{a_j,r_i}, num\_conflicts_{a_j,r_i} = SCHEDULE(r_i, a_j)$
        if $assignment_{a_j,r_i}$ *valid* **then**
          $ELIMINATE\_CONFLICTS(assignment_{a_j,r_i})$
          $of\_found_{a_j,r_i} = True$
          $reward_{a_j,r_i} = W_{difference}$ or W_{ratio}
          $BROADCAST(r_i, a_j, [of\_found_{a_j,r_i}, num\_conflicts_{a_j,r_i}, reward_{a_j,r_i}])$
        end
      end
    end
  end
end
return S;

4 EMPIRICAL EVALUATION

We used grid search to tune the values for $P_{initialize}$, P_{assign}, and $P_{unassign}$ for each algorithm. We searched the range of probabilities from 0.1 to 1.0 for the combination of probabilities that performed the best on a training problem of 10 Skysats scheduling 1,030 requests (634 point targets, 66 selected for daily observation) over 1 week (1 August 2021-8 August 2021) for a subset of BD algorithm variations (Table 2). Notably, the algorithms performed best with a low probability of initial assignment, $P_{initialize}$, and high probability of self-assignment, P_{assign}. If fewer agents are initially self-assigned to observe a request, more may self-assign only as needed with high certainty, without agent capacities reaching n_{cap} early in execution.

Table 2: Best probability values for a subset of algorithms after grid search.

BD <algorithm type>-<sort type>-<reward type>	$P_{initialize}$	P_{assign}	$P_{unassign}$
BD Contention-LFO-Difference	0.1	0.9	0.6
BD Contention-GFO-Ratio	0.1	1.0	0.2
BD Contention-RD-Ratio	0.1	1.0	1.0
BD Request Satisfaction-RD	0.1	0.9	0.7

We evaluate our algorithms on the following problem:

(1) Scheduling Horizon: 12 hours (00:00:00 1 August 2021 - 12:00:00 1 August 2021).
(2) Requests: 634 terrestrial point targets were each divided into 12 independent requests, corresponding to each hour of the scheduling horizon (7,608 total). Across all agents, there were non-zero overflights in the scheduling horizon for 3,139 of these requests, yielding the input set.
(3) Agents: 228 total, composed of Planet Skysats A, B, and C 1-19, plus 207 Doves available on the NORAD Celestrak site as of 8 February 2022.

The Centralized and each of the BD algorithms run until one of four termination conditions: 1) no improvement between consecutive iterations, 2) all requests satisfied, 3) all agents at capacity, or 4) the number of iterations hits some user-specified cap. In the results below, all multi-iteration algorithms terminated at condition 1. Figures 5 and 6 show results from Centralized, Highly Decentralized, and the subset of Broadcast Decentralized algorithm variations with the following naming convention: BD-<algorithm type>-<sort type>-<$P_{initialize}$ type>-<reward type>. Highly Decentralized and Centralized ran for one and three iterations respectively, but we plot their metrics from the time of their last iterations to time of the last iteration of the longest running algorithm for clearer comparison.

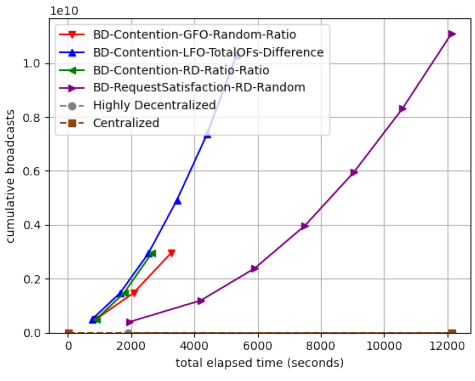

Figure 4: Broadcasts by iteration and algorithm.

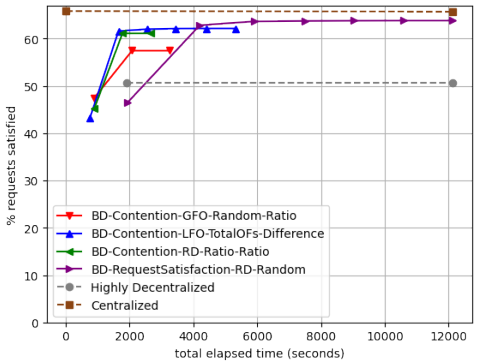

Figure 5: Request satisfaction by iteration and algorithm.

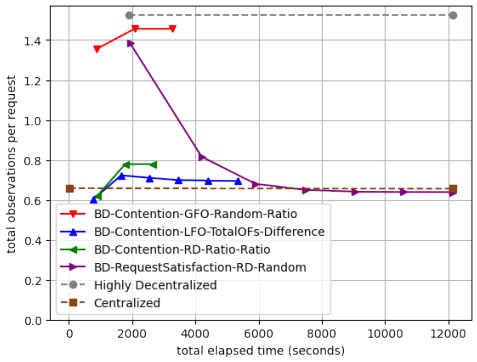

Figure 6: Total observations per request by iteration and algorithm.

In the Centralized algorithm, a master node only broadcasts a final schedule of requests to observe out to each agent. For the decentralized algorithms, the master node initially broadcasts each request to every agent and every agent broadcasts its current schedule of requests back to that node at the end of each iteration. In the iteration phase of the Broadcast Decentralized algorithms, the number of broadcasts is $O(n^2)$ in the number of agents (Figure 4). The size of the inter-agent broadcasts is only a few words, with agents broadcasting three times more information in BD Contention than BD Satisfaction (Table 1). A user may prefer BD Request Satisfaction to BD Contention if they aim to limit inter-agent communication.

Due to the lack of coordination among agents, the Highly Decentralized algorithm has the highest rate of redundant observations (Figure 6) and a satisfaction rate that lags behind the rest (Figure 5). The Centralized algorithm provides an upper bound to compare our distributed methods to but requires a single agent to manage state for all others and does not allow for computation to be parallelized unlike the distributed algorithms. The BD algorithms employ distributed heuristic search and exchange guaranteed optimality of solution quality in favor of speed of scheduling. Since our current results are from a single-threaded simulation of distributed execution across agents, there is an $O(n)$ inflation in runtime. For each iteration and each agent, the solutions for our instance of the continuous planning problem require only a few seconds.

Our short scheduling horizon and large number of repeat observations result in a highly conflicted scheduling problem. The better performing distributed algorithms have a lower $P_{initialize}$ and successfully eliminate redundant overflights every iteration until approaching a 1:1 ratio of total observations per request to proportion of requests satisfied (Figures 5, 6). This suggests that in highly conflicted scheduling problems, it is more effective for minimal agents to initially attempt to schedule a request and additional agents to self-assign as needed than for excess agents to unassign themselves between iterations.

5 CONCLUSIONS

Increased space observation capabilities represent opportunities for improved space-based monitoring of a wide range of Earth Science phenomena including but not limited to volcanoes, flooding, wildfires, cryosphere, and biosphere. Increased in-situ sensing via IoT also can provide significant information to study these phenomena. One challenge to such developments is that of allocating observation assets. We describe a range of centralized and decentralized methods for allocating satellite agents to observations.

In particular, we outline two Broadcast Decentralized algorithms that implement heuristic search approaches inspired by DCOP algorithms to search for quick, approximate solutions to the large-scale constellation request allocation problem with low data volume for agent coordination. Most coordination occurs exclusively between scheduling agents without the supervision of a master agent or the redundancy of scheduling without inter-agent communication.

We allow users to moderate the amount of information shared among agents when coordinating on request allocation, how agents determine whether to initially self-assign to requests, the order in which they consider requests, and the reward functions agents use to assess their request satisfaction. We plan to expand on this initial study of distributed algorithms to solve the request allocation problem by 1) exploring challenges in the continuous scheduling domain, where agents may be lost or compromised and requests may be modified, added, or removed during scheduling and 2) studying communications patterns between agents.

ACKNOWLEDGMENTS

Portions of this work were performed by the Jet Propulsion Laboratory, California Institute of Technology, under contract from the National Aeronautics and Space Administration.

REFERENCES

[1] Sean Augenstein, Alejandra Estanislao, Emmanuel Guere, and Sean Blaes. 2016. Optimal Scheduling of a Constellation of Earth-Imaging Satellites, for Maximal Data Throughput and Efficient Human Management. In *ICAPS 2016 (International Conference on Automated Planning Scheduling)*. London, UK. http://www.aaai.org/ocs/index.php/ICAPS/ICAPS16/paper/download/13173/12696

[2] James Boerkoel, James Mason, Daniel Wang, Steve A. Chien, and Adrien Maillard. 2021. An Efficient Approach for Scheduling Imaging Tasks Across a Fleet of Satellites. In *International Workshop on Planning Scheduling for Space (IWPSS)*. https://ai.jpl.nasa.gov/public/papers/Boerkoel_IWPSS2021_paper_23.pdf

[3] Andrew Branch, Steve A. Chien, Yulia Marchetti, Hui Su, Longtao Wu, James Montgomery, Maggie Johnson, Benjamin Smith, Lukas Mandrake, and Peyman Tavallali. 2021. Federated Scheduling of Model-Driven Observations for Earth Science. In *International Workshop on Planning Scheduling for Space (IWPSS)*. https://ai.jpl.nasa.gov/public/papers/Branch_IWPSS2021_paper_12.pdf

[4] Steve A. Chien, Benjamin Cichy, Ashley Davies, Daniel Tran, Gregg Rabideau, Rebecca Castaño, Rob Sherwood, Daniel Mandl, Stuart Frye, Seth Shulman, Jeremy Jones, and Sandy Grosvenor. 2005. An Autonomous Earth-Observing Sensorweb. *IEEE Intelligent Systems* (May/June 2005), 16–24. https://ai.jpl.nasa.gov/public/papers/chien_IEEEIS2005_AutonomousSenworweb.pdf

[5] Steve A. Chien, Ashley G. Davies, Joshua Doubleday, Daniel Q. Tran, David Mclaren, Wayne Chi, and Adrien Maillard. 2020. Automated Volcano Monitoring Using Multiple Space and Ground Sensors. *Journal of Aerospace Information Systems (JAIS)* 17:4 (2020), 214–228. https://doi.org/10.2514/1.I010798

[6] Steve A. Chien, Joshua Doubleday, David Mclaren, Ashley Davies, Daniel Tran, Veerachai Tanpipat, Siri Akaakara, Anuchit Ratanasuwan, and Daniel Mandl. 2011. Space-based Sensorweb Monitoring of Wildfires in Thailand. In *International Geoscience and Remote Sensing Symposium (IGRSS 2011)*. Vancouver, BC. https://ai.jpl.nasa.gov/public/papers/chien_igarss2011_spacebased.pdf

[7] Steve A. Chien, David Mclaren, Joshua Doubleday, Daniel Tran, Veerachai Tanpipat, and Royol Chitradon. 2019. Using Taskable Remote Sensing in a Sensor Web for Thailand Flood Monitoring. *Journal of Aerospace Information Systems (JAIS)* 16, 3 (2019), 107–119. https://doi.org/10.2514/1.I010672 arXiv:https://doi.org/10.2514/1.I010672

[8] Ferdinando Fioretto, Enrico Pontelli, and William Yeoh. 2018. Distributed Constraint Optimization Problems and Applications: A Survey. *Journal of Artificial Intelligence Research* 61 (Mar 2018), 623–698. https://doi.org/10.1613/jair.5565

[9] David E. Joslin and David P. Clements. 1999. Squeaky Wheel Optimization. *Journal of Artificial Intelligence Research* 10 (May 1999), 353–373. https://doi.org/10.1613/jair.561

[10] Rajiv T. Maheswaran, Jonathan P. Pearce, and Milind Tambe. 2004. Distributed Algorithms for DCOP: A Graphical Game-Based Approach. In *17th International Conference on Parallel and Distributed Computing Systems (PDCS-2004)*.

[11] Daniel Mandl, Stuart Frye, Pat Cappelaere, Matthew Handy, Fritz Policelli, Mc-cloud Katjizeu, Guido Langenhove, Guy Aube, Jean-Francois Saulnier, Rob Sohlberg, Julie Silva, Nataliia Kussul, Sergii Skakun, Stephen Ungar, Robert Grossman, and Jörg Szarzynski. 2013. Use of the Earth Observing One (EO-1) Satellite for the Namibia SensorWeb Flood Early Warning Pilot. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* 6 (04 2013), 298–308. https://doi.org/10.1109/JSTARS.2013.2255861

[12] Sreeja Nag, Alan S. Li, and James H. Merrick. 2018. Scheduling algorithms for rapid imaging using agile Cubesat constellations. *Advances in Space Research* 61, 3 (Feb. 2018), 891–913. https://doi.org/10.1016/j.asr.2017.11.010

[13] Sean Phillips and Fernando Parra. [n.d.]. *A Case Study on Auction-Based Task Allocation Algorithms in Multi-Satellite Systems.* https://doi.org/10.2514/6.2021-0185 arXiv:https://arc.aiaa.org/doi/pdf/10.2514/6.2021-0185

[14] Gauthier Picard. 2021. Auction-based and Distributed Optimization Approaches for Scheduling Observations in Satellite Constellations with Exclusive Orbit Portions. arXiv:2106.03548 [cs.AI]

[15] Vishwa Shah, Vivek Vittaldev, Leon Stepan, and Cyrus Foster. 2019. Scheduling the world's largest earth-observing fleet of medium-resolution imaging satellites. In *International Workshop on Planning and Scheduling for Space*. Organization for the 2019 International Workshop on Planning and Scheduling ..., 156–161.

[16] Weixiong Zhang, Guandong Wang, Zhao Xing, and Lars Wittenburg. 2005. Distributed Stochastic Search and Distributed Breakout: Properties, Comparison and Applications to Constraint Optimization Problems in Sensor Networks. *Artif. Intell.* 161, 1–2 (jan 2005), 55–87.