Search Applications for Integrated Planning and Execution of Satellite Observations using Dynamic Targeting

Akseli Kangaslahti[1,2], **Itai Zilberstein**[2], **Alberto Candela**[2], **Steve Chien**[2]
[1]University of Michigan, Ann Arbor, USA
[2]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, USA
akanga@umich.edu, itai.m.zilberstein@jpl.nasa.gov, alberto.candela.garza@jpl.nasa.gov, steve.a.chien@jpl.nasa.gov

Abstract

In the Dynamic Targeting (DT) planning problem, a satellite uses a look-ahead sensor to gain information on the upcoming environment. This information can be used to intelligently plan observations for the future as to maximize utility. The DT problem is an exciting application of automated planning to further Earth science missions. We build on previous work to analyze the performance of well-known search algorithms on realistic DT problem instances derived from data sets and operational constraints such as power and slewing. We also examine the strengths and weaknesses of different search algorithms and strategies and explore how they optimize the trade-off between planning time and execution time in this online planning application. We evaluate beam searches, partitioned depth-first searches, Monte-Carlo tree searches, A* searches, and a greedy approach on two DT problems. For comparison, we also evaluate an algorithm representative of most planning algorithms aboard Earth science missions today as well as an upper bounding algorithm. Empirical results and analysis highlight the dominance of slewing capability in the search for a high-utility observation plan, meaning successful planning algorithms need to be computationally efficient in order to maximize slewing capability at execution time.

Introduction

Intelligent planning aims to increase the performance of robotic systems across applications and domains. Remote sensing is one such domain. We examine the application of intelligent planning to a real remote sensing problem: Dynamic Targeting (DT).

The Dynamic Targeting planning problem is an emerging, exciting application of planning and scheduling to Earth science and space. In the DT problem, a satellite uses a look-ahead sensor to retrieve information on the upcoming environment. This look-ahead information is used to construct a short-term plan as to maximize the utility of observations taken by a primary sensor. Advancements in satellite technology raise the bar of potential science return, but in turn increase the complexity of planning problems. Agile satellites are satellites that can slew their primary sensor, which enables a greater region for potential observation. These satellites are used in the DT planning problem. In addition, the next generation of satellites will have a higher ceiling for on-board computation, using processors that enable deep learning (Swope et al. 2023; Chien et al. 2024). While DT is conceptualized for Earth-observing satellites, the concept extends beyond this application. DT applies to a variety of observing assets, whether in air, in water, in space, or on land.

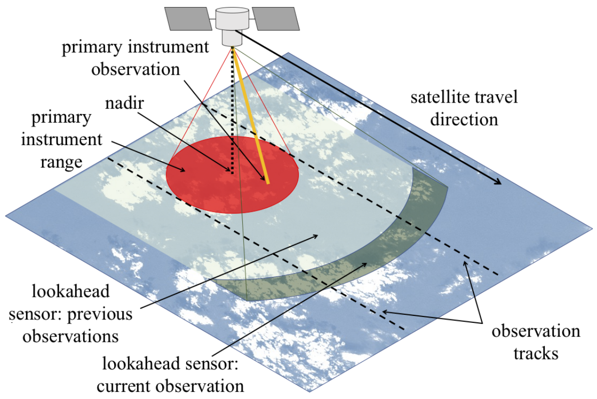
primary instrument observation
nadir
primary instrument range
satellite travel direction
lookahead sensor: previous observations
lookahead sensor: current observation
observation tracks

Figure 1: Dynamic Targeting uses a look-ahead sensor to plan intelligent observations. Figure is adapted from (Candela, Swope, and Chien 2023).

The goal of Dynamic Targeting is to utilize intelligent planning solutions to increase the quality of the science data from observations. The data from the look-ahead sensor enables avoidance of low utility observations, such as ones that are blocked by clouds, and identification of potential high utility observations. This information is used to plan and execute, within constraints, the actions of the spacecraft for the near future. Figure 1 illustrates the DT concept (Candela, Swope, and Chien 2023).

DT can be applied to a wide variety of Earth science missions. Furthermore, the ability of DT algorithms to improve science return has already been demonstrated (Candela, Swope, and Chien 2023). Thus, DT is expected to become common amongst future missions (Candela et al. 2022). However, much of the previous work on DT planning is not yet applicable to general missions. The majority of DT research has failed to account for operational constraints of onboard hardware and the complex utility models that are realistic to Earth science applications. Previous relaxations include infinite slewing capabilities and constant utility for repeat observations of the same target. Recently, Kangaslahti et al. worked towards a more realistic DT framework, which included a physics-based primary sensor slew model and complex utility functions to account for changing utility across observations and time (2024). Just considering slewing capabilities, resource constraints, and dynamically changing utility creates an exponentially large search space to plan over for a single look-ahead observation. The previous work also introduced some preliminary algorithms for planning over the search space, including a greedy algorithm and a tree-search algorithm, that were tested on two data sets (Kangaslahti et al. 2024). However, the DT problem remains largely unexplored. Very few planning algorithms have been tested, and there has been little insight into the performance of those algorithms across real instances of DT problems. Furthermore, the search algorithms deployed to plan observation paths can have long run-times. When considering online planning, the planning time reduces the amount of time that remains for execution during a fixed plan-execute cycle time. This imposes a trade-off between search thoroughness and execution time. Searches that are more thorough have the potential to produce plans that accrue more utility, but they must account for a decrease in the available time to slew and observe since their longer plan computations expend valuable cycle time. So far, there has also been minimal analysis regarding this trade-off.

In this paper, we aim to propel the research of the DT problem further by evaluating more search techniques on more real-world data sets, and crucially, uncovering the properties of planning techniques and problem instances that drive the performance of DT solutions.

We introduce variations of beam search, Monte-Carlo tree search (MCTS), and A* search to compare to a revised version of the previously presented partitioned depth-first search (Kangaslahti et al. 2024). All of these algorithms aim to search the look-ahead space to generate high-utility plans for the primary sensor. Our analysis includes a trade study of correlations between features of the problem instances and performance of the search algorithms. We also delve into optimizing the allocation of planning time and execution time and investigate how different problem instances can affect this trade-off.

In the next sections, we discuss related work relevant to the DT problem, describe the DT problem in full, outline a variety of observation planning algorithms, and present and analyze the performance of these algorithms on data sets.

Related Work

Dynamic targeting is part of a collective effort to increase the efficiency of satellite data collection. Prior to the DT concept, previous work rapidly screened observations for clouds and removed any cloudy observations to reduce data volume (Thompson et al. 2014). This method was demonstrated on the Airborne Visible-Infrared Imaging Spectrometer - Next Generation (AVIRIS-NG) instrument (Thompson et al. 2014). Early intelligent targeting methods include the imaging order scheduling algorithm that was used for the FORMOSA-2 satellite, which generated a nominal observation schedule based on weather conditions and forecasts that could be adjusted at execution time (Liao and Yang 2005). In addition, intelligent targeting onboard the Thermal and Near Infrared Sensor for Carbon Observation Fourier-Transform Spectrometer-2 (TANSO-FTS-2) was used to avoid observations blocked by clouds, resulting in more clear-sky observation scenes by a factor of 1.8 (Suto et al. 2021). Furthermore, the Earth Observing-1 (EO-1) satellite flew an algorithm that analyzed previous observations in order to intelligently schedule subsequent observations (Chien et al. 2005), although the subsequent observations were scheduled for the next orbital cycle, roughly 90 minutes later. Beaumet, Verfaillie, and Charmeau also presented a feasibility study that showed, in theory, how a look-ahead camera for cloud detection could be combined with a primary sensor using an online algorithm to better satisfy mission objectives (2011).

DT builds on all of these past works by utilizing the look-ahead sensor to schedule observations during a single overpass, only a few minutes later or less. A greedy algorithm for scheduling such observations based on information from previous observations has been demonstrated (Chien and Troesch 2015). The DT concept specifically has been demonstrated in simulation for both storm hunting (Swope et al. 2021; Candela et al. 2022) and cloud avoidance (Candela, Swope, and Chien 2022, 2023). It has also been conceptualized for mapping the planetary boundary layer (Candela et al. 2024). However, previous research has assumed infinite slewing capabilities and constant utility for repeat observations of the same target. One theoretical DT study represented slewing limitations by only allowing the primary sensor to move a certain spatial distance along each axis between observations (Hasnain et al. 2021). More recently, DT was simulated using a more realistic, physics-based slew model with a complex, varying utility function dependent on repeat observations (Kangaslahti et al. 2024). This study introduced the slew model and utility model that motivate the work in this paper. In addition, the authors presented a greedy algorithm and a partitioned depth-first search algorithm that we adapt in this paper (Kangaslahti et al. 2024). We analyze these algorithms and several additional algorithms in this paper.

Dynamic Targeting Problems

In this section we discuss the variations in real instances of Dynamic Targeting (DT) problems and the different aspects that define a problem instance. There are several features of DT that make it a challenging application of robotic planning:

- *Real-time planning*: as the satellite orbits Earth, the look-ahead sensor provides a constantly changing view of the upcoming environment. Planning must occur in real-time as to task the primary sensor before the satellite passes over the look-ahead view, resulting in constantly updating planning horizons with new problem views.
- *Real-world constraints*: a satellite is constrained by several driving factors including memory limits, computational power, slewing capabilities, and energy consump-

tion. All of these factors contribute to an exponentially large search space.

- *Dynamic environments*: the look-ahead sensor provides a current view of the upcoming environment. However, this environment may be volatile and change during the planning horizon.
- *Complex utility*: the utility is driven by the science return of the primary sensor. Science return can be hard to quantify, especially when the environment is changing. Taking an observation with the primary sensor can impact the utility of the local space, depending on the application. For example, some applications may want multiple repeat observations of the same target in order to track how it changes over time, while other applications may de-prioritize collecting repeat observations of the same geo-spatial region.

The aspects above can all shift drastically depending on the science mission, operational constraints, and other factors. We aim to understand how prominent search methods can apply to this planning problem, and what techniques serve best for different instantiations of the DT problem. The most basic components of a DT problem are:

- H: a planning horizon.
- D: the data obtained by the look-ahead sensor over time.
- C: the set of constraints of the satellite.
- A: the set of actions of the satellite.
- U: the utility function.

The goal of the problem is to construct a plan, $\pi = (a_1, ..., a_n)$ where $a_i \in A$, that does not violate C and maximizes the utility of the observations taken by the primary sensor, as defined by U, over the course of the planning horizon H. The next sub-sections discuss these components for different problem instances.

Look-Ahead Data

In our data sets, the look-ahead data $D = D_1, ..., D_n$ is a sequence of discrete views from the look-ahead sensor over the planning horizon. Each D_t is a matrix containing a map of the current environment with static utilities for each pixel in the look-ahead. We call the time that passes between t and $t+1$ the *cycle time*. Several factors influence the sampling of the look-ahead sensor including the angle of the sensor, the velocity of the satellite, and the re-sample frequency. This means the difference between the look-ahead view at time t and time $t+1$ can change across agents and problems. There are two data sets we evaluate:

1. *Global Precipitation Measurement (GPM)*: this data set contains look-ahead data generated from precipitation rate estimates retrieved by integrated multi-satellite retrievals and brightness temperature from the National Oceanic and Atmospheric Administration Climate Prediction Center, National Centers for Environmental Prediction, and the National Weather Service (Candela, Swope, and Chien 2023). An expert generated classification rules that label a pixel as $c \in \{0, 1, 2\}$ based on the measurements. The goal of this instance is to identify and observe storm clouds of interest, which are represented by higher label values.
2. *Moderate Resolution Imaging Spectroradiometer (MODIS)*: this data set contains data from a moderate resolution imaging spectroradiometer. This data determines the fraction of a pixel that is covered by a cloud. Thresholds in the cloud fractions determine the label of a pixel, $c \in \{0, 1, 2\}$, where higher label values represent more clear images. The aim of this DT instance is to avoid taking observations that are blocked by clouds.

These data sets have been used in previous work (Candela, Swope, and Chien 2023; Candela et al. 2022), and are becoming baselines for evaluating DT solutions on realistic look-ahead data. We refer the reader to (Candela, Swope, and Chien 2023) for more details on the classification approaches of the raw data, and other aspects of the data sets. Note that the standard we have enforced across the two data sets is for the classified data to be labelled $c \in \{0, 1, 2\}$ such that increasing label values classify pixels of increasing science utility.

Constraints and Actions

Across the data sets, the actions we consider are fixed. The available actions during each cycle are combinations of slewing and observing. The primary sensor can be slewed to point at a target with no energy penalty, however slewing requires time to perform the rest-to-rest maneuver. We model the satellite as a rigid body. Our slew model (Kangaslahti et al. 2024) is physics-based and defined by angular acceleration/deceleration values and maximum angular velocities of the primary sensor along the pitch and roll axes. Each axis is treated as independent of the other. To calculate a reachable set of observation targets in a given amount of time, we first calculate the maximum angular distance that can be slewed along each axis in the allotted time, given the angular acceleration/deceleration and maximum angular velocity along each axis. Then, one ray is cast down from the satellite to each ground observation target within the entire primary sensor range. The angular distance required to slew

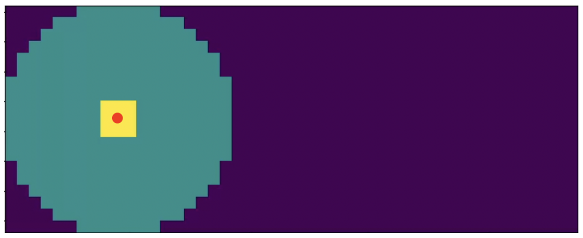

Figure 2: The reachable set of targets that can be reached within one cycle of slewing in our experiments. We use the slewing model first introduced in (Kangaslahti et al. 2024). The red dot in the middle is nadir, which is where the primary sensor is currently pointed. The yellow area represents all targets that can be slewed to within one cycle. The blue area shows the entire primary sensor range. The right edge of the purple area is the extent of the look-ahead range.

from the current primary sensor position to each ground target ray is then calculated for both axes using a spherical coordinate system. Any targets that fall within the previously calculated angular distance threshold for each axis are considered reachable observations within the originally specified amount of slewing time (Kangaslahti et al. 2024). Figure 2 shows how far the primary sensor can slew within one cycle in our experiments.

The primary sensor must be at rest when an observation is collected, so observations cannot be collected while the primary sensor is slewing. The satellite is also constrained by its energy consumption. Observations can be made a maximum of one time per cycle at the expense of energy. A constant amount of power is regenerated during each cycle from the solar panels on the satellite, regardless of whether an observation is collected or not. The satellite cannot make an observation if the current state of charge is below the energy cost of an observation. Additionally, there is processing time required to collect an observation.

We use previously presented values from DT research to define the common constraints of our satellite across data sets in Table 1 (Kangaslahti et al. 2024). We use realistic angular acceleration and maximum angular velocity values (Lappas, Steyn, and Underwood 2002).

Utility Model

The utility model we evaluate captures several real-world phenomena. Static utility functions are often too simple for remote sensing applications. Complex strategies that evolve over time with the observations and dynamism of the measured phenomena are required. Our utility model is a function of the previous observations and time, making it a suitably complex model to evaluate. A similar utility model was introduced in previous work (Kangaslahti et al. 2024). For all the data sets, we use a decrease-after-observation utility paradigm. The base utility of a point classified with label $c \in \{0, 1, 2\}$ is 4^c. After observation, the utility of the points around the observed point are decreased with a Gaussian kernel centered at the observation point. As time passes, the utility of decreased points gradually recovers to the initial utility. In addition, an off-nadir penalty is applied to the utility of an observation. In practice, the further off-nadir a satellite observes, the worse the observation quality. We decrease the utility of off-nadir observations linearly in the distance of the target from nadir. This off-nadir penalty means the same point in space observed at different points in time will accrue different utility, regardless of the previous observations. Figure 3 illustrates the effect of this utility model.

The motivation of this utility function was presented in (Kangaslahti et al. 2024), but we re-iterate the reasons here. For both storm hunting and oceanic studies, it is valuable to prioritize gathering information for a diverse set of events as opposed to detailed observations of a single event. Thus, observing multiple high utility regions is preferred to re-observing a single high utility region. Accruing utility in this manner applies to domains outside of DT. For example, rover exploration may prefer to limit repeat visits to the same location in quick succession as to gain more knowledge of the environment.

Table 1: Common run-time Parameters Across Testing (Kangaslahti et al. 2024)

Parameter	**Value**	**Description**
Primary Sensor Range	15^o	Max off-nadir angle for primary sensor, from SMICES (Swope et al. 2021)
Look-ahead Sensor Range	45^o	Off-nadir angle for look-ahead sensor, from SMICES (Swope et al. 2021)
Power Recharge Rate	1%/ cycle	Rate at which battery power increases
Power Discharge Rate	1%/ obs	Rate at which battery power decreases during cycles when an observation is made
Cycle Time	3s	Total time allocated for each cycle
Gaussian Kernel Size	75 km $\times$ 75 km	Size of Gaussian kernel used in utility models
Gaussian Kernel Standard Deviation	15 km	Standard deviation of Gaussian kernel used in utility models
Angular Acceleration	$1.08°/s^2$	Angular acceleration of slewing along the pitch and roll axes. The value we use is realistic for small satellites
Maximum Angular Velocity	$5.40°/s$	The maximum angular velocity of slewing along the pitch and roll axes. The value we use is realistic for small satellites
Processing Time	0.1 s	Amount of time allocated for observation acquisition and processing
Minimum Utility Multiplier	0.25	The minimum portion of the original base utility that a target must retain in our utility model
Maximum Off-Nadir Penalty	20%	Penalty for an observation as far off-nadir as possible

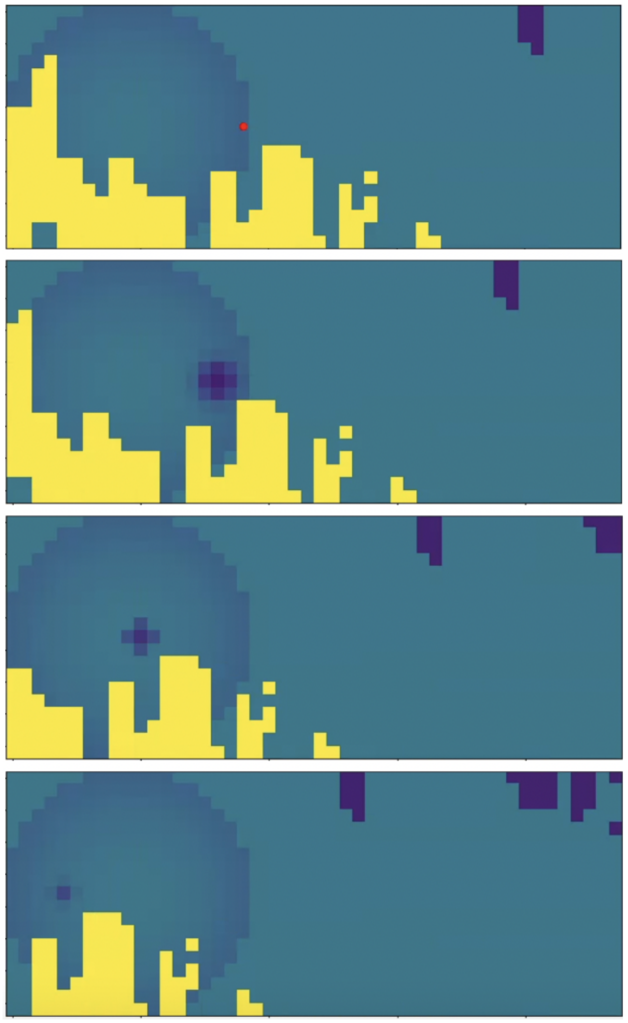

Figure 3: The effect of an observation on the utility of the corresponding target and nearby targets, under the decreasing utility model first introduced in (Kangaslahti et al. 2024). Brighter colors indicate greater utility. Note that the ring that forms on the left side of each figure is due to off-nadir penalties, which are applied to every target within the entire primary sensor range but have the greatest effect at points far from nadir. First, an observation is made, as indicated by the red dot (top). Then, the utilities of the corresponding target and nearby targets decline sharply for the next cycle (top middle). As time passes and the satellite flies to the right, the utility of these targets gradually recovers (bottom middle, bottom).

Planning Algorithms

At the center of Dynamic Targeting (DT) is a search to find high-utility, feasible plans based on the look-ahead sensor view of the upcoming environment. Prior work includes a greedy approach and a partitioned depth-first search (Kangaslahti et al. 2024). For a limited-width depth-first search such as the partitioned depth-first search from (Kangaslahti et al. 2024), little analysis has been conducted on the effects of different heuristic expansion of the search tree and the values of the limited-width and the maximum depth in terms of run-time and solution quality, which we aim to explore in this paper.

Along with the partitioned depth-first search, we analyze a beam search, a Monte-Carlo tree search (MCTS), and an A* search. The search trees for DT consist of nodes and edges where a node represents a problem state and an edge corresponds to actions between states. At depth d from the root, the nodes correspond to states that are d cycles in the future from the root node. Each node in the search tree consists of the current state of charge, p, the location of the primary sensor, (x, y), as well as the data read in by the look-ahead sensor and the current utility. At a node, the actions available include observing a reachable point at the next cycle or waiting, which expands the set of reachable points for child nodes since more time is allocated for slewing. The details of all evaluated algorithms are presented below.

1. *Nadir Only (NO)*. This algorithm simply observes the target directly at nadir whenever there is enough power available for an observation. This is representative of the scheduling algorithms aboard most Earth science missions today (Candela et al. 2022).
2. *Greedy (G)*. We adapt our greedy algorithm from previous work (Kangaslahti et al. 2024). This algorithm does not plan into the future, but rather selects a locally optimal action. If there is enough power available for an observation, the highest utility target within the set of targets that are reachable in one cycle is selected for observation. If there is not enough power available for an observation, the primary sensor is slewed towards the highest utility target within the entire primary sensor radius. Unlike previous work, this algorithm does not use any power allocation adjustments, as that heuristic does not generalize well to different data sets.
3. *Beam Search (BS)*. The beam search uses a limited-width depth-first search structure to generate a short-term observation plan that spans a pre-specified number of cycles. After the search is completed, the first observation in the highest utility plan is executed and the algorithm subsequently re-evaluates with the new problem view. Beam search is parameterized by a search depth, d, and a search width, k. During expansion, a lower bound of the utility for the node is calculated by simulating the greedy algorithm d cycles into the future. An upper bound is also calculated using the upper bound algorithm described below (algorithm number 7). These bounds are used to prune branches during the search that are suboptimal. At each node, the beam search selects the k highest utility actions to expand. Although utility is only accrued when

taking an observation, we can estimate the utility of not observing. We do this by first calculating the amount of power that is saved by reserving the current cycle for slewing. We divide this value by the energy cost of taking an observation, which equates to the number of additional observations that can be made if we reserve the current cycle for slewing. We then multiply this value by the mean target utility within the current reachable set of observation targets. This serves as an estimate of the utility that could be accrued if no observations are taken at the current cycle. Beam search recursively performs this expansion for each of the k highest utility actions until the search depth is reached. The highest utility observation plan is returned, and the first observation in that plan is executed. We repeatedly search with new problem views to generate subsequent plans.

4. *Partitioned Depth-First Search (PDFS).* PDFS resembles beam search with one major distinction. During expansion, instead of selecting the highest k utility actions, PDFS partitions the set of reachable targets into k sets of equal size and selects the highest utility action from each partition. The partitioning is done by angle after converting the rectangular coordinates of the reachable set to polar coordinates, thus creating partitions like slices of a pie. The idea behind this search is to enforce spatial diversity in observation candidates. A different version of this algorithm was first introduced in (Kangaslahti et al. 2024). As opposed to previous work on PDFS, this variation considers waiting (i.e., not making an observation) as an action that may or may not be among the top k actions by using the same utility estimation as the BS. Previous versions also used a different search structure that resulted in longer run-times without a significant improvement in output plan utilities (Kangaslahti et al. 2024).

5. *Monte-Carlo Tree Search (MCTS).* Our variation of MCTS is adapted for our look-ahead planning problem. During selection, we use the Upper Confidence Bound (UCB) (Kocsis, Szepesvári, and Willemson 2006) to identify the tree node to expand. We modify the formula of UCB so that q is the maximum utility over the explored paths rooted at the node. For expansion, we define the hyper-parameter k, which denotes the maximum search width, limiting the breadth of the tree. We select the $k-1$ observations with the current highest utility to expand in addition to waiting at the current cycle. We use a random rollout and perform back-propagation as normal. The first observation in the returned plan is executed. After each action, the search tree is pruned so that the root is the current state. The tree is continuously expanded both to reduce repeated computation and generate better estimates of the outcomes of actions that are still possible to take in the future.

6. *A*.* We implement a variation of A* that is adapted from Partial Expansion A* (PEA*), which we call limited-width A* (Yoshizumi, Miura, and Ishida 2000). As we are maximizing utility, we use a max priority queue for the frontier, where nodes are ordered based on their f values: $f(n) = g(n) + h(n)$. The upper bound algorithm, described below, serves as a suitable admissible and consistent heuristic, $h(n)$, for A*. We allow any node during our search to be considered a goal node and search for the highest utility path to any node over our planning horizon. This means that the first goal state reached will not terminate the search. To balance this increase in computation, we utilize a modified PEA* and expand only the top k child nodes based on their current utility. To fit within memory constraints, whenever a child node is added to the frontier, we prune nodes from the frontier with f values less than the expanded node's g value. The search terminates when the frontier is empty and the algorithm returns the highest utility plan. The plan is executed, and then the algorithm is called to plan over the new environment.

7. *Upper Bound (UB).* This algorithm, first presented in (Kangaslahti et al. 2024), is used to calculate an upper bound on the utility that can be accrued over a planning horizon. It is not a viable scheduling algorithm, as it assumes infinite slewing capabilities and does not necessarily abide by other hard operational constraints. First, this algorithm calculates o, the maximum number of observations that can be made during the simulation given the satellite's constraints. Then, a set of observations are constructed by selecting the highest utility target (considering static utility), in the entire primary sensor range during each cycle of the horizon. The o highest utilities of this set are then summed together to calculate the upper bound. This bound over-estimates the amount of obtainable utility and is not tight. Future work would explore ways to improve this bound.

Exhaustive search would have a complexity of $O((n^2)^d)$ for a single look-ahead view. Here, n is the diameter of the entire primary sensor reachability range and d is the search depth. In comparison, PDFS, BS, and A* have a complexity of $O(n^2 k^d)$, where k is the search width. Another benefit of these search strategies is that the $O((n^2)^d)$ complexity of the exhaustive search strategy is largely controlled by the spatial footprint of the instrument (i.e. proportional to n). The complexity of PDFS, BS, and A* is driven by k and d rather than n. This affords more power to the search parameters, making these algorithms generalizable to missions with differing sensors and spatial footprints and satellites with varying computational power (Kangaslahti et al. 2024).

In comparison to the greedy algorithm, we obtain the following result:

$$BS, PDFS, A^* \geq G$$

BS, PDFS, and A* are lower bounded by the performance of the greedy algorithm. This is a trivial result since these three algorithms are optimal over the sub-region of the search space they expand, which is guaranteed to contain the action selected by the greedy algorithm based on our implementations.

Experiments and Results

We evaluate all planning algorithms on both the MODIS and GPM data sets. The spatial resolution of the GPM data set is 4 km/pixel, which is much finer than the 11 km/pixel resolution of the MODIS data set. The utility available in the GPM data set is sparse compared to the MODIS data set. Recall that higher class pixels have exponentially higher utility. In the GPM data set, approximately 95.99% of pixels are class 0, 3.90% of pixels are class 1, and 0.11% of pixels are class 2. In the MODIS data set, 69.48% of pixels are class 0, 26.11% of pixels are class 1, and 4.41% of pixels are class 2. In order to select optimal search parameters (including maximum search width and depth) for each instance, we perform parameter selection using 1% of the data from each data set. By analyzing the parameter selection results, we uncover several underlying patterns and properties of these DT planning problem instances that determine the success of the various algorithms that we examine. We reserve the remaining 99% of the data for final testing using the optimal parameters. In both parameter selection and final testing, battery power starts at 0%. Although we only study the MODIS and GPM datasets in this work, these experiments could be replicated with different data sets to analyze other DT applications. We note that all experiments were executed using Python, so the reported run-times are subject to variation relative to what would be executed on-board a satellite.

Parameter Selection

Both the MODIS and GPM data sets contain 28400 cycles of data. We use 284 of these cycles for parameter selection. During parameter selection for a given algorithm and data set, we first sweep through all of our parameter combinations. For each parameter combination c, we record the run-time of the algorithm during each cycle in which it ran. We then find the maximum run-time r_c that the algorithm took to run across all cycles. Next, we run the same parameter selection sweep again, but for each parameter combination, we reserve r_c (simulated) time for computation during each cycle in which the planning algorithm runs. This means that planning algorithms that take longer to run are penalized in slewing time during plan execution, as the reserved computational time is subtracted from cycle time that could otherwise be spent slewing. This simplifies how real-time, on-board planning would function. As future work, we aim to execute these algorithms in a coupled planning and execution environment on simulated flight software.

We use the best performing parameter combination c' from this second parameter sweep in final testing. We also reserve $r_{c'}$ (simulated) time for computation during planning cycles in final testing. We repeat this process for every combination of search algorithm and data set.

Figure 4 shows the parameter selection results for all algorithms when run on the GPM data set. Likewise, Figure 5 shows the parameter selection results for all algorithms when run on the MODIS data set. In all figures, the total utility presented is from the second parameter sweep, which accounts for the run-time penalty. The color scale indicates the total utility accrued across all 284 cycles. White areas represent parameter combinations where either the maximum per-cycle run-time from the first parameter sweep exceeded the 3 second cycle time or the 16 GB memory limit was breached during the first parameter sweep. Note the differing search parameters that we iterated through across different algorithms.

Table 2: Optimal Parameters Found during Parameter Selection

	GPM Data set	MODIS Data set
BS	Depth = 3, Width = 3	Depth = 7, Width = 3
PDFS	Depth = 5, Width = 3	Depth = 5, Width = 5
MCTS	Depth = 4, Width = 2	Depth = 4, Width = 5
A*	Depth = 3, Width = 2	Depth = 7, Width = 4

Table 3: Maximum Single Cycle Run-Time during First Parameter Sweep for Optimal Parameters (ms)

	GPM Data set	MODIS Data set
BS	40.7457	475.0683
PDFS	172.4098	484.0328
MCTS	100.7869	30.8206
A*	36.8192	441.7007

Table 2 shows the best search width and depth combinations found during parameter selection. Table 3 provides the maximum run-times from the previously mentioned initial parameter sweep with no computational time subtracted from the available slewing time. During the second parameter sweep and final testing, these values indicate how much computational time is reserved during cycles in which the planning algorithm is executed.

First, we note that the in both BS and PDFS, parameters with a balance of search depth and width were optimal. This indicates that neither depth nor width dominates the search. These limited-width depth-first searches need to carefully balance exploration and exploitation in order to perform well. We also observe that the difference in search strategy between BS and PDFS does not have a great impact on performance, as the optimal parameters for the two algorithms accrued similar amounts of utility. Furthermore, BS and PDFS are relatively expensive in terms of run-time. In all of the BS and PDFS search parameter combinations that have no value in Figure 4 and Figure 5, it was the maximum per-cycle run-time that was exceeded rather than the memory limit. Especially in the GPM data set, the optimal parameters were relatively low values compared to other values that we swept through, meaning trading off search time for execution time is valuable. This suggests that the slewing capability, rather than search thoroughness, is a dominant factor in solution quality. Deeper and wider searches perform worse because their greater computational time limits the amount of time available to slew during the 3 second cycle.

For MCTS, we observe that the randomness of rollouts leads to varying performances across different sets of search parameters with seemingly little pattern. However, we do note that the optimal search parameters are again less thor-

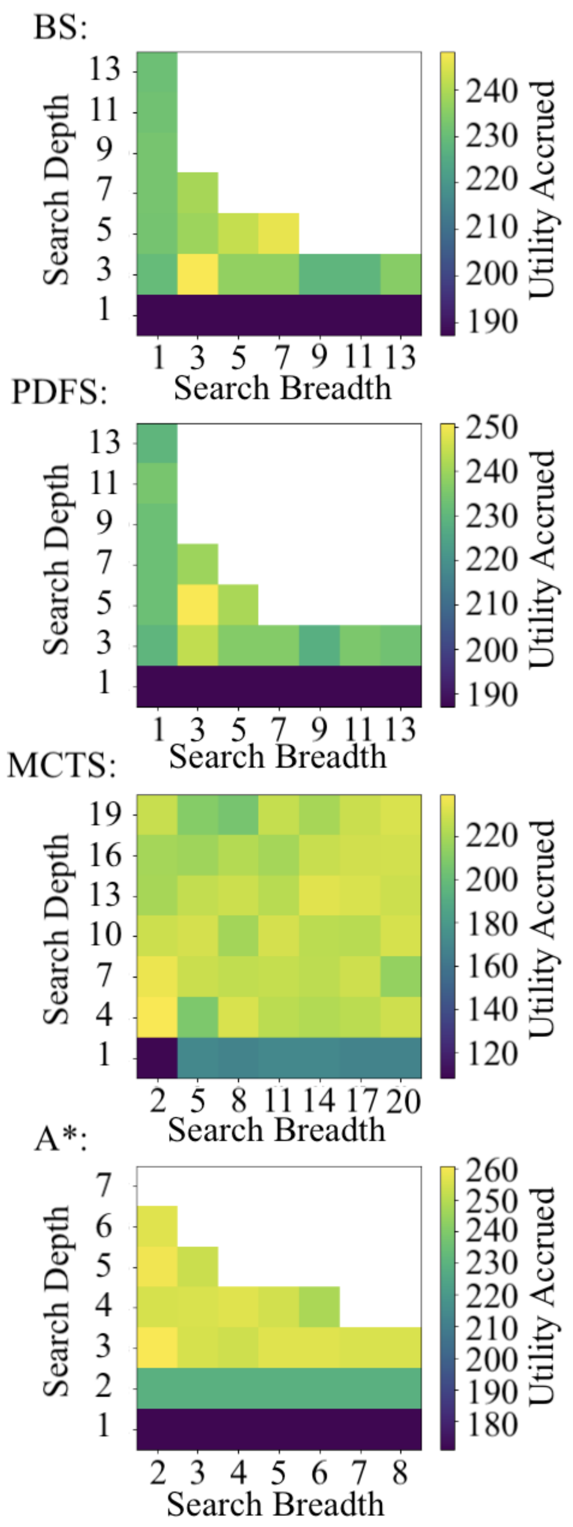

Figure 4: GPM data set parameter selection results for beam search (top), partitioned depth-first search (top middle), Monte Carlo tree search (bottom middle), and A* search (bottom).

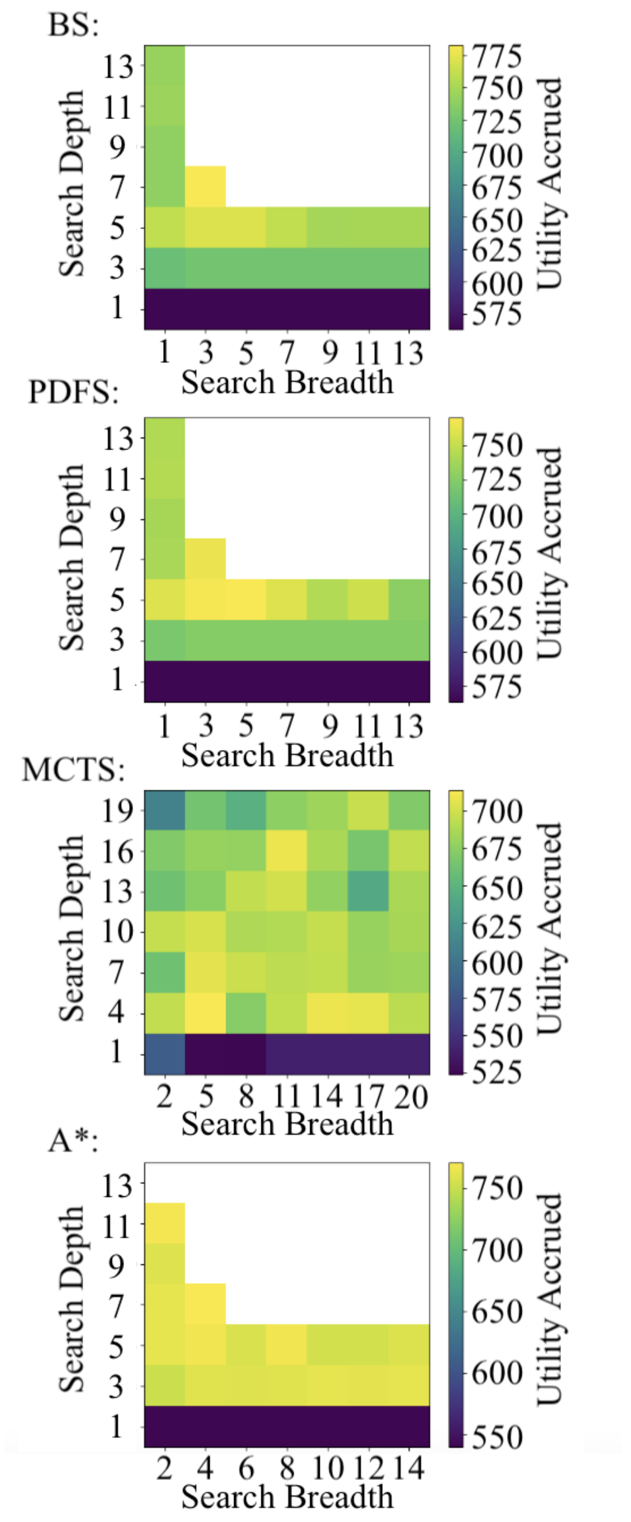

Figure 5: MODIS data set parameter selection results for beam search (top), partitioned depth-first search (top middle), Monte Carlo tree search (bottom middle), and A* search (bottom).

ough than most other parameters that we tried, which reinforces the idea that slewing capability, rather than search thoroughness, is crucial to solution quality.

In the A* search, we first note that the memory consumption of this search was far greater than the other searches. All parameter combinations with no value in Figure 4 and Figure 5 were combinations that exceeded memory limits, which did not occur in any other search even though the search parameters we tested for A* search did not have particularly high values compared to other searches. We also observe that there is less overall variation in performance across different search widths for the A* search compared to other search algorithms. We hypothesize that this is because the greediest option, according to the priority function that we use in the A* search, often ends up being the best choice. As a result, the optimal or near-optimal child nodes are added to the frontier early on during expansion, as they have the highest priority values. This means that increasing search width results in adding suboptimal nodes that get pruned from the frontier later. Therefore, after some small search width is reached, increasing search width has a minimal effect on run-time, slewing restriction, and overall performance, although it does increase memory consumption as nodes still need to be added to the frontier during expansion before they can be pruned. This is consistent with the parameter selection results. It is also consistent with the smaller variation in run-times that we recorded for the A* algorithm across varying search widths compared to other algorithms. For example, with depth held constant at 5, increasing the search width of the A* algorithm from 2 to 14 in the MODIS data set parameter selection only causes the maximum single cycle run-time to increase from 21.4202 ms to 634.4345 ms, while increasing the search width from 1 to 13 (with depth constant at 5) causes the maximum single cycle run-time of the partitioned depth-first search algorithm to increase from 13.3352 ms to 2813.6971 ms.

We also observe that overall, there were some search parameter combinations that were feasible for the MODIS data set but not the GPM data set, as they ran into run-time or memory issues with the GPM data set (e.g., depth = 5, width = 9-13 for the beam search). This is likely due to the fact that the GPM data set has a finer spatial resolution, meaning the data and search tree nodes simply have a greater volume and take longer to operate on. We also note that overall utility acquisition is far lower in the GPM data set parameter selection than in the MODIS data set parameter selection. This is most likely due to the sparsity of the GPM data set compared to the MODIS data set. Additionally, we observe that optimal search parameters tend to be less thorough and maximum per-cycle run-times tend to be lower for the sparser GPM data set compared to the MODIS data set. This suggests that slewing capability is especially dominant over planning time for solution quality in sparser data sets. In sparser applications, utility needs to be gathered whenever the opportunity arises, so slewing needs to be as flexible as possible in order to ensure that far-away high utility points can be reached. Deeper searches could theoretically be used to plan more proactively around such instances but fail to do so in practice due to longer run-times. Since planners re-evaluate after each observation, the cycles immediately following observations are the most affected by long algorithm run-times. This is problematic for searches with long run-times because in many scientific applications such as storm hunting and cloud avoidance, utility is distributed in clusters rather than uniformly. In the storm hunting application (the application of the GPM data set), these clusters are much smaller than in cloud avoidance (the application of the MODIS data set). This means that there are several brief windows throughout the GPM data set in which the planner needs to quickly observe multiple different high-utility points within a small cluster. Our results here support that quicker, more reactive searches with more slewing flexibility perform better in such windows than deeper, more restricted proactive searches. In applications with more abundant utility such as cloud avoidance (the application of the MODIS data set) utility is still often distributed in clusters. However, these clusters are larger, so it is less crucial to maximize slewing capability since high utility points are often not as rare or difficult to reach. Although run-time should still be kept to a minimum in such applications, our results show that deeper and more thorough searches help plan the order and timing of observations a little more proactively as to maximize the more abundantly available utility.

Table 4: Utility Accrued during Final Testing

	GPM Data set	MODIS Data set
NO	16098	34713
G	19901	37521
BS	23960	43555
PDFS	24614	43238
MCTS	22488	41273
A*	23992	43070
UB	37953	71038

Final Testing

During final testing, we run each algorithm on the remaining portions of each data set. This is the 99% of the data that was not used during parameter selection. We use the optimal search parameters found during parameter selection for each algorithm. We subtract the maximum per-cycle run-times that were found during parameter selection (which are shown in Table 3) from the allotted slewing time during cycles in which the planning algorithm is run in order to account for computational time.

Figure 6 shows the accumulated utility for each algorithm over time. Table 4 displays the total utility accrued by each algorithm during final testing.

In Figure 6, we observe that in GPM final testing, the line plots for the nadir only algorithm and the greedy algorithm appear almost linear, while those of the search algorithms have several small spikes in utility. This is because in the sparser GPM data set, algorithms need to plan to observe almost all of the rare high utility points, even when they are far away from nadir. This requires great slewing capability that the simpler planning algorithms do not have. Thus, the

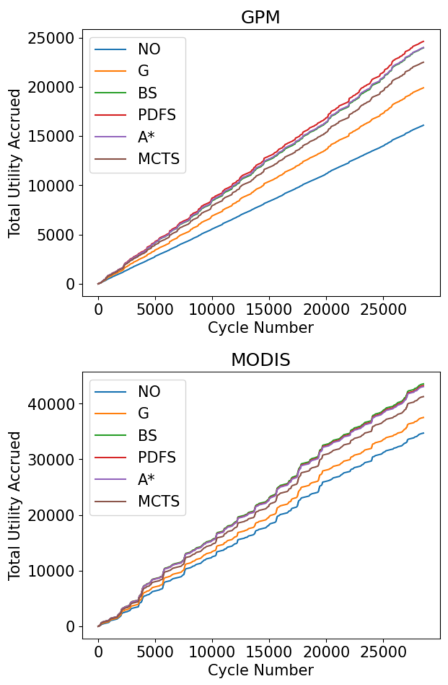

Figure 6: Total utility accrued vs. cycle number for all planning algorithms in GPM final testing (top) and MODIS final testing (bottom).

search algorithms can capitalize on the small clusters of high utility targets, causing small spikes in their line plots, while simpler planning algorithms observe low utility points almost every time, resulting in almost linear line plots. In contrast, all algorithm line plots have many large utility spikes throughout MODIS final testing. In applications with greater utility density and larger high-utility clusters, even the simpler algorithms can observe the high-utility targets that are easier to reach. The search algorithms just optimize the order and timing of observations during such windows.

As shown in Table 4, the greedy algorithm improves on the nadir only algorithm by about 24% for the GPM data set and about 8.1% for the MODIS data set. We attribute this to the sparsity of the GPM data set, as the nadir only algorithm can still observe some high utility targets by chance when utility is more abundant and high utility clusters are larger.

The partitioned depth-first search performs the best on the GPM data set with an improvement of 53% over the nadir only algorithm. It also accrues 65% of the intractable upper bound utility. The beam search algorithm performs the best on the MODIS data set, with an improvement of 25% over the nadir only algorithm. It accrues 61% of the intractable upper bound utility. However, in both problem instances, the BS, PDFS, and A* search strategies all result in relatively similar performances, with MCTS slightly underperforming compared to other algorithms. This aligns with the parameter selection results, where MCTS did not perform as well as other algorithms that do not use random rollouts. Furthermore, although DT offers significant improvement over current scheduling algorithms in both the MODIS and the GPM problem instances, we note that on a percentage basis, DT has greater science returns on the sparser GPM data set. We hypothesize that a significant portion of the disparity between the utility accrued by our best performing algorithms and the upper bound is a result of the difficulty in efficiently generating a tight upper bound on a planning problem with such a large search space. However, we aim to close this gap by improving both planning algorithms and upper bounding algorithms in future work.

Conclusion

Dynamic targeting (DT) is a challenging application of automated planning and scheduling. In working towards deploying DT solutions, we have analyzed different search strategies on different problem instances to uncover the most effective approaches to DT problems and understand why these approaches are effective.

Our results show that slewing capability is an essential factor in the search for a high-utility observation plan. Searches that prioritize run-time and slewing flexibility over thoroughness tend to perform best on the DT planning problem. We find that this is especially important for applications with sparse utility distributions (e.g. storm hunting), where high slewing capability is required to ensure that small clusters of high-utility observations can be collected whenever the opportunity arises, even when those targets are far away from the primary sensor. We also observe that applications with more abundant utility benefit less overall from DT on a percentage basis, as even simpler planners, like the nadir only planner, can accrue substantial utility by chance when clusters of high-utility targets are larger. Finally, we find that a variety of search strategies can be successful in accruing high amounts of utility, as long as parameters are selected appropriately. Based on the findings of this paper, we strive to continue boosting performance in future work by focusing on algorithm efficiency in order to maximize slewing capability while maintaining or even increasing search thoroughness. Developing anytime algorithms that can perform searches in any allotted amount of time are of particular interest, as current implementations waste extra time whenever the planning algorithm finishes early. We also aim to tighten our upper bound algorithm and use a coupled planning and execution environment in order to more realistically account for algorithm runtime in future work. Beyond simulation, we aim to utilize dynamic targeting in flight demonstrations, which will require further work in integrating planning and execution.

Nevertheless, we have found important performance trends in the DT problem and shown the potential for intelligent planning to greatly increase the data return of satellites compared to baseline observation collection schemes.

Acknowledgements

Portions of this research were carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration (80NM0018D0004). This work was supported by the Earth Science and Technology Office (ESTO), NASA.

References

Beaumet, G.; Verfaillie, G.; and Charmeau, M.-C. 2011. Feasibility of autonomous decision making on board an agile Earth-observing satellite. *Computational Intelligence*, 27(1): 123–139.

Candela, A.; Delfa Victoria, J.; Zilberstein, I.; Kurowski, M.; Yue, Q.; and Chien, S. 2024. Dynamic Targeting scenario to study the planetary boundary layer. In *IEEE Geoscience and Remote Sensing Symposium*.

Candela, A.; Swope, J.; and Chien, S. 2022. Dynamic targeting for cloud avoidance to improve science of space missions. In *16th Symposium on Advanced Space Technologies in Robotics and Automation*.

Candela, A.; Swope, J.; and Chien, S. 2023. Dynamic targeting to improve Earth science missions. *Journal of Aerospace Information Systems*, 20(11): 679–689.

Candela, A.; Swope, J.; Chien, S.; Su, H.; and Tavallali, P. 2022. Dynamic targeting for improved tracking of storm features. In *International Geoscience and Remote Sensing Symposium*. Kuala Lumpur, Malaysia.

Chien, S.; Candela, A.; Zilberstein, I.; Rijlaarsdam, D.; Hendrix, T.; and Dunne, A. 2024. Leveraging commerical assets, edge computing, and near real-time communications for an enhanced New Observing Strategies (NOS) flight demonstration. In *IEEE Geoscience and Remote Sensing Symposium*.

Chien, S.; Sherwood, R.; Tran, D.; Cichy, B.; Rabideau, G.; Castano, R.; Davis, A.; Mandl, D.; Frye, S.; Trout, B.; Shulman, S.; and Boyer, D. 2005. Using autonomy flight software to improve science return on Earth Observing One. *Journal of Aerospace Computing, Information, and Communication*, 2(4): 196–216.

Chien, S.; and Troesch, M. 2015. Heuristic onboard pointing re-scheduling for an Earth Observing Spacecraft. In *International Workshop on Planning & Scheduling for Space*. Buenos Aires, Argentina.

Hasnain, Z.; Mason, J.; Swope, J.; Vander Hook, J.; and Chien, S. 2021. Agile spacecraft imaging algorithm comparison for Earth science. In *International Workshop on Planning & Scheduling for Space*.

Kangaslahti, A.; Candela, A.; Swope, J.; Yue, Q.; and Chien, S. 2024. Dynamic Targeting of Satellite Observations Incorporating Slewing Costs and Complex Observation Utility. In *IEEE International Conference on Robotics and Automation (ICRA 2024)*. Yokohama, Japan.

Kocsis, L.; Szepesvári, C.; and Willemson, J. 2006. Improved monte-carlo search. *Univ. Tartu, Estonia, Tech. Rep*, 1: 1–22.

Lappas, V.; Steyn, W.; and Underwood, C. 2002. Attitude control for small satellites using control moment gyros. *Acta Astronautica*, 51(1): 101–111.

Liao, D.-Y.; and Yang, Y.-T. 2005. Satellite imaging order scheduling with stochastic weather condition forecast. In *IEEE International Conference on Systems, Man and Cybernetics*, 2524–2529.

Suto, H.; Kataoka, F.; Kikuchi, N.; Knuteson, R. O.; Butz, A.; Haun, M.; Buijs, H.; Shiomi, K.; Imai, H.; and Kuze, A. 2021. Thermal and near-infrared sensor for carbon observation Fourier transform spectrometer-2 (TANSO-FTS-2) on the Greenhouse gases Observing SATellite-2 (GOSAT-2) during its first year in orbit. *Atmospheric Measurement Techniques*, 14(3): 2013–2039.

Swope, J.; Chien, S.; Bosch-Lluis, X.; Yue, Q.; Tavallali, P.; Ogut, M.; Ramos, I.; Kangaslahti, P.; Deal, W.; and Cooke, C. 2021. Using intelligent targeting to increase the science return of a Smart Ice Storm Hunting Radar. In *International Workshop on Planning & Scheduling for Space*.

Swope, J.; Mirza, F.; Dunkel, E.; Candela, A.; Chien, S.; Holloway, A.; Russell, D.; Sauvageau, J.; Sheldon, D.; and Fernandez, M. 2023. Benchmarking space mission applications on the Snapdragon processor onboard the ISS. *Journal of Aerospace Information Systems*, 1–9.

Thompson, D. R.; Green, R. O.; Keymeulen, D.; Lundeen, S. K.; Mouradi, Y.; Nunes, D. C.; Castaño, R.; and Chien, S. A. 2014. Rapid spectral cloud screening onboard aircraft and spacecraft. *IEEE Transactions on Geoscience and Remote Sensing*, 52(11): 6779–6792.

Yoshizumi, T.; Miura, T.; and Ishida, T. 2000. A* with partial expansion for large branching factor problems. In *AAAI/IAAI*, 923–929.