(Preprint) AAS XX-XXX

REINFORCEMENT LEARNING FOR THE MULTI-SATELLITE EARTH-OBSERVING SCHEDULING PROBLEM

Adam Herrmann*, João Vaz Carneiro† and Hanspeter Schaub‡

This work explores the use of reinforcement learning (RL) for the multi-satellite, multi-target Earth-observing satellite (EOS) scheduling problem. In this work, a decision-making agent is trained independently in a stationary, single satellite environment on a fixed number of imaging targets that are updated as they are imaged and downlinked by the spacecraft. The trained agent is then deployed in a non-stationary, multi-satellite scenario where different spacecraft can share imaging targets. The distributed nature of the architecture simplifies the training process and required training time but requires cross-communication between satellites and is inherently suboptimal in terms of global reward due to the competition between agents. The prospect of this method is that the constellation design and size can scale up and down without requiring the decision-making agent be retrained. This method is evaluated and benchmarked for various Walker-delta constellations and target numbers. Furthermore, extrapolation of the trained agents to orbits outside of the training distribution is explored. An increase in spacecraft leads to an increase in global reward, but a decrease in reward per spacecraft due to the competition among agents. Furthermore, local reward is highest for orbits with a longitude of the ascending node close to the training distribution.

INTRODUCTION

Large Earth-orbiting satellite constellations require on-board planning and scheduling to reduce the burden on spacecraft operators and decrease the cost of operations. Reinforcement learning has been shown to be a viable method for developing autonomous decision-making agents for on-board planning and scheduling.[1–4] In prior work, the single satellite Earth-observing satellite (EOS) scheduling problem is cast as a Markov decision problem, and various algorithms are applied to generate near-optimal policies. However, the scalability of these methods to multi-satellite scenarios has not been demonstrated. The primary challenge of scaling reinforcement learning to multiple decision-making agents is non-stationarity. Multi-agent environments with limited to no communication between agents appear non-stationary to individual agents because other agents change the environment with their actions. The problem may be cast as a decentralized partially observable Markov decision process (Dec-POMDP) to account for the uncertainty in the environment due to other agents. Several multiagent robotics problems using macro-actions have demonstrated the

*PhD Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80309. AIAA Member.

†Masters Student, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, Boulder, CO, 80309. AIAA Member.

‡Professor, Glenn L. Murphy Chair in Engineering, Ann and H.J. Smead Department of Aerospace Engineering Sciences, University of Colorado, Boulder, 431 UCB, Colorado Center for Astrodynamics Research, Boulder, CO, 80309. AAS Fellow, AIAA Fellow.

success of such an approach.[5–7] However, finding an optimal solution for a finite-horizon Dec-POMDP is NEXP-complete.[8] If free communication between agents and full observability of the environment is assumed, the Dec-POMDP can be reduced to a multiagent Markov decision process (MMDP).[9] An MMDP is an MDP for multiagent systems in which there is a set of agents and a joint action space.[10] The state space, reward function, and transition function are a function of the joint action space. Finding an optimal solution for a finite horizon MDP is only P-complete,[11] but an MMDP joint action space is exponential in the number of agents. To avoid the computation required to solve a Dec-POMDP or MMDP formulation of the multi-satellite Earth-observing satellite scheduling problem, this work first formulates the problem as a single agent Markov decision process for training. During deployment in the environment, cross-link communication between spacecraft is utilized to communicate which ground targets have and have not been imaged, thereby providing agents with the ability to maintain a collective belief state. By training agents in a stationary, single-satellite spacecraft environment and deploying them in a non-stationary, multi-spacecraft environment with cross-link communication, the required training time is dramatically reduced. However, the agents are in competition with one another for the shared targets because they are trained using local reward functions that do not independently sum together to make up the global reward function. Therefore, the resulting global reward is inherently suboptimal, but the degree to which it is suboptimal is dependent on how sparse the targets are and how many spacecraft are in the constellation. As the number of targets increases and the number of spacecraft decreases, the competition decreases.

This paper begins by formulating the single satellite EOS scheduling problem as a Markov decision process. The training process which utilizes Monte Carlo tree search and supervised learning is shown. The nuances of applying the single-satellite MDP and training pipeline to a multi-satellite environment is then discussed. Specifically, the Walker-delta constellation design,[12] assumptions of the cross-link communication between spacecraft, generation of the ground targets, and multi-satellite simulation architecture used to model the problem are explained. Finally, the performance of the trained agents for different Walker-delta constellation designs is explored. The performance of the agents for different longitude of ascending nodes is also discussed.

SINGLE SATELLITE MARKOV DECISION PROCESS

In the Earth-observing satellite (EOS) scheduling problem, one or more spacecraft collect and downlink data to one or more ground stations over the specified planning interval, which is defined as the total amount of time considered for planning and scheduling. An example of the multi-satellite EOS scheduling problem may be found in Figure 1. For training purposes in this work, a single-satellite EOS scheduling problem is formulated where a satellite orbiting the Earth must choose between imaging different ground-based targets, charging its batteries, desaturating its reaction wheels, or downlinking data to any available ground stations. Over the duration of the planning horizon, the satellite has a set of targets along its flight path available for imaging. This set of targets is referred to as $\mathbf{T}$. Each target has its own priority, one being the highest priority and three being the lowest. The goal is to maximize the weighted sum of the targets in $\mathbf{T}$ imaged and downlinked over the planning horizon while minimizing resource management failures.

The EOS scheduling problem is formulated as a Markov decision process, a sequential decision-making process where an agent follows a mapping from states to actions, known as the policy $\pi : \mathcal{S} \times \mathcal{A}$, to take an action a_i in state s_i until the end of the planning horizon or early termination. The agent transitions to a new state s_{i+1} following a transition function $T(s_{i+1}|s_i, a_i)$, which is

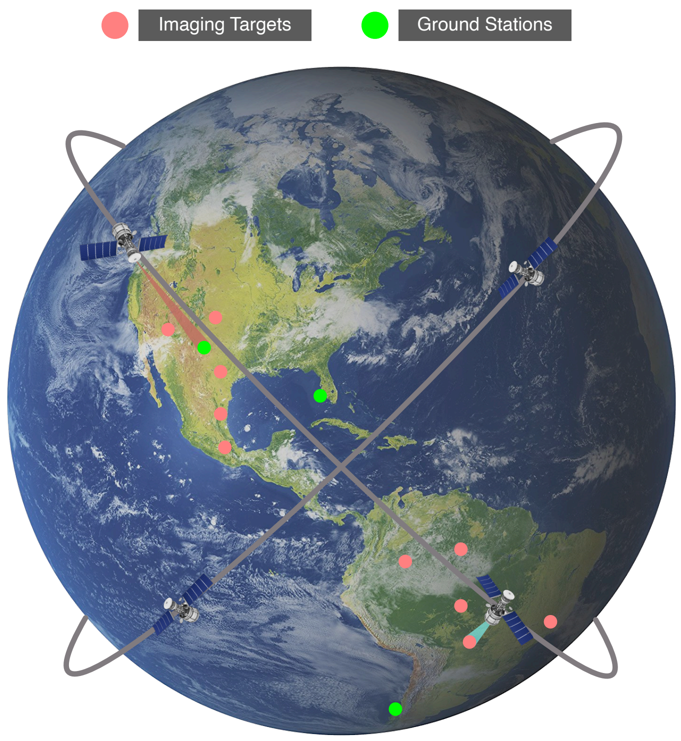


Figure 1: **Multi-satellite Earth-observing scheduling problem.**

the conditional probability of transitioning to state s_{i+1} given s_i and a_i. MDPs follow what is known as the Markov assumption, which states that the next state is conditioned on the current state and action alone. This is represented as $T(s_{i+1}|s_i, a_i) = T(s_{i+1}|s_i, a_i, s_{i-1}, a_{i-1}, ..., s_0, a_0)$. A generative transition function is one in which the next state is sampled from a distribution or computed via simulation, $s_{i+1} \sim G(s_i, a_i)$. A generative model is used in this work because of the difficulty in representing a complex spacecraft simulation using discrete probabilities. At each state, the agent receives a reward r_i based on the reward function $R : \mathcal{S} \times \mathcal{A} \to \mathcal{R}$. The goal is to maximize the sum of reward over the planning horizon. The described process is depicted in Figure 2.

In the EOS scheduling problem, hundreds of ground targets may be considered over a day of operations. However, the relative geometry of the targets and limited spacecraft resources prevent the collection of every target. As a result, the agent must make trade-offs between spacecraft resources and ground targets to maximize the total reward. In order to adhere to the Markov assumption, each target should be included in the state and action space. The problem with including each target in the state and action space is the increase in the wall-clock time required to solve the problem. A problem formulated using hundreds of targets can take days to solve for because of the problem complexity and the use of a high-fidelity astrodynamics simulation. Only a subset of the targets in $\mathbf{T}$ are available for imaging at each time step, though. In order to simplify the problem and approximately adhere to the Markov assumption, a subset of $\mathbf{T}$ that contains the next upcoming, unimaged targets are considered at each step. This subset is given in Equation 1, where J is the total number of targets in the state and action space and $\mathbf{D}$ is a subset of $\mathbf{T}$ containing the imaged or passed targets.

$$\mathbf{U} = \{c_j \in (\mathbf{T} - \mathbf{D}) \mid \forall\, j \in [1, J]\} \tag{1}$$

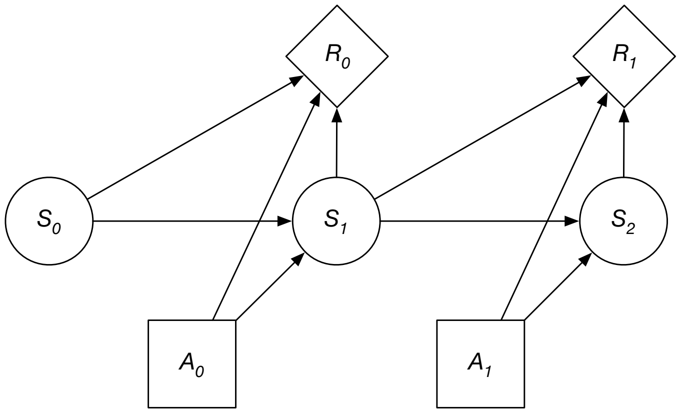


Figure 2: **Markov decision process.**[4]

State Space

The state space for the single-satellite EOS scheduling problem may be found below. The state space is designed to balance adherence to the Markov assumption and the total dimensionality of the state space.

- ECEF spacecraft position, ${}^{\mathcal{E}}\mathbf{r}$
- ECEF spacecraft velocity, ${}^{\mathcal{E}}\mathbf{v}$
- Image tuples for targets $c_j \in \mathbf{U}$
  - Target position in the spacecraft Hill frame, ${}^{\mathcal{H}}\mathbf{r}_j$
  - Priority, p_j
  - Imaged indicator, w_j
  - Downlinked indicator, d_j
- L^2 norm of attitude error, $||\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}||$
- L^2 norm of attitude rate, $||{}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}||$
- Reaction wheel speeds, $\boldsymbol{\Omega}$
- Battery charge, z
- Eclipse indicator, k
- Stored data in buffer, b
- Data transmitted, h
- Ground station access indicator g_i, $i = 1 : M$

The left superscript on a vector denotes the coordinate frame in which vector components are expressed.[13] The body-fixed frame is referred to as $\mathcal{B}$. The Earth-centered, Earth-fixed (ECEF) position and velocity of the spacecraft provide state information for upcoming downlink windows. The position of each target in the satellite's Hill frame provides spacecraft-relative state information for each target. A target priority, imaged indicator, and downlinked indicator are also provided for each target in order to provide reward-specific state information.

Several states are included for spacecraft resource management. The L^2 norm of the attitude error, L^2 norm of the attitude rate, and reaction wheel speeds provide state information for management of the attitude control subsystem. The battery charge and eclipse indicator provide state information for resource management of the power subsystem. Likewise, the data stored in the buffer, data transmitted, and ground station access indicators provide state information for the on-board data management subsystem.

For the purposes of function approximation, each state is normalized to a range of $[-1, 1]$ or $[0, 1]$. The normalization of each state variable is performed such that minimal state information is lost.

Action Space

This work takes a mode-based planning approach where low-level behaviors are dictated by high-level abstractions of actions. The mode-based planning approach is identical to the macro-actions described in the aforementioned Dec-POMDP formulations of robotics problems. Similarly, by discretizing the actions, the problem becomes tractable to solve in an MDP formulation. In this problem, the low-level behavior of each mode is dictated by the attitude reference and which spacecraft subsystems are on or off. The mode-based planning approach discretizes continuous behavior into discrete modes that are taken at each step through the environment. The spacecraft modes modeled are given below, which comprise the action space $\mathcal{A}$:

- Charge
- Desaturate
- Downlink
- Image target $c_1 \in \mathbf{U}$

$$\vdots$$

- Image target $c_j \in \mathbf{U}$

In the charge mode, the spacecraft turns off all instruments and transmitters and points its solar panels at the sun. In the desaturation mode, the spacecraft maintains a sun-pointing attitude reference and maps reaction wheel momentum to thruster firings. In the downlink mode, the spacecraft points in the nadir direction and turns on the transmitter. If a ground station is accessible, which is determined by a simple range and elevation model, the spacecraft downlinks data. In the imaging mode, the spacecraft points at target c_j. If the target is within elevation, range, and attitude error requirements, the spacecraft takes the image and the data is added to the on-board data buffer.

Transition Function

A generative transition function is implemented using the Basilisk astrodynamics software architecture which is an open-source software architecture for high-fidelity spacecraft simulations.[14] The Basilisk simulation is wrapped within a Gym environment. Gym provides a standard interface agents can interact with during training and deployment. The decision-making agent passes an action to the Gym environment, which turns the relevant models on or off and runs the simulation for a specified amount of time at each step. This process is depicted in Figure 3.

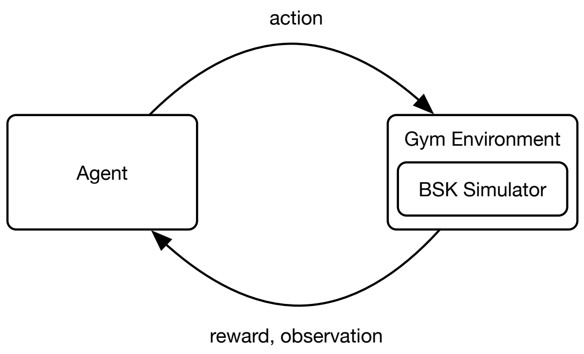


Figure 3: **Gym environment interface.**[4]

Reward Function

The reward function $R(s_i, a_i, s_{i+1})$ is given in Equation 2.

$$r_i = \begin{cases} -10 & \text{if failure} \\ \sum_j^{|\mathbf{T}|} H(d_j) & \text{if } \neg\text{failure} \wedge a_i \text{ is downlink} \\ 0.1H(w_j) & \text{if } \neg\text{failure} \wedge a_i \text{ is image } c_j \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

If the agent fails to manage its resource during the step, a failure penalty of -10 is returned and the episode terminates. The failure condition is true if the agent drains the spacecraft's batteries, exceeds the maximum reaction wheel speeds, or overflows the data buffer. Failure is computed as follows:

$$\text{failure} = (z = 0 \ \vee \ \text{any}(\hat{\mathbf{\Omega}} \geq 1) \ \vee \ b \geq 1) \tag{3}$$

If a failure does not occur and the downlink mode was initiated, each ground target in $\mathbf{T}$ is checked to determine if it was downlinked for the first time. A function $H(x_j)$ is created to check if a state variable x_j is false at step i and true at step $i+1$, returning one divided by the ground target's priority if true. This function is applied to the downlinked flag, and is given as follows:

$$H(x_j) = (1/p_j) \text{ if } \neg x_{j_i} \ \wedge \ x_{j_{i+1}} \tag{4}$$

If a failure does not occur and an image of a ground target was attempted, the same function is applied to the imaged flag. This reward is multiplied by a scaling factor of 0.1 to provide a small positive reward in the short term for imaging targets. This reward engineering aids convergence during training by making the reward less sparse.

TRAINING PROCESS

Solving Markov Decision Processes

The goal of a Markov decision process is to maximize the sum of future reward with each action taken. To solve a Markov decision process, a policy must be computed that maximizes the sum of

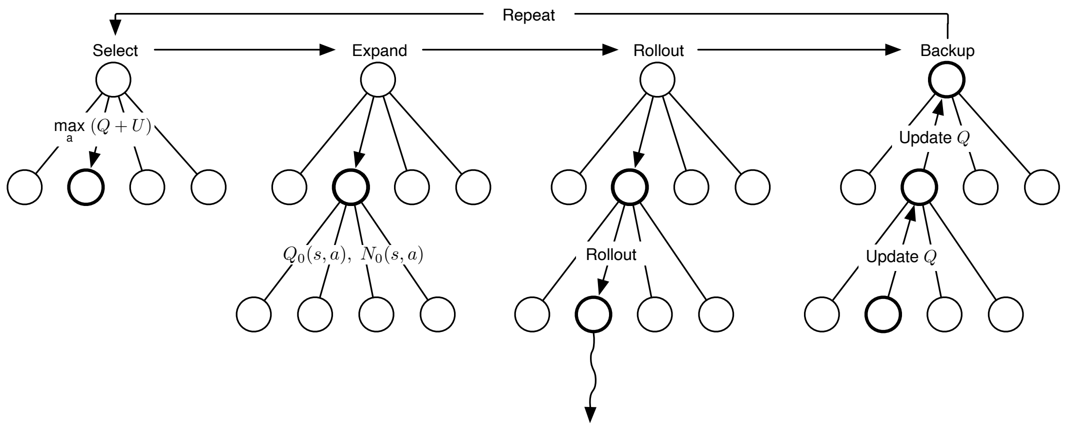


Figure 4: **Monte Carlo tree search.**[4]

future reward. This is known as the optimal policy, $\pi^*(s_i)$. The optimal policy can be computed as a function of the optimal value function, $V^*(s_i)$, which is the expected value of all future reward when following the optimal policy until the episode terminates.

$$\pi^*(s_i) = \arg\max_{\pi} V^{\pi}(s_i) \tag{5}$$

$$V^*(s_i) = \max_{\text{a}} \Big(R(s_i, a_i) + \gamma \sum_{s_{i+1} \in \mathcal{S}} T(s_{i+1} | s_i, a_i) V^*(s_{i+1}) \Big) \tag{6}$$

Instead of solving for $\pi^*(s_i)$ or $V^*(s_i)$, a third function may be solved for that can be used to find the optimal policy. The state-action value function $Q(s_i, a_i)$ is the expected sum of all future reward when taking action a_i in state s_i when following some policy. The optimal value function can be represented by the optimal state-action value function with the following equation

$$V^*(s_i) = \max_{\text{a}} Q^*(s_i, a_i) \tag{7}$$

If the optimal state-action value function is known, the optimal policy may be derived with the equation below.

$$\pi^*(s_i) = \arg\max_{\text{a}} Q^*(s_i, a_i) \tag{8}$$

This work solves for $Q^*(s_i, a_i)$ for a single spacecraft using Monte Carlo tree search, an online search algorithm, and state-action value function regression using artificial neural networks (ANNs).

Monte Carlo Tree Search

Monte Carlo tree search is an online search algorithm, meaning that it searches reachable states to maximize the expected reward. MCTS uses a combination of simulation and rollout to explore the state space, building an estimate of the state action value function, $\hat{Q}(s_i, a_i)$, as it interacts with the environment. The process is depicted in Figure 4.

During the selection step, MCTS selects the action that maximizes $\hat{Q}(s_i, a_i) + U(s_i, a_i)$, where $U(s_i, a_i)$ is an exploration bonus based on the number of times a particular state-action pair has been visited. The selection step repeats until a state that has not been visited before is reached.

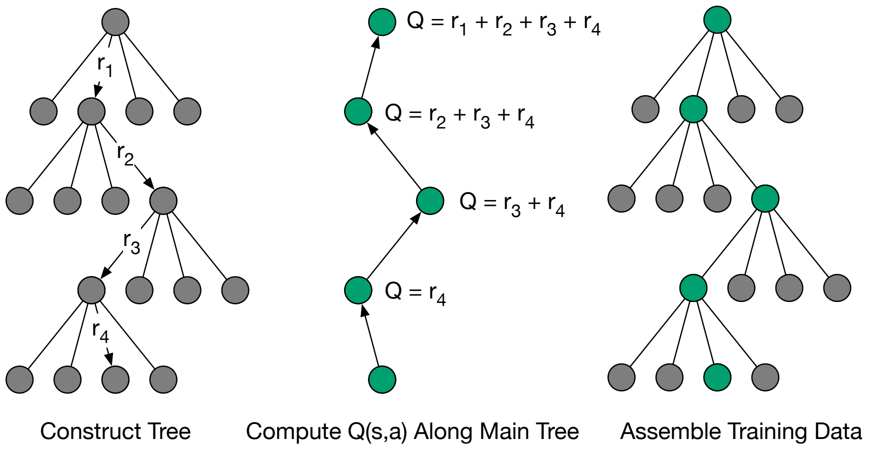


Figure 5: **Updating the state-action value function.**[4]

The expansion step is then taken, where $\hat{Q}(s_i, a_i)$ and $N(s_i, a_i)$ are initialized for each state-action pair. Then, a rollout is performed where actions are taken until the episode terminates. The rollout policy is a heuristic policy used by MCTS to find states with high reward that are more promising to search. The rollout policy is discussed in detail in past work.[4] In this problem, if the resource states are nominal, the spacecraft attempts to image the nearest target. Otherwise, the spacecraft takes an action to maintain the spacecraft resources such as power or the level in the data buffer. After rollout, the reward is backed up through each state-action pair to update $\hat{Q}(s_i, a_i)$ using an incremental averaging operator. The total return is denoted with q.

$$\hat{Q}(s,a) = \hat{Q}(s,a) + \frac{q - \hat{Q}(s,a)}{N(s,a)} \tag{9}$$

State-Action Value Function Approximation

Monte Carlo tree search is used to solve hundreds of planning problems, generating an estimate of $Q(s_i, a_i)$ for various initial conditions that is regressed over using an artificial neural network to approximate the state-action value function. This approximation is referred to as $Q_\theta(s_i, a_i)$. However, before the MCTS estimates of $Q(s_i, a_i)$ can be used for regression, the realized reward from stepping through the simulation must be backed up through the main tree such that the corresponding state-action values reflect the real reward found in the environment, not the averaged reward. This process is demonstrated in Figure 5.

After updating the state-action value estimates, the ANN training begins as depicted in Figure 6. First, several hundred episodes are solved for using the process described above. Then, the data is assembled and split into a training set and a validation set. 90% of the data is used for training. The remaining data is used for validation. This helps monitor for overfitting during training. Various hyperparameters (i.e. the activation function, depth, width) are used to generate a number of neural networks. These neural networks are then deployed in the environment and evaluated based on performance metrics such as reward, the number of targets imaged, the number of targets downlinked, and resource management failures.

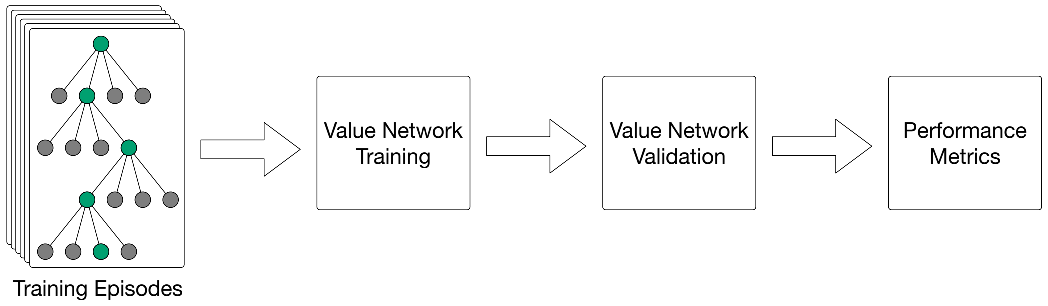


Figure 6: **Supervised learning process.**[4]

Table 1: Initial condition distributions during training.

Orbit Parameters	Value
Semi-Major Axis, a	6871 km
Eccentricity, e	$\mathcal{U}$[0, 0.01]
Inclination, i	$\mathcal{U}$[40, 60] deg
Long. of Ascend. Node, Ω	$\mathcal{U}$[0, 20] deg
Arg. of Periapsis, ω	$\mathcal{U}$[0, 20] deg
True Anomaly, f	$\mathcal{U}$[0, 360] deg
Spacecraft Parameters	
Disturbance Torque, $\boldsymbol{\tau}_{\text{ext}}$	2×10^{-4} Nm
Attitude Initialization, $\boldsymbol{\sigma}_{\mathcal{B}/\mathcal{R}}$	$\mathcal{U}$[0, 1.0] rad
Rate Initialization, ${}^{\mathcal{B}}\boldsymbol{\omega}_{\mathcal{B}/\mathcal{N}}$	$\mathcal{U}$[-1e-05, 1e-05] rad/s
Reaction Wheel Speeds	$\mathcal{U}$[-4000, 4000] RPM
Initial Battery Charge	$\mathcal{U}$[30, 50] Whr

Trained Agents

The training pipeline described in previous sections is applied to train agents which are deployed in the multi-satellite environment. For each number of targets included in the action space ($|\mathbf{U}| = \{1, 2, 3, 4\}$), MCTS is used to generate thousands of estimates of $Q(s_i, a_i)$, and a number of agents are trained using different artificial neural network parameters to compute an approximation of the state-action value function, $Q_\theta(s_i, a_i)$. The hyperparameter search conducted for the artificial neural networks is covered in past work.[15] However, the distributions of the initial conditions for training are included in Table 1. These distributions are selected to encompass a small range of low-Earth orbits with the same semi-major axis. Because the initial conditions do not cover all possible combinations of Walker-delta constellations, some overfitting of the trained agents on these initial conditions is anticipated once deployed in a multi-satellite scenario. This provides the opportunity to assess the degree to which agents are overfit or poorly extrapolate if not trained over all possible initial conditions.

The average reward for highest performing agents for each $|\mathbf{U}|$, along with the 95% confidence intervals for each agent, is provided in Figure 7. The average reward for each size of $|\mathbf{U}|$ provides an upper bound on the expected performance in the multi-satellite scenario. Note that by including additional targets in the action space that the agent can choose from, the average reward increases. As shown in prior work, this is largely due to the spacecraft selecting high-priority targets over low-priority targets, which is the desired behavior.

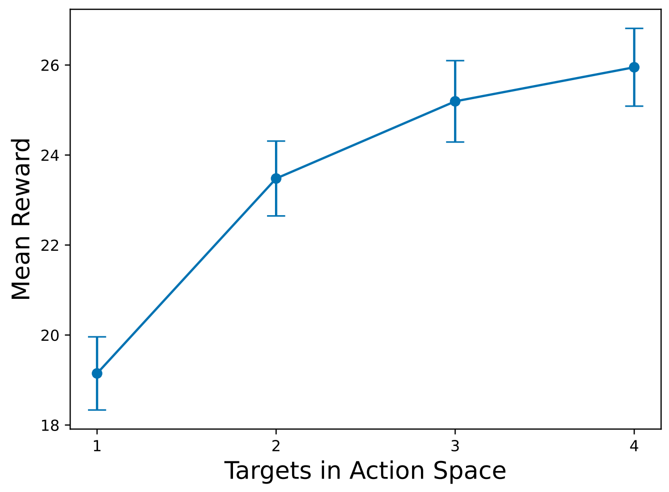


Figure 7: **Average reward for each agent after training.**[15]

MULTI-SATELLITE DEPLOYMENT

The trained agents are deployed in a multi-satellite spacecraft scenario. The design, communication assumptions, methods for target generation, and simulation architecture are described in this section.

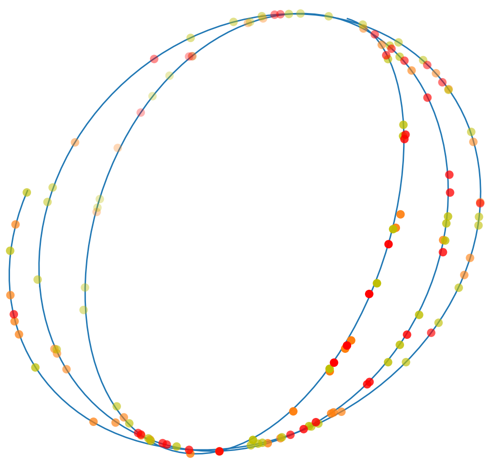
Walker-Delta Constellation Design

The constellations described in this work follow the Walker-Delta notation. The k satellites are distributed evenly between n orbit planes, which are distributed at $360/n$ deg intervals of the longitude of ascending node. A relative phase, g, between satellites in adjacent spacecraft planes is also specified. Each satellite in the constellation has the same semi-major axis and inclination, and each orbit is circular. By using the Walker-Delta constellation notation, different constellation architectures may be quickly prescribed and insights about performance as it relates to the training parameters can be easily obtained.

Cross-link Communication

Decentralized POMDPs operate under the assumption that there is limited to no communication between agents. In order to train and deploy agents using an MDP problem formulation, instantaneous cross-link communication between every spacecraft is added as an assumption such that communication is free. In large constellations where every spacecraft has a path to each other spacecraft via line-of-sight communication, this assumption holds well. The smaller the constellation, the less the assumption holds without adding uplink and downlink, but it is still useful for comparison purposes.

Each spacecraft k has a set of $\mathbf{T}_k$ targets available for imaging during the planning horizon. Each set $\mathbf{T}_k$ is a subset of a global set of targets shared between all spacecraft, $\mathbf{M}$. At the end of every step, the spacecraft come to a consensus on which targets in $\mathbf{M}$ have been imaged and which have not. If a target in $\mathbf{M}$ has been imaged, the corresponding target for all $\mathbf{T}_k$ is removed and added to $\mathbf{D}_k$ for each spacecraft, the subset of $\mathbf{T}_k$ containing the imaged or passed targets.


(a) **Training target distribution.**

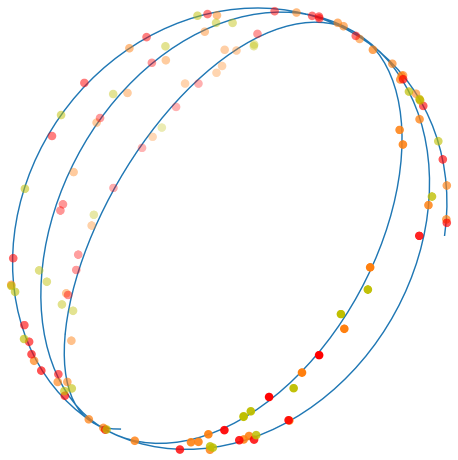

(b) **Deployment target distribution.**

Figure 8: **Target generation for a single spacecraft.**

Ground Targets

Ground targets are generated differently for training and deployment. In training, the spacecraft has access to all global targets. In deployment, the spacecraft has access to a subset of the global targets. During training, before MCTS solves the planning problem, the spacecraft translational state is propagated forward and rotated into an Earth-centered, Earth-fixed (ECEF) frame. Once rotated, points are randomly selected from the orbit. Zero mean Gaussian noise is added to the point, which is then projected onto the surface of the Earth. This point is added as a ground target and timestamped for when it is accessible to the spacecraft, and the process is repeated for the desired number of ground targets. For deployment, a set of ground targets is generated by projecting a number of random unit vectors onto the surface of the Earth. The spacecraft orbits are propagated, and once these points are accessible to the spacecraft based on a range requirement, they are timestamped and added to a target list. An example of the training targets and deployment targets may be found in Figure 8. The spacecraft orbits are shown projected onto the surface of the Earth. Qualitatively, the two methods generate nearly identical looking sets of targets. However, the deployment method of generation has more targets in the off-nadir direction because of the standard deviation selected to add noise to the training set.

Multi-Satellite Simulation Architecture

The multi-satellite EOS scheduling problem is simulated using the Basilisk* Astrodynamics and Flight Software Framework.[14] Basilisk is chosen due to the speed at which is can run a high-fidelity spacecraft simulation. The speed of Basilisk allows for rapid training and validation, and its ability to simulate real flight software lends itself to flying this work on-board real spacecraft. Its modularity also facilitates the rapid creation of a simulation with spacecraft that have the precise characteristics needed.

*`http://hanspeterschaub.info/basilisk`

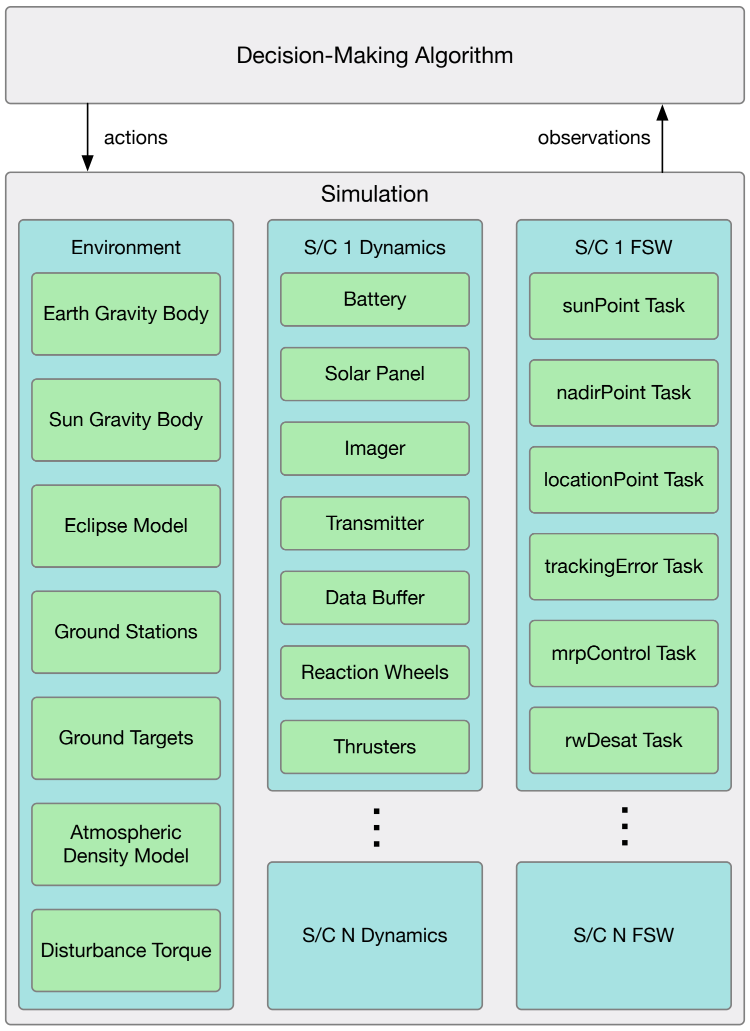


Figure 9: **Simulation architecture.**

Single spacecraft Basilisk simulations are described in detail in past work.[4] In this work, the most significant change to the simulation is its integration into a multi-satellite architecture, which is described in Figure 9. The simulation modules are split into three different classes: environment, dynamics, and flight software (FSW). While each spacecraft has its own dynamics and FSW classes, the environment class is shared between all spacecraft. This architecture is built to be easily scalable to any number of satellites, as one only needs to add dynamics and FSW classes for each spacecraft in the simulation and set the initial conditions for each. This amounts to changing values of the Walker-delta parameters in a single line of code.

The environment class contains the modules that are not spacecraft specific, but instead describe the simulation environment. The gravity field is modeled to include J_2 perturbations from the Earth and third-body perturbations from the sun. The effect of drag is modeled through the use of an atmospheric density model. The environment class also contains several ground locations which return an access state to each spacecraft. The ground stations for downlinking data are modeled using the ground locations, as are the ground targets for imaging. During each integration step, the environment class is the first to be updated, and the outputs of its modules are shared with all spacecraft via the dynamics and FSW classes.

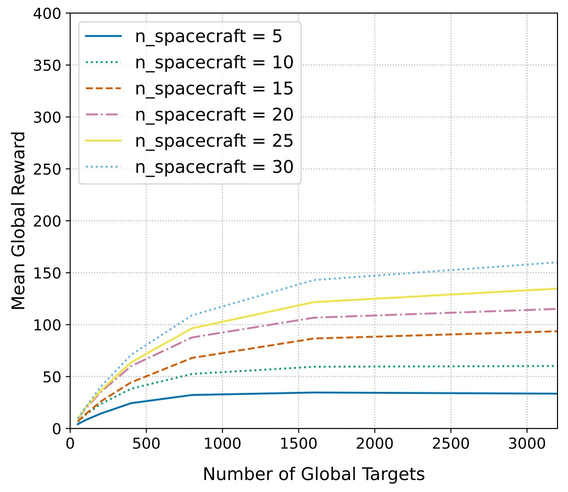
The dynamics class contains the modules that represent each spacecraft and its components. For simplicity, all spacecraft models are identical, but the initial conditions for each are set to different values. A comprehensive power module is simulated, leveraging Basilisk's high fidelity dynamics capabilities to accurately compute power consumption and generation. Solar panels take into account incidence angle, eclipse regions, and distance from the Sun to calculate power generation, which is stored in a battery. This battery is connected to power-consuming devices, such as the imager, transmitter, and reaction wheels. The latter are part of the attitude control system, and are modeled after the Honeywell HR16 reaction wheels. Thrusters are also implemented to allow for momentum-dumping maneuvers in the desaturation mode and are modeled after the Moog Monarc-1 thrusters. Attitude perturbations are added to the simulation using random disturbance torques and drag-induced torques. An on-board data management system is also modeled, which takes the data generated by the instrument during imaging modes and stores it in a data buffer. The transmitter then downlinks this data to the ground stations specified in the environment class when access is available.

The FSW class contains the modules responsible for control of the various spacecraft subsystems. It sets the attitude modes, which consist of Sun pointing for battery charging, nadir pointing for downlinking data, target pointing for imaging, and momentum dumping to remove some of the momentum in the reaction wheels. This class also contains the logic for the attitude control system. A reference attitude and attitude rates are computed depending on the current attitude mode, and the attitude error is calculated by comparing the reference and the current attitude states. This error is fed into an MRP-based controller,[13] which outputs the necessary control torques to converge to the reference attitude. This control torque is then mapped into the reaction wheel configuration, which yields a requested motor torque for each. The thruster commands are also defined in this class, where a module maps the reaction wheel momentum to thruster on-time commands to remove momentum from the reaction wheels if the momentum dumping mode is active. Finally, the instrument control is implemented, which sends an image command to the imager once the ground target is accessible for imaging.

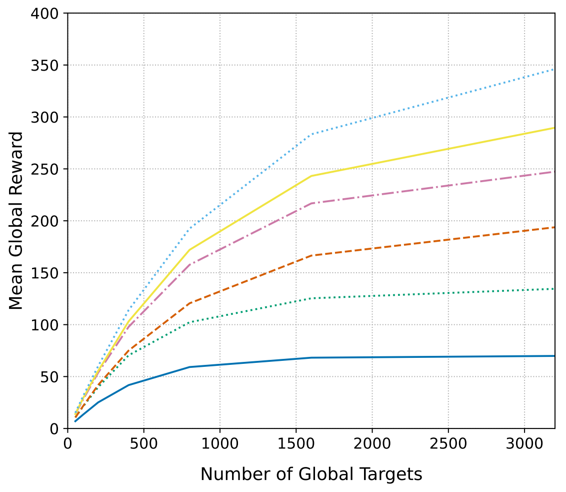
RESULTS

Three separate experiments are conducted to explore the performance of the trained agents when deployed in a multi-satellite environment. For each experiment, results are generated for each size of $\mathbf{U}$, the total number of targets in the action space. The global reward of each experiment refers to the sum of local rewards of each spacecraft, which is computed using the reward function for the single satellite MDP. However, reward is not duplicated for imaging or downlinking the same target. If two spacecraft image a target at the same time, the reward for doing so is only added to the global reward once. As a result, the performance of the trained agents is inherently suboptimal as the only coordination between agents is sharing which targets are still available for imaging or downlinking. Please note that all results are preliminary as this work is ongoing.

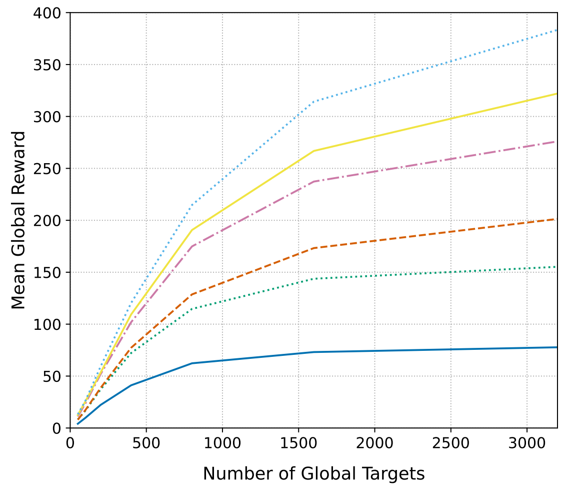
In the first experiment, an increasing number of spacecraft (from 5 to 30) are placed in 5 orbit planes to validate that the reward grows with an increase in global of targets for all constellations and sizes of $\mathbf{U}$. The change in reward as more spacecraft are added is also investigated. As more spacecraft are added, the global reward should increase, but the reward-per-spacecraft should decrease



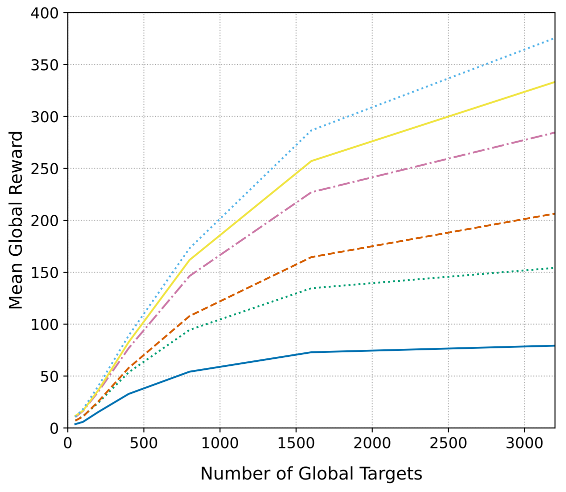
(a) **1 target in action space.**



(b) **2 targets in action space.**



(c) **3 targets in action space.**

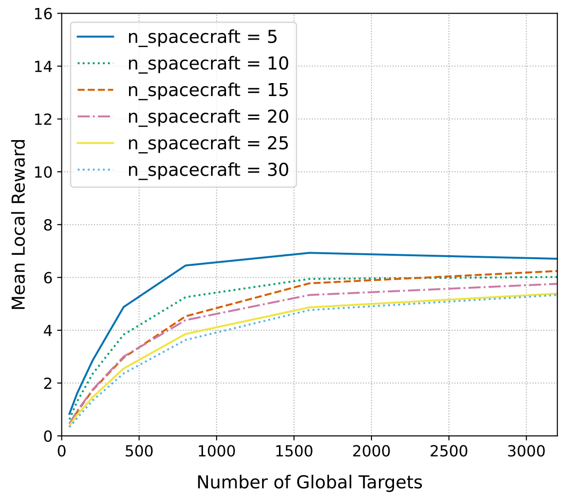


(d) **4 targets in action space.**

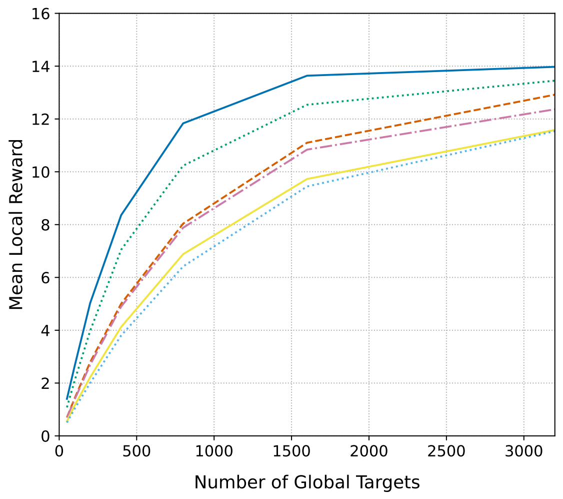
Figure 10: **Global reward vs. the number of global targets for 6 orbit planes.**

due to an increase in competition. In Figure 10, the mean global reward is plotted for 6 separate constellation designs and 4 different sizes of $\mathbf{U}$. As expected, for all sizes of $\mathbf{U}$, the mean global reward increases with both the number of spacecraft and number of global targets. Reward begins to plateau with the increase in the number of global targets. The frequency of downlink windows and size of the data buffer constrains the maximum possible global reward at this point, and the addition of more targets results in less of an increase in reward. Interestingly enough, $|\mathbf{U}| = \{2, 3, 4\}$ are similar in mean global reward. However, $|\mathbf{U}| = 1$ generates a mean global reward about half of the other sizes of $\mathbf{U}$. The reason for this is unclear, but two possibilities are presented: an error in the way in which targets are distributed or poor extrapolation to orbits outside of the training distribution exacerbated by fewer target tuples in the state space. Future work will investigate this further.

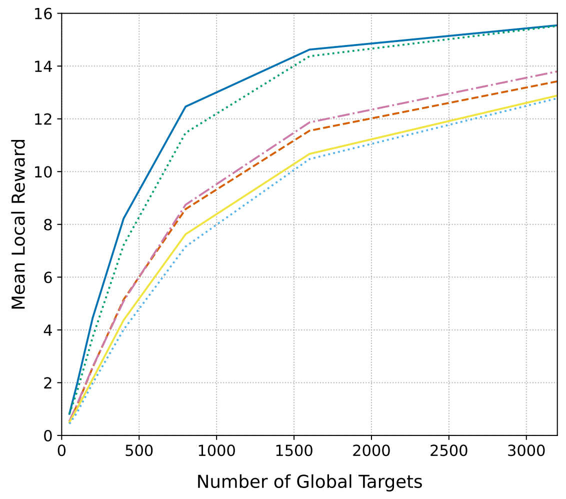
In Figure 11, the local reward (reward per spacecraft) for experiment 1 is plotted. Plotting the



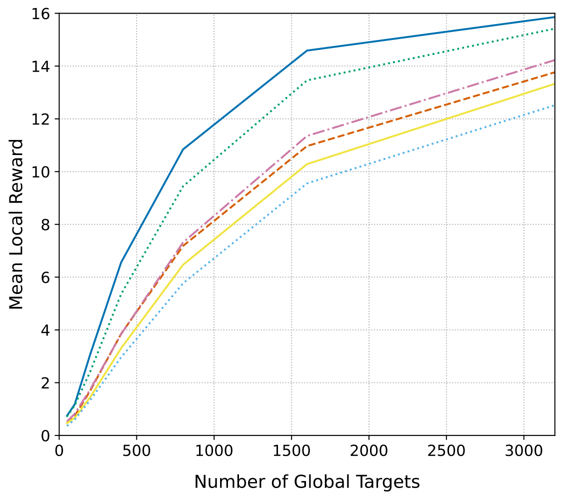
(a) **1 target in action space.**



(b) **2 targets in action space.**



(c) **3 targets in action space.**

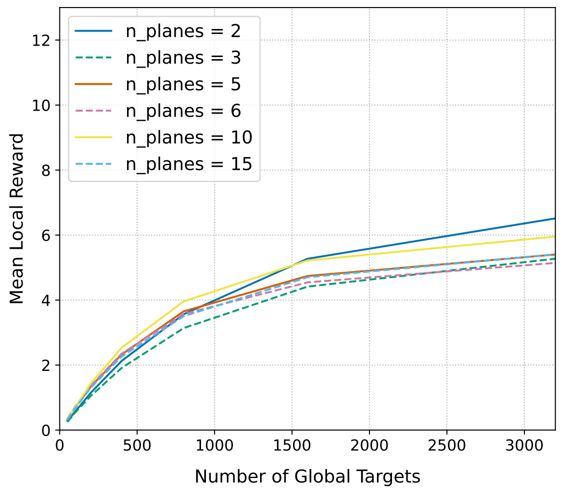


(d) **4 targets in action space.**

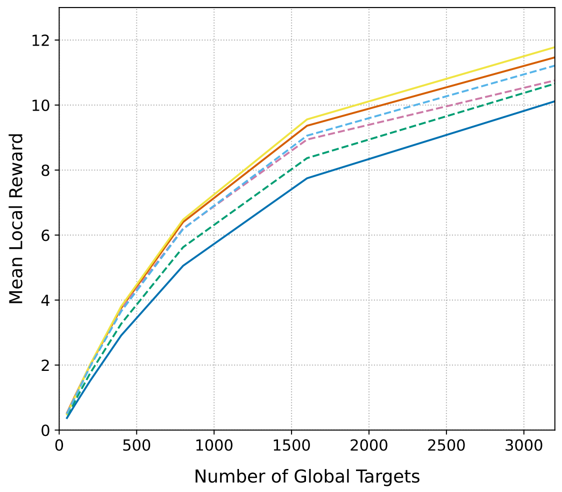
Figure 11: **Reward per spacecraft vs. the number of global targets for 6 orbit planes.**

mean local reward can give insights into how efficiently each spacecraft is being utilized when more spacecraft are introduced. Note that when compared to Figure 7, the mean local reward for each spacecraft is much lower than during training. The reason for this is covered in experiment 3. For all sizes of $\mathbf{U}$, the mean local reward decreases as more spacecraft are added to the simulation, which is expected. As more spacecraft are added to the simulation, the competition for ground targets increases, which leads to a decrease in the availability of targets to image for a given spacecraft and an increased probability that two spacecraft will attempt to image the same target. It should be noted that the probability of the latter occurrence is low, but possible.

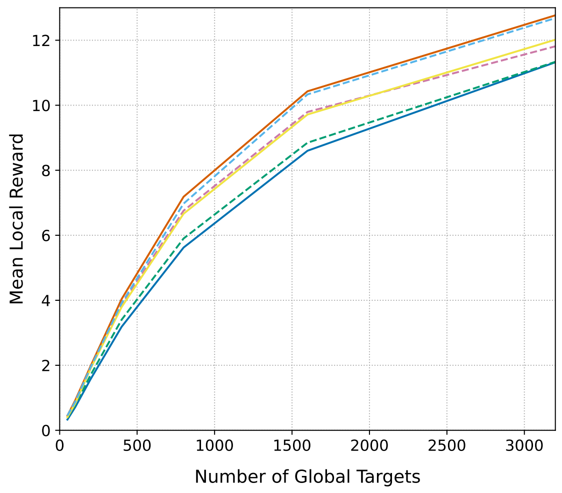
In the second experiment, the number of spacecraft is fixed at 30, and the spacecraft are evenly distributed among the following number of planes: $\{2, 3, 5, 6, 10, 15\}$. In this experiment, it is expected that mean local reward will display a dependency on the number of planes. In Figure



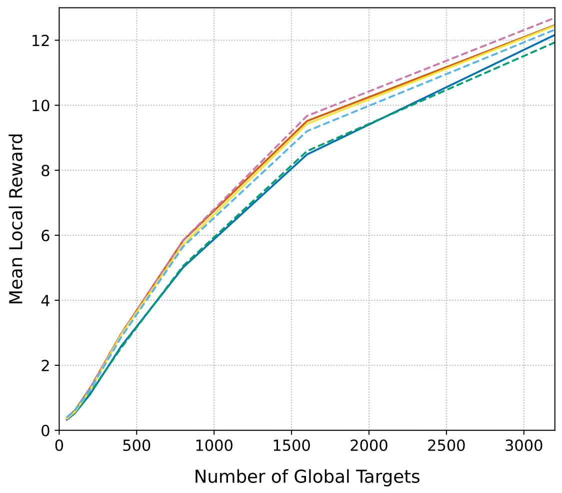
(a) **1 target in action space.**



(b) **2 targets in action space.**



(c) **3 targets in action space.**



(d) **4 targets in action space.**

Figure 12: **Reward per spacecraft vs. the number of global targets for 30 spacecraft.**

12, the mean local reward is plotted for the 6 separate constellations. The same reward plateau as the number of global targets increases shown in experiment 1 remains. However, there is no consistent pattern between all sizes of $\mathbf{U}$ as the number of planes increase. Furthermore, there is no pattern between complementary retrograde and prograde orbits (planes = $\{2, 6, 10\}$). This lack of clear pattern was not anticipated because of relationships between the number of orbital planes and constellation metrics such as maximum revisit time and mean coverage time.[16] It appears that the total number of spacecraft has a larger impact on the mean local reward than the total number of orbital planes, but future work should explore this in greater depth.

Finally, in experiment 3, the local reward for each spacecraft plane is explored for 21 spacecraft, each in its own orbital plane. During training, each episode begins with a spacecraft in an orbit sampled from uniform distributions of Keplerian orbital parameters as shown in Table 1. In particular, the longitude of the ascending node is sampled from a uniform distribution of 0 to 20 degrees. In

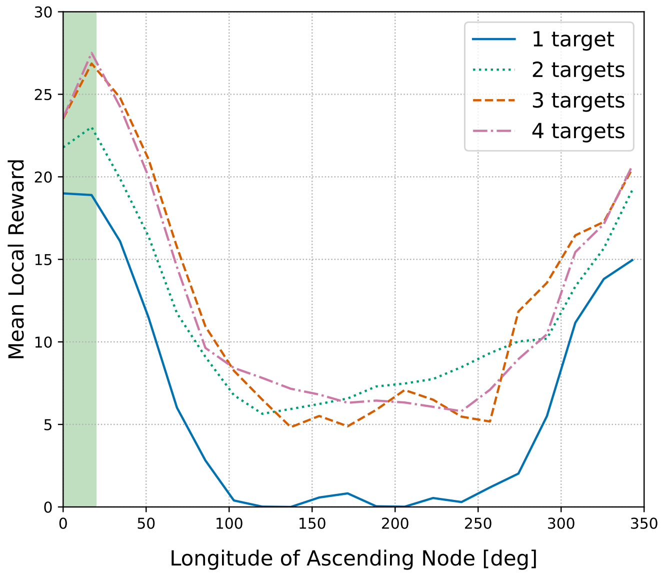


Figure 13: **Reward per spacecraft vs. longitude of the ascending node.**

Figure 13, the local reward is plotted vs. the longitude of the ascending node. The distribution sampled from during training is highlighted in green. Furthermore, a line for each size of $\mathbf{U}$ is included. The maximum local reward for all $|\mathbf{U}|$ occurs within the training distribution and matches the maximum reward during training shown in Figure 7. As the longitude of the ascending node increases, the local reward decreases until reaching a minimum value in the range of 100 - 250 deg. The reward then begins to increase as the longitude of the ascending node approaches 0 degrees once more. The decrease in reward for initial conditions outside of the training distribution explains the low average local reward in Figures 11 and 12. This plot demonstrates two things: a dependency on the training distribution and a gradual degradation in performance the farther away from the training distribution the spacecraft's orbit is. Two possible solutions are presented to remedy this issue, which will be addressed by future work. First, the training distributions should cover all possible Keplerian orbital parameters experienced by the spacecraft. Second, alternative state spaces, particularly regarding the spacecraft's position and velocity, should be explored. Two candidates should be considered: normalized Keplerian parameters and 4D-spherical coordinates.

CONCLUSION

This work shows that an agent trained in a stationary, single-spacecraft environment for the Earth-observing satellite (EOS) scheduling problem can be deployed as a part of a large constellation of spacecraft with the addition of cross-link communication between spacecraft. Several experiments are performed to explore the performance of the agents when deployed in such a constellation. It is shown that for all constellation designs, global and local reward plateaus as the number of targets available to the agents increases because of data buffer and ground station constraints. Furthermore, as more spacecraft are added to the constellation, the competition to image ground targets increases, so local reward (reward per spacecraft) is lower when compared to a constellation with fewer spacecraft. Finally, the performance of each agent is explored for different longitudes of the ascending node to determine how well the agents extrapolate to orbits outside of the training distribution. As the longitudes of ascending node get farther away from those of the training distribution, performance decreases until the nominal training distribution is approached once again.

Future work should explore alternative state spaces and increase the upper bound of the training distribution for the longitude of the ascending node to attempt to eliminate this issue. Additionally, future work will tie the performance of the agents to typical evaluation metrics for Walker-delta constellation designs such as maximum revisit time and mean daily visibility time. Future work should also compare these results to other problem formulations such as multi-agent Markov decision processes (MMDPs) or decentralized partially-observable Markov decision processes (Dec-POMDPs). The addition of a global reward function during training would result in more coordination between spacecraft. In the MMDP formulation, communication between agents is free and always available. In the Dec-POMDP formulation, a belief state would need to be maintained over the probability that a given target has been imaged or downlinked.

REFERENCES

[1] A. Harris, T. Teil, and H. Schaub, "Spacecraft Decision-Making Autonomy Using Deep Reinforcement Learning," *AAS Spaceflight Mechanics Meeting*, Maui, Hawaii, January 13–17 2019. Paper No. AAS-19-447.

[2] A. Harris and H. Schaub, "Deep On-Board Scheduling For Autonomous Attitude Guidance Operations," *AAS Guidance, Navigation and Control Conference*, Breckenridge, CO, January 30 – Feb. 5 2020. AAS 02-117.

[3] A. Harris and H. Schaub, "Spacecraft Command and Control with Safety Guarantees using Shielded Deep Reinforcement Learning," *AIAA SciTech*, Orlando, Florida, January 6–10 2020.

[4] A. P. Herrmann and H. Schaub, "Monte Carlo Tree Search Methods for the Earth-Observing Satellite Scheduling Problem," *Journal of Aerospace Information Systems*, 2021, pp. 1–13, 10.2514/1.I010992.

[5] C. Amato, G. Konidaris, G. Cruz, C. A. Maynor, J. P. How, and L. P. Kaelbling, "Planning for Decentralized Control of Multiple Robots under Uncertainty," *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 1241–1248.

[6] S. Omidshafiei, A.-A. Agha-Mohammadi, C. Amato, S.-Y. Liu, J. P. How, and J. Vian, "Decentralized Control of Multi-Robot Partially Observable Markov Decision Processes using Belief Space Macro-actions," *The International Journal of Robotics Research*, Vol. 36, No. 2, 2017, pp. 231–258.

[7] L. Matignon, L. Jeanpierre, and A.-I. Mouaddib, "Coordinated Multi-Robot Exploration under Communication Constraints using Decentralized Markov Decision Processes," *Twenty-sixth AAAI conference on artificial intelligence*, 2012.

[8] F. A. Oliehoek and C. Amato, *A Concise Introduction to Decentralized POMDPs*. Springer, 2016.

[9] M. J. Kochenderfer, T. A. Wheeler, and K. H. Wray, *Algorithms for Decision Making*. MIT Press, 2022.

[10] C. Boutilier, "Sequential Optimality and Coordination in Multiagent Systems," *IJCAI*, Vol. 99, 1999, pp. 478–485.

[11] C. H. Papadimitriou and J. N. Tsitsiklis, "The Complexity of Markov Decision Processes," *Mathematics of Operations Research*, Vol. 12, No. 3, 1987, pp. 441–450.

[12] J. R. Wertz, D. F. Everett, and J. J. Puschell, *Space Mission Analysis and Design*. 2015.

[13] H. Schaub and J. L. Junkins, *Analytical Mechanics of Space Systems*. Reston, VA: AIAA Education Series, 4th ed., 2018, 10.2514/4.105210.

[14] P. W. Kenneally *et al.*, "Basilisk: A Flexible, Scalable and Modular Astrodynamics Simulation Framework," *7th International Conference on Astrodynamics Tools and Techniques (ICATT)*, DLR Oberpfaffenhofen, Germany, Nov. 6–9 2018.

[15] A. Herrmann and H. Schaub, "Autonomous On-board Planning for Earth-orbiting Spacecraft," *IEEE Aerospace Conference*, Big Sky, MT, March 5-12 2022.

[16] C. A. Hinds, "A Pareto-Frontier Analysis of Performance Trends for Small Regional Coverage LEO Constellation Systems," Master's thesis, California Polytechnic State University, 2014.