(Preprint) AAS 20-686

BROAD TRAJECTORY SEARCHES USING MONTE CARLO TREE SEARCH WITH THE INCLUSION OF $\triangle$VEGA TRAJECTORIES

Burton A. Yale[*], Jehosafat J. Cabrera[†], Rohan D. Patel[‡], and Navid Nakhjiri[§]

Multiple flybys of the inner planets and the application of V_∞ leveraging are essential trajectory design techniques to reduce the required launch energy for interplanetary missions. These trajectories are often difficult to formulate and require extensive computational resources. However, this problem can be classified as a sequential planning task which can be solved by a Monte Carlo Tree Search (MCTS) method. In this paper, a MCTS algorithm is developed and tested with a focus on reducing required ΔV from the spacecraft. This method balances exploration and exploitation of the search space, and the algorithm's performance is assessed by tweaking the selection policy parameter. Optimizations of several cases are studied to prove the feasibility of the tree search results. The algorithm will allow for inner-planetary flyby search planning for outer planetary missions.

INTRODUCTION

Broad trajectory searches are one of the first steps in mission planning, allowing for the selection of candidates given a list of constraints and a search space. As is the nature with combinatorial problems, every additional flyby adds another dimension to the search space. Brute force methods lead to expensive but thorough results. By pruning possible choices before the each selection, the overall computation time can be decreased, at the expense of losing possible results. There have been various approaches to this method, from evolutionary/genetic algorithms, to particle swarm optimization, to enumerated searches, all with their own benefits and drawbacks. The method of interest for this algorithm was enumerated searches, a subset of grid searches, as recent work has proven their ability to quickly and accurately find multi-leg interplanetary trajectories.[1]

Monte **C**arlo **T**ree **S**earch (MCTS), is a heuristics-based adaptation of the enumerated search, has the ability to both explore and exploit its environment. This allows the algorithm to find the narrow bands where multiple planet combinations are possible and exploit them to complete its goal. As such, larger search spaces can be employed without exponential growth in computation time.

This algorithm will deliver a set of planetary sequences and the dates associated that meet mission criteria, such as destination, launch windows, total fuel budgets, etc. These results can then be passed on to the next phase of the design, optimization

[*]Undergraduate Student, Aerospace Engineering, Cal Poly Pomona, 3801 W Temple Ave, E-mail: bayale@cpp.edu
[†]Undergraduate Student, Aerospace Engineering, Cal Poly Pomona, 3801 W Temple Ave, E-mail: jehosafatc@cpp.edu
[‡]Undergraduate Student, Aerospace Engineering, Cal Poly Pomona, 3801 W Temple Ave, E-mail: rohanpatel@cpp.edu
[§]Assistant Professor, Aerospace Engineering, Cal Poly Pomona, 3801 W Temple Ave, E-mail: nnakhjiri@cpp.edu

BACKGROUND

Interplanetary mission design is an iterative process. First, a sequence of encounter bodies is selected and is followed by evaluating all possible trajectories for that specified sequence. Due to the increasing complexity of gravity assists and combinatorial solutions, a low-fidelity tool is generally utilized first to find regions of interest. This is then followed by a high-fidelity tool which searches through prospective and highly valued regions. In literature, this is often referred to as pathfinding and path-solving respectively [3]. Pathfinding is a sequence optimization problem that provides with a diverse set of mission options. Path-solving, on the other hand, embodies a single-leg transfer trajectory optimization problem where two types of transfer models, low-thrust and multi-impulse, are mainly considered [Deep Networks as Approximators of Optimal Transfers Solutions in Multitarget Missions]. Low thrust trajectories can be optimized using direct or indirect methods. Direct methods convert the optimal control problem to a parameter optimization problem through discretization. The optimal solution is then found through nonlinear programming which requires a long computation time. Indirect methods involve solving a two-point boundary value problem using calculus of variations [Practical techniques for low thrust trajectory optimization with homotopic approach]. Artificial Intelligence techniques have been proposed on numerous occasions over the past few decades by a diverse number of scientists as pathfinding algorithms and routines to tackle the multi impulse optimization problem. Some of these techniques include evolutionary algorithms, machine learning, evolutionary neuro-controllers, and tree searches methods [machine learning and evolutionary techniques in interplanetary trajectory design].

Evolutionary algorithms are optimization routines that make use of heuristics and Darwinian evolution to solve complex optimization problems. Three of the most popular type evolutionary algorithms are genetic algorithms, Differential Evolution (DE), and Particle Swarm Optimization (PSO). DE has been successfully used as an optimization routine to design Halo Orbits and optimal trajectory to Halo Orbits in [Precise Halo orbit Design and Optimal Transfer to Halo Orbits from Earth Using Differential Evolution] and as a interplanetary mission design algorithm [Interplanetary Mission Design Using Differential Evolution]. The Cassini and Galileo missions were considered as case studies to analyze the performance of the algorithm. One each, the DE routine was able to come within appropriate bounds of both trajectories and found minima to the flight constraints applied. PSO was used by Zhuang and Huang in combination with Legendre pseudo-spectral method for solving time-optimal trajectory planning problems. The PSO algorithm was robust enough to handle random initial values but was deemed slow at convergence. Thus, when the change in fitness function of the algorithm is smaller than a predefined value, the searching algorithm is switched the Legendre pseudospectral method for fast convergence [Time-Optimal Trajectory Planning for Under- actuated Spacecraft Using a Hybrid Particle Swarm Optimization Algorithm].

The application of Machine Learning algorithms to the design of interplanetary trajectory has not been extensively researched as evolutionary or tree search techniques have. The less applicability of these methods to the problems encountered during in trajectory design cause them to not be as effective for use. Machine Learning algorithms require data sets as initializers to the problem. The lack of which throughout the aerospace community as open source information restrict the ability to use these techniques [Machine learning and evolutionary techniques in interplanetary trajectory design].

Grid-search, for instance has been previously used as a search algorithm to find possible trajectories to KBOs [4]. Grid search is a type of search algorithm with the ability to map the entire search space so that there are no regions of interest missed. This method has been used to properly

parametrize the time of flight between encounter bodies. In this manner, Earth having a period of 365 days will have the same number of possible encounters as Jupiter with a period of approximately 12 years [2]. Thus, the output grid will be that of an angular grid as opposed to a cartesian grid. This type of search algorithm, however, can be expensive in terms of computing power and time. To alleviate both of these constraints, heuristics can be implemented into the program.

Beam Search (BS), a heuristic tree search algorithm, can be used as a searching criterion to find a sequence of planets from which gravity-assist from [4]. BS uses the method of Breadth-First Search (BFS) to search the tree of possibilities. BS builds each layer of the tree and orders the nodes in accordance to their heuristic cost. However, it only chooses those nodes with a maximum value to build from. Depth-First Search (DPS), alternatively, is used as a searching criterion to traverse down the tree [5]. Opposite to BFS, DPS does not search the tree at every level but rather explores a branch until the termination criteria is met. After which, moving in reverse updates the branch and moves towards the root node to start the process again. Both the BFS and DPS, search and prune the search- space consecutively. Using the Lazy Race Tree Search, the search space is pruned, and nodes ranked through the time of flight [5]. The use of heuristics avoids finding redundant solutions and increases the efficiency of the method used. Izzo et. al. [2] proposed the use of the Monte Carlo Tree Search to find fast solutions to interplanetary trajectories.

Monte Carlo methods have been suggested extensively for games with a random behavior and minimal observability. The nature of the Monte Carlo methods, however, allow them to be applied to deterministic games with faultless information [A survey of monte Carlo tree search methods]. Starting from the initial state, a large amount of games and actions are simulated until the end of the game. In the majority of cases, actions are chosen at random with no link to the game theory. What this means, is that even if the iterative process is executed for an extended period of time, the move selected and path selected might not the most optimum.

The search sensitivity criteria of DPS, BS, and BFS are explored in conjunction with that of the MCTS and compared. These searches, however, did not take into account Deep Space Maneuvers (DSMs) and led to trivial solutions when trying to validate the Rosetta Trajectory. The validation and correct recreation of this mission will be the basis of the algorithm taking into account DSMs to fully model Leveraging Orbits.

MONTE CARLO TREE SEARCH IMPLEMENTATION

The tree is built upon a set of individual nodes, that are all connected to each other through their parent and children, much like a family tree. Each node has an corresponding state to represent a leg in the tour of a spacecraft, with a planetary body, and the time at which it is encountered. As the algorithm builds the tree, the process can be characterized by four essential steps: selection, expansion, simulation, and backpropagation, as depicted in Figure 1.

At the initialization of the tree, an array of launch nodes is specified based upon user input. If the sample time span is larger than a year, the number of corresponding nodes will be modified in order to even sample the search space. From this point, the algorithm will begin its core loop to search out possible trajectory sequences, until the iteration budget is depleted.

Selection

At the start of each iteration, the program initializes at the root of the tree $id = 0$, and from its children, select the child with the highest associated UCB1 value. With X being the future reward

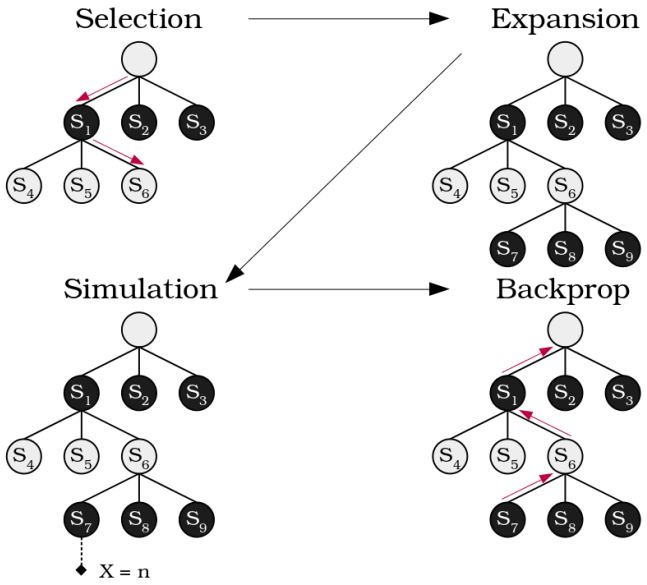

Figure 1. 4 Main Steps for the Creation of a Monte Carlo Tree

from exploring the node, N being the number of visits the node has recieved, and n being the number of visits the parent node has recieved.

$$\text{UCB1 Node Value: } X + C_p\sqrt{\frac{\ln n}{N}} \tag{1}$$

In the case of C_p, the exploration-exploitation parameter, a value of $C_p = \frac{1}{\sqrt{2}}$ was selected due to its performance demonstrated by Hennes.[1] This reward policy allows the tree to explore all available nodes, while still exploiting nodes that return high enough future rewards.

Once a child node has been selected, this process will repeat until a leaf node, a node with no children, has been reached. As the function navigates down a branch, the string of nodes leading to a leaf, if all of a selected nodes children are deemed terminal, such as exceeding the total ΔV budget, the current node is also declared terminal. When this condition is met, the select function will restart its search process at root of the tree. With this implementation, the tree is able to further constrict the search space, reducing computation time. Once the selection process leads to a leaf node, the node id will be passed to the expansion function for the next step of building the tree.

Expansion

Before the creation of the new set of nodes, the function will first look at the state of the current node being expanded from as a base. As planetary motion is one of conic sections, cartesian grid position sampling is not viable, thus angular divisions are required to efficiently sample a planets position. For each of the possible planets to visit, the following policy will be implemented.

$$E = \begin{pmatrix} n*(\tau_1+\tau_0)+t_0 \\ \vdots \\ m*(\tau_1+\tau_0)+t_0 \end{pmatrix} \text{ for } \begin{cases} n = 0.1, & m = 1.0 & \text{if } a_1 < 2 \text{ AU} \\ n = 0.05, & m = 0.25 & \text{if } a_1 \geq 2 \text{ AU} \end{cases} \tag{2}$$

For the ephemeris time array E, the ephemeris time of the child node is dictated by the period child node's planet, τ_1, and the period of the parent node's planet, τ_0. This array is bounded using the parameters n and m, which are defined by the semi-major axis of the child node's planet, a_1. The values for n and m for the inner planets, $a_1 < 2$ AU, were selected upon their performance in Hennes.[1] These values were not ideal for travel to the outer planets, as it lead to Lambert arc time of flights in excess of 12 years for travel to Jupiter. Additionally, if Jupiter were to be used for a graviational assist to another outer body, faster transfers were required. Based on their performance in simulations, the values of $n = 0.05$ and $m = 0.25$ lead to more transfers that satisified mission requirements.

With the upper and lower bounds of the array set, an additional parameter, defined by the user, d specifies the length of the array, also defining the resolution of the time steps. Additionally, this allowed for dynamic angular detail in the ephemeris time arrays. The inner planetary sequences are more sensitive to small time disturbances, and could lead to missed transfers. Thus, the fixed array length made the time steps smaller for inner planets, and coarser for the outerplanets, which were much less sensitive to the time differences of each node. For the purpose of this paper, a value of $d = 16$ was found to be sufficient for all calculations as a balance between computation time and detail.

Furthermore, children nodes that revisit the same parent planet have additional epheremis node tweaks to allow for better performance. n and m have their values set to 0.9, and 1 accordingly, for the set values in k. This allows the tree to search differing, non-impulsive, leveraging orbits of 2:1, 3:1, and 4:1 resonances. At this time, only leveraging sequences with revolutions of less than 1 were allowed.

$$E = \begin{pmatrix} k * n * \tau + t_0 \\ \vdots \\ k * m * \tau + t_0 - 3 \text{ days} \end{pmatrix} \quad \text{For } K \text{ in } [2, 3, 4] \tag{3}$$

Once the list of states has been established, a new set of children nodes are created to match all combinations of ephemeris times and planetary NAIF ID pairs. In order to further prune the tree to reduce unnecessary calculations, Lambert arcs are calculated to each state pair to evauluate their associated ΔV cost. Any node that exceeds the mission budget is considered terminal and not used in any further exploration.

In order to calculate mission ΔV usage, the simulation uses powered flybys, as characterized by [Wei] **MORE HERE LATER**

$$a_{\text{in/out}} = -\frac{\mu_p}{v^2_{\infty-\text{in/out}}}$$

$$\delta = \cos^{-1}\left(\frac{\vec{\mathbf{v}}_{\infty-\text{in}} \cdot \vec{\mathbf{v}}_{\infty-\text{out}}}{v_{\infty-\text{in}} \cdot v_{\infty-\text{out}}}\right)$$

$$f =$$

$$\frac{\mathrm{d}f}{\mathrm{d}e_{out}} =$$

$$r_p = a_{in}(1 - e_{in}) = a_{out}(1 - e_{out})$$

$$\Delta V_{fb} = \left| \sqrt{v^2_{\infty-\text{in}} + \frac{2\mu_p}{r_p}} - \sqrt{v^2_{\infty-\text{out}} + \frac{2\mu_p}{r_p}} \right|$$

After the states have been created, and the nodes pruned, the expand function selects the new node with the lowest associated ΔV cost to continue onto the Simulation step.

Simulation

On the tree's arrival to an unvisited leaf node, the node's future expected reward, X, must be calculated. A group of temporary state pairs are expanded from the leaf, as starting point for the random exploration. For each new node, the program will randomly walk through additional state pairs, generated in the same fashion as in the expand function, calculating the ΔV in the process, until a termination state is reached. When either the budget is expended or the target planet is reached, the associated cost is calculated using the equations below.

$$X = \max\left(0.1 \cdot l, \ \frac{\Delta V_{budget} - \Delta V_{used}}{\Delta V_{budget}}\right) \tag{4}$$

Upon termination, the reward is balanced between two criteria: ΔV usage, and number of flybys completed, l. In the case were the final desination is not reached within the allowed budget, the algorithm will still provide a small benefit to a simulation completing Lambert arcs. After all temporary states have been simulated from, the reward returned from each is averaged and output from the function to be passed back up the branch.

Backpropagation

At the end of the core MCTS loop, the backpropagation step communicates the expected reward from a leaf node back up the branch to reflect a more accurate reward of a branch. The X of each node in the branch is adjusted based of a weighted average of its previously observed rewards.

$$X_{node,new} = \frac{X_{node,old} \cdot N_{node} + X_{new}}{N_{node} + 1} \tag{5}$$

Once the cost is propagated up the tree, the loop restarts at another iteration, exploring a new set of state pairs. This process will repeat until the computation budget is depleted. At completion, the tree will evaluate all leaf nodes that meet the mission constraints to passed to the next step in the trajectory design process, optimization.

ΔVEGA MANEUVER IMPLEMENTATION

A ΔVEGA orbit, also referred to as a leveraging orbit, launches from Earth with the intent to gravity assist off the body for a higher post-flyby heliocentric energy. Leveraging orbits are classified by their nominal resonance period multiple with respect to Earth's heliocentric orbit, and we will refer to this number as "k." For example, a leveraging orbit that has a period roughly 3 times as large as Earth's orbit will have a $k = 3$ which is represented as a 3:1 ΔVEGA trajectory. The actual period of these orbits will vary either greater than or less than the nominal time due to flying by the body at a different Earth heliocentric true anomaly. Trajectories with a larger period are referred to as k:1^{+}ΔVEGA and those with a shorter are k:1^{-}ΔVEGA trajectories. To modify the flyby Earth true anomaly a maneuver at the aphelion (deep space maneuver) is executed.

The deep space maneuver (DSM) and subsequent Earth launch characteristics are calculated before the tree generation in order to reduce the required number of computations within the search.

We approximate the ΔV requirements for both maneuvers, and their inclusion reduces the discontinuous trajectory endpoint velocities needed to patch Earth-Earth flyby sequences. Having the required aphelion ΔV can also aid the optimization process initial guess. A lookup table sorted by the nominal resonance multiple (k) and encounter true anomalies (θ_E) contains the resulting leveraging orbit properties and maneuver magnitudes. Due to being a rough approximation, the circular Earth orbit and coplanar trajectories assumptions are used. Finding the ΔVEGA orbit parameters for a specific encounter true anomaly and k begins by assuming a nominal orbit period launch and its associated V_∞. The state elements are computed at Earth and the resulting aphelion radius and velocity are found. Because the intercept true anomaly is fixed, its location and the time it takes to reach the point are known. Eq. (6) is the difference in time from the DSM maneuver location to the Earth gravity assist (EGA) point where T_E is the orbital period of Earth.

$$dt = kT_E \pm T_E(\theta_E/2\pi) \tag{6}$$

A Lambert arc is computed between these points and the resulting initially velocity change is used

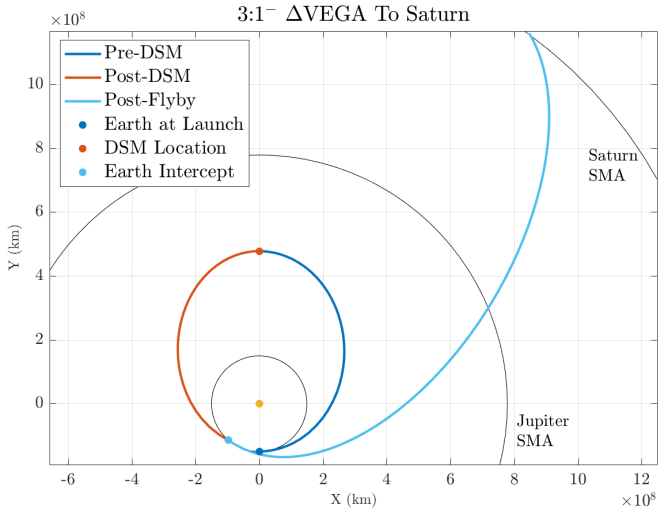

Figure 2. Example of a computed 3:$1^-$$\Delta$VEGA trajectory to Saturn's semi-major axis. The launch V_∞ is 6.97 km/s and the required DSM ΔV is 0.39 km/s. The EGA flyby altitude was constrained to 200 km, which yielded the highest post flyby energy.

to find the DSM vector. The final velocity vector is assumed to be the heliocentric velocity of the leveraging orbit at the EGA. The relative velocity, $\vec{V_\infty}$ is computed and a planar flyby of Earth can now be calculated from the tree search algorithm. Fig. 2 illustrates an example leveraging orbit being calculated for θ_E = 40.7°. An energy maximizing flyby and propagation to the new aphelion is added to the end of the ΔVEGA orbit. From testing, we noticed that as $\mid \theta_E \mid$ increased, a normal component of the DSM ΔV appeared and grew larger. To limit the ΔV to only a tangential component, and in return to reduce the total ΔV required, a minimizer can be employed. This optimization comes at the expense of a higher launch energy and a longer flight time for trajectories with large $\mid \theta_E \mid$. Differing trends from Sims et. al. analysis of V_∞ leveraging[2] were only noticed in high total ΔV cases for each k leveraging orbit family. These solutions were discarded from the lookup table due to delivering lower aphelion radii post-flyby when compared to lower total ΔV leveraging orbits of the same family. The case presented in Fig. 2 matches that discussed by Sims et. al.[3]

Now that the ΔVEGA orbit properties are known, the lookup table solution can be extended to the actual solar system model in the tree search. The table values are represented in a relative frame with respect to Earth's state vector at the launch epoch. A subsequent transformation of the departure velocity and pre-EGA incoming $\vec{V_\infty}$ can be done in order to find their specific components corresponding to an Earth epoch in the Ecliptic J2000 frame. From the initial node in the tree, a set of Earth leveraging time-of-flight nodes are created corresponding to their respective k and θ_E values parameters. The number of these leveraging nodes included in the initial flight time layer of the tree is directly related to the angular spacing of θ_E. As the resolution becomes finer, the estimated ΔV becomes more accurate. This, however, increases of the number of tree nodes created, and so for a rough idea of the trajectory search space, a coarse resolution is preferred. The discontinuous ΔV post-EGA required to patch the incoming leg from the leveraging orbit and outgoing leg to the next planet node will determine if the leveraging node and its performance is effective for the transfer. Using this method, a distinction between the ΔVEGA trajectory families to different outer planets can be observed.

PERFORMANCE EVALUATION

As this program was developed with no basis for its astrodynamics, multiple studies on the validation of the program were conducted. Three major cases were explored to highlight the different core functionalities of the algorithm. The first case is a search to find Europa Clippers non-SLS trajectory involving a sequence of 4 planetary flyby's, to test MCTS's ability to find long sequence trajectories. The second study is designed to push the limits of powered flyby ΔV theory, by finding Galileo's low budget trajectory to Jupiter with similar efficiency. The final case is not based off any historical mission, but built to test the flexibility of the ΔVEGA theory by finding a variety of the launch possibilities for a Trident-like mission to Neptune.

Case 1: Europa Clipper Trajectory Recreation

For the first case, a search was conducted of Europa Clipper's launch window. The goal in mind was to find the EEVEEJ trajectory, as shown by Buffington.[4] The nominal trajectory has the spacecraft set to launch on June 03, 2022, and arriving at Jupiter on January 15, 2030 with an additional 3 Earth flybys and 1 Venus flyby. The primary challenge for this mission was the complexity of the sequence, as the difficulty of the combinational problem grows non-linearly with length.

Table 1. Constrain Input Table for Europa Clipper Trajectory Search

Input Name	Input Value
Arrival Planet	Jupiter
Launch Window	March 01, 2022 — September 01, 2022
Iterations	75,000
ΔV Budget	10 km/s
Max C3	10 km^2/s^2
Detail (d)	24

Table 1 provides the search criteria for the broad trajectory search. For general trajectory searches, it was found that an iteration budget of 50,000 was found to be sufficient for most trajectories, but

due to the length of the planetary sequence for the nominal trajectory, the iteration limit had to be adjusted. Runs with a $d = 16$ were also conducted, but due to the coarseness in the separation of state pairs, the exact solution was deemed infeasible. As described earlier, this behavior is one of the drawbacks of heuristic searches as not all possible solutions can be found.

At the completion of the search, the algorithm had conducted 20 million Lambert arcs, and created one million tree nodes. Of the 1.9 billion possible combinations possible for a 6 layer tree, only 275 solutions were deemed feasible for the input criteria. Figure 3 (Left) shows the top 100 trajectories, sorted by their unoptimized mission ΔV with the nominal circled and shown on the (Right). The targeted sequence was the ninth highest result in terms of unoptimized ΔV usage behind an EEVVEJ and EEVEJ sequence. The trajectories non-podium finish is down to the final Earth flyby, which even under ideal dates requires a flyby ΔV of 3.23 $^{km}/_{s}$. Among all top performing trajectories, the initial two flybys of Earth and Venus were held in common sharing similar dates. Of the EEVEEJ solutions found, the most optimal utilized a 3:1 orbital resonance for the final Earth flyby, while the nominal trajectory utilized a 2:1. The 2:1 solutions also exist within the results, but do not appear until later is the data set, utilizing an additional 250 $^{m}/_{s}$ of ΔV. Even with this deviation, the Jupiter arrival occured only one month later than targeted and with a lower incoming V_∞.

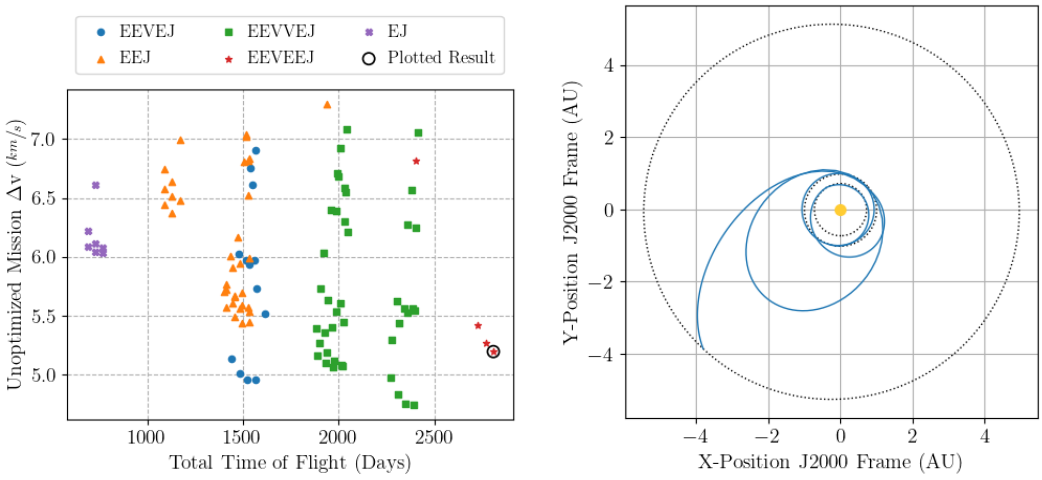

Figure 3. (Left) Top 100 results from built broad search colored by sequence (Right) Circled, unoptimized trajectory from results that matches targeted sequence from Buffington[4]

Utilizing the highest performing EEVEEJ transfer, the dates MCTS supplied were optimized using MALTO (Mission Analysis Low Thrust Optimizer) **REF**. While this tool is primarily designed for low-thrust trajectories, the application is still capable of taking optimizing ballistic and chemical propulsion trajectories. Table **??** shows the inputs to MALTO and their associated bounds for the optimizer to search. **CONTINUE AFTER MALTO SEARCH**

Case 2: Galileo Trajectory Recreation

For the second validation case, the nominal trajectory of Galileo has the spacecraft set to launch from Earth with a C3 of 13-17 $^{km^2}/_{s^2}$ on October 18, 1989, and arriving at Jupiter on December 07, 1995 with a Venus flyby and a Earth 2:1 resonance sequence. The major challenge behind

this sequence was for MCTS to be able to find the low required ΔV solution as described by D'Amario.[5] For Galileo's interplanetary leg of its trajectory design, the spacecraft was only budgeted 100 $^{m}/_{s}$ of ΔV for TCMs, trajectory correction maneuvers.

Table 2. Constrain Input Table for Galileo Trajectory Search

Input Name	Input Value
Arrival Planet	Jupiter
Launch Window	Jun 01, 1989 — Dec 31, 1989
Iterations	50,000
ΔV Budget	3 $^{km}/_{s}$
Max Arrival V_∞	7.5 $^{km}/_{s}$
Max C3	20 $^{km^2}/_{s^2}$
Detail (d)	16

Table 2 lays out the inputs to the search algorithm, which fall more in line with the average use case. In order to constrain the solution to the low ΔV solution, the ΔV budget was set to 3 $^{km}/_{s}$, and the maximum C3 was rounded to a value of 20 $^{km^2}/_{s^2}$. An additional constrain was also placed on the search, requiring a low energy arrival to Jupiter, to once again constrain the solution to that of Galileo. As a silent constraint for all searches, solutions will also be required to have a radially V_∞ greater than -3 $^{km}/_{s}$ or less than +3 $^{km}/_{s}$ for radial in and out arrivals accordingly. This was implemented so that solutions contained the correct arrival V_∞ direction, as well as value were found.

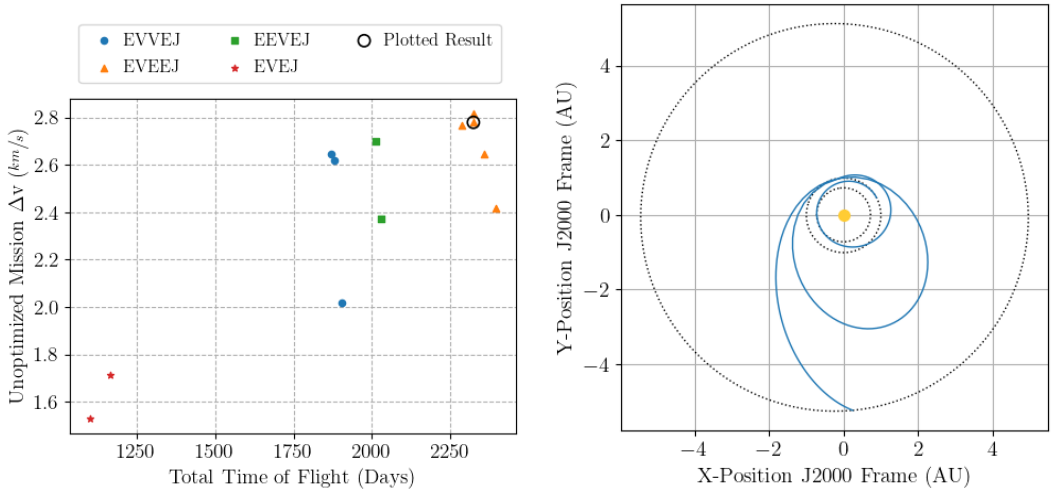

Figure 4. (Left) Top results from built broad search colored by sequence (Right) Circled, unoptimized trajectory from results that matches targeted sequence from D'Amario[5]

At the conclusion of the search, the tree made 867,000 nodes and conducted 10 million Lambert arcs, in order to find the 12 solutions to the constraints. Of those solutions, the eighth was the nominal targeted trajectory. Figure 4 (Left) shows the 12 solutions ranked by their unoptimized mission ΔV usage, and the (Right) shows the circled nominal trajectory. In the same fashion as the

Europa Clipper case, the optimal found trajectory deviates from what Galileo originally performed. The final Earth flyby was conducted in a 3:1 resonance rather than a 2:1 resonance and lead to an overall smaller incoming V_∞. The 2:1 resonance case also exists within the found data set, but is the 11 of 12 that satisfies the criteria. Once again, similar to Europa Clipper, with the additional 1 year resonance, the spacecraft only arrives at Jupiter 3 months after targeted. With the trajectory leg dates, this solution will be optimized using MALTO to find the optimized version of this trajectory. **MORE ONCE MALTO RUN IS COMPLETE**

Case 3: Triton ΔVEGA Opportunity Search

FOR ROHAN TO COMPLETE LATER

APPENDIX A: RAW MCTS RUN RESULTS

Table 3. Top 23 results from broad search of Europa Clipper's launch window. Gray row shows nominal solution.

Result #	Sequence	C3 ($\frac{km^2}{s^2}$)	$\Delta V_{unopt}(\frac{km}{s})$	ToF (Days)	Arrival $V_\infty(\frac{km}{s})$
0	EEVVEJ	3.12	4.75	2392.28	6.62
1	EEVVEJ	3.12	5.07	2019.14	6.30
2	EEVVEJ	3.12	5.08	2013.15	6.36
3	EEVEJ	3.12	4.96	1561.77	6.54
4	EEJ	35.83	5.44	1533.57	6.15
5	EEJ	34.30	5.54	1533.57	6.10
6	EEVVEJ	3.12	5.45	2025.13	6.27
7	EEJ	28.63	5.57	1523.57	6.17
8	EEVEEJ	3.12	5.20	2810.10	6.58
9	EEVVEJ	3.12	4.76	2351.48	7.02
10	EEVVEJ	3.12	5.12	1978.34	6.71
11	EEVVEJ	3.12	5.06	1972.34	6.76
12	EEVEJ	3.12	4.96	1520.96	6.94
13	EEVEJ	3.12	5.52	1612.78	6.41
14	EEJ	35.83	5.44	1492.77	6.51
15	EEJ	34.30	5.59	1492.77	6.45
16	EEVVEJ	3.12	5.60	2007.16	6.45
17	EEJ	28.63	5.56	1482.76	6.53
18	EEVVEJ	3.12	5.55	2398.27	6.57
19	EEVVEJ	3.12	5.54	1984.33	6.66
20	EEVVEJ	3.12	5.40	1966.35	6.84
21	EEVVEJ	3.12	5.56	2386.29	6.68
22	EEVEEJ	3.12	5.26	2769.29	7.01

Table 4. All results from broad search of Galileo's launch window. Gray row shows nominal solution.

Result #	Sequence	C3 ($\frac{km^2}{s^2}$)	$\Delta V_{unopt}(\frac{km}{s})$	ToF (Days)	Arrival $V_\infty(\frac{km}{s})$
0	EVEJ	19.71	1.71	1162.28	6.64
1	EVEJ	19.71	1.53	1099.71	7.38
2	EVEEJ	20.75	2.42	2395.81	6.73
3	EVVEJ	19.71	2.02	1904.48	7.27
4	EVEEJ	15.94	2.65	2360.62	6.67
5	EEVEJ	27.93	2.37	2030.56	7.17
6	EVEEJ	28.73	2.82	2325.42	6.72
7	EVEEJ	21.75	2.78	2325.42	6.87
8	EEVEJ	28.97	2.70	2013.05	7.09
9	EVVEJ	15.42	2.62	1879.77	7.24
10	EVEEJ	22.03	2.77	2289.24	7.20
11	EVVEJ	15.42	2.65	1869.28	7.33

REFERENCES

[1] D. Hennes and D. Izzo, "Interplanetary trajectory planning with Monte Carlo tree search," IJCAI International Joint Conference on Artificial Intelligence, Vol. 2015-Janua, No. Ijcai, 2015, pp. 769–775.

[2] J. Sims and J. Longuski, Analysis of V(infinity) leveraging for interplanetary missions, 10.2514/6.1994-3769.

[3] A. Sims, Jon; Longuski, James; Staugler, "V(infinity) Leveraging for Interplanetary Missions Multiple-Revolutuion Orbit Techniques," Journal of Guidance, Control, and Dynamics, Vol. 20, No. 3, 1997, pp. 409–415.

[4] B. Buffington, "Trajectory design for the europa clipper mission concept," AIAA/AAS Astrodynamics Specialist Conference 2014, 2014.

[5] L. A. D'Amario, L. E. Bright, and A. A. Wolf, "Galileo trajectory design," Space Science Reviews, 1992, 10.1007/BF00216849.