# AA228 Final Project
# Agile Earth Observation Satellite Tasking

Emily Bates
*Dept. of Aeronautics and Astronautics*
*Stanford University*
Stanford, USA
ebates2@stanford.edu

Anshuk Chigullapalli
*Dept. of Aeronautics and Astronautics*
*Stanford University*
Stanford, USA
anshuk@stanford.edu

Yuji Takubo
*Dept. of Aeronautics and Astronautics*
*Stanford University*
Stanford, USA
ytakubo@stanford.edu

***Abstract*—This paper develops an on-board autonomous decision-making algorithm for tasking an Earth observation satellite. The problem incorporates model uncertainty in the attitude dynamics (slew angle) during each attempted observation and the rewards associated with each target. Monte-Carlo Tree Search (MCTS) is adopted in a receding horizon manner to enable the spacecraft to efficiently explore the potential target sites while maximizing the total reward. The performances of two MCTS implementations are compared with a greedy baseline policy, where we discovered the effectiveness of considering a long time horizon as the problem becomes a more sparse-reward environment.**

***Index Terms*—Earth observation, remote sensing, spacecraft autonomy, decision-making under uncertainty**

## I. Introduction

As satellite constellation systems keep increasing in size and complexity, further enhancements of on-board decision-making capabilities are needed to limit demands on human operators and ground infrastructure. One such on-board capability is a decision-making system (i.e., autonomy) in a satellite system, in which the satellite makes a reactive decision based on the observations that are collected periodically.

In this paper, the agile tasking of an Earth observation satellite (EOS) is investigated. While most EOS scheduling problems in the real world are solved on the ground and then uplinked to each individual satellite, on-board scheduling would reduce the demands the ground operators would face, especially if the satellite system involves a large number of satellites.

Previous studies have shown successful results in formulating the EOS problem as a Markov decision process (MDP)-based representation, leveraging the robustness of the state uncertainty and model uncertainty with a large action space that Mixed-Integer programming [1] or dynamic programming [2] cannot handle. The first proposal for the usage of MDP in satellite tasking problems was proposed by Bensana et al. [3]. Deep reinforcement learning was adopted in [4], which slightly outperformed the heuristic-based method. Eddy developed semi-MDP, which the state can "jump" multiple time steps with one transition step, successfully reducing the dimensionality of the problem to solve a large-scale scheduling via Depth-first search and Monte Carlo tree search (MCTS) [5]. In order to represent the state-action value function (i.e., Q-function), Herrmann et al. further leverage MCTS to generate data to be regressed in the neural network for the function approximation with a high-fidelity orbital dynamics simulator [6]. This approach to leverage MCTS as a baseline of the algorithm is further investigated and used as a training dataset for reinforcement learning (RL) to accelerate the on-board decision-making capability by order of three [7], [8]. This RL agent is deployed to the multiple spacecraft constellation and the performance was validated in [9]. Additionally, Nazmy et al. addressed shielded RL [10] in order to satisfy safety-critical constraints, which is a huge implication in the spacecraft application.

In this paper, an MCTS algorithm is developed and integrated for a single-agent EOS problem. MCTS is well-suited to this problem because it balances exploration and exploitation while limiting computational complexity for a large action space. Furthermore, it has been proven successful in prior literature. Model uncertainty is included in the attitude dynamics and the rewards associated with each successful observation.

## II. Problem Formulation

In the EOS scheduling problem, a satellite maximizes the sum of rewards associated with observing the Earth's surface. In this formulation, we consider a satellite with a narrow field of view that must point its camera at discrete targets on the Earth's surface to observe them. The observations occur during a single pass over a region of the Earth. We assume a known set of target observation sites, represented by fixed coordinates, where each site has a reward associated with its observation. At each time step, the satellite attempts to make observations and collect the resulting rewards, and the problem asks the agent to maximize the total reward in the given observation window. This problem can be formulated as a finite-horizon MDP.

This study assumes the satellite uses an active slewing capability (changes its attitude) to observe each target, and its attitude control system limits the maximum slew angle per each time step. The satellite can only observe one site at each step, even though multiple sites are within its slew range. There are more potential observation targets than the satellite can possibly observe, requiring it to select an optimal attitude trajectory to visit the most valuable targets.

Rewards vary by target, and each reward is awarded stochastically at the observation time. This represents potential changes in the quality of the observation due to clouds or fog. Furthermore, stochastic errors are included in the attitude due to unexpected perturbations or attitude control system imprecision.

The state and action spaces are represented by:

$$\mathcal{S} : \{\boldsymbol{\alpha}, \theta_C, \theta_T, t, \text{(target list)}, \text{(observed list)}\} \quad (1)$$

$$\mathcal{A} : \{\text{index of target site, 0 if not observing}\} \quad (2)$$

where $\boldsymbol{\alpha}$ represents the Keplerian orbital elements (KOE) of the satellite and $t$ is the time elapsed from the initial state. $\theta_C$ and $\theta_T$ are the angles of the observation vector projected along the cross-track direction and along-track direction, defined with respect to the local nadir direction in the RTN frame. The RTN frame is defined such that unit vector $\hat{R}$ points in the radial direction, $\hat{N}$ points in the anti-orbit normal direction (the direction of the orbit's angular momentum), and $\hat{T}$ points roughly along the velocity direction for a circular orbit. Look vectors are transformed into the RTN frame using a procedure taken from the SatelliteDynamics.jl package. [11]. Likewise, $\theta_C$ is positive in the anti-orbit normal direction and $\theta_T$ is positive in the velocity direction. This definition is illustrated in Fig. 1 with the satellite's RTN frame.

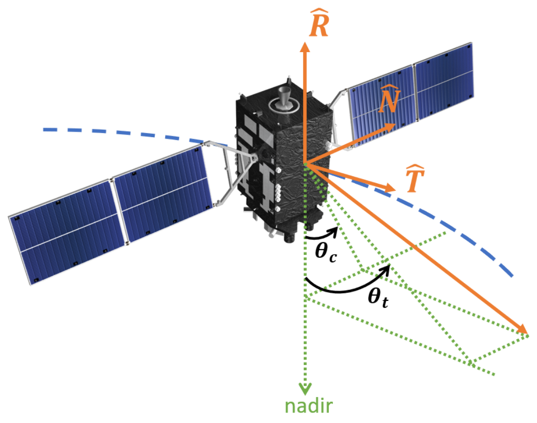

Fig. 1. Satellite attitude geometry and its RTN frame.

The target list is a list of tuples containing the latitude and longitude of the targets, their earth-fixed X, Y, and Z positions, and their mean reward and associated standard deviation. The observed list is a binary vector that indicates if the target site has been observed.

The transition and reward function used in this work is oulined in Algorithm 1. This function determines if a maneuver is permitted, changes the attitude accordingly, determines if the observation is successful, and calculates the resulting reward. Regardless of the attitude manuvers and observation attempt, the time is advanced and the satellite's orbit is propagated at the end of the process.

**Algorithm 1:** Transition and Reward function

```
input : state $s$, action $a$
output: state $s$, reward $R$
$r_{obs}, v_{obs}$ = KoeToCart($s.\boldsymbol{\alpha}$) ;
if $a$ == 0 then
    Maintain attitude;
else
    $r_{tar}$ = ECEFToECI($s.target\_list[a][1:3]$);
    if Target beyond horizon | Target already observed then
        Maintain attitude;
    else if Target beyond slew bounds then
        // Slew as far as possible towards
           target
        $s.\theta_C \leftarrow s.\theta_C + max\_slew +$ rand(Normal($0, \sigma_{slew}$));
        $s.\theta_T \leftarrow s.\theta_T + max\_slew +$ rand(Normal($0, \sigma_{slew}$));
    else if Target inside slew bounds then
        // Slew to target
        $s.\theta_C \leftarrow s.\theta_{c,tgt} +$ rand(Normal($0, \sigma_{slew}$));
        $s.\theta_T \leftarrow s.\theta_{t,tgt} +$ rand(Normal($0, \sigma_{slew}$));
    end
    if Target within FOV then
        $s.observed\_list[a] = 1$;
        $R$ = rand(Normal($\mu_a, \sigma_a$));
    else
        $R = 0$ ;
    end
end
$t \leftarrow t + time\_step$;
$s.\alpha \leftarrow$ Propagate($s.\alpha$);
```

We assume that the attitude control system nominally maintains the satellite's attitude relative to its RTN frame, and it is capable of slewing at an additional fixed rate. As a result, the slew maneuver is constrained by $|\theta_{C,t+1}| \leq \theta_{C,t} + \theta_{C,max}$ and $|\theta_{T,t+1}| \leq \theta_{T,t} + \theta_{T,max}$ where $\theta_{T,max} = \theta_{C,max} = \frac{\text{slew rate}}{\Delta t}$.

If the selected target is within the slew bounds, the satellite will slew directly to the target. If the target is out of bounds, but not past the horizon, the satellite will slew the maximum allowed amount towards the target. Each attitude maneuver also incorporates some small random error in the two attitude angles. After a maneuver, the function checks whether the target falls within the camera field of view (FOV). If it does, the observed list is updated and a reward is calculated. Table I outlines key parameters used in the transition function.

TABLE I
TRANSITION FUNCTION PARAMETERS

| Time step | Slew rate | $\theta_{C,max}$ | $\theta_{T,max}$ | FOV | Slew error |
|---|---|---|---|---|---|
| 30 s | 30°/min | 15° | 15° | 5° | $\mathcal{N}(0, 1°)$ |

The reward function for observation site $i$ is defined as follows. Considering the uncertainty in the image quality (due to clouds, air pollution, etc.), the reward function is expressed as a normal distribution.

$$R_i = \mathcal{N}(\mu_i, \sigma_i) \quad (3)$$

In this formulation, the observation angle from the nadir direction does not affect the reward.

The transition function propagates the orbit based on simple two-body orbital dynamics (i.e., $M(t) = M(0) + \sqrt{\mu/a^3}t$). The initial KOE of the satellite is presented in Table II.

This orbit is designed to represent a sun-synchronous orbit (common with earth-observing satellites) and has a period of 98 minutes. The trajectory is propagated for 2000 seconds. Fig. 2 shows the resulting ground track. The blue dotted line shows the ground track of one orbit (6000s). The area of interest in this observation campaign is shown in the orange parallelogram, which has a width of 20° in longitude and a height of 60° in latitude. The list of the target sites is generated randomly from the uniform distribution of $\mu_r \sim [0, 1]$ and $\sigma_r \sim [0, 0.3]$.

TABLE II
KEPLERIAN ORBITAL ELEMENTS OF THE SATELLITE

| $a$ [km] | $e$ | $i$ [°] | $\Omega$ [°] | $\omega$ [°] | $M$ [°] |
|---|---|---|---|---|---|
| 7057.0 | 0.000879 | 98.12 | 255.09 | 225.37 | 75.0 |

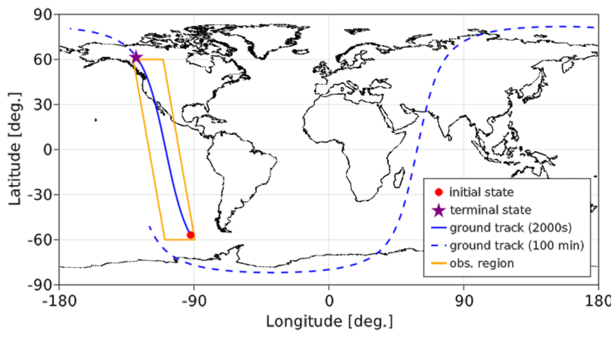

Fig. 2. Ground track of the satellite.

## III. Monte Carlo Tree Search

MCTS is an online sequential planning algorithm that explores a tree structure and solves for an approximate optimal MDP policy [12].

A predefined upper bound in the exploration iteration provides the explicit trade-off between exploration and computational efficiency, which is beneficial for various resource-limited contexts.

The fundamental action policy for the MCTS is the maximization of the following function (UCB-1):

$$Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}}, \quad N(s) := \sum_a N(s,a), \qquad (4)$$

where $c$ is the exploration constant that weighs the utility of exploration to the current Q-function, and $N(s, a)$ is the visit count of the state-action pair $(s, a)$ during the search.

For each iteration of the search, the algorithm chooses the action that maximizes the above function, and then the state is updated using the transition function. At some point, the algorithm goes into the maximum depth or a new state. Then, the visit count and the $Q(s, a)$ are reset to zero, and the estimate of the utility function is returned by performing a rollout to a selected depth $d_r$ before the search is terminated. By repeating this search, the agent finds "promising branches" which are further explored to the deeper layer of the tree. Finally, the action with the highest utility is chosen and run in the outer simulation. The key parameters of the implemented MCTS are shown in Table III.

Due to the potentially large action space of the problem, MCTS may suffer from the curse of dimensionality in a feasible amount of time. So, we tested two versions of the MCTS algorithm that differ in the rollout method:

- **Random rollouts**: To estimate the Q value at the leaf node of the MCTS exploration, the agent chooses random actions for depth $d_r$. The feasibility of the selected action at each time step is not guaranteed.
- **One-step lookahead**: $Q(s, a)$ at the leaf node is estimated with a one-step lookahead, where the action is chosen greedily from among the target sites that are guaranteed to be observable.

Hermann et al. [6] also implement a similar heuristic action search in the rollout at the end of the tree depth.

TABLE III
MCTS PARAMETERS

| Parameter | Value |
|---|---|
| Exploration constant, $c$ | 4000 |
| Search depth, $d$ | 40 |
| Rollout depth after the terminal state, $d_r$ | 15 (random)/ 1 (heuristic) |
| Search Iteration, $M$ | 500 |
| discount rate, $\gamma$ | 0.95 |
| time step [s], $\Delta t$ | 30 |

## IV. Results and Analysis

We compare our MCTS implementation with a baseline greedy policy. The baseline policy is a myopic policy that selects the highest reward target that is accessible from the satellite's current state. This is the same greedy policy is used in the rollout for the MCTS with one-step lookahead.

The MCTS implementations and the baseline policy were tested on datasets with a varying number of target sites for the satellite to image. Within the same region on the Earth's surface, three datasets with 50, 100, and 1000 target points were generated, as shown in Fig. 3. The size of the marker indicates the mean reward expected by observing that target.

Sections IV-A, IV-B, and IV-C present the results of running the different on the datasets. Section IV-D analyzes the trends in the results.

### A. 50 Target Dataset

With the 50 target dataset, we ran the baseline greedy policy, the MCTS with random rollouts, and the MCTS with one-step lookahead. MCTS was implemented with the parameters specified in Table III. The results of these experiments are given in Table IV.

In a sparse layout of targets, the baseline greedy policy struggles to find feasible targets that can provide high rewards because of its myopic strategy. In this scenario, both MCTS implementations outperform the baseline, with the MCTS that uses random rollouts gathering a higher reward and observing

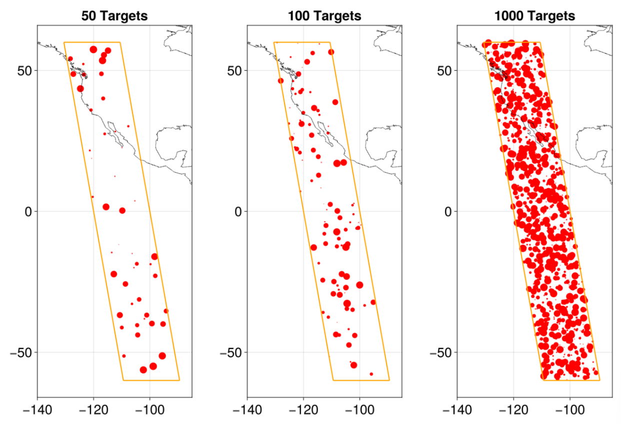

Fig. 3. Distribution of targets in the testing datasets.

TABLE IV
COMPARING AVERAGE RESULTS OF POLICIES ON 50-TARGET DATASET

| Policy | # Runs | $\mu(R)$ | Median sites observed |
|---|---|---|---|
| Baseline (Greedy) | 10 | 8.875 | 17 |
| MCTS (random) | 10 | **17.658** | **31** |
| MCTS (lookahead) | 10 | 13.01 | 19.5 |

more states than the MCTS with one-step lookahead. With only 50 possible actions, the MCTS implementations are able to effectively explore the full action space.

Fig. 4 compares the results of the baseline policy and MCTS with random rollouts.

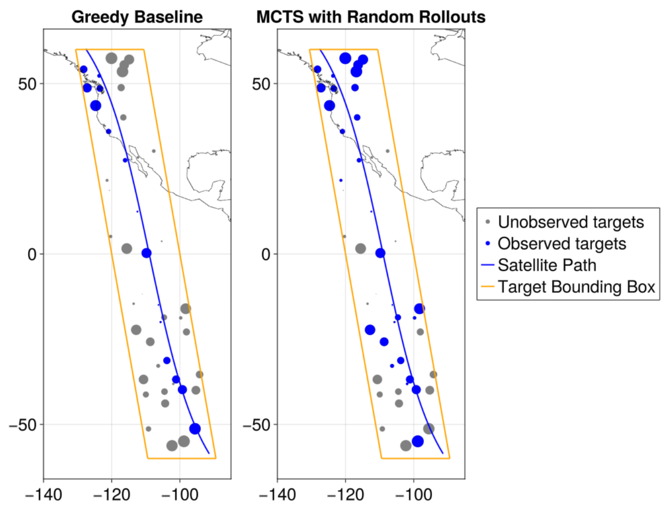

Fig. 4. Observed target sites from the baseline and MCTS policy for the 50 target dataset.

## B. 100 Target Dataset

As with the 50 target dataset, we ran the baseline greedy policy, the MCTS with random rollouts, and the MCTS with the one-step lookahead for the 100 target dataset using the MCTS parameters specified in Table III. The results of these experiments are shown in Table V.

TABLE V
COMPARING AVERAGE RESULTS OF POLICIES ON 100-TARGET DATASET

| Policy | # Runs | $\mu(R)$ | Median no. of sites observed |
|---|---|---|---|
| Baseline (Greedy) | 10 | 15.422 | 31.5 |
| MCTS (random) | 10 | **26.237** | **44.0** |
| MCTS (lookahead) | 5 | 24.019 | 40.0 |

The results of the 100 target simulations closely resemble those of the 50 target simulations, with MCTS with random rollouts performing the best. However, the advantage of running MCTS over the baseline policy decreased: using MCTS with random rollouts gave a reward that is 1.7 times larger than the baseline reward, while MCTS in the 50 target simulation doubled both the reward and the median number of targets observed. MCTS cannot explore as many potentially useful portions of the decision tree with a larger action space and fixed search iterations.

Notably, MCTS with the one-step lookahead rollout improves in performance from the 50-target simulation, almost matching the performance of MCTS with random rollouts.

Fig. 5 compares the results of the baseline policy and the MCTS with random rollouts.

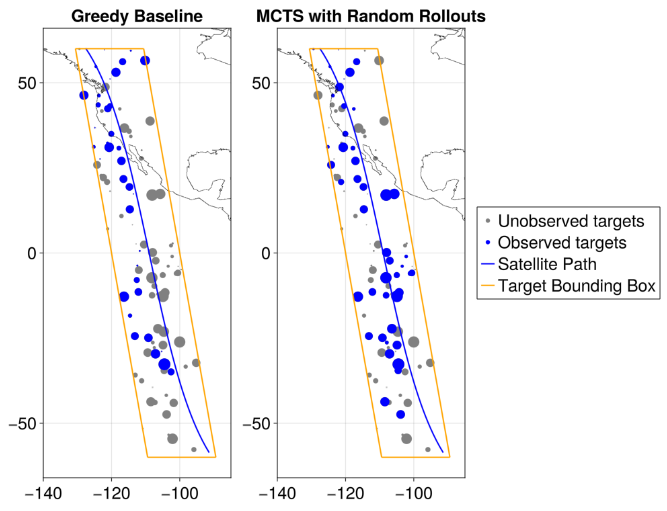

Fig. 5. Observed target sites from the baseline and MCTS policy for the 100 target dataset.

## C. 1000 Target Dataset

To test the limits of our MCTS implementation, we ran the policies against a dataset with 1000 targets, still using the MCTS parameters specified in Table III. The results of these experiments are given in Table VI.

TABLE VI
COMPARING AVERAGE RESULTS OF POLICIES ON 1000-TARGET DATASET

| Policy | # Runs | $\mu(R)$ | Median no. of sites observed |
|---|---|---|---|
| Baseline (Greedy) | 10 | 33.261 | **66** |
| MCTS (random) | 5 | 17.739 | 23 |
| MCTS (lookahead) | 2 | **49.152** | 59.5 |

Only two runs were performed for MCTS with one-step lookahead due to the large computational effort taken to calculate the feasible greedy action at every rollout.

The greedy policy begins to perform significantly better with the high density of the large dataset. This is because it almost always has a new reachable target within its slew range at every time step. However, due to its myopic nature, it does not maximize the total reward.

The MCTS implementations only search through the decision tree 500 times, and therefore they explore at most 500 possible root actions. Therefore, these implementations only utilize around half of the action space and the MCTS with random rollouts performs poorly in this situation. It has the lowest final reward and a low number of targets observed.

MCTS with one-step lookahead, however, captures a higher total reward despite of the slightly fewer targets observed than the baseline greedy policy. It appears that the one-step lookahead step provided a better approximation of $Q$ function at the root nodes than the random rollouts, allowing more effective planning. The downside of this method is the increase in the computation time, where the current implementation takes 60 times longer than the baseline.

Fig. 6 compares the results of the baseline policy and the MCTS with the one-step lookahead rollout.

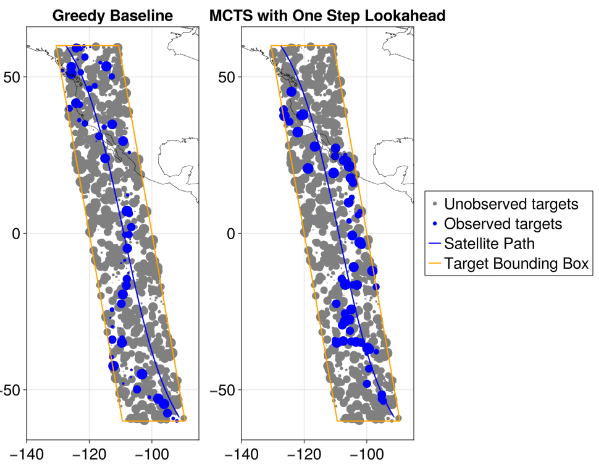

Fig. 6. Observed target sites from the baseline and MCTS policies for the 1000 target dataset.

## D. Analysis of Results

The trends in the performance of different policies with varied dataset sizes are summarized in Fig. 7; the number of targets observed in each experiment is also normalized in the same manner and shown in Fig. 8. The rewards from Tables IV, V, and VI are normalized (divided by the L-2 norm of the sum of the rewards across the three policies) in the figures, and the additional results from datasets of sizes 20, 30, and 200 are also presented in the figures.

In scenarios with smaller datasets, the MCTS with random rollouts earns the highest reward with a policy that observes the most number of targets. This is because a sparse reward environment with fewer target sites requires the agent to consider a longer time horizon to reach the rewards, thereby making the greedy heuristic ineffective, whereas the random rollout policy effectively explores the action space. The MCTS with one-step lookahead strikes a balance between the two by using MCTS to explore decision trees but using leaf node action values estimated purely from the greedy policy.

On the contrary, the greedy policy steadily improves in its performance (relative to the other two methods) with an increase in the size of the dataset, as shown in Fig 7. This is because the greedy policy usually finds access to feasible high-reward targets in the environment where high-reward target sites are densely populated. This diminishes the relative value of foreseeing the long time horizon.

A similar effect is observed with the MCTS with a one-step lookahead. As the density of the dataset increases, the estimate for the $Q$ function from the one-step lookahead improves in quality. This is because choosing nearby feasible actions with high rewards becomes a better estimate of utility in the denser environment. Eventually, the quality of this estimate surpasses that of the estimate for the $Q$ function generated from random rollouts, which perform no such check for feasibility. In Fig 7, we see that this occurs around the target space of size 200.

Both MCTS with one-step lookahead and MCTS with random rollouts face the issue of not iterating through enough decision trees to explore the entire action space in the 1000 target space dataset. However, because the one-step lookahead with feasible actions becomes a better approximator of the utility than the random rollouts, MCTS with one-step lookahead delivers better results. The greedy policy also picks feasible targets at every time-step, but lacks the added advantage of first exploring various decision trees that is contributed by MCTS.

With larger computation resources, it could be possible to increase the number of decision tree iterations run by MCTS and improve its results. Alternatively, one could employ a strategy of down-selecting the allowed actions in the MCTS tree to ones that are reachable in each time step.

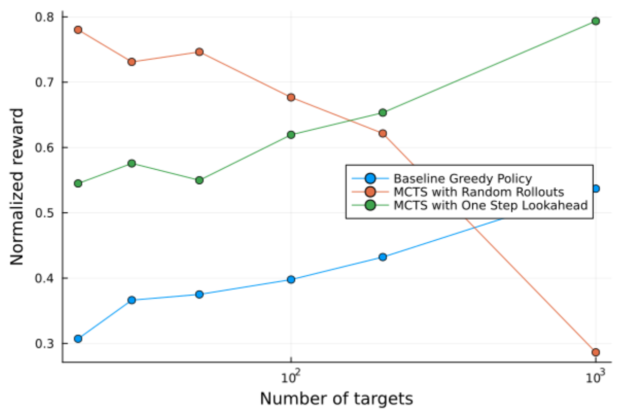

Fig. 7. Final rewards of the tested policies on different target space sizes.

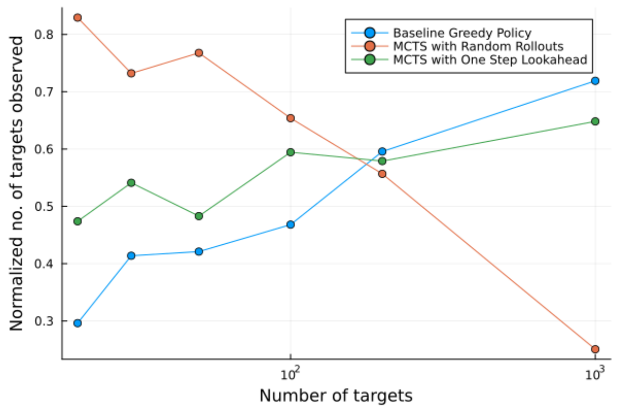

Fig. 8. Normalized total number of observed targets of the tested policies on different target space sizes.

## V. Conclusions and Future Work

The on-board EOS scheduling problem for a single satellite is developed, and the performance of two implementations of the MCTS algorithm is compared with a baseline greedy policy. The uncertainty in attitude dynamics and the target site reward is considered and is represented in the transition function of the MDP formulation.

We observe trends in the performance of the policies tested, over different sizes of the potential target/action spaces. Notably, the MCTS with random rollouts surpasses the other planning algorithms in smaller target number situations (i.e., sparse reward environment), but was unable to deliver results with larger target spaces where it could not explore the complete action space. A myopic one-step lookahead at the MCTS's leaf node is shown to improve the results by forcing feasible actions.

Future work includes the sectioning of the larger action spaces by additional heuristics. This would help reduce computation time when running MCTS with heuristic rollouts, while allowing for more exploration of potential actions. Other future directions include the introduction of a partially observable environment and the application of the cooperative multi-agent context.

## VI. Team Member Contributions

Emily Bates implemented and validated the transition and reward function along with supporting geometry calculations. Anshuk Chigullapalli integrated the custom formulation into an MCTS architecture and identified effective MCTS parameters. Yuji Takubo set up the problem scenario (orbits, test data, and ground track plotting) and developed the initial implementation of MCTS. All members contributed equally to the final report and analysis of the results.

## VII. Project Code

The code repository for the work in this paper can be found at https://github.com/GigaVoltFlash/AA228-Final-Project.

## References

[1] S. Spangelo, J. Cutler, K. Gilson, and A. Cohn, "Optimization-based scheduling for the single-satellite, multi-ground station communication problem," *Computers & Operations Research*, vol. 57, pp. 1–16, 2015.

[2] S. Nag, A. S. Li, and J. H. Merrick, "Scheduling algorithms for rapid imaging using agile cubesat constellations," *Advances in Space Research*, vol. 61, no. 3, pp. 891–913, 2018.

[3] E. Bensana, G. Verfaillie, C. Michelon-Edery, and N. Bataille, "Dealing with uncertainty when managing an earth observation satellite," in *Artificial Intelligence, Robotics and Automation in Space*, vol. 440, 1999, p. 205.

[4] A. Hadj-Salah, R. Verdier, C. Caron, M. Picard, and M. Capelle, "Schedule earth observation satellites with deep reinforcement learning," *arXiv preprint arXiv:1911.05696*, 2019.

[5] D. Eddy, "Task planning for earth observing satellite systems," Ph.D. dissertation, Stanford University, 2021.

[6] A. P. Herrmann and H. Schaub, "Monte carlo tree search methods for the earth-observing satellite scheduling problem," *Journal of Aerospace Information Systems*, vol. 19, no. 1, pp. 70–82, 2022.

[7] A. Herrmann and H. Schaub, "Reinforcement learning for the agile earth-observing satellite scheduling problem," *IEEE Transactions on Aerospace and Electronic Systems*, 2023.

[8] ——, "A comparison of deep reinforcement learning algorithms for earth-observing satellite scheduling," in *AAS Spaceflight Mechanics Meeting, Austin, TX*, 2023, pp. 23–116.

[9] A. Herrmann, M. A. Stephenson, and H. Schaub, "Single-agent reinforcement learning for scalable earth-observing satellite constellation operations," *Journal of Spacecraft and Rockets*, pp. 1–19, 2023.

[10] I. Nazmy, A. Harris, M. Lahijanian, and H. Schaub, "Shielded deep reinforcement learning for multi-sensor spacecraft imaging," in *2022 American Control Conference (ACC)*, 2022, pp. 1808–1813.

[11] D. Eddy, "Satellitedynamics.jl." [Online]. Available: https://sisl.github.io/SatelliteDynamics.jl/latest/

[12] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for decision making*. MIT press, 2022.