An Integer Programming Approach to Observation Scheduling for Space Domain Awareness

Grant Nations
KBR
985 Space Center Drive
Colorado Springs, CO 80915
grant.nations@us.kbr.com

Justin Fletcher
United States Space Force Space Systems Command
550 Lipoa Pkwy
Kihei, HI 96753
justin.fletcher.14.ctr@us.af.mil

***Abstract*— Satellites underpin all economic, military, and scientific activity in space. Ground-based telescope observations provide much of the tracking information that enables collision avoidance and space traffic management, but recent growth in the size and dynamism of the satellite population divides the available observing capacity between multiple competing observation objectives. The task of selecting an optimal assignment of telescope observing capacity to observation objectives is NP-hard and involves domain-specific considerations such as sequence-dependent transition times, observability constraints, occlusion avoidance, and astrodynamic limitations. These features conspire to prevent direct and efficient application of traditional scheduling approaches over long time horizons.**

This work maps the task of scheduling ground-based sensor observations to an unrelated parallel machine scheduling problem with sequence-dependent transition times, restricted time windows, and a fixed time horizon. We contribute an integer programming model of the task, complete with inequality formulations of domain-specific constraints, and a decomposition algorithm that discovers reward-optimal schedules. Using the open-source CBC solver, the integer programming model can reliably be solved for problem instances with up to 6 sensors and 40 targets for a 4 minute time horizon within an equal runtime limit. To address more challenging problem instances, we propose a decomposition algorithm that supports early termination and returns suboptimal schedules along with an associated optimality gap for instances with up to 6 sensors and 200 targets over a 15 minute time horizon.

TABLE OF CONTENTS

1. INTRODUCTION

Space Domain Awareness (SDA) is the field of study concerned with establishing and maintaining an accurate world model that encompasses both natural and artificial objects in space. SDA enables many human activities in space, such as manned space flight, space exploration, communication, and military operations. Scheduling ground-based observations in support of SDA objectives is a complex task due to several factors. One of the major challenges is that space observations are subject to numerous constraints, such as meeting target position windows and staying within limits on Sun separation, galactic latitude, and Moon illumination. However, the scheduling problem is not just about meeting these constraints but also recognizing priorities assigned to observations and adjudicating between competing objectives. Some observations are of higher importance than others, and an optimal schedule must take into account both the constraints and the priority of the observations.

Often, scheduling algorithms for SDA are designed to maintain accurate kinematic state estimates of known targets (examples are given in Section 2). These schedulers are given a set of targets or objects and determine the observation frequency for each target to ensure that the kinematic state estimates of all targets are maintained with minimal error or kept below a chosen threshold. The approach taken in this work is different. Instead of providing the scheduler with a set of targets, we provide a set of "observation blocks". These blocks are wrappers around a target observation that contain additional constraint information (e.g., limiting the target to only be scheduled above a certain altitude), reward, and observation duration; an observation block can only be scheduled once or not at all. This enables the application of general-purpose scheduling approaches to SDA sensor tasking for arbitrary goals, such as catalog maintenance, periodic target revisits, region search, pattern-of-life analysis, and spectral positive identification. The proposed solution for this unique approach is an integer programming (IP) model that treats the ground-based SDA scheduling problem as an unrelated parallel machine scheduling problem with sequence-dependent setup times, restricted windows, and a fixed time horizon.

The objective of the proposed model is to maximize the total observational reward subject to all restricted windows placed on a block. The restricted windows prevent the assignment of each block to a range of time periods, and they are derived from the constraints that each block is subject to (details on this step in section 5). The following assumptions are made for the IP model:

- The time horizon is discretized into periods whose lengths are equal to the schedule's time resolution.
- The deadline of each block is the end of the final time period.
- Each block can start no earlier than the beginning of the first time period.
- All blocks are available to be assigned at the beginning of the first time period.
- A sensor can process no more than one block at a time.

979-8-3503-0462-6/24/$31.00 ©2024 IEEE

- Preemption is not allowed.
- Block durations are the same across all sensors.
- Sequence-dependent transition times between blocks are specific to the sensor to which the blocks are assigned and the time period in which the transition begins.
- The time to transition to the first block is 0.
- The time to transition from the final block is 0.
- A block's restricted windows are sensor-specific and define the periods in which a block may not be scheduled on the given sensor.
- A block can be assigned to at most one sensor.

This paper is arranged as follows: Section 2 first presents a review of considerations of scheduling for SDA, then of similar approaches to the problem in existing literature. Section 3 presents the IP model for the parallel machine scheduling problem with sequence dependent setup times and a fixed time horizon. Section 4 introduces the decomposition of the IP model. Section 5 presents an overview of the entire scheduling program. Section 6 explains the computational experiments performed, including an overview of the computational environment. Finally, section 7 summarizes the outcomes of this research and suggests future areas of study for this problem.

2. BACKGROUND AND RELATED WORK

Scheduler Considerations

Optimal SDA sensor scheduling remains intractable, even after many simplifications and assumptions. In [1], an overview of the main challenges within SDA is given, with one being predictive analytics or "predictive Space Situational Awareness" (SSA). Predictive SSA involves using mathematical models and algorithms to predict the future behavior of objects in space, such as satellites, debris, and other space objects. Sensor tasking and scheduling lie at the cornerstone of predictive SSA, as accurate predictions cannot be made without the collection of comprehensive data. Scheduling for SDA is a strongly NP-hard problem, and as such has been the subject of research with many different approaches.

Abstracting the scheduling problem for SDA to achieve limited tractability requires the introduction of simplifications. There are several critical areas subject to simplifications, including the following:

1. *Transition times.* An exact determination of the transition time between target observations would typically require the consideration of orbital parameters, including position and velocity for both targets, the observing sensor's slewing speed and stabilization mechanisms, and the simultaneous motion of the Earth. In this research, a simplification is adopted by assuming that the positional change of the target does not vary significantly during the sensor's slewing duration. Consequently, only the target positions at the beginning of the transition (i.e., the end of the initial target observation) are taken into account, as well the settle time of the sensor. This simplification effectively reduces the computational complexity while still maintaining a high level of accuracy.

2. *Observation durations.* In this context, a duration refers to the time required to collect data for the observation of a target. The duration is the product of integration time, which is the amount of time per frame the sensor spends collecting photons, and frame number. The integration time and frame number depend on various factors, including the desired measurement precision, the characteristics of the target, the sensitivity of the sensor, and the prevailing noise levels. Longer integration times typically yield superior data quality for dimmer targets, while brighter targets need shorter integration times to avoid image saturation. The frame number is influenced by the orbital motion of the target, where more frames are required for targets with a higher relative velocity. In effect, the ideal duration of a target observation relies on the sensor, target, and observation time, and may vary throughout a scheduling window. However, to decrease solution time and computational complexity, the models in this research assume that durations are constant in time and sensor designation, but may vary across targets.

3. *Observational constraints.* Observations in SDA are subject to various constraints, including those considering atmospheric conditions, sensor coverage, the positions of various celestial bodies, and sensor maneuvering. Astroplan [2], an open-source observation planning package for astronomers, offers methods for dealing with these constraints. This Python package provides convenient abstractions of many observational constraints that allow determining target observation feasibility from a specific sensor at a given time. In this research, an Astroplan fork is utilized to simplify the complex constraints faced in SDA observations. The considered constraints include limiting target altitude and azimuth, restricting an observation to be outside of Earth's shadow, Moon and Sun separation windows, observing the target at night, using a specific sensor, and observing the target within a defined time window.

4. *Scheduler objective function.* In the context of scheduling for SDA, there are various objectives to consider. These include goals such as minimizing the mean time since updating target Two-Line Elements (TLEs), minimizing the average target position error covariance, or maximizing the targets found in a search. Achieving high-quality schedules often involves multi-objective optimization, however this research uses a single objective function of maximizing the total reward, subject to all relevant constraints. This makes the scheduler versatile, allowing different objectives to be represented through the combination of individual blocks, their constraints, and their reward values.

5. *Resource Allocation.* Efficient utilization of resources is another important factor in scheduling for SDA. Resources may include costs associated with sensor operations, data storage costs, or physical hardware limitations. In response, a scheduler may include objectives that allocate tasks across sensors proportionately to match their load capabilities. In this paper, resource constraints are not considered.

Related Work

Scheduling for SDA is a widely studied problem with many different approaches. Common approaches fall under the categories of greedy algorithms, reinforcement learning, and evolutionary algorithms.

Under the category of greedy algorithms, one approach is the block greedy algorithm, which maximizes the number of targets seen while minimizing slewing and image storage costs and target revisit time, subject to observational constraints [3]. This algorithm works by discretizing the time axis, then greedily scheduling "blocks" of time periods within the schedule. Contrary to traditional greedy approaches, which schedule in temporal order, all time periods are considered when selecting which targets to schedule within a block. This approach considers constraints for signal-to-noise ratio (SNR) threshold, sensor memory, and target score threshold.

This approach also evaluates transition times with sensor-specific slew rates applied over a slewing distance, assuming that target positions remain largely unchanged during transition. For the same experimental trials, the results from the block greedy approach outperformed a traditional greedy approach, a genetic algorithm, random search, and a weapon-target-assignment algorithm. This was evaluated over tests including 3 sensors and 970 targets for a 24-hour time horizon with a 60-second time resolution.

Another greedy approach is the auction-bidding algorithm, which is used to minimize the Rényi α-divergence of resident space object (RSO) state covariances [4]. The objective function also allows for certain RSOs to be prioritized with a "Tactical Importance Function" that scales its original reward. The auction algorithm works by performing "bidding" rounds that progress toward each sensor being paired with the object that is most valuable. The termination of the bidding results in the approximate optimal assignment of objects across all sensors [5]. It considers observational constraints for sensor elevation, sensor range, apparent target magnitude, sky darkness, sun-station-satellite phase angle, Sun angle, Moon angle, and radius limit, as well as the resource constraint of a laser link budget for laser sensors. Transition times are included in determining the visibility of an RSO at time periods in the schedule based on the slew rate of the sensor. As of 2019, this scheduler was in use at the Space Environment Research Centre in Mt. Stromlo Facility in Canberra, Australia, and is capable of scheduling observations for 20,000 objects across 6 sensors with a 5-second time resolution.

Within machine learning, multiple approaches have shown to be effective. One such approach is ant colony optimization (ACO), which is a versatile path finding algorithm capable of producing good solutions to complex variations of the travelling salesman problem (the SDA scheduling problem is a variation of the multi-travelling salesman problem) [6]. This algorithm mimics the biological process of an ant colony finding the shortest path between two locations. In the case of scheduling for SDA, the "ants" are agents that explore the space of potential sensor viewing directions to maximize the total reward of target observations.

Similarly to ACO, distributed Q-learning (DQL) uses multiple agents to seek the optimal solution. These two approaches are compared in [7] for the problem of scheduling a single ground-based sensor. In this comparison, the objective is to maximize the set of weighted viewing directions, where the weighting is determined by the expected reward of the target to be observed. In comparison to other approaches (including the one in this paper), the inclusion of a set of discrete pointing directions as a grid field increases the dimensionality of the search space. Additionally, because the objective function is in terms of sensor pointing directions instead of sensor-target pairs, observation durations are not explicitly defined. Finally, functions determining the probability of detecting a target and the probability that a target is visible within a grid field replace the role of observational constraints. This approach highlights the stark differences between potential solution methods toward this problem.

The comparison gives a simulation containing 1231 geosynchronous targets for a schedule duration of 10 hours and 47 minutes, which shows ACO observing 508 targets compared to DQL observing 425 targets. Additionally, a simple greedy algorithm was tested (which always chose the highest reward grid field at each time step) that observed 496 targets. however, that a greedy approach outperformed reinforcement learning is not a general trend. It was shown in [8] that deep reinforcement learning agents performed significantly better than greedy approaches across multiple criteria.

Within the category of evolutionary algorithms, one example is that proposed in [9]. This approach optimizes the two objectives of maximizing target priority and balancing the load of all sensors, utilizing a multi-objective evolutionary decomposition algorithm [10]. This approach presents both sensor constraints and target constraints. The sensor constraints include restrictions on the observation range, elevation, and azimuth, as well as the resource constraint of sensor data storage. The target constraints include time windows and adjacent observation time requirements, which enforce additional observations of a target. The final observation constraint places a minimum observation duration for a given target. This approach was simulated using 3 ground-based sensors with default constraints and 1000 RSO targets.

Another evolutionary approach is presented in [11], where a genetic algorithm is used to maximize the expected Shannon Information Content [12] of observations, which results in lower target position covariance. This approach considers the visibility constraints of maximum phase angle, minimum Moon distance, and Earth shadow distance. Simulations showed the viability of the scheduler for 3 sensors and 436 targets over a 1700-minute time horizon.

3. INTEGER PROGRAMMING MODEL

Table 1: **Notation for the IP scheduling model**

Sets	
D	Set of sensors
J	Set of blocks
T	Set of consecutive time periods spanning the time horizon
W_{dj}	Set of restricted windows for sensor $d \in D$ and block $j \in J$
Parameters	
b_w	Beginning period of restricted window $w \in W_{dj}$, $d \in D$, $j \in J$
e_w	Ending period of restricted window $w \in W_{dj}$, $d \in D$, $j \in J$
p_j	Observation duration of block $j \in J$; $p_j \in \mathbb{R}_{>0}$
r_j	Reward of block $j \in J$; $r_j \in \mathbb{R}_{>0}$
$\tau_{djj't}$	Transition time in periods between block $j \in J$ and block $j' \in J$ on sensor $d \in D$, beginning at time period $t \in T$
Decision Variables	
y_{dj}	1 if block j is assigned to sensor d
z_{djt}	1 if sensor d begins processing block j at time t

The notation for the presented programming models is given in Table 1. The IP model in this section draws inspiration from the formulations introduced by Gedik et al. in [13], however, our model differs in a few significant ways. Firstly,

Gedik et al. assume that the restricted windows are universal across machines, whereas our problem requires that they are specific to each sensor. Consequently, their proposed cuts cannot be applied to the decomposition of the IP model presented in the next section. Secondly, we do not include a budget or a sensor-specific operation rate in our problem definition, (also affecting the cuts). Finally, the transition times in our problem are determined not only by the starting and ending blocks, but also by the assigned sensor and the beginning time period, resulting in a significantly higher number of constraints. The IP model for this problem, denoted as IPM, is as follows:

$$\max \sum_{d\in D}\sum_{j\in J} r_j\, y_{dj}$$

Subject to: (IPM)

$$\sum_{d\in D} y_{dj} \le 1, \quad j \in J \tag{1}$$

$$\sum_{t\in T} z_{djt} = y_{dj}, \quad d \in D,\, j \in J \tag{2}$$

$$\sum_{t=\max(0,b_w-p_j)}^{e_w} z_{djt} = 0, \quad w \in W_{dj},\, d \in D,\, j \in J \tag{3}$$

$$\sum_{t'=t}^{min\left\{|T|-1,\, t+p_j+\tau_{djj'}\min\{|T|-1,\, t+p_j\}\right\}} z_{dj't'} \le 1 - z_{djt},$$
$$j \in J,\, j' \in J,\, j \ne j',\, d \in D,\, t \in T \tag{4}$$

$$(t+p_j)z_{djt} \le |T|, \quad d \in D,\, j \in J,\, t \in T \tag{5}$$

$$y_{dj} \ge 0, \quad d \in D,\, j \in J \tag{6}$$

$$z_{djt} \in \{0,1\}, \quad d \in D,\, j \in J,\, t \in T \tag{7}$$

The objective of IPM is to maximize the total reward of assigned blocks. Constraints (1) ensure that block j is processed by at most one sensor. Constraints (2) ensure that if block j is assigned to sensor d, j has a start time on d. Constraints (3) ensure that no two blocks j and j' are processed simultaneously on sensor d. Constraints (4) ensure that no block j is scheduled in a restricted period. Constraints (5) ensure that the completion time of a block j does not occur beyond the time horizon.

4. IP MODEL DECOMPOSITION

To increase the solvable problem size, an algorithm closely following logic-based Benders decomposition (LBBD) is proposed for IPM, with the introduced master problem, subproblems, and cuts drawing from those presented in [14]. Benders decomposition operates in partitioning the decision variables of a problem into two sets. Additionally, the problem is broken into a master problem and a subproblem. The algorithm works by exploring the space of potential solutions, where after each iteration a subsection of potential solutions is eliminated by adding additional constraints to the master problem until the optimal solution is found. Essentially, this works as follows:

First, the master problem is solved to optimality (using a numerical solver) to find a trial solution of one set of variables. From this trial solution, subproblems are created with these variables fixed. The subproblems are then solved to find the values of the other variables. If any of subproblems are infeasible, or the combined objective of the subproblems is worse than the objective of the master problem, this solution is a "nogood" and cuts are generated to exclude it and others from being a solution to the master problem. The master problem is re-solved, and because the previously optimal solution is no longer feasible, it is forced to find another. If all subproblems are feasible and the combined objective of the subproblems is at least as preferable as the objective of the master problem, the solution is globally optimal. For more detail, the reader is directed to the original paper discussing LBBD [15].

Similarly to LBBD, the decomposition algorithm presented in this section partitions the decision variables into two sets, y and z. Mainly, the master problem eliminates the constraint that prevents overlapping blocks, relaxes the z variables to the domain of non-negative reals, and restricts the y variables to the binary domain. In effect, the solution to the master problem provides a speculative assignment of blocks to sensors, without guarantee of a feasible block sequence. The subproblems are then created from the solution to the master problem and solved with the necessary constraints to ensure a feasible assignment and sequence of blocks.

This algorithm differs slightly from LBBD in that the subproblems do not fix the assignment variables obtained from a master problem solution. Instead, the y variables are reduced to only those assigned to the subproblem of each sensor in the master problem solution, and their domain remains binary. The subproblems also restrict the z variables to the binary domain and enforce the constraints that prevent block overlap. Not fixing the y variables is necessary, as a solution to a subproblem may result in one or more blocks no longer fitting within the time horizon after sequencing constraints are enforced. In this case, fixing the y variables would result in an infeasible solution, preventing early termination from the algorithm as well as generating the cuts, which are introduced later in this section.

The decomposition algorithm uses the following master problem, denoted as DMP:

$$\max \sum_{d\in D}\sum_{j\in J} r_j\, y_{dj}$$

Subject to: (DMP)

CUTS (8)

$$\sum_{d\in D} y_{dj} \le 1, \quad j \in J \tag{9}$$

$$\sum_{t \in T} z_{djt} = y_{dj}, \quad d \in D,\, j \in J \tag{10}$$

$$\sum_{t=\max(0,b_w-p_j)}^{e_w} z_{djt} = 0, \quad w \in W_{dj},\, d \in D,\, j \in J \tag{11}$$

$$y_{dj}\left(p_j + \sum_{w \in W_{dj}} (e_w - b_w)\right) \leq |T|, \quad d \in D,\, j \in J \tag{12}$$

$$\sum_{j \in J} y_{dj} p_j \leq |T|, \quad d \in D \tag{13}$$

$$y_{dj} \in \{0,1\}, \quad d \in D,\, j \in J \tag{14}$$

$$z_{djt} \geq 0, \quad d \in D,\, j \in J,\, t \in T \tag{15}$$

Constraints (9), (10), and (11) perform the same operations as (1), (2), and (3) respectively. Constraints (12) prevent assigning block b to sensor d if the sum of the total length of the block's restricted windows and the block's duration is greater than the number of time periods. Constraints (13) ensure that the total processing time of sensor d does not exceed the time horizon. Finally, constraints (14) restrict the y variables to the binary domain, while constraints (15) relax the z variables to the non-negative real domain.

A solution to the master problem on iteration h is denoted as (f^h, y^h, z^h), where f is the objective value. From this solution, one subproblem is created per sensor, where the blocks considered in each subproblem are those assigned to the subproblem's sensor in the master solution. In other words, a subproblem for each sensor $d \in D$ is created with the set of block assignments J_d^h, where J_d^h is the set of blocks $\left\{j \in J :\, y_{dj}^h = 1\right\}$. A subproblem for sensor d, denoted as DSP, is represented by the following:

$$\max \sum_{j \in J_d^h} r_j y_j$$

Subject to: (DSP)

$$y_{dj} \leq 1, \quad j \in J_d^h \tag{16}$$

$$\sum_{t \in T} z_{djt} = y_{dj}, \quad j \in J_d^h \tag{17}$$

$$\sum_{t'=t}^{min\left\{|T|-1,\, t+p_j+\tau_{djj'}\min\{|T|-1,\, t+p_j\}\right\}} z_{dj't'} \leq 1 - z_{djt},$$
$$j \in J_d^h,\, j' \in J_d^h,\, j \neq j',\, t \in T \tag{18}$$

$$\sum_{t=\max(0,b_w-p_j)}^{e_w} z_{djt} = 0, \quad w \in W_{dj},\, j \in J_d^h \tag{19}$$

$$(t+p_j) z_{djt} \leq |T|, \quad j \in J_d^h,\, t \in T \tag{20}$$

$$y_{dj} \geq 0, \quad j \in J_d^h \tag{21}$$

$$z_{djt} \in \{0,\, 1\}, \quad j \in J_d^h,\, t \in T \tag{22}$$

The constraints of the subproblem are the same as in IPM, except that there is now only one sensor. The combination of subproblem solutions, denoted as (f^{h*}, y^{h*}, z^{h*}), gives a feasible solution to IPM. However, only when the combined subproblem rewards are equal to the master problem reward is the solution optimal, i.e., when $f^h = f^{h*}$. Otherwise, a cut is added to the master problem on iteration h with the following definition:

$$f \leq f^{h*} + \sum_{d \in D} \sum_{j \in \left(J - J_d^h\right)} r_j y_{dj}$$

As demonstrated in [16], in order for a cut to be valid, it must satisfy two conditions: 1) it must remove the current solution from the master problem, and 2) it must not remove any globally optimal solutions. The proposed cut satisfies these conditions, with proofs given in the appendix.

After a cut is added to the master problem, a new iteration starts by re-solving the master problem. The algorithm continues in this way until the optimal solution is found or the time limit is reached. Algorithm 1 gives pseudocode for the decomposition scheduling algorithm (DSA).

5. PROGRAM OVERVIEW

Starting data

The input data to the scheduler is mission-agnostic, i.e., the scheduler does not distinguish between observation blocks created with different goals in mind. The original goals of some mission (such as the periodic revisit of a target) are instead represented through the generated blocks' constraints and rewards. In this way, the scheduler may be populated with observing blocks that map to a heterogeneous set of missions, although this information is irrelevant to the scheduler (likewise the concept of missions may not apply at all).

Preprocessing

The first task of the scheduler is to convert the provided observation blocks into data that is consumable by the IP model. This is done in the preprocessing step, which essentially translates observation block constraints and sensor constraints to restricted windows, and evaluates transition times between targets for all sensors.

Computing the restricted windows during which a block cannot be scheduled on a sensor requires evaluating the astrodynamic constraints placed on the block for all time periods within the time horizon. As mentioned beforehand, constraint evaluations are performed using a modified version

Algorithm 1: Pseudocode of DSA. $preprocess$ is a function that initializes the restricted windows and parameters used in the model. $time\_limit$ denotes the maximum wall time given to the scheduler.

$W_{d,j}, b_w, e_w, p_j, \tau_{djj't} = preprocess(S, J, T)$
$time\_remaining = time\_limit$
$optimal\_found = False$
$best\_reward = 0$
$best\_schedule = None$
while not $optimal\_found$ or $time\_remaining > 0$ **do**
  $(f^h, y^h, z^h) = master.solve()$
  $(f^{h*}, y^{h*}, z^{h*}) = None$
  for d in D **do**
    $(f_d^{h*}, y_d^{h*}, z_d^{h*}) = subproblem\_d.solve(J_d^h)$
    $(f^{h*}, y^{h*}, z^{h*}) \mathrel{+}= (f_d^{h*}, y_d^{h*}, z_d^{h*})$
  end for
  if $f^{h*} \geq best\_reward$ **then**
    $best\_reward = f^{h*}$
    $best\_schedule = (f^{h*}, y^{h*}, z^{h*})$
  end if
  if $f^h == f^{h*}$ **then**
    $optimal\_found = True$
  end if
  update $time\_remaining$
end while
return $best\_schedule$

of the Astroplan package [2]. These calculations are done in parallel using a separate process for each sensor. Additionally, the topocentric positions of the Sun, Moon, and all targets across the time horizon are memoized for each sensor to avoid repeating expensive propagations. The result is a three-dimensional boolean matrix that stores the time periods that each block can be scheduled in on each sensor, from which the restricted windows W_{dj} are generated. Then, the transition times are calculated for the transitions which are valid.

As stated in section 2, transition time calculations are simplified by assuming that a target's position does not change significantly during a sensor's slewing duration. Additionally, these experiments assume that the transition $\tau_{djj't}$ is equal to the transition $\tau_{dj'jt}$. This is a safe assumption, as transition times are calculated by finding the angular separation between j and j' and dividing by the slew rate of sensor d, then adding the sensor settle time. Similarly to evaluating constraints, transition time calculations are performed in parallel with a shared target positions cache. At this point, the sets and parameters used in the IP model are populated.

Schedule generation

The schedule generation step amounts to solving the IP model either directly with an IP solver, or with the IP decomposition approach. The algorithm that solves the IP model directly is referred to as the IP scheduler (IPS) while the latter is denoted as the decomposition scheduling algorithm (DSA). The solution to the IP model is a valid schedule of blocks (targets) assigned to sensors and time periods.

6. COMPUTATIONAL EXPERIMENTS

Setting

Both IPS and DSA were implemented with Python, using the Coin-OR Branch and Cut (CBC) open-source solver [17] in combination with Pyomo [18], [19].

The experiments in this section are conducted to resemble general scheduling cases for SDA as closely as possible. Because SDA operations mostly operate on a minute-based timescale, a time resolution of 60 seconds is used in all experiments. The experiments use various combinations of number of sensors, number of blocks, and time horizon to evaluate the decomposition scheduler's performance in each of these situations compared to the IP scheduler. The schedules generated in each experiment are given a start time of June 17, 2023 at 10:00:00 AM UTC.

The experimental instances use the notation of $(|D|, |J|, |T|)$, corresponding to the number of sensors, number of blocks, and time horizon, respectively. Ten runs are performed for each instance, each with the same sensors but newly generated blocks. The sensors are chosen from a set of 6 research sensors, with locations and observational constraints given in the appendix. The blocks are assigned at pseudorandom a reward from 1 to the number of blocks used that trial, a duration from 6 to 30 seconds, and a target from a pool of 287 satellites. The IDs used in accessing the TLEs of these satellites are given in the appendix. No astrodynamic constraints are placed on the blocks aside from those enforced by the sensors (given in the appendix). The experiments are carried out in an Ubuntu 22.04.2 LTS WSL2 instance running on a Dell Precision 5570 laptop with a 14-core 12th Gen Intel Core i7-12800H processor at 2.40 GHz, with 32 GB RAM. Each instance run is given a time limit of its time horizon.

The metrics recorded for each IPS instance are run time, schedule reward, and the completion rate. These are compared with the run time, number of iterations, and optimality gap reported for each instance of DSA. All metrics are averaged across the 10 runs performed for each problem instance. In the case of run time, the standard deviation is also given. The optimality gap presented for DSA instances is defined as the percent difference of the best feasible schedule and the lowest master problem reward with respect to the lowest master problem reward across all iterations of the decomposition algorithm:

$$\text{gap} = \frac{\text{lowest master reward} - \text{best schedule reward}}{\text{lowest master reward}}$$

In other words, the gap gives the difference between the upper and lower bound as a percentage of the upper bound. If the upper bound and lower bound are equal, the gap is 0 and the schedule is globally optimal.

Results

Tables 2 and 3 give results for experiments with $|D| = 3$ and $|D| = 6$, respectively. An instance with a run time greater than its time horizon implies that the scheduler reached its time limit and was stopped.

In these experiments, IPS successfully found optimal solutions up to instances with 40 blocks and a 240 second time resolution for both 3 and 6 sensors. DSA failed to converge to zero gap for most instances, excluding edge cases with only one time period (i.e. instances with a time horizon of

Table 2: Comparison of IPS and DSA for instances with $|D| = 3$

Instance	DSA					IPS			
	Run time (s)	σ	**Reward**	**Iterations**	**Gap**	**Run time (s)**	σ	**Reward**	**Completion**
(3, 20, 60)	1.35	0.08	54.20	1.00	0.00	1.24	0.10	54.20	1.00
(3, 20, 120)	110.80	19.80	53.00	265.10	0.19	2.36	0.30	53.00	1.00
(3, 20, 180)	155.12	52.31	54.30	145.80	0.23	6.90	5.25	54.30	1.00
(3, 20, 240)	240.32	0.31	87.80	225.30	0.26	7.28	2.88	87.80	1.00
(3, 20, 300)	269.16	64.75	93.10	279.10	0.19	14.56	7.12	93.10	1.00
(3, 20, 600)	306.39	294.25	137.80	462.30	0.03	304.66	296.28	65.30	0.50
(3, 20, 900)	37.37	19.60	139.80	10.10	0.00	12.74	1.71	139.80	1.00
(3, 40, 60)	2.04	0.13	115.60	1.00	0.00	2.55	0.30	115.60	1.00
(3, 40, 120)	120.51	0.18	111.50	197.50	0.43	18.06	5.40	111.50	1.00
(3, 40, 180)	180.52	0.21	134.80	78.50	0.46	39.60	19.62	139.50	1.00
(3, 40, 240)	240.57	0.30	209.30	110.10	0.41	39.04	21.34	209.30	1.00
(3, 40, 300)	300.59	0.24	210.70	66.90	0.48	236.10	89.82	83.80	0.40
(3, 40, 600)	600.80	0.57	379.30	311.40	0.26	602.20	1.24	0.00	0.00
(3, 40, 900)	900.82	0.24	417.30	53.40	0.17	903.13	1.50	0.00	0.00
(3, 60, 60)	3.03	0.30	170.70	1.00	0.00	4.62	1.38	170.70	1.00
(3, 60, 120)	120.58	0.24	168.30	171.80	0.43	38.52	20.23	168.30	1.00
(3, 60, 180)	180.62	0.16	200.90	72.50	0.54	148.17	49.20	99.40	0.40
(3, 60, 240)	240.53	0.31	336.80	100.30	0.45	149.38	51.15	302.80	0.90
(3, 60, 300)	300.69	0.28	325.50	47.30	0.53	301.53	0.23	0.00	0.00
(3, 60, 600)	600.57	0.29	579.00	134.50	0.44	603.61	2.04	0.00	0.00
(3, 60, 900)	900.70	0.24	701.00	14.30	0.35	906.17	3.94	0.00	0.00
(3, 80, 60)	3.70	0.35	234.20	1.00	0.00	14.22	2.21	234.20	1.00
(3, 80, 120)	120.54	0.27	231.40	155.20	0.46	109.80	17.66	69.50	0.30
(3, 80, 180)	180.66	0.31	265.40	71.60	0.57	168.10	27.20	70.50	0.20
(3, 80, 240)	240.66	0.31	449.80	91.40	0.46	242.03	0.67	0.00	0.00
(3, 80, 300)	300.74	0.32	449.00	42.90	0.55	302.78	0.99	0.00	0.00
(3, 80, 600)	600.71	0.31	863.60	93.20	0.51	605.55	2.25	0.00	0.00
(3, 80, 900)	900.93	0.29	1068.80	6.60	0.47	909.12	4.31	0.00	0.00
(3, 100, 60)	4.95	0.46	290.60	1.00	0.00	28.70	12.37	262.40	0.90
(3, 100, 120)	120.67	0.22	290.10	146.10	0.47	121.73	0.41	0.00	0.00
(3, 100, 180)	180.52	0.17	338.90	69.70	0.57	182.57	0.88	0.00	0.00
(3, 100, 240)	240.74	0.29	566.50	94.80	0.47	243.15	1.07	0.00	0.00
(3, 100, 300)	300.77	0.23	566.30	43.20	0.57	303.43	1.42	0.00	0.00
(3, 100, 600)	600.58	0.24	1095.70	71.30	0.53	607.01	3.81	0.00	0.00
(3, 100, 900)	900.82	0.34	1381.80	5.30	0.52	905.50	2.68	0.00	0.00
(3, 150, 60)	10.88	1.84	442.90	1.00	0.00				
(3, 150, 120)	120.68	0.25	444.30	124.90	0.48				
(3, 150, 180)	180.71	0.23	549.20	64.20	0.56				
(3, 150, 240)	240.70	0.23	872.50	89.70	0.48				
(3, 150, 300)	300.69	0.32	865.80	41.70	0.57				
(3, 150, 600)	600.86	0.44	1685.60	58.30	0.55				
(3, 150, 900)	900.96	0.31	2029.70	2.20	0.60				
(3, 200, 60)	12.42	3.16	589.60	1.00	0.00				
(3, 200, 120)	120.80	0.28	590.80	108.30	0.49				
(3, 200, 180)	180.80	0.26	698.30	58.10	0.59				
(3, 200, 240)	240.88	0.19	1170.90	83.50	0.49				
(3, 200, 300)	300.84	0.28	1164.80	41.20	0.58				
(3, 200, 600)	601.14	0.29	2290.10	44.10	0.57				
(3, 200, 900)	901.54	0.77	2081.40	1.80	0.72				

Table 3: Comparison of IPS and DSA for instances with $|D| = 6$

Instance	DSA					IPS			
	Run time (s)	σ	**Reward**	**Iterations**	**Gap**	**Run time (s)**	σ	**Reward**	**Completion**
(6, 20, 60)	1.59	0.11	70.90	1.00	0.00	2.04	0.61	70.90	1.00
(6, 20, 120)	120.32	0.20	68.60	167.80	0.30	3.32	0.30	68.60	1.00
(6, 20, 180)	180.39	0.20	70.40	101.70	0.34	9.75	5.93	70.60	1.00
(6, 20, 240)	240.54	0.31	108.30	371.80	0.22	10.93	4.15	108.30	1.00
(6, 20, 300)	300.63	0.42	118.90	409.60	0.16	21.54	8.44	118.90	1.00
(6, 20, 600)	77.83	174.41	159.50	113.50	0.00	92.68	180.45	138.00	0.90
(6, 20, 900)	387.08	419.36	159.70	421.70	0.04	16.06	2.17	159.70	1.00
(6, 40, 60)	2.78	0.13	153.10	1.00	0.00	3.58	0.32	153.10	1.00
(6, 40, 120)	120.63	0.29	148.30	127.50	0.44	25.60	7.59	148.30	1.00
(6, 40, 180)	180.71	0.26	172.40	43.10	0.51	83.24	60.11	138.10	0.80
(6, 40, 240)	240.54	0.32	271.20	139.50	0.39	65.84	30.87	271.20	1.00
(6, 40, 300)	300.66	0.29	273.70	107.40	0.46	292.83	26.96	27.60	0.10
(6, 40, 600)	600.64	0.25	457.70	337.70	0.20	604.23	2.45	0.00	0.20
(6, 40, 900)	900.81	0.30	544.00	94.00	0.17	905.45	3.04	0.00	0.00
(6, 60, 60)	4.10	0.19	226.40	1.00	0.00	6.71	1.29	226.40	1.00
(6, 60, 120)	120.82	0.22	222.60	116.20	0.45	68.48	36.08	176.40	0.80
(6, 60, 180)	180.84	0.46	264.40	41.00	0.56	173.42	17.38	60.90	0.20
(6, 60, 240)	240.83	0.21	441.70	109.60	0.44	210.63	41.66	219.80	0.50
(6, 60, 300)	300.70	0.30	428.20	50.10	0.52	302.65	0.25	0.00	0.00
(6, 60, 600)	601.17	0.73	751.60	134.10	0.39	606.88	3.34	0.00	0.00
(6, 60, 900)	900.73	0.12	945.80	22.00	0.35	905.39	3.51	0.00	0.00
(6, 80, 60)	6.31	0.64	308.40	1.00	0.00	18.62	3.94	308.40	1.00
(6, 80, 120)	120.67	0.27	309.90	102.20	0.47	122.29	0.35	0.00	0.00
(6, 80, 180)	180.75	0.32	353.60	43.50	0.57	182.19	3.05	45.40	0.10
(6, 80, 240)	241.03	0.53	594.60	92.40	0.46	244.43	1.88	0.00	0.00
(6, 80, 300)	300.91	0.31	589.00	28.90	0.54	305.55	2.97	0.00	0.00
(6, 80, 600)	600.94	0.26	1083.80	101.90	0.47	613.13	9.14	0.00	0.00
(6, 80, 900)	901.49	0.17	1237.20	6.20	0.52	913.13	9.14	0.00	0.00
(6, 100, 60)	7.54	0.57	386.60	1.00	0.00	29.90	10.38	386.60	1.00
(6, 100, 120)	120.95	0.25	386.20	98.80	0.47	122.85	0.59	0.00	0.00
(6, 100, 180)	180.89	0.28	401.30	41.80	0.62	184.26	1.40	0.00	0.00
(6, 100, 240)	241.12	0.31	747.40	79.60	0.46	246.18	2.55	0.00	0.00
(6, 100, 300)	301.11	0.22	754.30	27.90	0.56	307.49	3.82	0.00	0.00
(6, 100, 600)	602.27	1.00	1440.80	64.70	0.50	605.76	4.19	0.00	0.00
(6, 100, 900)	902.31	0.48	1593.30	4.80	0.60	904.30	2.62	0.00	0.00
(6, 150, 60)	9.94	0.64	588.70	1.00	0.00				
(6, 150, 120)	120.91	0.28	591.20	85.60	0.49				
(6, 150, 180)	180.77	0.25	683.00	40.20	0.59				
(6, 150, 240)	240.89	0.28	1153.00	85.70	0.48				
(6, 150, 300)	301.08	0.38	1149.20	26.70	0.57				
(6, 150, 600)	601.60	0.31	2222.20	54.90	0.54				
(6, 150, 900)	902.29	0.61	2830.60	2.40	0.63				
(6, 200, 60)	12.96	0.77	784.30	1.00	0.00				
(6, 200, 120)	120.75	0.35	787.10	75.70	0.49				
(6, 200, 180)	181.10	0.45	878.80	38.00	0.61				
(6, 200, 240)	241.15	0.27	1552.00	80.50	0.49				
(6, 200, 300)	301.59	0.32	1545.40	29.50	0.58				
(6, 200, 600)	601.45	0.22	3027.00	36.70	0.56				
(6, 200, 900)	902.21	0.38	2382.10	1.60	0.79				

60 seconds). Regardless, with one exception, in all instances with an IPS completion rate of 1, DSA reported the same average reward as IPS (the exception being instance (3, 40, 180)). In these instances, DSA maximally raised the lower bound on the objective function, but failed to lower the upper bound to the same value. This suggests that the proposed cuts do not eliminate enough of the solution space in each iteration of DSA to combat the combinatorial growth rate of increasing problem sizes. This is further supported by the DSA average gap increasing with problem size. However, in instances where IPS failed to report solutions to all or many of the performed runs, DSA returned feasible schedules. Beyond 200 blocks, the CBC solver reported inconsistent errors, preventing further evaluation.

The number of observation blocks that IPS and DSA can handle are significantly lower than the number of targets other schedulers consider, such as those listed in section 2. Likewise the time horizons that DSA and IPS can schedule are only 15 minutes and 4 minutes, respectively, fractions of an 8 to 10-hour full night of observations. Moreover, DSA barely generated a single feasible schedule for 15-minute time horizons with 200 blocks, with average iteration numbers of 1.80 and 1.60 for $|D| = 3$ and $|D| = 6$, respectively. The results of these experiments show most significantly that, on their own, IPS and DSA are not suitable for scheduling a full night of observations. The total reward of generated schedules using IPS and DSA are not compared to existing approaches for this reason.

7. Conclusion

This work presents a unique approach to scheduling for SDA with an integer programming solution. Experiments using real sensors, satellites, and observational constraints show that the presented IP model can routinely find optimal schedules for instances of up to 6 sensors and 40 observation blocks over a 240-second time horizon. The proposed decomposition algorithm it is capable of finding feasible schedules for instances of up to 6 sensors and 200 blocks over 900 second time horizon, although failing to converge to optimality in the general case.

Compared to the existing approaches presented in section 2, DSA and IPM fail to compete with the number of scheduled observations. However, this comes as no surprise, as NP-hard problems are intractable even at small sizes. This research offers a new tool to be used in future scheduling algorithms, which may combine the use of heuristics to achieve high quality schedules. Future research includes investigating these schedulers, such as a locally optimal/greedily selective scheduler, similar to the block-greedy approach presented in [3].

Additionally, this research serves as a starting point for future investigations of solving the SDA scheduling problem with integer programming. In this direction, further research may include a reformulation of the IP model and decomposition with improved cuts that lead to faster convergence, or evaluating the benefit of a commercial solver in IPS and DSA.

Appendices

A. Proof of Cut Validity

The following proofs establish the validity of the proposed cut for the decomposition algorithm presented in section 4.

Proposition 1. *The proposed cut removes the current solution from the master problem.*

Proof: Assume there exists a solution from iteration k with block assignments from the master problem represented by J^k that satisfies the cut. Given that the block assignments from the master problem of iteration h are given by J^h, if the solution from iteration h is identical to the solution from iteration k, then $J^k = J^h$. The proof proceeds to show a contradiction. Starting from the definition of the cut:

$$f^k \leq f^{h*} + \sum_{d \in D} \sum_{j \in \left(J - J_d^h\right)} r_j y_{dj}^k$$

$$f^k \leq f^{h*} + \sum_{d \in D} \sum_{j \in \left(J - J_d^k\right)} r_j y_{dj}^k$$

Using the definition of J_d^k, $J - J_d^k$ is the set of blocks $\left\{j \in J : y_{dj}^k = 0\right\}$. Thus,

$$\sum_{d \in D} \sum_{j \in \left(J - J_d^k\right)} r_j y_{dj}^k = 0$$

$$f^k \leq f^{h*} \tag{23}$$

However, if $J^k = J^h$, then $f^k = f^h$, and by (23), $f^h \leq f^{h*}$. Because a cut was generated for iteration h, we know that $f^h > f^{h*}$, which is a contradiction and proves the first proposition. □

Proposition 2. *The proposed cut does not eliminate any globally optimal solutions.*

Proof: Suppose a globally optimal solution is found in some iteration $k, k > h$ that violates the proposed cut from iteration h. Because the solution in iteration k is different than the solution in iteration h, there exists a set of sensors $\Delta \subseteq D$, such that $\forall\ \delta \in \Delta$, $J_\delta^k \neq J_\delta^h$. In other words, in the optimal solution there will be a set of sensors whose block assignments from the master problem solution for iteration k are not the same as those from iteration h. We assume that the cut generated from iteration h is violated under these circumstances, then proceed to show a contradiction. Starting from the violation of the cut:

$$f^k > f^{h*} + \sum_{d \in D} \sum_{j \in \left(J - J_d^h\right)} r_j y_{dj}^k$$

$$f^k > f^{h*} + \sum_{d \in D, d \notin \Delta} \sum_{j \in \left(J - J_d^h\right)} r_j y_{dj}^k + \sum_{\delta \in \Delta} \sum_{j \in \left(J - J_\delta^h\right)} r_j y_{\delta j}^k$$

We can partition $J - J_\delta^h$ into $J - J_\delta^h = \left(\left(J - J_\delta^h\right) - J_\delta^k\right) \cup$

$\left(J_\delta^k \cap \left(J - J_\delta^h\right)\right)$.

$$f^k > f^{h*} + \sum_{\delta\in\Delta} \sum_{j\in\left(\left(J-J_\delta^h\right)-J_\delta^k\right)} r_j y_{\delta j}^k + \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^k$$

$$f^k > f^{h*} + \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^k$$

Because the solution from iteration k is globally optimal, we know that $f^k = f^{k*}$ and $y^k = y^{k*}$.

$$\begin{aligned} f^k = f^{k*} &= \sum_{d\in D, d\notin\Delta} \sum_{j\in J_d^k} r_j y_{dj}^{k*} + \sum_{\delta\in\Delta} \sum_{j\in J_\delta^k} r_j y_{\delta j}^{k*} \\ &> f^{h*} + \sum_{\delta\in D} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^{k*} \end{aligned}$$

$$\begin{aligned} \sum_{d\in D, d\notin\Delta} \sum_{j\in J_d^k} r_j y_{dj}^{k*} + \sum_{\delta\in\Delta} \sum_{j\in J_\delta^k} r_j y_{\delta j}^{k*} &> \sum_{d\in D, d\notin\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*} \\ + \sum_{d\in\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*} + \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^{k*} & \end{aligned}$$

$$\sum_{\delta\in\Delta} \sum_{j\in J_\delta^k} r_j y_{\delta j}^{k*} > \sum_{d\in\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*} + \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^{k*}$$

$$\sum_{\delta\in\Delta} \sum_{j\in J_\delta^k} r_j y_{\delta j}^{k*} - \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap\left(J-J_\delta^h\right)\right)} r_j y_{\delta j}^{k*} > \sum_{d\in\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*}$$

$$\sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap J_\delta^h\right)} r_j y_{\delta j}^{k*} > \sum_{d\in\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*} \tag{24}$$

(24) is interpretable: The combined subproblem reward for sensors $\delta \in \Delta$ in iteration k, considering only blocks also assigned to these sensors from the master problem in iteration h, is greater than the combined subproblem reward for the same sensors in iteration h. We continue to show this is false for a globally optimal solution at iteration k.

$$0 > \sum_{d\in\Delta} \sum_{j\in J_d^h} r_j y_{dj}^{h*} - \sum_{\delta\in\Delta} \sum_{j\in\left(J_\delta^k\cap J_\delta^h\right)} r_j y_{\delta j}^{k*}$$

$$0 > \sum_{d\in\Delta} \sum_{j\in\left(J_d^h-\left(J_d^k\cap J_d^h\right)\right)} r_j y_{dj}^{h*} - \sum_{d\in\Delta} \sum_{j\in\left(J_d^k\cap J_d^h\right)} r_j \left(y_{dj}^{h*} - y_{dj}^{k*}\right) \tag{25}$$

To arrive at a contradiction, we can use the fact that if y^k is globally optimal, $\forall\ d \in D, \forall\ j \in J_d^k, y_{dj}^k = y_{dj}^{k*} = 1$. This enforces that

$$\sum_{d\in\Delta} \sum_{j\in\left(J_d^k\cap J_d^h\right)} r_j \left(y_{dj}^{h*} - y_{dj}^{k*}\right) \leq 0$$

and thus

$$\sum_{d\in\Delta} \sum_{j\in\left(J_d^h-\left(J_d^k\cap J_d^h\right)\right)} r_j y_{dj}^{h*} - \sum_{d\in\Delta} \sum_{j\in\left(J_d^k\cap J_d^h\right)} r_j \left(y_{dj}^{h*} - y_{dj}^{k*}\right) \geq 0$$

which is a contradiction to (25). □

B. Experimental Instance Data

Satellites

The IDs of satellites used in the computational experiments are given in Table 4. The TLEs of this satellites are accessed using `celestrak.org` [20].

Sensors

The slew rate, settle time, and location for all sensors used in the computational experiments are listed in Table 5. The observational constraints for each sensor are listed in Table 6.

Table 4: **IDs of satellites used in experiments**

42709	39070	37737	28884	37843	37951	40146	37210	41589	27566	01328	07646
08820	16908	22195	33105	38077	25772	25872	51511	00858	02608	02639	02969
03029	03431	04068	04250	04297	04353	04902	05587	05588	05709	05851	06691
07466	07544	07648	07815	08132	08516	08746	08747	09009	09855	10669	10855
10953	11568	11669	11940	11964	12447	12994	13035	13056	13069	13431	13624
13631	13643	14077	14133	14195	14234	14365	14951	14985	15144	15560	15574
15677	15946	15993	15994	16650	17561	17873	18387	18631	19400	19483	19484
19508	19621	20168	20203	20391	20662	20873	20923	20926	20946	21129	21227
21392	21639	21703	21762	21803	21805	22027	22116	22175	22205	22723	22796
22883	22911	22931	22966	23168	23185	23200	23426	23467	23522	23536	23553
23615	23639	23670	23712	23717	23741	23781	23880	24208	24307	24313	24315
24652	24748	24786	24916	24957	25010	25019	25110	25152	25258	25312	25318
25404	25491	25516	25894	25924	25954	26052	26089	26298	26356	26382	26554
26580	26608	26624	26639	26643	26715	26719	26766	26863	27298	27399	27403
27426	27444	27499	27566	27603	27632	27691	27711	27714	27715	27718	27831
27854	27954	28082	28089	28161	28187	28234	28240	28252	28378	28446	28472
28659	28868	28884	28903	28911	28924	28935	29014	29162	29236	29494	29495
29643	29648	29656	31102	31862	32252	32729	32951	33207	33274	33275	33376
33453	33596	33749	35491	35496	35756	36131	36411	36499	36516	37185	37210
37218	37737	37748	37806	37834	37843	37951	38072	38093	38356	38551	38978
39035	39070	39122	39127	39206	39222	39256	39360	39688	40146	40333	40425
40663	40746	40882	40892	40946	41471	41589	41591	41794	41869	41879	41893
41937	42070	42709	42741	42967	43226	43271	43633	44048	44231	44333	44582
44868	46114	47240	47613	49115	49817	50001					

Table 5: **Attributes of sensors used in experiments**

Sensor	Slew rate (deg/s)	Settle time (s)	Latitude (deg)	Longitude (deg)	Elevation (m)
RME01	3.2	15.0	20.7462	-156.4317	111.1
RME03	3.2	15.0	20.7464	-156.4315	106.9
RME04	3.2	15.0	20.7464	-156.4315	103.8
ABQ01	3.2	15.0	34.9631	-106.4973	1725.0
PR01	3.2	15.0	18.4847	-67.1489	100.0
LMNT01	3.2	15.0	-22.218683	114.10265	20.1

Table 6: **Observational constraints for sensors used in experiments**

Sensor	Altitude (deg)	Solar altitude (deg)	Sun separation (deg)	Moon separation (deg)	Earth shadow
RME01	Min: 21.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered
RME03	Min: 21.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered
RME04	Min: 25.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered
ABQ01	Min: 21.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered
PR01	Min: 21.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered
LMNT01	Min: 21.0	Max: -6.0	Min: 5.0	Min: 5.0	Considered

REFERENCES

[1] M. J. Holzinger and M. K. Jah, "Challenges and potential in space domain awareness," *Journal of Guidance, Control, and Dynamics*, vol. 41, no. 1, pp. 15–18, 2018. [Online]. Available: https://doi.org/10.2514/1.G003483

[2] B. M. Morris, E. Tollerud, B. Sipőcz, C. Deil, S. T. Douglas, J. Berlanga Medina, K. Vyhmeister, T. R. Smith, S. Littlefair, A. M. Price-Whelan, W. T. Gee, and E. Jeschke, "astroplan: An Open Source Observation Planning Package in Python," *The Astronomical Journal*, vol. 155, no. 3, p. 128, Mar. 2018.

[3] N. O. Fahrner, "A regional greedy algorithm for space domain awareness resource allocation," 2021.

[4] D. Shteinman, M. Yeo, A. Ryan, S. Dorrington, J. Bennett, and M. Lachut, "Design & development of an optimized sensor scheduling & tasking program for tracking space objects," in *Advanced Maui Optical and Space Surveillance Technologies Conference*, 2019, p. 83.

[5] D. P. Bertsekas, "Auction algorithms." *Encyclopedia of optimization*, vol. 1, pp. 73–77, 2009.

[6] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[7] B. D. Little and C. Frueh, "Space situational awareness sensor tasking: Comparison of machine learning with classical optimization methods," *Journal of Guidance Control and Dynamics*, vol. 43, pp. 262–273, 2020.

[8] P. M. Siew and R. Linares, "Optimal tasking of ground-based sensors for space situational awareness using deep reinforcement learning," *Sensors*, vol. 22, no. 20, p. 7847, 2022.

[9] X. Long, W. Cai, L. Yang, and L. Yang, "Moea/d based multi-sensor collaborative task scheduling for space domain awareness," in *International Conference on Guidance, Navigation and Control*. Springer, 2022, pp. 4288–4298.

[10] Q. Zhang and H. Li, "Moea/d: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[11] A. Hinze, H. Fiedler, and T. Schildknecht, "Optimal scheduling for geosynchronous space object follow-up observations using a genetic algorithm," in *Advanced Maui Optical and Space Surveillance Technologies Conference (AMOS)*. Maui Economic Development Board Maui, HW, 2016.

[12] C. D. Rodgers, *Inverse methods for atmospheric sounding: theory and practice*. World scientific, 2000, vol. 2.

[13] R. Gedik, C. Rainwater, H. Nachtmann, and E. A. Pohl, "Analysis of a parallel machine scheduling problem with sequence dependent setup times and job availability intervals," *European Journal of Operational Research*, vol. 251, no. 2, pp. 640–650, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0377221715010693

[14] L. Fanjul-Peyro, R. Ruiz, and F. Perea, "Reformulations and an exact algorithm for unrelated parallel machine scheduling problems with setup times," *Computers & Operations Research*, vol. 101, pp. 173–182, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0305054818301916

[15] J. Hooker and G. Ottosson, "Logic-based benders' decomposition," *Mathematical Programming, Series B*, vol. 96, pp. 33–60, 04 2003.

[16] Y. Chu and Q. Xia, "A hybrid algorithm for a class of resource constrained scheduling problems," in *Integration of AI and OR Techniques in Constraint Programming*, 2005.

[17] J. Forrest, T. Ralphs, H. G. Santos, S. Vigerske, J. Forrest, L. Hafer, B. Kristjansson, jpfasano, EdwinStraver, M. Lubin, Jan-Willem, rlougee, jpgoncal1, S. Brito, h-i gassmann, Cristina, M. Saltzman, tosttost, B. Pitrus, F. MATSUSHIMA, and to st, "coin-or/cbc: Release releases/2.10.10," Apr. 2023. [Online]. Available: https://doi.org/10.5281/zenodo.7843975

[18] W. E. Hart, J.-P. Watson, and D. L. Woodruff, "Pyomo: modeling and solving mathematical programs in python," *Mathematical Programming Computation*, vol. 3, no. 3, pp. 219–260, 2011.

[19] M. L. Bynum, G. A. Hackebeil, W. E. Hart, C. D. Laird, B. L. Nicholson, J. D. Siirola, J.-P. Watson, and D. L. Woodruff, *Pyomo–optimization modeling in python*, 3rd ed. Springer Science & Business Media, 2021, vol. 67.

[20] D. T. Kelso. (2023) Celestrak. [Online]. Available: https://celestrak.org/NORAD/elements

BIOGRAPHY

Grant Nations *is a research and development intern at KBR and an undergraduate computer science student at the University of Utah. At KBR, his work has focused on creating novel scheduling solutions for space observations on a distrubuted network of ground-based sensors. His current research interests include scheduling for Space Domain Awareness and improving efficiency in electroaerodynamic thruster arrays.*

Justin Fletcher *received the M.S. degree in computer science from the Air Force Institute of Technology, Dayton, OH, USA, in 2016, with a concentration in machine learning. Mr. Fletcher is an advisor and subject matter expert on the enterprise adoption of deep learning technologies for the United States Space Force, Space Systems Command, and serves as a Major in the United States Air Force reserves. His current research interests include applications of computer vision to space object sensing, reinforcement learning for optical system actuation, deep learning framework optimization on high performance computing systems, and low size, weight, and power deep learning applications.*