Optimal Cislunar Architecture Design Using Monte Carlo Tree Search Methods

Michael Klonowski
The University of Colorado at Boulder
Marcus J. Holzinger
The University of Colorado at Boulder

Naomi Owens Fahrner
Ball Aerospace

ABSTRACT

A novel multi-objective Monte Carlo Tree Search (MO-MCTS) algorithm is developed and implemented for use in architecture design problems. This algorithm is implemented on two well-known problems with known solutions in order to verify its performance. It is then used in a highly nonlinear cislunar architecture design problem with no known analytical solutions. The results of this implementation display the ability of MO-MCTS to effectively navigate the state space of mixed integer nonlinear programming problems and emphasize the versatility of MO-MCTS for designing critical cislunar architecture.

1. INTRODUCTION

With the ever-increasing interest in cislunar space from commercial, military, and scientific entities alike, it is imperative that space domain awareness architecture occupying this region is sufficiently capable of providing operators with effective support for their missions [8]. It is equally important that the cost of installation and maintenance of such an architecture is realistic given a budget and specific mission goals. This problem is exponentially more difficult than designing near-Earth architectures given the complex non-Keplerian dynamics governing the region [15]. All these factors combine to create an intractable mixed integer nonlinear programming problem (MI-NLP) for which sophisticated numerical methods must be used to generate a solution near the global optimum [3].

Work done by [29] proposed placing satellites in periodic orbits near Earth-Moon Lagrange points to provide near theoretical near global coverage of the Earth and Moon. In [26], the authors analyzed the performance of satellites in Earth-Moon synodic period resonant periodic orbits observing space objects within a specified volume of cislunar space. Other work in this area including [19], [11], and [9], places repeated emphasis on the importance of space-based observers in cislunar periodic orbits to provide coverage within the cislunar region. Gaps within the literature, however, occur in the process of designing said observer architectures. Given the immense cost and vast complexities associated with missions within cislunar space, architecture in the region must be able to meet certain performance and cost objectives with great confidence. The cislunar region is the new frontier in space. As such, traditional intuition surrounding space architecture design must be reevaluated considering the expanse of unknowns to be encountered in future research and exploration.

Monte Carlo Tree Search (MCTS) methods have been successfully applied to a multitude of MINLP design problems, including optimal wind farm layouts [2], structural design [22], stream processing instance placement [16], interplanetary mission design [14], and numerous others. Results from these works have displayed MCTS's performance as competitive with and even outperforming other state-of-the-art numerical methods. While in most of these design problems there are multiple competing objectives to be considered in an optimal solution, the standard MCTS algorithm is not fully equipped to handle them. Rather than looking at a linear combination of objectives as a single objective, [27] proposes a multi-objective (MO) approach using the hyper-volume indicator to guide action selection in the exploration vs. exploitation step of MCTS.

To approximately solve nonlinear mixed integer programming problems, we propose a novel implementation of a multi-objective Monte Carlo Tree Search (MO-MCTS) algorithm that effectively and efficiently explores the feasible

state space of architectural design problems and returns a set of approximate Pareto-optimal solutions. This contribution builds on the MO-MCTS algorithm proposed by [27] while adding mechanisms specific to architecture problems with large branching factors. We validate our approach by looking at two well-known "toy" problems, and present a solution to a representative cislunar architecture problem to demonstrate the utility of the method.

2. MONTE CARLO TREE SEARCH AND MULTI-OBJECTIVE OPTIMIZATION

2.1 Markov Decision Processes

A Markov Decision Process (MDP) is a discrete-time stochastic process in which the probability of each state in the process depends only on the previous state. An MDP is defined by the tuple (S, A, T, R, γ), with state space S, action space A, state transition probability function T, reward function R, and discount factor γ. The goal of an MDP is to find a policy function $\pi(s) : S \rightarrow A$ that maximizes some function of the cumulative future rewards. MDPs are by definition fully observable, and thus can be solved using a variety of algorithms when no time constraint is given. However, for MDPs with large state and action spaces, solving for the optimal policy, $\pi^*(s)$, quickly becomes intractable due to the so-called "curse of dimensionality." Numerous methods have been developed that use random sampling combined with heuristics to find sufficiently good approximations to these problems.

2.2 Monte Carlo Tree Search

Monte Carlo Tree Search is a reinforcement learning algorithm that combines the structure of trees with the randomness of Monte Carlo methods to obtain an approximate solution to an MDP with guarantees of convergence to the global optimum over time [18]. In the tree iteratively generated by the MCTS algorithm, states are represented by tree nodes and actions are represented as tree edges. MCTS methods have gained increasing popularity in recent years due to its implementation in *AlphaGo* that beat a human in a game of computer Go [23], and its proceeding successes across numerous other domains.

The steps of MCTS have been heavily covered in many publications [6], [5], [25]. For brevity, only a brief description is provided here. MCTS consists of four distinct steps: selection, expansion, simulation, and backpropagation. These are visualized in Fig. 1 from [25], and are described as follows:

Selection Starting from the root node, the algorithm traverses the tree using some exploration vs. exploitation policy until it reaches a node that has not yet been expanded (a leaf node).

Expansion Unless the leaf node is a terminal state, from the leaf node a single (or multiple) child node is created from a chosen action (uniformly or heuristically).

Simulation From the new child node, a Monte-Carlo simulation is run using a rollout policy until a terminal state or some other condition is reached.

Backpropagation The resultant reward of this simulation is used to update the statistics of all the nodes visited during the tree traversal. These statistics include the node's estimated value and visit counts.

Arguably the most powerful aspect of MCTS is its ability to efficiently explore new parts of the tree while also relying on previous solutions in a process called exploration vs. exploitation, performed in the selection phase. While there are many methods used to balance this trade off, the most common algorithm used is the upper confidence bound for trees (UCT) algorithm [18]. UCT guides exploration vs. exploitation by treating the selection process as a multiarmed bandit problem, with the value of a node being the expected reward as approximated by the simulation phase. UCT selects a child node by choosing an action a that maximizes

$$Q(s,a) + C\sqrt{\frac{\ln N(s)}{N(s,a)}} \tag{1}$$

where $Q(s,a)$ is the action value estimate of taking action a at state s, $N(s)$ is the number of times s has been visited, $N(s,a)$ is the number of times a has been selected at s, and C is a constant typically chosen as $\sqrt{2}$ for rewards defined in the interval $[0,1]$. The first term in this equation represents the exploitation aspect of UCT, where the likelihood of

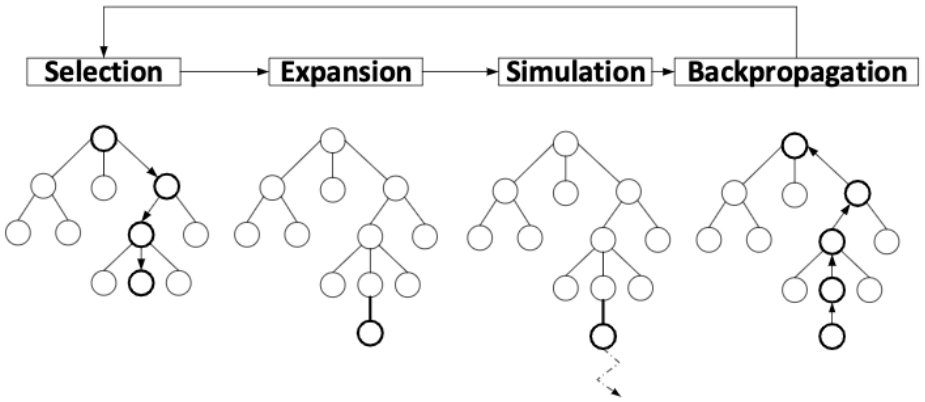


Fig. 1: Visualization of the Monte Carlo Tree Search steps from [25].

revisiting a child node is proportional to its expected value. The second term gives weight to child nodes that have not often been visited. Typically, when $N(s,a) = 0$, this term becomes infinite leading to the default policy of always selecting actions that have not yet been selected [25].

2.3 Multi-Objective Optimization

Multi-objective optimization (MOO) is the process of optimizing a collection of objective functions subject to some pre-defined constraints. Typically, there is no globally optimal solution to a MOO problem and thus a set of points must meet a predetermined definition to qualify as an optimal solution. Pareto-optimality is a common approach for characterizing solutions as optimal. It is how we will determine an optimal set of solutions for the duration of this work. Pareto-optimality is defined as follows from [20]:

Definition 2.1 (Pareto-optimal) *Given a MOO problem with objective function* $\mathbf{F}(\mathbf{x}) = [F_1(\mathbf{x}), F_2(\mathbf{x}), ..., F_k(\mathbf{x})]^T$ *a solution* $\mathbf{x}^*$ *is Pareto-optimal iff there does not exist another point* $\mathbf{x}$*, such that* $\mathbf{F}(\mathbf{x}) \leq \mathbf{F}(\mathbf{x}^*)$*, and* $F_i(\mathbf{x}) < F_i(\mathbf{x}^*)$ *for at least one* $F_i \in \mathbf{F}$*. The set of Pareto-optimal solutions is referred to as the* ***Pareto set*** *(P), and* $\mathbf{x}^*$ *is said to be* ***non-dominated*** *in P.*

2.4 Multi-Objective Monte Carlo Tree Search

Multi-objective Monte Carlo Tree Search leverages the power of MCTS to return a set of Pareto-optimal solutions. Since UCT relies on action value estimates defined in $\mathbb{R}$, we implement the method described by [27] that modifies UCT such that the action selected maximizes the hyper-volume of the union of the most recently updated Pareto set solutions with the action value estimate, summed with the typical UTC exploration term:

$$U(s,a) = HV(Q(s,a) \cup P) + C\sqrt{\frac{\ln N(s)}{N(s,a)}}. \tag{2}$$

Here, $HV(\cdot)$ is the hyper-volume indicator function from [4] and P is the Pareto set. As summarized in [21], the hyper-volume indicator, also known as the Lebesgue measure, is a metric that measures the size of the objective space covered by a set of solutions with respect to a reference point. In this work all objectives are formulated as reward functions in the interval $[0,1]$ to be maximized. Thus, the hyper-volume reference point for each objective is 0. As MO-MCTS progresses through its iterations and new Pareto-optimal solutions are found, the hyper-volume indicator of the Pareto set will increase, as visualized in Fig. 2.

Furthermore, since the hyper-volume indicator does not decrease with the union of dominated point, a penalty $d(\cdot)$ is added to equation (2) for a dominated $Q(s,a)$ with respect to P defined as the Euclidean distance from $Q(s,a)$ to the piecewise linear envelope of the points in P:

$$W(s,a) = \begin{cases} U(s,a) & \text{if } Q(s,a) \text{ is non-dominated in } P \\ U(s,a) - d(Q(\cdot)) & \text{otherwise} \end{cases} \tag{3}$$

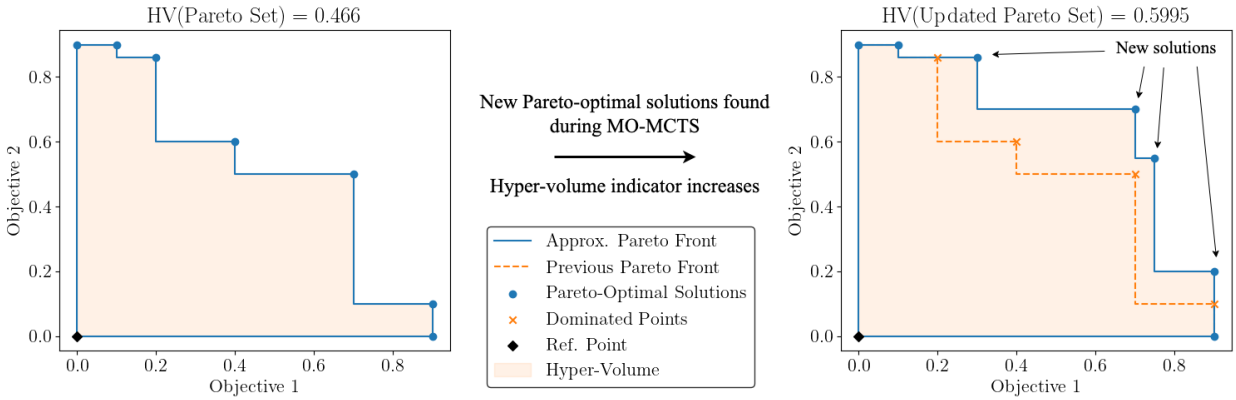


Fig. 2: In this representative example, the inclusion of new non-dominated (Pareto-optimal) solutions found during MO-MCTS increases the hyper-volume indicator of the Pareto set.

Without the addition of this penalty, MO-MCTS would end up exploring the state space near non-dominated points more often than it likely should.

A critical aspect of this method is consistently updating the solutions within the Pareto set to ensure that equation (3) can properly function. For the sake of computational efficiency, especially in the case of problems with a high branching factor, we typically update the Pareto set anywhere from every 100 to 1,000 MCTS iterations. We have found that varying this value does not significantly impact the quality of solutions returned by the algorithm. Fig. 3 describes the steps of our implementation of this algorithm.

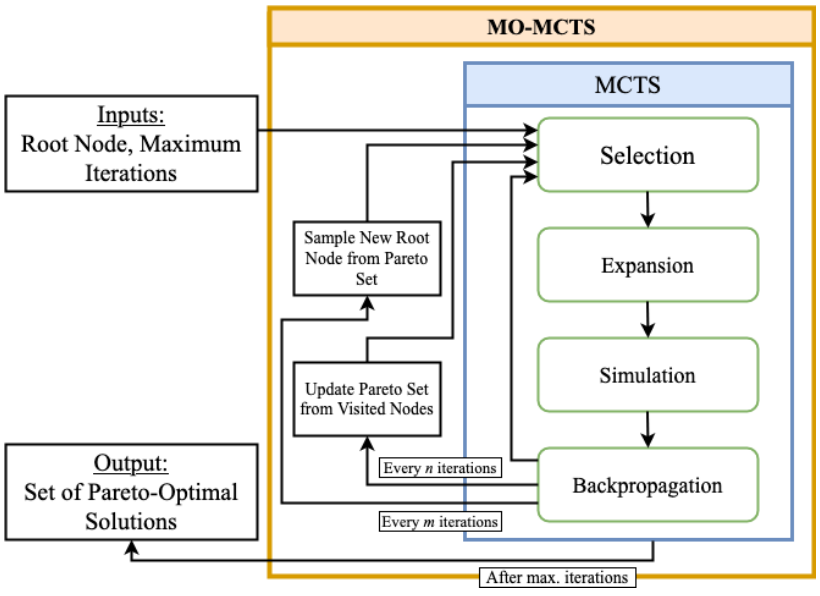


Fig. 3: Block diagram of the MO-MCTS algorithm.

2.4.1 MO-MCTS for Architecture Design

For architecture design problems dealing with the placement of assets in specific locations, actions that determine the evolution from one state to another are better interpreted as proxies for the states themselves. Put another way, in architecture problems where the order of placement is not critical, a designer does not care about the value of specific actions as discovered throughout the tree [14]. Rather, they care about the quality of a state that represents a specific

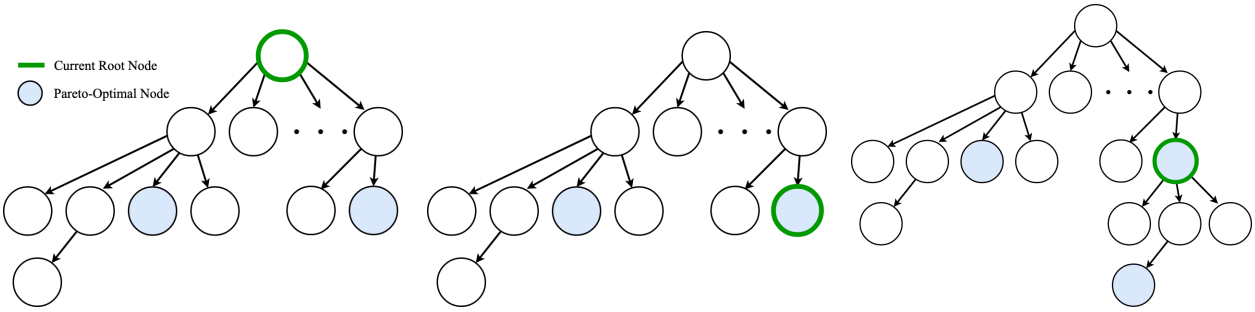
architecture design. In applications of single objective MCTS to games, the expected output of a run is an optimal action to be taken at the root node based on the traversal performed by the algorithm in a specified amount of time. This action is performed at the root node (by a user or in a simulation), and the process continues until a terminal constraint is met.

In architecture problems where the state represents a certain layout, each state visited throughout the tree has a deterministic value. That is, the immediate quality of an architecture design at any depth during the tree traversal can be directly determined through evaluation of the objective functions. Furthermore, since the goal of a multi-objective architecture design problem is to obtain a set of Pareto-optimal solutions (as layouts), we can ignore the action values completely and search through every visited node during the MCTS run and chose those that are Pareto-optimal. With this in mind we are relying on the modified UCT algorithm to reach quality and unique states from which we can determine our Pareto set.

It is often the case with MO-MCTS that once a good enough solution has been reached through selection, the algorithm may become trapped near this locally optimal solution. This seems to happen with greater frequency in problems that have a large branching factor and sparse Pareto-optimal solutions such as those considered in this work. Inspired by [28], we introduce a process by which the root node is resampled after some number of MCTS iterations. By randomly choosing a new state from the most recently updated Pareto set, we see dramatic increases in the hypervolume indicator. Practically speaking, resampling the root node from the Pareto set is a way of pseudo-pruning the search tree by forcing the algorithm to ignore for some time parts of the tree that do not appear to be successful. Tests have shown this method to be consistent in generating diverse sets of solutions faster than otherwise. We describe this method in the following conjecture, to be proved in future work:

Conjecture 2.1 *Given a tree with sparse Pareto-optimal nodes, randomly resampling the tree's root node from the current set of Pareto-optimal nodes allows MO-MCTS to quickly reach new Pareto-optimal nodes, while maintaining the asymptotic convergence guaranteed through MCTS with UCT.*

Fig. 4 visualizes how this method allows MO-MCTS to diversify the set of Pareto-optimal solutions, and explore new areas of the tree.



(a) MO-MCTS has decided to explore nodes near the left optimal node rather than looking at other parts of the tree.

(b) Choose the right optimal node as the new root node and continue MO-MCTS from there.

(c) MO-MCTS is able to find a new Pareto-optimal node from the new root node.

Fig. 4: Visualization of root node resampling method.

3. VALIDATION

3.1 Problems with Known Solutions

As an initial test of the MO-MCTS algorithm, we looked at two "toy" problems: the placement of satellites in geosynchronous (GEO) orbits to provide maximal coverage of the Earth's surface, and the combination of orbital elements for an orbit that maximizes time-averaged coverage of the United States and Russia. These problems both have known solutions. The solution to the former is three satellites equally spaced in a geosynchronous orbit and the solution to the latter is a Molniya orbit. We expect the output of the MO-MCTS algorithm to include these known solutions.

3.2 The GEO Problem

We begin with the following simple problem: what is the optimal placement of a number of satellites in geosynchronous orbit that maximizes coverage of the Earth's surface while considering the cost of each satellite? This problem has a well-known and simple solution: three satellites equally spaced in the GEO belt will provide near full coverage of the Earth's surface (save for a small area of land around each pole).

3.2.1 MDP Formulation

To begin, we formulate the GEO problem as a Markov Decision Process. First, the position of a satellite in any orbit exists in a continuous state space and as such must be discretized to be used in our MO-MCTS algorithm. For our implementation, the GEO belt is represented by a tuple of length 360 representing 360 equally spaced "slots" in the belt. Each element of the tuple can be 1 or 0 indicating that a slot is either occupied by a GEO satellite (hereby referred to as a "GEO") or is empty:

$$\underbrace{[\overbrace{1}^{\text{GEO}}, 0, \ldots, \overbrace{0}^{\text{Empty}}]}_{\text{360 Slots}}$$

Actions are defined as adding a GEO to a slot that is unoccupied, until either full coverage is reached or a maximum number of GEOs has been placed. The reward function takes the current state tuple as input and returns a two-dimensional reward vector containing the instantaneous coverage of the Earth's surface within $\pm 60^\circ$ latitude and the inverse of the number of GEOs in the state. Both rewards lie in the interval $[0, 1]$ and both are to be maximized.

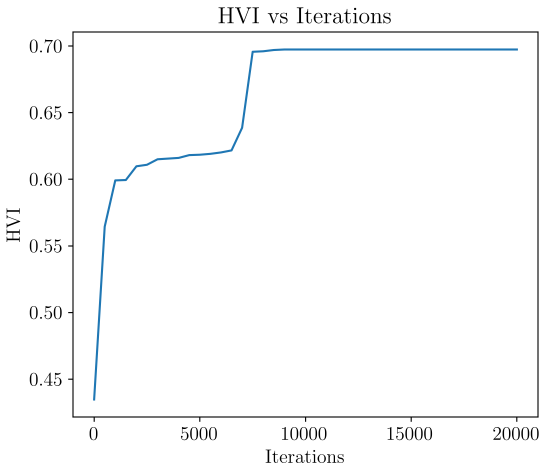


Fig. 5: MO-MCTS HVI vs iterations for the GEO problem.

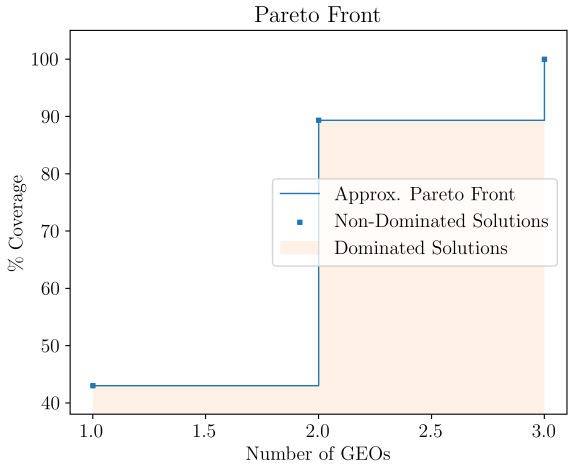


Fig. 6: Resultant approximate re-scaled Pareto front for the GEO problem.

3.2.2 Results

With the problem initialized as a single GEO occupying an arbitrary slot, we run MO-MCTS for 20,000 iterations, updating the Pareto set every 500 iterations, and resampling the root node every 1,000 iterations. Running on a single thread on a 2021 Apple MacBook Pro with an M1 Pro processor (hereby referred to as the M1 Mac), the algorithm completes running after approximately 9 minutes. Fig. 5 clearly shows that the hyper-volume indicator increases rather steadily over time. This is a clear indication that the solutions within the Pareto set are improving with respect to our objective functions over time. Furthermore in this plot, we see that the hyper-volume of the Pareto set quickly increases and plateaus around 7,000 iterations, at which point little progress is made since a sufficiently optimal set of

solutions have been found. Fig. 6 shows the resultant approximate Pareto front with one point representing the well-known solution of three equally spaced GEOs providing 100% coverage of a selected portion of the Earth's surface. Furthermore, Fig. 6 also includes an optimal placement of two GEOs providing approximately 90% coverage of the selection region of interest. Finally, we plot these placements in Fig. 7 to further verify that the solutions returned are as expected.

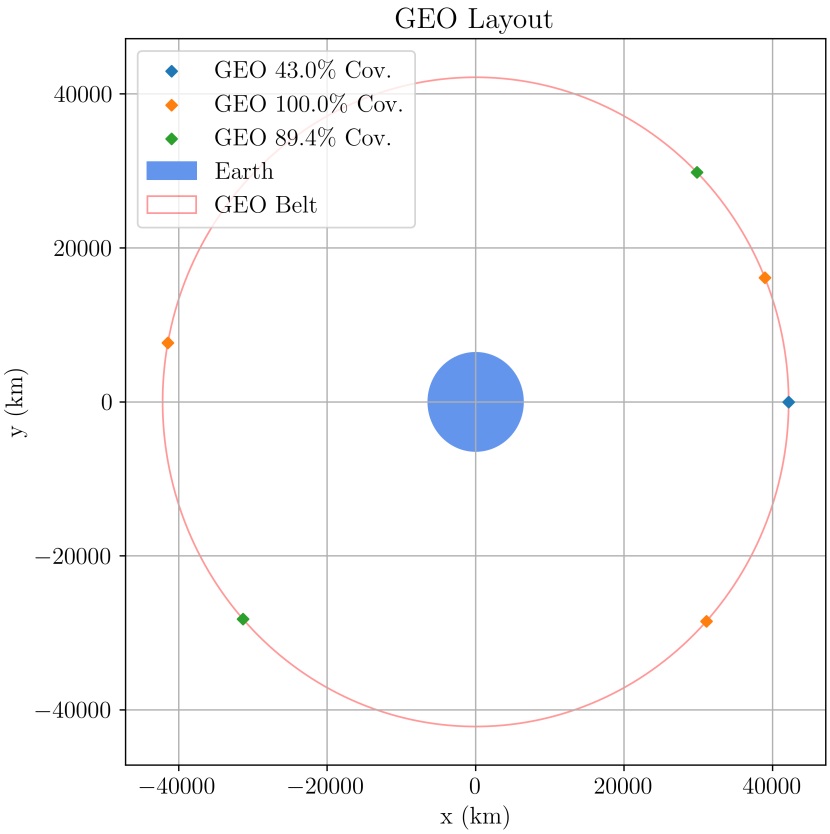


Fig. 7: Approximate Pareto-optimal GEO layouts.

3.3 The Molniya Problem

The Molniya orbit is well known for the amount of time it spends over the Earth's northern hemisphere. It is characterized by its inclination, argument of periapsis, radius of periapsis, and radius of apoapsis, as seen in Table 1. It was named after a series of Russian satellites that were first launched into the orbit in the early 1960s intending to provide effective communication to eastern parts of the Soviet Union. The Soviet Union also found this orbit to be highly effective for military purposes [17], and possibly used some of the early Molniya satellites to scout for cloud-free zones for its Zenit spy satellites [13]. Therefore, for this toy problem we aim to use MO-MCTS to find a combination of orbital elements that maximizes the time averaged coverage of the United States and Russia over a 48-hour time span using basic Keplerian dynamics. To make this problem multi-objective, we introduce an estimate of the Δv necessary to maneuver a spacecraft from a circular parking orbit at an altitude of 400km with an inclination of 0° to the state's current orbit.

Table 1: Characteristic orbital elements for a typical Molniya orbit.

Inclination (°)	ω (°)	$r_{per.}$ (km)	r_{ap} (km)
63.4	270	600	39,700

3.3.1 MDP Formulation

Once again, we begin by formulating the Molniya problem as a Markov Decision Process. We represent the state as a tuple with orbital elements we wish to alter:

$$[r_{\text{periapsis}}, r_{\text{apoapsis}}, i, \omega, \Omega]$$

where i is inclination, ω is argument of periapsis, and Ω is right ascension of the ascending node. Actions are defined as altering a single orbital element such that all orbital elements are within their defined ranges, $r_{per}, r_{ap} \in [400, 46000]$ km, and $r_{per} \leq r_{ap}$. Because these orbital elements take on continuous values, we discretize them to reduce the action space to a reasonable size. The reward function takes the current state tuple as input and returns a two-dimensional reward vector containing a scaled time-averaged coverage of the United States and Russia, $r_{\bar{c}}$, and a scaled Δv reward both in the interval $[0, 1]$ to be maximized. Since 100% time-averaged coverage is impossible given the position of the United States and Russia, we scale the coverage by a theoretical maximum of 21%. Like in the GEO problem, since we wish to minimize Δv, it is also scaled by a theoretical maximum value, here chosen as $\Delta v_{max} = 7$ km/s. Then the Δv reward to be maximized is defined as

$$r_{\Delta v} = 1 - \frac{\Delta v}{\Delta v_{max}}. \tag{4}$$

The Δv's calculated for this implementation are merely representative of the actual Δv needed, and as such are not optimized specifically for each individual transfer.

3.3.2 Results

We discretize the Molniya problem action space into 100 equally spaced orbital element alterations, and run MO-MCTS for 50,000 iterations, updating the Pareto set every 100 iterations, and resampling the root-node every 500 iterations. The algorithm completes running after approximately 8 hours on the M1 Mac. Fig. 8 shows the improvement of the hyper-volume indicator over iterations. Once again, we see an initial steep increase in the hyper-volume followed by an eventual plateau. This behavior indicates that either a sufficient amount of Pareto-optimal solutions have been found, or that the algorithm is unable to determine the remaining solutions.

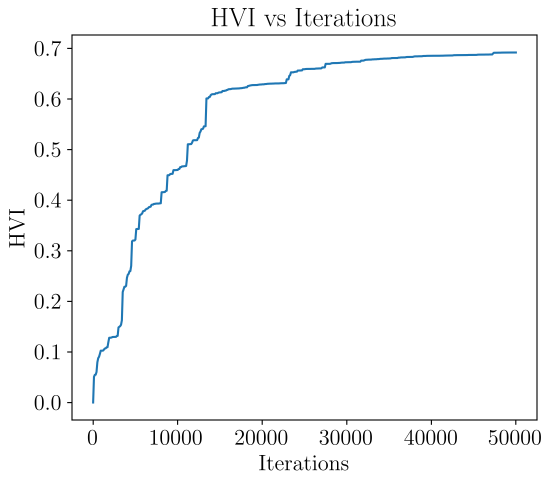


Fig. 8: MO-MCTS HVI vs iterations for the Molniya problem.

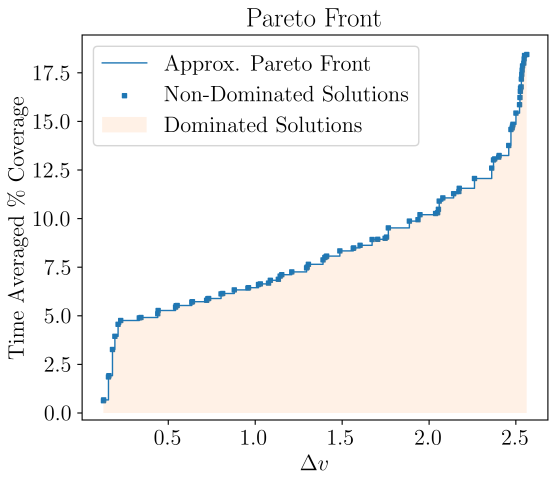


Fig. 9: Resultant approximate re-scaled Pareto front for the Molniya problem.

Looking at the approximate Pareto front in Fig. 9, we see a multitude of solutions deemed Pareto-optimal by the algorithm, showing the trade off between percentage coverage and Δv needed to access the orbit. Having an abundance

of solutions to a problem may be useful for some applications. However, if we are looking to the MO-MCTS algorithm for high-level assistance for designing architecture, we may not be as interested in the minuscule differences between optimal states. If anything, such a crowded Pareto set may be negatively affecting the ability of MO-MCTS to obtain a diverse set of solutions in an allowed amount of time. For this problem, crowding could be reduced by reducing the discretization factor, thus reducing the size of the action and state space. We will investigate the effect of crowding in future work.

Table 2 lists a sample of the resultant Pareto-optimal solutions with a Molniya-esque result in bold. Looking at the solution in bold, we see many similarities to the known Molniya orbit orbital elements. Of importance is that the MO-MCTS algorithm chose to raise inclination, raise apoapsis, and rotate the argument of periapsis such that the satellite spent as much of its simulation time over the United State and Russia as possible. Furthermore, in every other Pareto-optimal solution returned, the algorithm maintained this pattern of low periapsis, high apoapsis, high inclination, and argument of periapsis around 270°.

Table 2: Sample of outputs from MO-MCTS run with Molniya-esque orbit in bold.

$r_{periapsis}$ (km)	$r_{apoapsis}$ (km)	Inclination (°)	ω (°)	Ω (°)	% Coverage	Δv (km/s)
400.00	**40933.33**	**59.09**	**269.09**	**7.27**	**18.45**	**2.56**
400.00	39551.52	57.27	269.09	7.27	17.86	2.54
400.00	39551.52	50.91	269.09	7.27	15.86	2.52
400.00	34945.45	63.64	269.09	149.09	14.86	2.48
400.00	28957.58	66.36	269.09	116.36	13.03	2.37
400.00	24351.52	70.00	269.09	120.00	12.07	2.26
400.00	12836.36	66.36	269.09	101.82	9.05	1.75
400.00	9151.52	70.91	269.09	98.18	8.34	1.49
400.00	6387.88	75.45	269.09	116.36	7.26	1.21
400.00	5006.06	70.91	269.09	283.64	6.61	1.02
400.00	1321.21	75.45	269.09	116.36	4.92	0.34
400.00	860.61	28.18	269.09	152.73	0.64	0.13

4. CISLUNAR ARCHITECTURE DESIGN

4.1 Problem Formulation

Finally, we define a basic architecture design problem within the cislunar region. We define the problem as follows: what layouts of up to five observers in cislunar space provide maximal custody maintenance of a space object (SO) in a predefined Earth to Moon transfer reference trajectory while minimizing a total cost? There are numerous factors that go into custody maintenance, including sensor specifications, mission goals, SO properties, maneuver detection, solar phase angle, and many others. Thus, for this implementation, we will make a number of assumptions. First, circular restricted three-body problem (CR3BP) dynamics are used to propagate the reference trajectory and observer orbits. Second, SO in the reference trajectory is assumed to be spherical with specular and diffuse reflection constant for all wavelengths as in [26]. Third, each observer uses an off-the-shelf Finger Lakes Instruments Kepler KL4040 camera. Finally, it is assumed that each observer has perfect tracking of the SO with no streaking, and that on average the SO occupies the same amount of pixels in the sensor during each exposure.

4.1.1 Observer Orbits

In [26] it was found that periodic orbits in the CR3BP with periods resonant with the Earth-Moon synodic period (about 29.5 days) had significant advantages for observing cislunar space. As such, in this formulation we choose candidate observer orbits that have periods approximately resonant with the synodic period. We consider orbits with 1:1, 1:2, 1:3, 1:4, 2:1, 3:1, 3:2, 4:3, 5:2, 5:3, 5:4, and 6:5 resonance with the synodic period. These candidate orbits include Lyapunov, Halo, Axial, and Vertical orbits near the L1 and L2 CR3BP equilibrium points, Axial, Planar, and Long orbits near L4 and L5, Earth-Moon Distant Retrograde orbits [24], and a single GEO orbit.

As important as the resonance of the observer orbit selected, [26] also found that initial observer orbit phasing played a significant part in observation quality. Combined with the initial epoch date of the simulation, this phasing describes the relative geometry between the observer and the solar phase angle. In a configuration with multiple observers in orbits with varying degrees of resonance with the synodic period, we expect to generate a vast set of observing geometries.

To ensure that we consider as many relative geometric layouts as possible, it is helpful to calculate how long a configuration would take to return to its initial layout. For a configuration with three observers in orbits with 1:2, 3:2, and 5:3 resonance with the synodic period, as seen in Fig. 10, we must propagate the system for six synodic periods for each observer to return to its initial position at the same time. In general, with for a configuration with j observer orbits of resonance $m_1 : n_1, ..., m_j : n_j$, the period of the layout is $lcm(n_1, ..., n_j)$ synodic periods, where $lcm(\cdot)$ is the least common multiple. Of all the orbits considered in this work, the longest possible layout's period is 60 synodic periods. As such, we consider the state history of each observer for 60 synodic periods in order to effectively and equally compare the performance of each configuration.

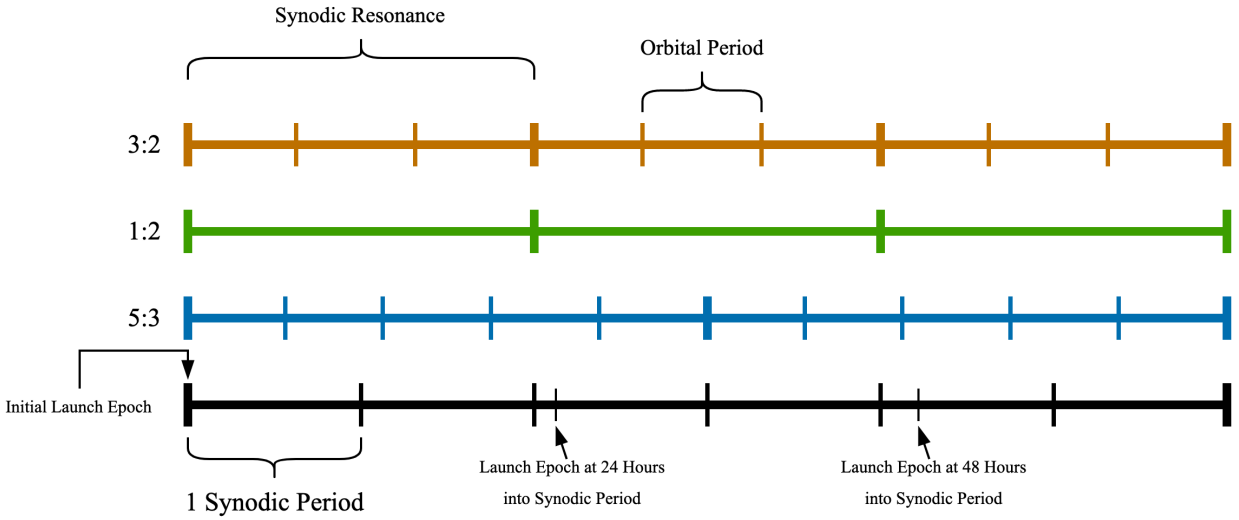


Fig. 10: Resonance of a cislunar architecture configuration consisting of three observers with 3:2, 1:2, and 5:3 resonance with the Earth-Moon synodic period. Every other synodic period the architecture configuration is evaluated at a different epoch, each separated by 24 hours relative to the synodic period. The pattern repeats for 60 synodic periods.

4.1.2 State Representation

We define an observer with the following tuple:

$$Observer = [Orbital\,Family,\ Orbit\,Index,\ Initial\,Phasing,\ Telescope\,Aperture]$$

where *Orbital Family* is the name of the periodic orbit family the observer occupies, *Orbit Index* specifies the orbit within the family that the observer occupies, $Initial\,Phasing \in [0,1]$ specifies the initial position of the observer along the orbit as a fraction of its period, and $Telescope\,Aperture \in \{200\text{mm}, 300\text{mm}, 500\text{mm}\}$ specifies the size of the observer's optics. Then, a state representing a configuration is defined as a tuple of observers:

$$State = [Observer_1,\ Observer_2,\ \ldots]$$

Therefore, a state represents a particular architecture configuration and provides all the information necessary to simulate its effectiveness.

4.1.3 Actions

Once again with the idea in mind that actions are merely a means of getting from one state to another, we define the action space for this problem much like that from the GEO problem. We define an action as adding a new observer to the architecture configuration in a specific orbit, with a specific initial phasing, and a specific telescope aperture. We place no restrictions on observers occupying orbits within the same families, nor occupying the same orbit with different phasings. Restrictions like these could be easily implemented by limiting the action space at each node but were not considered for this demonstration.

4.1.4 Objective Functions and Rewards

Finally, we define two metrics with which to determine the quality of a particular architecture. First, we define a cost metric that takes into account the number of observers in the layout and the size of their respective apertures. We define the cost of each telescope apertures as follows:

$$
\begin{aligned}
C_{200mm} &= 1 \\
C_{300mm} &= 2.25 \\
C_{500mm} &= 6.25
\end{aligned}
$$

We then define the reward in the interval $[0,1]$ to be maximized as follows:

$$
1 - \frac{C_{200mm}n_{200mm} + C_{300mm}n_{300mm} + C_{500mm}n_{500mm}}{C_{500mm}n_{max}} \tag{5}
$$

where n_{200mm}, n_{300mm}, and n_{500mm} are the number of observers with telescope apertures of 200mm, 300mm, and 500mm, respectively, and n_{max} is the maximum allowed number of observers in the constellation, here chosen to be 5.

Next, we define a process by which we determine average custody of the SO in its reference trajectory as launched at multiple epochs during the Earth-Moon synodic period. To start, we obtain solar phase angle history in the CR3BP frame for approximately 60 synodic periods from JPL's SPICE Toolkit through `SpiceyPy` [1]. We also pre-propagate each observer candidate orbit to obtain a state history with time step size matching that of the SO reference trajectory and the solar phase angle history.

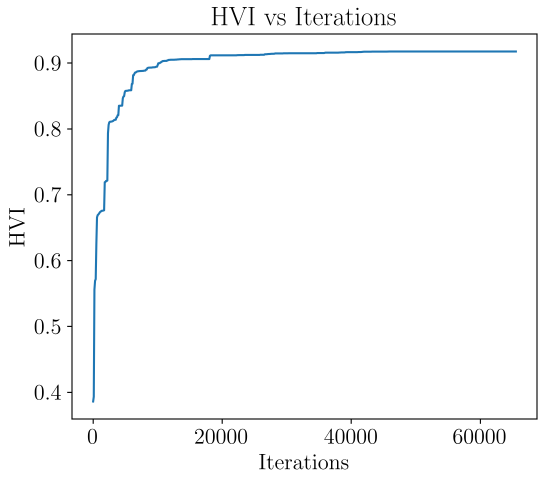


Fig. 11: MO-MCTS HVI vs iterations for the cislunar architecture problem.

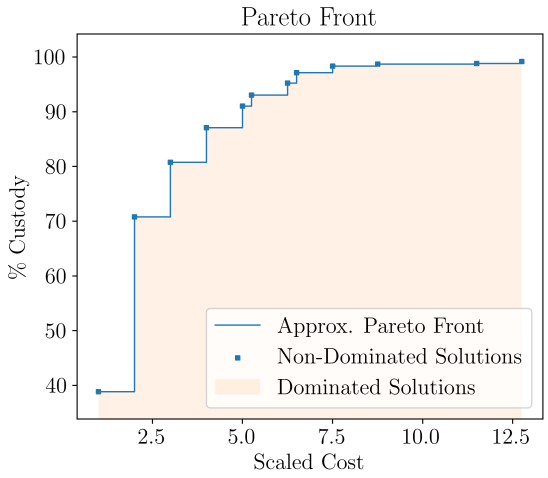


Fig. 12: Resultant approximate re-scaled Pareto front for the cislunar architecture problem.

We consider approximately 30 different initial epochs throughout the total 60 synodic periods of simulation time, with each separated in the synodic period by about 24 hours as seen in Fig. 10. By doing this we can evaluate the performance of a configuration with varying degrees of SO illumination and occlusion by the Earth and Moon. Then during each reference trajectory propagation we calculate the photometric SNR [7] obtained by each observer and determine if any reach the minimum SNR for detection, here chosen to be 6. So, for each reference trajectory propagation at each epoch, a total percentage custody maintained is returned. We take the average of percentage custody from each simulation and define this as our second reward. Thus, an optimal layout may be one that maximizes the custody maintained at various epochs during the Earth-Moon synodic period.

4.2 Results

We run MO-MCTS with the cislunar architecture design formulation for 65,000 iterations, updating the Pareto set every 100 iterations, and resampling the root node every 500 iterations. The algorithm takes approximately 8 hours to complete running on the M1 Mac. Fig. 11 shows the change in hyper-volume indicator over iterations, showing that after about 10,000 iterations, solutions found are only slightly improving with respect to the objective functions. Fig. 12 shows a fairly crowded approximate Pareto front, with many solutions past scaled cost of 5 showing only slight improvement in custody maintained over the simulation epochs. This behavior is reminiscent of the GEO problem Pareto front in Fig. 6, where a point came when improvement in coverage was small compared to the relative cost increase.

(a) Configuration statistics from Fig. 13a

Total Custody (%)	Total Cost	Min. Custody (%)	Max. Custody (%)
99.198	12.75	96.203	100
	Epoch	2023 NOV 02 23:00:00	2023 JAN 01 23:00:00

(b) Configuration statistics from Fig. 13b

Total Custody (%)	Total Cost	Min. Custody (%)	Max. Custody (%)
70.801	2.0	56.962	89.873
	Epoch	2023 MAY 03 23:00:00	2026 JUL 06 00:00:00

(c) Configuration statistics from Fig. 13c

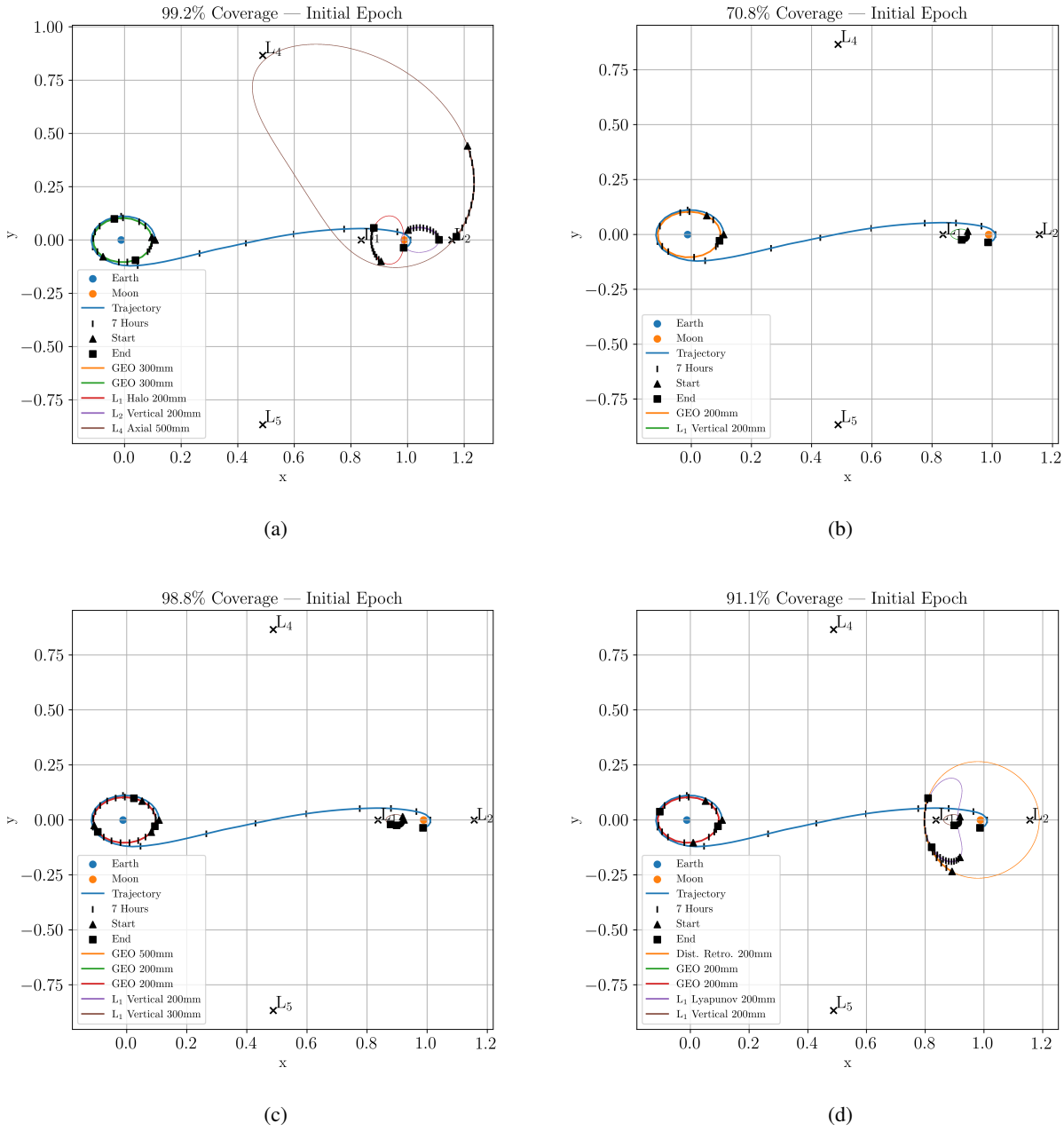
Total Custody (%)	Total Cost	Min. Custody (%)	Max. Custody (%)
98.819	11.5	91.139	100
	Epoch	2025 SEP 04 00:00:00	2022 MAR 02 23:00:00

(d) Configuration statistics from Fig. 13d

Total Custody (%)	Total Cost	Min. Custody (%)	Max. Custody (%)
91.055	5.0	82.278	98.734
	Epoch	2024 JAN 02 23:00:00	2025 NOV 03 23:00:00

Table 3: Configuration statistics from Pareto-optimal outputs in Fig. 13.

Fig. 13 shows a sample of the Pareto-optimal solutions found from the MO-MCTS algorithm. These solutions are further detailed in Table 3, including the epochs at which minimum and maximum custody of the SO in the reference trajectory is maintained. Of initial interest is the overwhelming tendency of the algorithm to pick multiple GEO observers in all solutions providing custody maintenance for the initial portion of the reference trajectory. This result is as expected, since observers in Lagrange point periodic orbits will have difficulty observing a space object as it starts its trajectory in a near-Earth orbit. Fig. 14 further displays this behavior and demonstrates the ability of the MO-MCTS algorithm to piece together observers in different orbits to maintain maximal custody of the space object in the reference trajectory.



(a) (b)

(c) (d)

Fig. 13: Four resultant Pareto-optimal cislunar architecture layouts with trajectory starting at the initial simulation epoch.

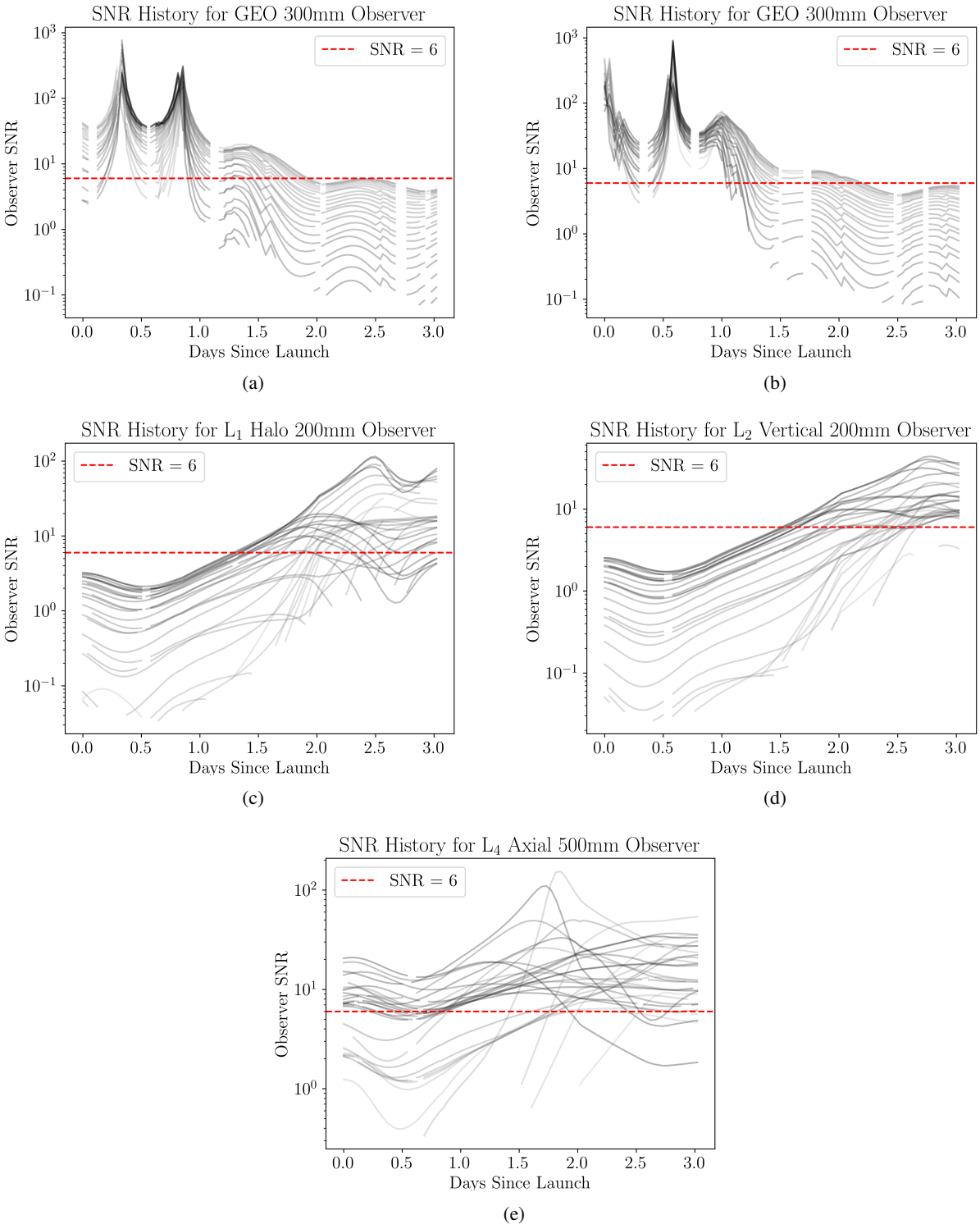


Fig. 14: SNR history for all epochs from architecture layout in Fig. 13a

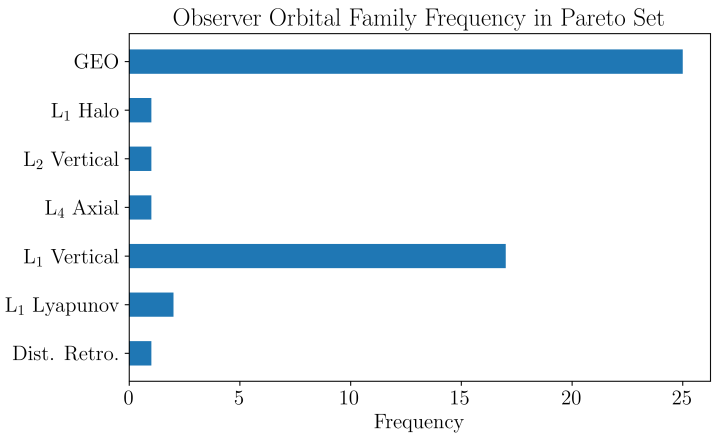


Fig. 15: Frequency of orbit families used in solutions in the Pareto set returned by MO-MCTS.

Fig. 15 presents a histogram of the types of orbits found optimal to be used in the Pareto-optimal configurations. Interestingly, MO-MCTS finds that in only one case is it Pareto-optimal to include an observer in an L4 orbit, and in no cases does it find it optimal to place an observer in an L5 orbit. This is somewhat expected given this particular reference trajectory and the assumptions made about perfect tracking. In general, many of the Pareto-optimal solutions returned from MO-MCTS place observers in the same orbital families while making slight variations to the initial phasing and telescope aperture. In future implementations where perfect tracking may not be assumed, we might expect to see a greater frequency of including observers in L4 and L5 orbits. Observers in these orbits would intuitively provide an architecture configuration with a more diverse set of viewing angles to the SO, thus increasing the amount of information needed for custody maintenance or maneuver detection [12], [9]. Our current formulation, however, does not include metrics to account for the quality of a detection by an observer.

As a representative cislunar architecture problem, these results serve mainly to demonstrate the ability of our MO-MCTS implementation to obtain approximate solutions to a highly nonlinear mixed integer programming problem. As such, there is significantly more nuance that goes into characterization, custody maintenance, and maneuver detection. Future work with the MO-MCTS algorithm applied to cislunar architecture design will consider more of these nuances, including the diversity of viewing angles and angular rates, full state observability, and others inspired by the literature [26], [9], [12], [10].

5. CONCLUSION

As the development of cislunar space continues, it becomes increasingly important to develop effective and optimal architecture for conducting space domain awareness throughout the region. However, given the breadth of complexities associated with cislunar space, and the necessity of considering multiple design variables and objectives, the problem quickly becomes intractable. To obtain approximate Pareto-optimal solutions to these problems, we introduced a novel multi-objective Monte Carlo Tree Search algorithm. This algorithm was validated through two problems with known solutions, and then used in a representative cislunar architecture design problem. In this final problem, MO-MCTS returned a Pareto-optimal set of configurations of observers in cislunar space that maximized percentage custody maintained of a space object on a known trajectory while minimizing a total cost. The versatility of this algorithm opens up various avenues for future research, including implementation with more highly dimensional architecture design problems. Furthermore, by implementing methods such as multiprocessing in future work we expect to make significant progress on minimizing computation time and resources.

6. ACKNOWLEDGMENTS

The authors would like to give special thanks to Ball Aerospace for funding this project, and to Patrick Handley, Shez Virani, and Sam Fedeler for their assistance in numerous aspects of this research.

7. REFERENCES

[1] Andrew M Annex, Ben Pearson, Benoît Seignovert, Brian T Carcich, Helge Eichhorn, Jesse A Mapel, Johan L Freiherr Von Forstner, Jonathan McAuliffe, Jorge Diaz Del Rio, Kristin L Berry, et al. Spiceypy: A pythonic wrapper for the spice toolkit. *Journal of Open Source Software*, 5(46):2050, 2020.

[2] Fangyun Bai, Xinglong Ju, Shouyi Wang, Wenyong Zhou, and Feng Liu. Wind farm layout optimization using adaptive evolutionary algorithm with monte carlo tree search reinforcement learning. *Energy Conversion and Management*, 252:115047, 2022.

[3] Pietro Belotti, Christian Kirches, Sven Leyffer, Jeff Linderoth, James Luedtke, and Ashutosh Mahajan. Mixed-integer nonlinear optimization. *Acta Numerica*, 22:1–131, 2013.

[4] J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

[6] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, volume 4, pages 216–217, 2008.

[7] Ryan D Coder and Marcus J Holzinger. Multi-objective design of optical systems for space situational awareness. *Acta Astronautica*, 128:669–684, 2016.

[8] Laura Duffy and Jim Adams. Cislunar systems architectures survey paper. In *2022 IEEE International Systems Conference (SysCon)*, pages 1–8. IEEE, 2022.

[9] Samuel Fedeler, Marcus Holzinger, and William Whitacre. Sensor tasking in the cislunar regime using monte carlo tree search. *Advances in Space Research*, 2022.

[10] Erin E Fowler, Stella B Hurtt, and Derek A Paley. Observability metrics for space-based cislunar domain awareness. In *AAS/AIAA Astrodynamics Specialist Conference*, 2020.

[11] C Frueh, K Howell, K DeMars, S Bhadauria, and M Gupta. Cislunar space traffic management: Surveillance through earth-moon resonance orbits. In *8th European Conference on Space Debris*, 2021.

[12] Jesse A Greaves and Daniel J Scheeres. Observation and maneuver detection for cislunar vehicles. *The Journal of the Astronautical Sciences*, 68(4):826–854, 2021.

[13] Bart Hendrickx. A history of soviet/russian meteorological satellites. *Journal of the British Interplanetary Society*, 57(1):56, 2004.

[14] Daniel Hennes and Dario Izzo. Interplanetary trajectory planning with monte carlo tree search. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[15] MJ Holzinger, CC Chow, and P Garretson. *A Primer on Cislunar Space*. Air Force Research Laboratory, 2021.

[16] Xi Huang, Ziyu Shao, and Yang Yang. MIPS: instance placement for stream processing systems based on monte carlo tree search. *CoRR*, abs/2008.00156, 2020.

[17] Stephen Barry Johnson et al. *Space Exploration and Humanity: A Historical Encyclopedia [2 volumes]: A Historical Encyclopedia*, volume 1. ABC-CLIO, 2010.

[18] Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European conference on machine learning*, pages 282–293. Springer, 2006.

[19] Sang-Uk Lee, Jae-Hoon Kim, and Seong-Pal Lee. Communications satellite system by using moon orbit satellite constellation. *Journal of Astronomy and Space Sciences*, 20(4):313–318, 2003.

[20] R Timothy Marler and Jasbir S Arora. Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, 26(6):369–395, 2004.

[21] Nery Riquelme, Christian Von Lücken, and Benjamin Baran. Performance metrics in multi-objective optimization. In *2015 Latin American computing conference (CLEI)*, pages 1–11. IEEE, 2015.

[22] Leonardo Rossi, Mark HM Winands, and Christoph Butenweg. Monte carlo tree search as an intelligent search tool in structural design problems. *Engineering with Computers*, pages 1–18, 2021.

[23] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[24] Thomas R Smith and Natasha Bosanac. Motion primitives summarizing periodic orbits and natural transport mechanisms in the earth-moon system. In *AAS/AIAA Astrodynamics Specialist Conference*, 2020.

[25] Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. Monte carlo tree search: A review of recent modifications and applications. *arXiv preprint arXiv:2103.04931*, 2021.

[26] Jacob K Vendl and Marcus J Holzinger. Cislunar periodic orbit analysis for persistent space object detection capability. *Journal of Spacecraft and Rockets*, 58(4):1174–1185, 2021.

[27] Weijia Wang and Michele Sebag. Multi-objective monte-carlo tree search. In *Asian conference on machine learning*, pages 507–522. PMLR, 2012.

[28] Di Weng, Ran Chen, Jianhui Zhang, Jie Bao, Yu Zheng, and Yingcai Wu. Pareto-optimal transit route planning with multi-objective monte-carlo tree search. *IEEE Transactions on Intelligent Transportation Systems*, 22(2):1185–1195, 2020.

[29] Ming Xu, Jinlong Wang, Shengli Liu, and Shijie Xu. A new constellation configuration scheme for communicating architecture in cislunar space. *Mathematical Problems in Engineering*, 2013, 2013.