Dynamic Targeting of Satellite Observations Incorporating Slewing Costs and Complex Observation Utility*

Akseli Kangaslahti[1], Alberto Candela[2], Jason Swope[2], Qing Yue[2], Steve Chien[2]

***Abstract*— Maximizing the utility of limited Earth observing satellite resources is a difficult ongoing problem. Dynamic Targeting is an approach to this challenge that intelligently plans and executes primary sensor observations based on information from a lookahead sensor. However, current implementations have failed to account for realistic satellite operational constraints and have used static utility for repeat observations of the same target. To address these limitations, we implement a more general Dynamic Targeting framework that comprises a physics-based slew model, a dynamic model of observation utility, and an algorithm for gathering high-utility observations. To demonstrate this framework, we also supply complex dynamic utility models that are applicable to many missions and new algorithms for intelligently scheduling observations with slewing restrictions and changing utility, including a greedy algorithm and a depth-first search algorithm. To evaluate these algorithms, we test their performance across simulated runs through two datasets and compare to the performance of an algorithm representative of most scheduling algorithms aboard Earth science missions today as well as an intractable upper bound. We show that our algorithms have great potential to improve science return from Earth science missions.**

I. INTRODUCTION

One challenge in remote sensing is making the most of a limited number of observations. This problem is applicable to many missions as the utility of satellites in general is often limited by operational constraints, including swath, revisit rate, and battery power. Dynamic Targeting (DT) is a method for intelligent observation scheduling that leverages information from a lookahead sensor to identify targets ahead of time for the primary sensor of a satellite to observe. The lookahead sensor views the ground below the future orbit path of the satellite. Then, a planning algorithm or model is used to select target points for the primary sensor to observe once the satellite passes over that point. This way, the satellite's resources are efficiently allocated and more targets of high value are observed. Fig. 1 illustrates this concept [1].

The DT concept is applicable to many Earth science missions and its vast potential to increase the efficiency of observations has already been demonstrated (e.g., [1]). As such, we expect DT to become very common amongst future Earth science missions [2]. However, current DT systems are not yet fully generalizable, as they fail to account for operational constraints and they do not change the utility of targets that have already been observed, which is unrealistic. To address this, we have designed a new, more general DT framework that accounts for many operational constraints including slewing capability, calculates utility changes for repeat observations, and runs a DT algorithm to intelligently schedule observations. Our slew model calculates the reachable set of observations during each cycle based on physical satellite constraints that are set to match those of a given mission. We supply complex example dynamic utility models that either increase or decrease utility near observation targets that already been observed, depending on the specific application. This combination of limited slewing capability and dynamic utility introduces a difficult problem of efficiently searching for a high-utility observation path. To tackle this problem, we demonstrate algorithms for our new, general Dynamic Targeting framework, including a greedy algorithm and a bounded depth-first search algorithm. We demonstrate this framework and the performance of our algorithms with a simulation study using two datasets from the Smart Ice Cloud Sensing (SMICES) project [3], [4]. However, our framework remains general enough to replace our proposed utility models and algorithms if desired.

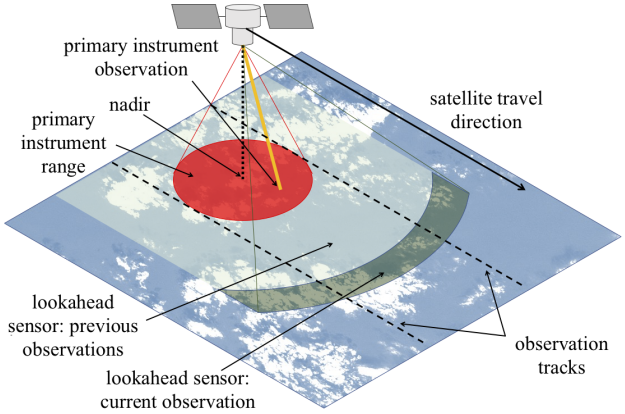

Fig. 1. Dynamic Targeting uses information from a lookahead sensor to intelligently plan primary sensor observations (adapted from [1]).

II. PRIOR WORK

One prior approach to more efficient data collection was rapid screening for clouds and subsequent compression and removal of cloud data in order to reduce data volume [5].

*The research was carried out at 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.

[1]A. Kangaslahti is with the University of Michigan, Ann Arbor, Ann Arbor, MI 48109, USA and the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91011, USA. akanga@umich.edu

[2]A. Candela, J. Swope, Q. Yue, and S. Chien are with the Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91011, USA. {alberto.candela.garza, jason.swope, qing.yue, steve.a.chien}@jpl.nasa.gov

This was demonstrated in real-time onboard the Airborne Visible-Infrared Imaging Spectrometer - Next Generation (AVIRIS-NG) instrument [5]. Another method for onboard observation data analysis and filtering to specifically reduce transmission bandwidth has also been demonstrated [6]. Other approaches have used intelligent targeting for agile instruments on Earth-observing missions. For example, the imaging order scheduling algorithm used for the now decommissioned FORMOSA-2 satellite used weather conditions and forecasts to generate a nominal observation schedule that could then be adjusted during execution [7]. Furthermore, a feasibility study presented by Beaumet, Verfaillie, and Charmeau [8] demonstrated in theory how an online algorithm for an agile Earth-observing satellite could combine a lookahead camera for cloud detection and a primary instrument for observations to better satisfy mission objectives. Additionally, a method for mission planning with cloud avoidance using a cloud-detecting sensor has been demonstrated in simulation [9]. Intelligent targeting has also been used aboard the Thermal and Near Infrared Sensor for Carbon Observation Fourier-Transform Spectrometer-2 (TANSO-FTS-2) for avoiding observations blocked by cloud cover [10], which resulted in an increase in clear-sky observation scenes by a factor of 1.8. Additionally, the Autonomous Sciencecraft Software (ASE) that flew aboard the Earth Observing-1 (EO-1) satellite used information from previous observations to schedule subsequent observations [11]. However, these subsequent observations were during the next orbital cycle, roughly 90 minutes later, while DT aims to direct observations during one overpass, only a few minutes or less later. A greedy scheduling algorithm for redirecting observations of satellites within the same overpass based on previous observations that were just acquired was also demonstrated [12]. DT specifically has previously been demonstrated in simulation for cloud avoidance [13] and storm hunting [2]. Additionally, DT algorithms have been developed and tested in simulation studies for deep convective ice storm hunting in the SMICES project [3], [4]. However, these projects all assumed infinite slewing capabilities. Other theoretical DT work has represented slewing limitations by only allowing the position of the primary sensor to be adjusted by a certain spatial distance along each slewing axis between timesteps [14]. In comparison, our new physics-based slew model calculates slewing ranges on each cycle based on the primary sensor position and the physical constraints of the satellite, which is more realistic. Our work is also the only project that has accounted for changes in target utilities depending on which targets have previously been observed, which, along with limited slewing capability, introduces a difficult utility maximization problem.

III. DATASETS

In our experiments, we use two datasets containing storm cloud data intended for the SMICES project, which aims to study deep convective ice storms [3]. One of these datasets is a tropical dataset from the Caribbean with 15 km/pixel resolution and the other is a nontropical dataset from the Atlantic Coast of the United States with 1 km/pixel resolution. These datasets are known to contain deep convective storms and are generated using a Weather Research and Forecasting (WRF) Digital Twin Simulation, as described in [4]. In this process, clustering is used to generate ground truth labels for the WRF brightness temperature data in each pixel based on hidden variables including ice water path, particle size, and cloud top height [4]. These labels are cloud classes from 0 to 4 (inclusive) of increasing interest and value.

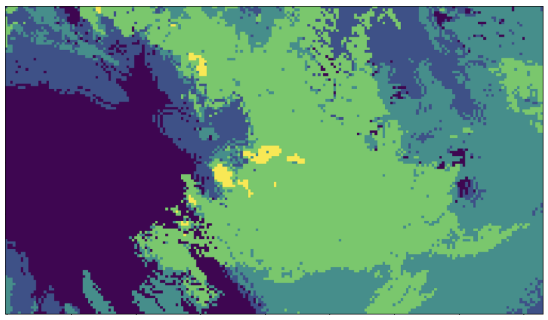

Fig. 2. A sample portion of the tropical dataset [3], [4]. Cloud classes 0-4 shown in increasingly bright colors.

Then, we train one random decision forest classifier on the WRF data and the corresponding ground truth labels for each of our two datasets [3], [4]. We use roughly a third of each dataset for this training, leaving the rest for parameter selection and testing. These classifiers are used to classify pixels in the remaining portions of the datasets. This is done to simulate classification error that would occur on a real mission. In our case, these classifiers performed at an overall classification accuracy of 72% for the tropical dataset and 73% for the nontropical dataset, compared to the ground truth labels. We account for classification error when evaluating the performance of our algorithms by always granting the utility associated with the ground truth cloud class rather than the classified cloud class. Although we only use these two datasets in our experiments, our framework is general enough to use any dataset of similar format.

IV. UTILITY MODELS

We have implemented two example utility models to demonstrate the capabilities of our new DT framework, both of which involve utility that changes based on observation locations. Although we only use these utility models in our experiments, our framework is general enough to schedule intelligent observations based on any model of utility.

In both models, we use a utility function of 4^C, where C is the class of the pixel ($C \in \{0,1,2,3,4\}$ in our case), to assign the original utility of each pixel before any observations are made. This is to ensure that even observations of the lowest possible value have positive utility while the observations of the highest possible value have substantially greater utility than other observations. On each cycle, a penalty proportional to distance from nadir is calculated to discourage lower quality off-nadir observations. Our framework permits substitution of this utility function if desired.

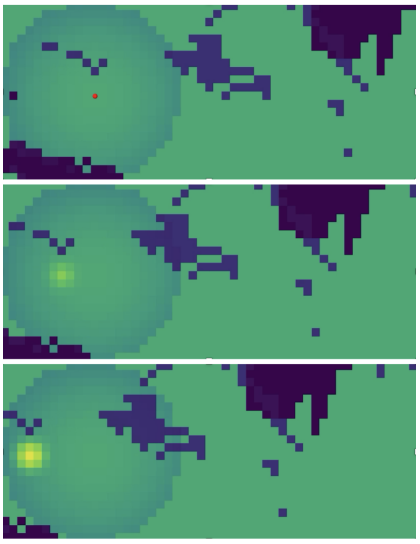

Fig. 3. The effect of an observation on nearby utility under the increasing utility model. In all three parts of the figure, brighter colors indicate higher utility targets. The ring that appears on the left side of all three parts is caused by off-nadir penalties. This ring also represents the overall slewing range of the primary sensor. During the first point in time (top), a target is observed, as indicated by the red dot. During subsequent points in time (middle, bottom), the compounding Gaussian kernel causes the utility of that target and nearby targets to increase as the satellite flies to the right.

The first utility model we use increases utility near targets that have already been observed. In this model, each point in the dataset is associated with a utility multiplier that starts at 1. First, an upper bound is set on the utility multiplier. Once an observation is made, a Gaussian kernel is used to increase the utility multiplier of that target and targets nearby. This increase compounds on every cycle until the target is observed again. If the same target is observed more than once, a Gaussian kernel is used to decrease the utility multipliers of that target and nearby targets after each repeat observation. Then, the utility multipliers of that target and nearby targets begin compounding again. The utility multiplier of targets located in multiple compounding Gaussian kernels will be compounded by all kernels but may never increase past the upper bound on the utility multiplier. Figure 3 illustrates how this utility model works.

The purpose of this utility model is to encourage repeat observations of spatially similar targets, especially from varying points in time and, implicitly, varying viewing geometries. This model is appropriate for any Earth science application with similar desired behavior, including some storm hunting applications in which tracking the changes of a storm as the satellite passes over it is important.

The second utility model that we use decreases utility near targets that have already been observed. In this model, each point in the dataset is associated with a utility multiplier that starts at 1. A lower bound is set on the utility multiplier. Once an observation is made, a Gaussian kernel is used to sharply decrease the utility multiplier of that target and targets nearby. Once utility multipliers decrease, they start to recover, compounding back to but never over 1. If a target is observed more than once, a Gaussian kernel is used to sharply decrease the utility multipliers of that target and nearby targets each additional time it is observed, after which the utility multipliers of that target and nearby targets begin compounding again. The utility multiplier of targets located in multiple compounding Gaussian kernels will be compounded by all kernels but may never increase past 1. The purpose of this utility model is to discourage repeat observations of spatially similar targets, especially from similar points in time and, implicitly, similar viewing geometries. This model is appropriate for any Earth science application with similar desired behavior, including many oceanic studies, in which observation targets do not change very much in the time that the satellite passes over them. Some storm hunting applications in which observing many different points in a storm is more important than tracking the same points may also benefit from using this utility model.

V. MODELING OPERATIONAL CONSTRAINTS

Our DT framework is capable of accounting for several physical satellite operational constraints. First, the orbit altitude, the maximum off-nadir angle of the pointable primary sensor, and the off-nadir angle of the lookahead sensor are used to determine a radius, in kilometers, for the ranges of the lookahead sensor and the primary sensor. These values, in addition to the cycle time, the spatial resolution of the data, and the orbit velocity of the satellite, are then used to determine how many pixels the satellite flies over in one cycle. The spatial resolution of the data is assumed to be equal to the spatial footprint of the primary instrument. In practice, if the lookahead sensor generates data at a spatial resolution that is different from the spatial footprint of the primary sensor, an additional data downsampling or upsampling step could be incorporated to meet this assumption. Furthermore, the initial state of charge, the power consumption per observation, and the power recharge per cycle are used to model the state of charge and determine whether an observation is possible during each cycle. Finally, the slewing capabilities of the primary sensor of the satellite are used to calculate a reachable set of observation targets during each cycle. We assume that the satellite is a rigid body. Slewing capabilities are inputted to our slew model as a constant angular acceleration/deceleration rate that the control moment gyroscopes (CMGs) aboard the satellite produce as well as a maximum angular velocity for the roll and pitch axes. To calculate the reachable set of observations during a cycle, we first compute the maximum angular distance that can be slewed along each axis based on the previously mentioned slewing parameters and the cycle time. Then, we project that angular distance to a physical distance on the ground based on the orbit altitude of the satellite and the current primary sensor position. Reachable targets are any pixels that fall within the allowed physical distance from the previous target along both axes. The roll and pitch axes are treated as independently of each other when calculating reachable sets of observation targets and yaw is assumed to remain constant at 0. Our experiments are based

on slewing capabilities of a small Earth observing satellite. Computational and processing times are also subtracted from slewing time when appropriate while calculating reachable sets of observation targets. Computational time is reserved for utility calculations and for running the DT planning algorithm. Processing time is reserved for observation acquisition and processing whenever an observation is collected by the primary sensor.

Many of the values for these operational constraints that we use in our testing are realistic for satellites in low Earth orbit that use CMGs [3], [15], [16]. These values, along with other parameters held constant during testing, are outlined in a later section in Table I.

VI. ALGORITHMS

We present two algorithms for observation scheduling in this DT framework, a greedy algorithm and a bounded depth-first search algorithm. We also provide a nadir only algorithm for a lower bound on performance and a separate calculation of an upper bound on performance.

A. Nadir Only (NO)

This algorithm simply observes the point at nadir whenever there is power available, resulting in observations that are indifferent to the data. This provides a lower bound on performance and is representative of the scheduling algorithms of most current Earth science missions [2].

B. Greedy (G)

This algorithm schedules the locally optimal action during each cycle. Before the algorithm is run, the power allocated for the current cycle is calculated by decreasing the amount of power currently available. This adjustment is proportional to the ratio of the total raw (before any multipliers or off-nadir penalties) utility within the reachable set of targets to the total raw utility within the combined primary sensor and lookahead range. This is to ensure that more power is used when the available utility is high compared to the utility coming up in the near future. The adjustment is also proportional to the ratio of the amount of raw utility in the entire primary sensor range to the mean amount of raw utility in the maximum primary sensor range during previous cycles. This is to ensure that more power is used when the nearby utility is high compared to previous cycles. Then, the greedy algorithm selects a locally optimal action. If enough power has been allocated for an observation or battery is at max capacity, an observation is made at the highest utility point within the reachability range of the current cycle. Otherwise, the primary sensor is slewed as far as possible towards the highest utility target within the radius of the primary sensor without making an observation in order to capture high-utility observations in the coming cycles. Both of these possible actions are rest-to-rest maneuvers. This achieves an $O(n^2)$ runtime complexity for each cycle, where n is the diameter of the entire primary sensor range in pixels.

C. Depth-First Search (DFS)

First, power allocation is adjusted similarly to the greedy algorithm described in the previous subsection. However, the adjustment is proportional to the ratio of the amount of raw utility within the entire primary sensor range (as opposed to the reachable set of targets for the current cycle) to the total raw utility within the combined primary sensor and lookahead range. This is to account for the extended range of slewing capabilities when slewing through multiple timesteps, which this search algorithm is capable of planning. Next, this algorithm calculates the best observation path it can find for a certain number of cycles into the future, schedules the first observation in that path, then re-evaluates once the observation is made. If no observations are scheduled in the best path found, then the primary sensor slews and stabilizes after the first cycle, then re-evaluates. The observation path is calculated using a recursive depth-first search algorithm. The desired search depth in cycles, t, and the desired search breadth, k, are inputs to the algorithm. First, a lower bound on utility is calculated by simulating the greedy algorithm described in the subsection above t cycles into the future. Next, the first recursive call is made. The recursive call calculates an upper bound on utility using an algorithm similar to the upper bound algorithm described in the next subsection. Then, for each whole number of cycles c such that $1 \quad c \quad$ the number of remaining cycles to plan, the reachable set is calculated with c cycles of slewing time with the processing and computational time subtracted if there is enough power allocated for an observation. At this point, the optimal strategy would be to make a recursive call where each of the $O(n^2)$ pixels in the reachable set are selected for observation in a separate branch, where n is the diameter, in pixels, of the entire primary sensor range. However, this has a very slow $O((tn^2)^t)$ runtime complexity. Instead, we partition the reachable set into k partitions and make k recursive calls, one for observing the highest utility point within each partition. The lower bound on utility is updated whenever a path is found with higher utility than the lower bound. This achieves an $O(n^2(kt)^t)$ runtime complexity, which is preferred to the optimal strategy not only because it is faster but also because a greater portion of the $O(n^2(kt)^t)$ runtime complexity is controlled by user preference of search thoroughness (k and t), while the $O((tn^2)^t)$ runtime complexity for the optimal strategy is in large part controlled by the spatial footprint of the instrument (which is proportional to n). This affords more power to the search parameters, making this algorithm suitable to a wider range of spatial footprints and thus a wider range of missions.

D. Upper Bound (UB)

We also provide an upper bound algorithm for putting performance into perspective. However, with changing utility and limited slewing capabilities, a tight upper bound is difficult to calculate efficiently. Note that upper bound algorithms should be specific to the utility model. For both of our utility models, we start by calculating the maximum number of observations that can be made during the entire simulation

Fig. 4. The reachable range of the primary sensor for a rest-to-rest maneuver with one cycle of slewing time and no computational time reserved, shown in 15 km/pixel resolution. The small red circle shows nadir, which is the starting location of the primary sensor. The yellow area shows the positions that the primary sensor could slew to within one cycle, in the satellite reference frame. The blue area shows the entire range of locations that the primary sensor is capable of slewing to given the 15 maximum off-nadir angle of the primary sensor, in the satellite reference frame. The right edge of the purple area represents the end of the lookahead range.

given power constraints, o. In the case of our increasing utility model, we compound the multipliers of all targets in the combined primary sensor and lookahead range with the maximum compounding factor during each cycle and add the utility of the highest utility target in the entire primary sensor range to a set of possible observations. At the end, we sum the utility of the o highest utility observations to calculate the upper bound. In the case of our decreasing utility model, we leave all utility multipliers at 1 (their maximum value) throughout the simulation. On each cycle, we add the utility of the highest utility target within the primary sensor range to a set of possible observations. Finally, we sum the utility of the o highest utility observations to find the upper bound.

VII. EXPERIMENTS AND RESULTS

All parameters other than the dataset, the utility model, and the algorithm were kept constant during our experiments. These common parameters are shown in Table I. Fig. 4 shows the slewing capabilities of the satellite during our testing.

Before testing our algorithms, we reserved about another third of each dataset for parameter selection for the depth-first search algorithm, leaving the final third for testing. We swept through search depth and breadth values of 1-14 for a combination of the two that achieves good performance and a maximum runtime that is tractable given the cycle time. This could be emulated on a real mission by trying a few different depth and breadth parameters when the satellite is first launched into orbit, or by running this parameter sweep on a sample dataset for the mission before launch. We ran a parameter sweep for every combination of our two utility models and our two datasets. Table II shows the parameters that our sweeps yielded. We also record the maximum cycle runtime during this parameter sweep and reserve that portion of the cycle time for computation during testing. This results in less slewing time and thus a more limited reachable set for the primary sensor when the DT algorithm is more computationally expensive. We do this for the greedy algorithm as well but omit the maximum cycle runtimes from this report because they are minimal in comparison to the depth-first search algorithm. Table III shows these maximum cycle runtimes for the depth-first search algorithm. Note that all testing was performed using Python, so all runtimes are subject to some variation.

The performance of each algorithm on the test data sets are shown in Table IV and Table V. All runtimes include the algorithm runtime and other time that needs to be reserved for computation during each cycle, including utility calculations for each target. Recall that all testing was performed using Python, so all runtimes are subject to some variation. Additionally, recall that the maximum per-cycle runtimes from Table III are subtracted from the amount of time that the primary sensor is allowed to slew during each cycle. The initial power was set to 0% at the beginning of all tests.

In general, our testing shows that the greedy algorithm performs far better than the nadir only algorithm that is representative of the algorithms aboard most current Earth science missions [2] and the depth-first search algorithm further improves on the performance of the greedy algorithm. Specifically, the greedy algorithm improves on the performance of the nadir only algorithm by an average of 65% across both utility models in the tropical dataset and 58% in the nontropical dataset. The depth-first search algorithm further improves on the performance of the greedy algorithm by an average of 9.1% across both utility models in the tropical dataset and 5.2% in the nontropical dataset. In the tropical dataset test with decreasing utility, the depth-first search algorithm accrues 78% of the upper bound utility. However, this algorithm only accrues 36% or less of the upper bound utility in all other tests. We believe that this is because utility is far more difficult to bound in the increasing utility case due to the great potential of utility increase in that utility model. Utility may also be harder to bound in the nontropical dataset because the finer spatial resolution means there are more observation options to consider when bounding. Additionally, the nadir only algorithm accrues equal utility across utility models in each dataset. This is because with the operational constraints that we set, every observation is outside of the range of the increasing or decreasing effect of the previous observation. We also observe that searches that are more thorough and tests on the nontropical dataset, which has a finer spatial resolution, correspond to increased runtimes, as expected. Finally, we observe that less utility is accrued in the nontropical dataset overall, which is likely due to differences in data volume across the two datasets.

VIII. CONCLUSIONS

Making the most of a limited number of satellite observations is a difficult and important problem. Dynamic Targeting is an approach to this challenge that leverages information from a lookahead sensor to schedule valuable observations for the primary sensor of the satellite. We generalize this concept with a framework capable of accounting for slewing constraints and dynamic utility models. We also implement and evaluate scheduling algorithms, including a greedy algorithm and a depth-first search algorithm, that take these conditions into account. Through this work, we have

TABLE I

COMMON RUNTIME PARAMETERS ACROSS TESTING

Parameter	Value	Description
Primary Sensor Range	15	Max off-nadir angle for primary sensor, from SMICES [3]
Lookahead Sensor Range	45	Off-nadir angle for lookahead sensor, from SMICES [3]
Power Recharge Rate	1%/cycle	Rate at which battery power increases
Power Discharge Rate	2%/ observation	Rate at which battery power decreases during cycles when an observation is made
Gaussian Kernel Size	75 km × 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 (Pitch Axis)	1.08 /s^2	The angular acceleration of slewing along the pitch axis. The value we use is realistic for small satellites [15]
Angular Acceleration (Roll Axis)	1.08 /s^2	The angular acceleration of slewing along the roll axis. The value we use is realistic for small satellites [15]
Maximum Angular Velocity (Pitch Axis)	5.40 /s	The maximum angular velocity of slewing along the pitch axis. The value we use is realistic for small satellites [15]
Maximum Angular Velocity (Roll Axis)	5.40 /s	The maximum angular velocity of slewing along the roll axis. The value we use is realistic for small satellites [15]
Cycle Time	4 s	Amount of time allocated for each cycle
Processing Time	0.1 s	Amount of time allocated for observation acquisition and processing
Satellite Orbit Velocity	7.5 km/s	Velocity of satellite orbit. The value we use is realistic for satellites in low Earth orbit [16]
Satellite Orbit Altitude	800 km	Altitude of satellite orbit. The value we use is realistic for satellites in low Earth orbit [16]
Maximum Utility Factor	4	Maximum value of the utility multiplier in the increasing utility case, or the reciprocal of the minimum utility multiplier in the decreasing utility case
Maximum Off-Nadir Penalty	20%	Penalty for an observation as far off-nadir as possible

TABLE II

PARAMETER SELECTION RESULTS

Dataset	Utility Model	Depth	Breadth
Tropical	Increasing	2	5
Tropical	Decreasing	6	2
Nontropical	Increasing	2	3
Nontropical	Decreasing	4	2

TABLE III

MAXIMUM CYCLE RUNTIMES DURING PARAMETER SELECTION

Dataset	Utility Model	Maximum Runtime (ms)
Tropical	Increasing	81.7
Tropical	Decreasing	425
Nontropical	Increasing	928
Nontropical	Decreasing	1340

TABLE IV

UTILITY ACCRUED BY EACH ALGORITHM ON THE TEST DATASETS

	Tropical Dataset, Increasing Utility	Tropical Dataset, Decreasing Utility	Nontropical Dataset, Increasing Utility	Nontropical Dataset, Decreasing Utility
NO	24851	24851	3383	3383
G	43462	38470	5433	5284
DFS	47274	42131	5706	5565
UB	183105	53745	50928	15371

improved the state of the DT concept by accounting for more realistic satellite operational constraints and utility models, thus bringing it closer to deployment on real missions.

IX. FUTURE WORK

In future work, we aim to try more approaches to scheduling the best observations possible. These approaches include more search algorithms such as a beam search and a Monte Carlo tree search. We also plan to implement a reinforcement learning approach to scheduling observations. Furthermore, one inefficiency is that the maximum runtimes are far greater than the mean runtimes for our depth-first search algorithm. This means that a lot of time that could be spent slewing is often being wasted because the algorithm finishes running in much less time than it is allotted. We intend to address this inefficiency by spending any surplus computational time considering other paths to see if we can beat the original path. Additionally, we are working on developing algorithms for tighter upper bounds on utility, especially for our increasing utility model. We also aim to further generalize this framework to DT with 3D data that accounts for altitude as well. Finally, there are currently projects in development that would fly our DT software on real low Earth orbit satellites.

TABLE V

MEAN RUNTIME FOR EACH ALGORITHM ON THE TEST DATASETS (MS/CYCLE)

	Tropical Dataset, Increasing Utility	Tropical Dataset, Decreasing Utility	Nontropical Dataset, Increasing Utility	Nontropical Dataset, Decreasing Utility
NO	0.198	0.191	15.1	15.1
G	0.672	0.640	46.9	45.1
DFS	9.37	56.6	246	126
UB	0.224	0.221	9.26	7.31

REFERENCES

[1] A. Candela, J. Swope, and S. Chien, "Dynamic Targeting to Improve Earth Science Missions," *Journal of Aerospace Information Systems*, vol. 20, no. 11, pp. 679–689, 2023. [Online]. Available: https://doi.org/10.2514/1.I011233

[2] A. Candela, J. Swope, S. Chien, H. Su, and P. Tavallali, "Dynamic Targeting for Improved Tracking of Storm Features," in *International Geoscience and Remote Sensing Symposium (IGARSS 2022)*, Kuala Lumpur, Malaysia, July 2022. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/IGARSS-2022-Candela-DT.pdf

[3] J. Swope, S. Chien, X. Bosch-Lluis, Q. Yue, P. Tavallali, M. Ogut, I. Ramos, P. Kangaslahti, W. Deal, and C. Cooke, "Using Intelligent Targeting to increase the science return of a Smart Ice Storm Hunting Radar," in *International Workshop on Planning & Scheduling for Space (IWPSS)*, July 2021. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/Swope-SMICES-targeting-IWPSS-2021.pdf

[4] S. Chien, J. Swope, Q. Yue, J. Lluis-Bosch, and W. Deal, "Using a Digital Twin Weather Research and Forecasting (WRF) Model for Machine Learning of Deep Convective Ice Storms," in *Proceedings of the Fall Meeting of the American Geophysical Union*. Washington, DC, USA: American Geophysical Union, 2021. [Online]. Available: https://agu.confex.com/agu/fm21/meetingapp.cgi/Paper/804752

[5] D. R. Thompson, R. O. Green, D. Keymeulen, S. K. Lundeen, Y. Mouradi, D. C. Nunes, R. Castaño, and S. A. Chien, "Rapid Spectral Cloud Screening Onboard Aircraft and Spacecraft," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 52, no. 11, pp. 6779–6792, 2014.

[6] C. Schwartz, I. Sander, R. Jordão, F. Bruhn, M. Persson, J. Ekblad, and C. Fuglesang, "On-board satellite data processing to achieve smart information collection," in *Optics, Photonics and Digital Technologies for Imaging Applications VII*, P. Schelkens and T. Kozacki, Eds., vol. 12138, International Society for Optics and Photonics. SPIE, 2022, p. 121380I. [Online]. Available: https://doi.org/10.1117/12.2620955

[7] D.-Y. Liao and Y.-T. Yang, "Satellite imaging order scheduling with stochastic weather condition forecast," in *2005 IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, 2005, pp. 2524–2529 Vol. 3.

[8] G. Beaumet, G. Verfaillie, and M.-C. Charmeau, "FEASIBILITY OF AUTONOMOUS DECISION MAKING ON BOARD AN AGILE EARTH-OBSERVING SATELLITE," *Computational Intelligence*, vol. 27, no. 1, pp. 123–139, 2011. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.2010.00375.x

[9] C. Zhang, L. Yuan, M. Xie, S. Zhang, and J. Li, "Autonomous mission planning of earth observation satellite based on onboard cloud detection," *Advances in Space Research*, vol. 70, no. 8, pp. 2178–2194, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0273117722005816

[10] H. Suto, F. Kataoka, N. Kikuchi, R. O. Knuteson, A. Butz, M. Haun, H. Buijs, K. Shiomi, H. Imai, and A. Kuze, "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*, vol. 14, no. 3, pp. 2013–2039, 2021. [Online]. Available: https://amt.copernicus.org/articles/14/2013/2021/

[11] S. Chien, R. Sherwood, D. Tran, B. Cichy, G. Rabideau, R. Castano, A. Davis, D. Mandl, S. Frye, B. Trout, S. Shulman, and D. Boyer, "Using Autonomy Flight Software to Improve Science Return on Earth Observing One," *Journal of Aerospace Computing, Information, and Communication*, vol. 2, no. 4, pp. 196–216, 2005. [Online]. Available: https://doi.org/10.2514/1.12923

[12] S. Chien and M. Troesch, "Heuristic Onboard Pointing Re-scheduling for an Earth Observing Spacecraft," in *International Workshop on Planning & Scheduling for Space (IWPSS)*, Buenos Aires, Argentina, 2015. [Online]. Available: https://ai.jpl.nasa.gov/public/papers/chien_iwpss2015_heuristic.pdf

[13] A. Candela, J. Swope, and S. Chien, "Dynamic Targeting for Cloud Avoidance to Improve Science of Space Missions," in *16th Symposium on Advanced Space Technologies in Robotics and Automation*, June 2022. [Online]. Available: https://ai.jpl.nasa.gov/public/documents/papers/Candela-DT-ASTRA-2022.pdf

[14] Z. Hasnain, J. Mason, J. Swope, J. Vander Hook, and S. Chien, "Agile Spacecraft Imaging Algorithm Comparison for Earth Science," in *International Workshop on Planning & Scheduling for Space (IWPSS)*, July 2021. [Online]. Available: ai.jpl.nasa.gov/public/papers/Hasnain_IWPSS2021_paper_13.pdf

[15] V. Lappas, W. Steyn, and C. Underwood, "Attitude control for small satellites using control moment gyros," *Acta Astronautica*, vol. 51, no. 1, pp. 101–111, 2002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0094576502000899

[16] The European Space Agency. (2020, Mar) Types of Orbits. Online. Accessed August 26, 2023. [Online]. Available: www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits