(Preprint) AAS 20-686

BROAD TRAJECTORY SEARCHES USING MONTE CARLO TREE SEARCH WITH THE INCLUSION OF ΔVEGA TRAJECTORIES

Burton A. Yale [*†], Rohan D. Patel[*‡], Jehosafat J. Cabrera[*§], 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 combinatorial task which can be solved by the Monte Carlo Tree Search (MCTS) method. In this paper, an MCTS algorithm is developed and tested. The tree search is able to incorporate V_∞ leveraging of Earth (ΔVEGA), which will allow the algorithm to find an addtional set of feasible sequences. Several cases are optimized and the tree search's performance and accuracy is discussed. This algorithm will allow for the inner-planetary flyby search planning for outer planet missions.

INTRODUCTION

Broad trajectory searches are one of the first steps in mission planning, allowing for the selection of candidate sequences given a list of constraints. 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. Also, by pruning possible choices before the each selection, the overall computation time can be significantly decreased, at the expense of not finding all possible results. The method of interest for this paper is enumerated searches, a subset of grid searches, as recent work has proven their ability to quickly and accurately locate multi-leg interplanetary trajectories.[1] **M**onte **C**arlo **T**ree **S**earch (MCTS), is a heuristics-based adaptation of the enumerated search, and has the ability to both explore and exploit its environment. With this, it is possible for the algorithm to efficiently narrow down possible search paths for future iterations to explore. 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 ΔV budgets, etc. These results can then be passed on to the next phase of the design, optimization.

In this paper, we discuss and implement the MCTS method to the multi-flyby sequence problem. We consider Earth V_∞ leveraging in the search space by approximating the required ΔV and flyby conditions of ΔVEGA trajectories. This paper demonstrates the capability of the algorithm to find ballistic or ΔVEGA multi-flyby sequences to the outer planets.

[*]Aerospace Engineering, Cal Poly Pomona, 3801 W Temple Ave., Pomona, California, 91786, USA
[†]Undergraduate Student, E-mail: bayale@cpp.edu
[‡]Undergraduate Student, E-mail: rohanpatel@cpp.edu
[§]Undergraduate Student, E-mail: jehosafatc@cpp.edu
[¶]Assistant Professor, E-mail: nnakhjiri@cpp.edu

BACKGROUND

Multi-flyby sequence generation begins with a set of target encounter bodies to visit and the evaluation of the feasibility of transfers between them. 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 map out the search space. Once a set of possible sequences and their respective encounter epochs are known, solutions of interest can be optimized in further detail. This is often referred to as pathfinding and path-solving, respectively.[2] Pathfinding is a sequence optimization problem that provides 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.[3] Some common techniques for pathfinding and generating search spaces include evolutionary algorithms, machine learning, evolutionary neuro-controllers, and tree searches methods.[4]

Evolutionary algorithms are optimization routines that make use of heuristics and Darwinian evolution to solve complex optimization problems. Three popular type of 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 trajectories to Halo Orbits, and as a interplanetary mission design algorithm.[5,6] The Cassini and Galileo missions were considered as case studies to analyze the performance of these algorithm. The DE routine was able to come within appropriate bounds of both trajectories and found optimal sequences with the given flight constraints. PSO was used by Zhuang et al.[7] in combination with Legendre pseudo-spectral method for solving time-optimal trajectory planning problems. The PSO algorithm was robust enough to handle random and changing the fitness function of the algorithm is smaller than a predefined value, the searching algorithm is switched the Legendre pseudo-spectral method for fast convergence. When considering non-evolutionary algorithms for pathfinding, a common method is the use of the grid-search technique. This has been previously used as a search method to find possible trajectories to Kuiper belt objects (KBOs).[8] Grid search is a type of search algorithm with the ability to map the entire search space so that no regions of interest missed. This method has been used to properly parameterize the time-of-flight between encounter bodies. 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. Beam Search (BS), a heuristic tree search algorithm, can be used as a method to find a sequence of planets for gravitational assists.[8] BS uses the method of Breadth-First Search (BFS) to find the tree of possibilities. With each layer of the tree and orders the nodes in accordance to their heuristic cost. It only chooses the nodes with a maximum value to build from. Depth-First Search (DPS), alternatively, is used as a searching criterion to traverse down the tree.[9] Contrary to BFS, DPS does not search the tree at every level but rather explores a branch until the termination criteria is met. After, it propagates up each parent node updating branches to start the process again. Both the BFS and DPS, search and prune the environmental choices consecutively. Using the Lazy Race Tree Search, the search space is pruned, and nodes ranked through the time of flight.[9] The use of heuristics avoids finding redundant solutions and increases the efficiency of the method used. Izzo et al.[1] 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.[10] 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 is not optimal. The search sensitivity criteria of DPS, BS, and BFS are explored in conjunction with that of the MCTS and compared. Currently, there exists a deficiency in these algorithms to take advantage of the ΔVEGA trajectories and there derivatives. This paper will present a solution to this missed opportunity.

MONTE CARLO TREE SEARCH IMPLEMENTATION

The tree is built upon a set of individual nodes, connected to each other through their parent and children, much like a family tree. Each node has a corresponding state to represent a leg in the trajectory of the 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: ① selection, ② expansion, ③ simulation, and ④ 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 evenly sample the launch window. From this point, the algorithm will begin its core loop to search out possible trajectory sequences, until the, user defined, iteration budget is depleted.

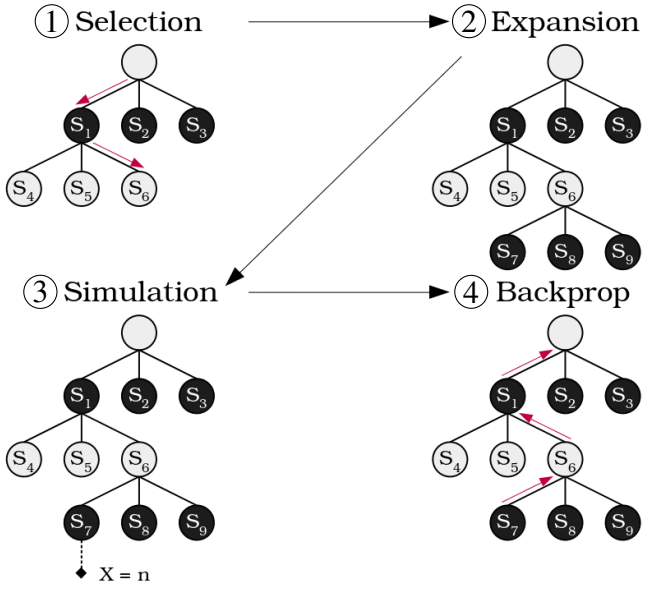

Figure 1. 4 Main steps for the creation of a Monte Carlo Tree

Selection

At the start of each iteration, we initializes at the root of the tree, $id = 0$, and from its children, select the node with the highest associated UCB1 reward, with X being the future reward from exploring of the node and its children, N being the number of visits the node has received, and n being the number of visits the parent node has received.

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

The exploration exploitation parameter, C_p, has a value of $1/\sqrt{2}$ which was selected due to its performance demonstrated by Hennes.[1] This reward policy allows the tree to explore all valuable 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, if, at any point, 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 next step, the expansion function.

Expansion

When a leaf node is reached and has not been deemed terminal, the next step of the algorithm is to create a new set of state pairs to explore. Before initializing a new group of nodes, we take into account the state of arrival, the child node, and the departure, the parent node. As planetary motion is of conics, Cartesian grid position sampling is not viable, thus angular divisions are required to efficiently sample a planet's position. For each of the possible planets to visit, the following policy is implemented:

$$E = \begin{pmatrix} n*(\tau_1+\tau_0)+t_0 \\ \vdots \\ m*(\tau_1+\tau_0)+t_0 \end{pmatrix} \text{ for } \begin{cases} n=0.10, & m=1.00 & \text{if } a_1 < 2 \text{ AU} \\ n=0.05, & m=0.25 & \text{if } a_1 \geq 2 \text{ AU} \end{cases}$$

For the ephemeris array E, the ephemeris state of the child node is dictated by the period of the node's associated 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 based upon their performance in Hennes.[1] These values however are not ideal for travel to the outer planets. For example, with the previous values, Lambert arcs time of flights could be in excess of 12 years for a leg to Jupiter. Additionally, if Jupiter are to be used for a gravitational assist to another outer body, faster transfers are required. Based on their performance in simulations, the values of $n = 0.05$ and $m = 0.25$ lead to more transfers that satisfied mission requirements.

With the upper and lower bounds of the array set, an additional parameter for detail, d, specifies the length of the array, also defining the resolution of the time steps. Additionally, this allows for dynamic angular detail in the ephemeris arrays. The inner planetary sequences are more sensitive to large time steps which could lead to missed transfers. Thus, the fixed array length lead to smaller time steps for inner planets, and larger for outer planets. 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, child nodes that revisit the same parent planet have additional epheremis node tweaks to allow for better performance. The n and m values are set to 0.9, and 1 accordingly, for the set of 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. To avoid the singularity solution of the Lambert problem for angles of 0° and 180°, the last node is offset by three days. The implementation of Earth specific leveraging orbits are discussed in the next section in greater detail.

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

Once the list of states has been established, a new set of children nodes are created to match all combinations of ephemeris 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 evaluate their associated ΔV cost. Any node that exceeds the mission budget is considered terminal and is not used in any further exploration. To calculate unoptimized mission ΔV usage, the simulation uses powered flybys, as characterized by Wagner.[11]

The powered flyby geometry consists of an incoming and outgoing spacecraft velocity vectors given by the solutions to the Lambert's problem. A graphical overview of this condition is shown in Figure 2. This maneuver is used to either bend the trajectory to a desired direction and/or further increase/decrease the magnitude of $V_{\infty - out}$, the relative velocity out of the planet. By manipulating the bend angle and velocity of a gravitational assist, it is possible to patch together two separate Lambert transfers. Therefore, a small ΔV can be used in order to increase the performance and efficiency of the gravity assists. While this maneuver can be performed at any location along the incoming and outgoing velocity vectors, there is a maxima in terms of efficiency. Its effectiveness, moreover, is derived from the Oberth effect,[12] which states that a ΔV maneuver is more efficient to change the overall orbital energy at high velocities. For this reason, it is common to design the maneuver at periapsis.[13] While this maneuver can be utilized to enhance trajectory characteristics, it serves a different purpose for the tree search, who's goal is to minimize this value. To mitigate the coarse nature of the search, we will use this powered flyby to evaluate the effectiveness of the epoch in which the flyby occurs. For values that exceed the total budget, it can be surmised that the flyby is at a non-optimal time. In addition, while smaller ΔV flybys lead to ballistic, or near ballistic, solutions, there have been cases where even high ΔV solutions, $> 1\ ^{km}/_{s}$, have been optimized.

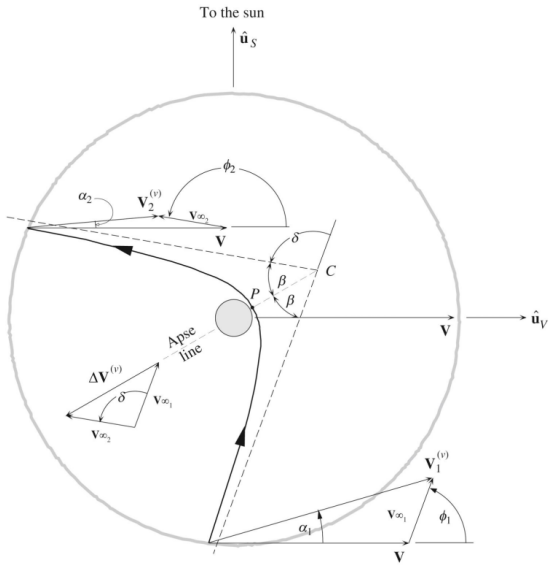

Figure 2. Geometry of a patch conics flyby (Courtesy of Curtis[14])

From this we arrive at the formulation of the boundary value problem, as described by Wagner.[11] It is assumed that the flyby is coplanar, and the plane can be defined by the incoming and outgoing spacecraft velocities and the center of the encounter planet. The conditions that define the BVP are the incoming and outgoing V_∞ and the bend angle, δ. The solution to the BVP is the ΔV and the value of the maneuver periapsis radius, r_p. The first initial condition to the problem is the semi-major axis, a, of the incoming and outgoing hyperbolic trajectories which is defined as:

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

Where μ_p is the standard gravitational parameter of the flyby planet. Additionally, the required turn angle from the flyby, δ, is another condition that must be met. This value is found by the following function finding the angle between the incoming and outgoing V_∞ vectors:

$$\delta = \cos^{-1}\left(\frac{\vec{\mathbf{V}}_{\infty-\text{in}} \cdot \vec{\mathbf{V}}_{\infty-\text{out}}}{V_{\infty-\text{in}} \cdot V_{\infty-\text{out}}}\right) \tag{3}$$

With e_{out}, the eccentricity of the outbound hyperbolic flyby orbit, as the iterated value, a function, and its derivative, can be defined. In order to solve for e_{out}, the root finding method of Newton Raphson is used to find the eccentricity that coincides with the two V_∞ vectors. For each full loop, an initial e_{out} value of 1.5 is used due to its performance in Wagner.[11]

$$f = \left(\frac{a_{out}}{a_{in}}(e_{out} - 1)\right)\sin\left(\delta - \sin^{-1}\left(\frac{1}{e_{out}}\right)\right) \tag{4}$$

$$\frac{\mathrm{d}f}{\mathrm{d}e_{out}} = \left(\frac{a_{out}}{a_{in}}e_{out} - a_{out}a_{in} + 1\right)\frac{\cos\left(\delta - \sin^{-1}\left(1/e_{out}\right)\right)}{e^2_{out}\sqrt{1 - 1/e^2_{out}}} + \frac{a_{out}}{a_{in}}\cos\left(\delta - \sin^{-1}\left(1/e_{out}\right)\right) \tag{5}$$

Once a value of e_{out} has been found, it can then be used to find the actual periapsis radius, which is then used to find the required maneuver ΔV. In addition, the periapsis radius is checked again body radius to prevent solutions that impact the planet's surface.

$$r_p = a_{out}(1 - e_{out}) \tag{6}$$

$$\Delta V_{GA} = \left|\sqrt{v^2_{\infty-\text{in}} + \frac{2\mu}{r_p}} - \sqrt{v^2_{\infty-\text{out}} + \frac{2\mu}{r_p}}\right| \tag{7}$$

While this results in an increased computation cost at each step of the node creation process, this method of ΔV calculation allows for a better approximation of actual trajectory performance. In addition to powered flyby ΔV's, budget usage was also calculated for launch energies higher than the specified limit. If, for example, the max launch energy was defined as $V_\infty = 6\ ^{km}/_s$ and the spacecraft launched at a V_∞ = 7 $^{km}/_s$, the incurred ΔV cost would be 1 $^{km}/_s$. 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 the 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 following equation:

$$X = \max\left(0.1 \cdot l, \ \frac{\Delta V_{budget} - \Delta V_{used}}{\Delta V_{budget}}\right) \tag{8}$$

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

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. Equation (9) is used to calculate the new reward, where X is the current reward value at each node. This value is then adjusted by a weighted average based off the number of current visits, N, to each node of the branch.

$$X_{node,new} = \frac{X_{node,old} \cdot N_{node} + X_{new}}{N_{node} + 1} \tag{9}$$

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 pass to the next step in the trajectory design process, optimization.

ΔVEGA MANEUVER IMPLEMENTATION

A ΔVEGA orbit, also referred to as a leveraging orbit specifically of Earth, launches with the intent to gravity assist off the body for a higher post-flyby heliocentric energy.[15] Leveraging orbits are classified by their nominal resonance period multiple with respect to the flyby body'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 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 encountering the body at a different Earth heliocentric true anomaly. Trajectories with a larger period are referred to as k:$1^{+}\Delta V$EGA and those with a shorter are k:$1^{-}\Delta V$EGA trajectories. To modify the Earth flyby true anomaly a maneuver at the aphelion, referred to as a 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 number of computations within the search. We

approximate the launch V_∞ and DSM ΔV requirements. The inclusion of the DSM and intercept Earth encounter location 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. Being a rough approximation, the circular Earth orbit and coplanar trajectories assumptions are used. Finding the ΔVEGA orbit parameters for a specific θ_E and k begins with 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. (10) is the difference in time from the DSM maneuver location to the Earth gravity assist (EGA) point where τ_E is the orbital period of Earth.

$$dt = k\tau_E \pm \tau_E(\theta_E/2\pi) \tag{10}$$

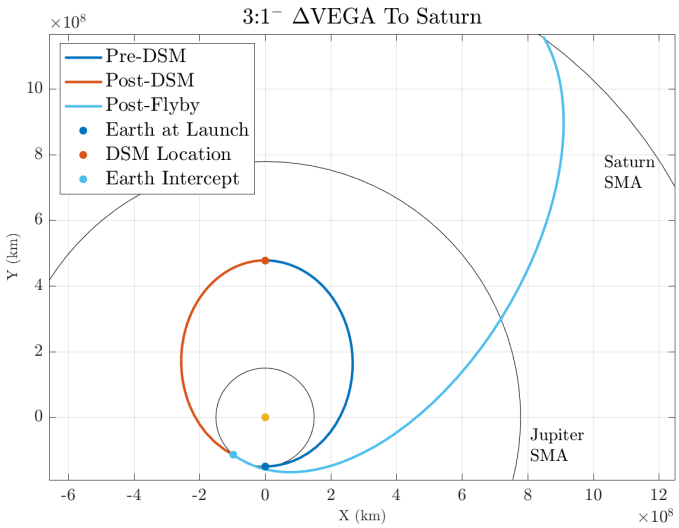

Figure 3. Example of a computed 3:1$^-$$\Delta V$EGA trajectory with a final aphelion radius roughly that of Saturn's semi-major axis. The launch V_∞ is 6.97 km/s and the required DSM ΔV is 0.39 km/s. The EGA flyby altitude was constrained to 200 km, which yielded the highest post flyby energy.

A Lambert arc is computed between these points and the resulting initial velocity change is used 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, $\vec{\mathbf{V}}_\infty$, is computed and a planar flyby of Earth can now be calculated from the tree search algorithm. Figure 3 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 to show the resulting trajectory.

From testing, we noticed that as $\mid \theta_E \mid$ increased, a normal component in the orbit plane 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 is employed. This optimization comes at the expense of a higher launch energy and a longer flight time for trajectories requiring large $\mid \theta_E \mid$. Differing trends from Sims et al. analysis of V_∞ leveraging[16] 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 Figure 3 is intended to match that discussed by Sims et al[17] and has nearly identical results. The inclusion of deep space maneuvers, not specific to ΔVEGAs, are not implemented in the current form of the tree search. However, because the algorithm patches the two conics forming the leveraging orbit using the Lambert's method, the possibility to extend DSM maneuvers for off-tangent V_∞ departures and targeting for the flyby can be implemented. This inclusion adds another dimension to the lookup table, but offers greater flexibility in the placement of leveraging maneuvers in the sequence. It can also include V_∞ leveraging of other planets.

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 rotating relative frame with respect to Earth's state vector at the launch epoch. A subsequent transformation of the departure velocity and pre-EGA incoming $\vec{\mathbf{V}}_\infty$ 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 parameters. The number of these leveraging nodes included in the tree is directly related to the angular spacing of θ_E. If the resolution becomes finer by increasing the *detail* input, d, the estimated ΔV becomes more accurate. This, however, increases of the number of tree nodes created, and so for an estimation 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

To validate the methods presented in this paper, multiple case studies were tested and evaluated. Three major cases were explored to highlight the different core functionalities of the algorithm. The first case is a search to find Europa Clipper's EELV[18] trajectory involving a sequence of 4 planetary flybys to test MCTS's ability to find long sequence trajectories. The second study is designed to test 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 evaluates the ΔVEGA theory by finding a variety of the launch possibilities for a Trident-like mission to Neptune.

In addition to roughly matching existing trajectories, search results will be used as initial conditions for an optimization process to validate the tree search's astrodynamics. While this software is primarily built for low-thrust applications, the program is capable of optimizing for ballistic and chemical propulsion trajectories.[19] For each leg of the mission, the optimizer was allowed $\pm$ 100 days, unless otherwise stated, on either side of MCTS's initial guess to mitigate the coarse nature of the broad searches constraints.

Case 1: Europa Clipper Trajectory Recreation

The first case is a search to find a sequence similar to Europa Clipper's interplanetary trajectory (15F9-A22 EEVEEJ) utilizing multiple gravitational assists of Earth and Venus to reach Jupiter.[18] The nominal trajectory has the spacecraft set to launch on June 03, 2022, and arriving at Jupiter on January 15, 2030 with an 3 Earth flybys and 1 Venus flyby. The primary challenge for this mission is 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	Value
Arrival Planet	Jupiter
Launch Window	March 01 — 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 Monte Carlo Tree Search. For general trajectory searches, an iteration budget of 50,000 was found to be sufficient, but due to the length of the planetary sequence for the nominal trajectory, we adjusted iteration limit. Runs with a $d = 16$, the ephemeris array length, 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 grid searches as not all possible solutions can be found.

At the completion of the search, 20 million Lambert arcs had been conducted, and a tree one million tree nodes was created. Of the 1.9 billion combinations possible for a 6 layer tree, only 275 solutions were deemed feasible for the input criteria. Figure 4 (Left) shows the top 100 trajectories, sorted by their unoptimized mission ΔV with the nominal solution 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 trajectory's low placement is down to the final Earth flyby, which, even under ideal conditions, requires an unoptimized flyby ΔV of 3.23 $^{km}/_{s}$. Among all top trajectories, the initial two flybys of Earth and Venus were found to share 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, to a 3:1 orbit, the Jupiter arrival occurred only one month later than targeted and has a lower $V_{\infty-in}$.

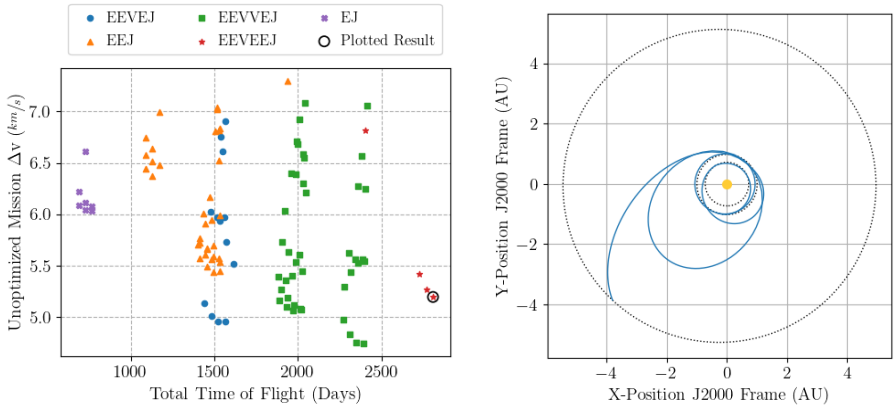

Figure 4. (Left) Top 100 results from built broad search colored by sequence (Right) Circled, unoptimized trajectory from results that matches targeted sequence[18] with the outermost dashed line is the semi-major axis of Jupiter, the middle dashed line is Earth's, and the innermost is Venus's.

Utilizing the highest performing EEVEEJ transfer, the dates the tree search supplied were optimized. Table 2 shows initial guess supplied to the optimizer by MCTS as well as the post optimized dates returned. To reduce the $V_{\infty-in}$ at Jupiter, the node's date bounds were adjusted to be $\pm$ 500 days. Upon completion the optimization run, the solution on Figure 5 was converged. The resulting trajectory was well within the bounds placed on the optimizer for the flybys and arrival dates, as well, the arrival V_∞ was reduced. In addition, the largest departure from the initial guess dates, apart from Jupiter arrival, was the third flyby of Earth. The date had transitioned 1.5 months later than initially chosen, leading to an orbit of more than 1 complete revolution. With this test complete, it can be shown that this Monte Carlo Tree Search algorithm is capable in searching out long sequence interplanetary trajectories that need minimal tweaking in later steps of the optimization process.

Table 2. MCTS output dates vs optimized dates for Europa Clipper

Planetary Node	Dates	
	MCTS	Optimized
Launch	June 04, 2022	May 12, 2022
Earth Flyby #1	Jun 02, 2023	May 12, 2023
Venus Flyby	November 23, 2023	November 22, 2023
Earth Flyby #2	October 22, 2024	October 2, 2024
Earth Flyby #3	October 20, 2027	December 2, 2027
Jupiter Arrival	February 13, 2030	March 17, 2031

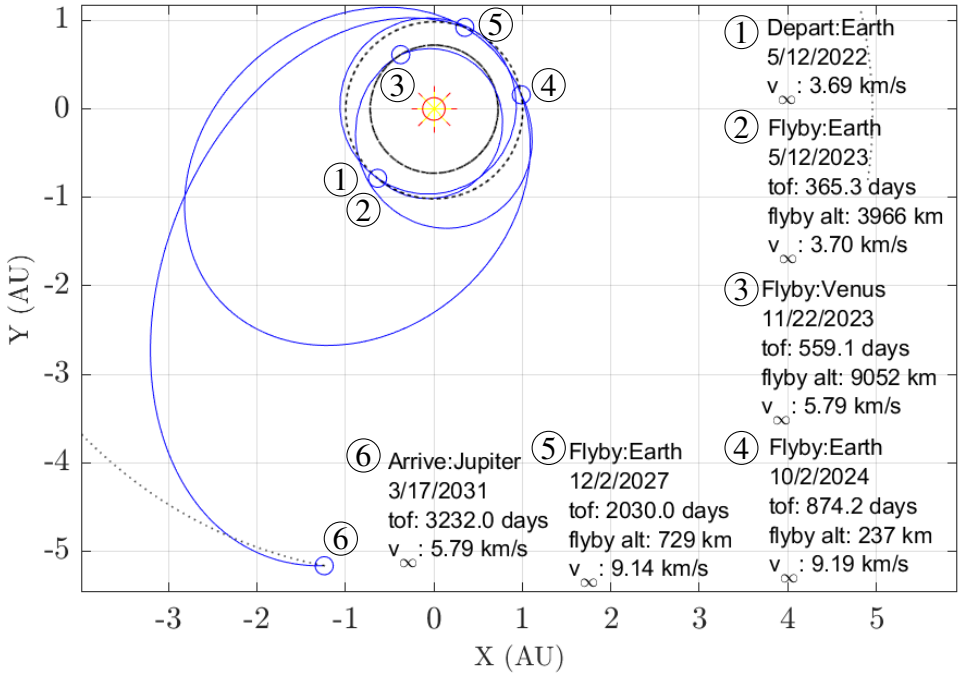

Figure 5. Optimized Europa Clipper trajectory from tree search results

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 an Earth 2:1 resonance sequence. The major challenge behind this sequence was for the algorithm to be able to find the low required ΔV solution as described by D'Amario.[20] 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 3. Constrain Input Table for Galileo Trajectory Search

Input	Value
Arrival Planet	Jupiter
Launch Window	Jun 01 — 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 3 lays out the inputs to the search algorithm, which fall more in line with the average use case. In order to constrain the results 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 further prune the results 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.

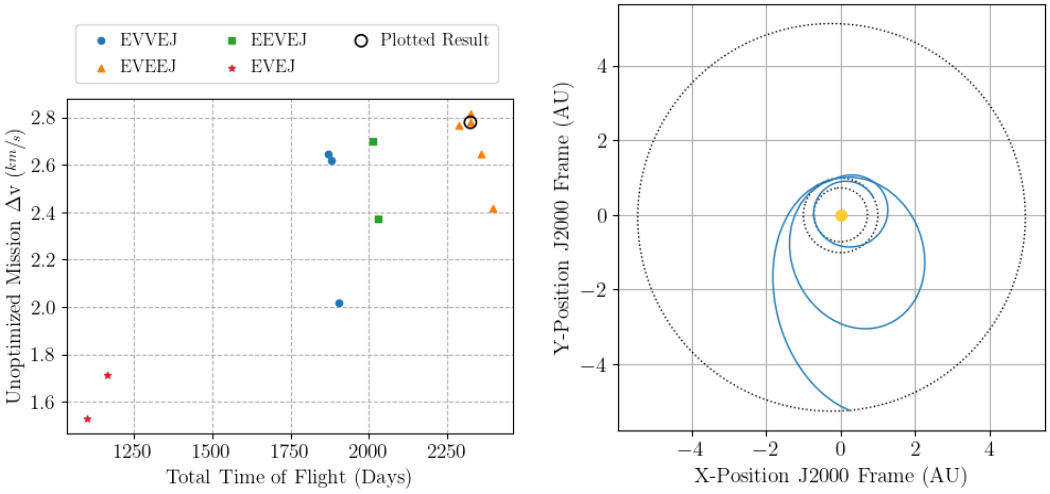

Figure 6. (Left) Top results from built broad search colored by sequence (Right) Circled, unoptimized trajectory from results that matches targeted sequence[20] with the outermost dashed line is the semi-major axis of Jupiter, the middle dashed line is Earth's, and the innermost is Venus's.

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 6 (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.

Table 4. MCTS output dates vs optimized dates for Galileo

Planetary Node	Dates	
	MCTS	Optimized
Launch	October 21, 1989	October 06, 1989
Venus Flyby	February 27, 1990	February 14, 1990
Earth Flyby #1	December 29, 1990	December 15, 1990
Earth Flyby #2	December 26, 1993	December 14, 1992
Jupiter Arrival	March 03, 1996	June 24, 1996

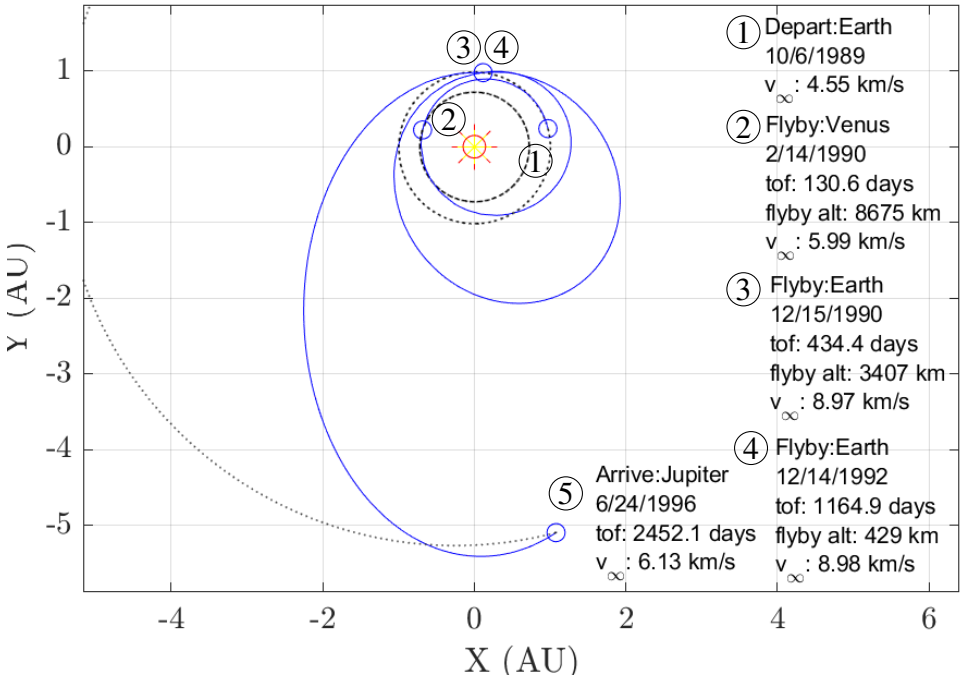

Figure 7. Optimized Galileo trajectory from tree search results

When inputting trajectory dates into MALTO, it was found the 3:1 sequence suggested was not as viable compared to other options. When converging the trajectory, it was found that the performance of the 3:1 trajectory was too high for MALTO to properly capture at Jupiter. Thus by increasing the bounds of the final Earth flyby to bounds of $\pm$500 days to give more leeway, MALTO converged on a solution that utilized a 2:1 orbit as opposed to the 3:1. This result can be found in Figure 7. This result aligns with the findings by Sims[16] where 3:1 orbits were capable of reaching aphelion heights of Uranus SMA, while 2:1 trajectories can reach aphelion heights of 8 AU. Doing so allowed, the work of the Jupiter transfer to be shared between the two Earth flybys without exceeding their energy capacities. As in Table 4, even after taking a year off the total flight time to the last Earth flyby, the Jupiter arrival was only set back 113 days.

Case 3: Triton ΔVEGA Opportunity Search

Now that the sequencing ability of the tree search algorithm has been verified, we can now extend its use to an exploratory context. With the interest in exploring of the solar system's icy moons, a sequence to Neptune, and in turn Triton, is explored.[21] An arbitrary launch period was selected for this evaluation, and the search is limited to trajectories with the inclusion of a ΔVEGA to evaluate its integration and performance. The number of iterations was set at 15,000 as this search has a maximum depth of 4 nodes, greatly reducing the size of the tree. To achieve fast transfer times to Neptune, the use of Jupiter for gravitational assists is critical, as the energy Jupiter can add to a heliocentric orbit is significantly more efficient than any direct launch, in terms of ΔV usage.[22] It is desired to have a higher incoming relative velocity at Jupiter in order to have the appropriate outgoing energy to reach Neptune. With this in mind, the intent of the search was to find ΔVEGA trajectories that deliver a high post-EGA semi-major axis so the encounter velocity is maximized. The *detail* is set to 16 which creates 8 k:1$^-$ and 8 k:1$^+$ nodes for each k family. This spread is a balanced compromise between computation time, due to the number of nodes created, and the accuracy of the DSM's ΔV estimate. The launch C3 is set to 60 km^2/s^2 which is set to explore the use of mainly 2:1 and 3:1 ΔVEGAs with the possibility of 4:1 leveraging families, at an additional ΔV cost. The results of the input search space are summarized in Table 5:

Table 5. Inputs for ΔVEGA Trajectories to Neptune via Jupiter

Input Name	Input Value
Arrival Planet	Neptune
Launch Window	Jan 01, 2029 — Jan 01, 2030
Iterations	15,000
ΔV Budget	3 km/s
Max C3	60 km^2/s^2
Detail (d)	16

The search completed and resulted in 65 sequences from the 704,000 nodes created and 7 million Lambert arcs performed. As expected, most solutions used the 2:1 or 3:1 leveraging maneuvers, and a few 4:1 sequences were also found, but with larger unoptimized ΔV values. Figure 8 shows the four different ΔVEGA sequence types that comprise a majority of the search results. As a general trend, the 3:1 solutions yielded higher energy transfers to Jupiter, and allowed them to achieve faster transfer times onto Neptune.

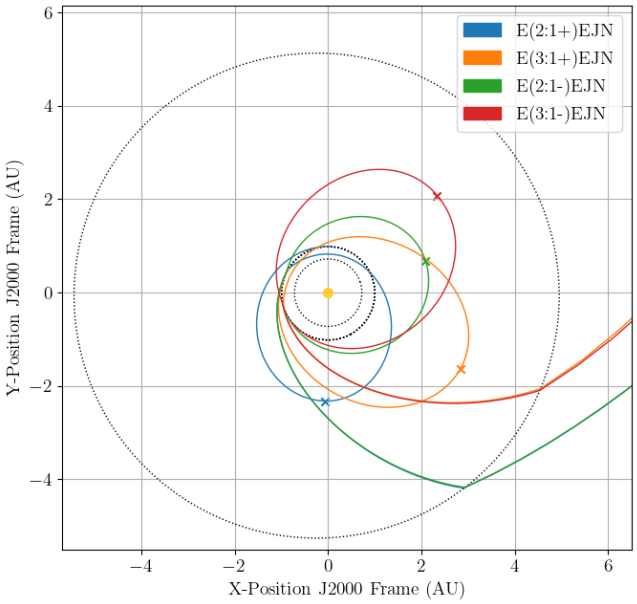

Figure 8. Different resulting E(k:1$^{+/-}$)EJN sequence types from the search. The outermost dashed line is the semi-major axis of Jupiter, the middle dashed line is Earth's, and the innermost is Venus's. The ΔVEGA type is labeled above and the mark on the leveraging orbit denotes the approximate DSM location.

Optimization of select tree search solutions are conducted to determine the accuracy of the sequences found and to further investigate the deep space maneuver performance. The 2:1 and 3:1 ΔVEGA trajectories are of interest as these families are able to reach Jupiter with favorable flight times, launch energies, and incoming relative velocities. The flight time and encounter epochs from the tree search are compared to the optimized solution to better understand the performance and limitations of the search algorithm. For the optimization process, all initial guess conditions were taken from the tree search output and the following bounds were set for encounter epochs. For Earth encounters and the DSM epoch, a plus or minus 30 day span is used to keep the sequence relatively similar to the tree search conditions. For the Jupiter and Neptune flybys 50 and 300 days respectively are used. This looser date bound accounts for the larger spacing in epochs for the outer planets' nodes in the tree. The maximum launch V_∞ is allowed to be 0.1 km/s over the predicted requirement from the search, and the limits on the DSM ΔV are plus or minus 0.4 km/s from the estimated value from the lookup table solution. Figure 9 shows examples of two optimized trajectories resulting from the search. The left plot is an example of a 2:1$^+$ ΔVEGA which requires a launch V_∞ magnitude of 5.20 km/s (as opposed to the 5.15 km/s from the prediction). The 2:1 leveraging trajectory has an incoming V_∞ to Jupiter of around 9 km/s which yields a 4900 day (13.4 year) flight time to Neptune. This case utilized an intercept θ_E of 42° which has an estimated DSM ΔV of 0.43 km/s, while the optimizer's computed 0.66 km/s. The right plot from Figure 9 shows a 3:1$^-$ trajectory with an Earth departure V_∞ of 6.98 km/s and a DSM ΔV of 0.41 km/s. The estimated performance from the tree search is 6.97 km/s and 0.40 km/s for the launch relative velocity and DSM respectively. The similarities between the tree results and the optimized cases presented indicate that the search algorithm is generally able to produce useful initial guesses for the optimization process.

Table 6. MCTS output dates vs optimized dates for ΔVEGA missions to Neptune

Planetary Node	$2:1^{+}$		$3:1^{-}$	
	MCTS	**Optimized**	**MCTS**	**Optimized**
Launch	Jan 01, 2029	Jan 11, 2029	May 01, 2029	Apr 28, 2029
Earth Flyby	Mar 07, 2031	Mar 05, 2031	Mar 07, 2032	Mar 20, 2032
Jupiter Flyby #1	Jan 08, 2033	Nov 16, 2032	Jul 25, 2033	Aug 14, 2033
Neptune Arrival	Jun 26, 2042	Jun 12, 2042	May 06, 2042	Aug 13, 2042

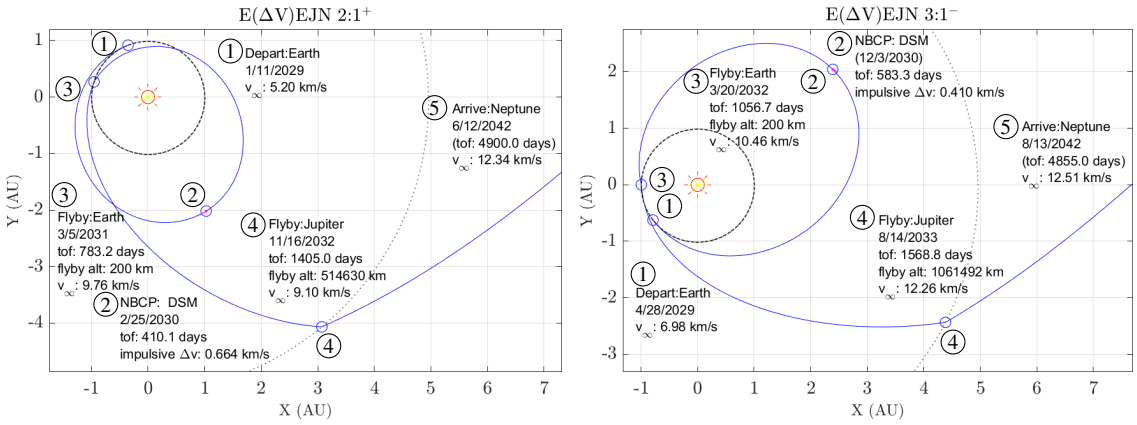

Figure 9. Optimized trajectory examples from the search. The left plot is an example of a $2{:}1^{+}$ and the right is a $3{:}1^{-}$. The $2{:}1^{+}$ is able to reach Neptune with a much lower V_∞ due to a slower relative velocity into Jupiter when compared to the $3{:}1^{-}$ sequence.

Certain characteristics of the results were noticed when optimizing and comparing many cases from this tree search run. The deep space maneuver magnitude and direction general vary from the search results due to various factors. The optimization process takes into account the change in declination for the outgoing flyby so depending on the transfer an additional Z-component of velocity is expected. Another source of discrepancy is in the maneuver location. In the lookup table, the assumption is that the deep space maneuver is conducted at the aphelion of the leveraging orbit. However, the optimizer is able to shift the location of the maneuver within a reasonable time bound. In order to keep the look-up table two dimensional and to minimize the required ΔV at the aphelion, the minimizer was employed to reduce the normal component of velocity of the maneuver. This, in turn, limited solutions that may potentially have higher DSM ΔVs for higher energy Earth gravity assists. It was also noticed that some trajectories from the search were exploiting the planar assumption resulting in lower launch velocities, DSM magnitudes, and flight times to Neptune. These limitations were primarily seen in the $2{:}1^{-}$ set of solutions, which in theory have the lowest ΔV cost of any other trajectory type. However, with the solar system model used in MALTO, these sequences fall short due to energy limitations. The largest discontinuous distance and velocity across all the segments of the trajectory was under 60,000 km and 0.2 km/s. The flight time and V_∞ was also seen to reach the lower bound indicating a lower energy trajectory to Neptune (which is roughly 0.9 AU below the ecliptic plane in 2041).

FUTURE WORK

In its current capacity, the tree search algorithm provides coarse solutions to trajectories around the solar system. As the premise of this paper is to validate both the tree search algorithm as a whole, and implementation of the theory for ΔVXGA (X being E for Earth, V for Venus, etc.), the scope was limited to just Earth leveraging maneuvers. Future revisions of this algorithm can include leveraging orbits of Venus, as seen in missions like Cassini. Additionally, to minimize calculation time and complexity, the algorithm employs 2D approximation of patch-conics flybys, only optimizing the bend angle, δ. Taking into account 3D flybys, as seen in Strange et al.,[23] would increase the fidelity and feasibility of output results.

CONCLUSION

The use of Monte Carlo Tree Search in the field of broad trajectory searches provides an efficient and effective method of solving the inherently difficult time-depended combinational problem. Through its ability to both explore and exploit its search space, the algorithm is able to build a tree of solutions that are compatible with the constraints placed on its environment. The addition of ΔVEGA orbits also allows for more flexible and efficient trajectory sequences to the outer planets.

APPENDIX A: RAW MCTS RUN RESULTS FOR EUROPA CLIPPER AND GALILEO

Table 7. Top 22 results from broad search of Europa Clipper's launch window. Gray row shows nominal solution.

#	Sequence	Dep. Date*	C3 ($\frac{km^2}{s^2}$)	ΔV_{unopt}($\frac{km}{s}$)	ToF (Days)	Arr. V_∞($\frac{km}{s}$)
1	EEVVEJ	8190	3.12	4.75	2392	6.62
2	EEVVEJ	8190	3.12	5.07	2019	6.30
3	EEVVEJ	8190	3.12	5.08	2013	6.36
4	EEVEJ	8190	3.12	4.96	1561	6.54
6	EEJ	8279	34.30	5.54	1533	6.10
5	EEJ	8270	35.83	5.44	1533	6.15
7	EEVVEJ	8190	3.12	5.45	2025	6.27
8	EEJ	8279	28.63	5.57	1523	6.17
9	EEVEEJ	8190	3.12	5.20	2810	6.58
10	EEVVEJ	8190	3.12	4.76	2351	7.02
11	EEVVEJ	8190	3.12	5.12	1978	6.71
12	EEVVEJ	8190	3.12	5.06	1972	6.76
13	EEVEJ	8190	3.12	4.96	1520	6.94
14	EEVEJ	8190	3.12	5.52	1612	6.41
15	EEJ	8270	35.83	5.44	1492	6.51
16	EEJ	8279	34.30	5.59	1492	6.45
17	EEVVEJ	8190	3.12	5.60	2007	6.45
18	EEJ	8279	28.63	5.56	1482	6.53
19	EEVVEJ	8190	3.12	5.55	2398	6.57
20	EEVVEJ	8190	3.12	5.54	1984	6.66
21	EEVVEJ	8190	3.12	5.40	1966	6.84
22	EEVVEJ	8190	3.12	5.26	2769	7.01

Table 8. All results from broad search of Galileo's launch window. Gray row shows nominal solution.

#	Sequence	Dep. Date*	C3 ($\frac{km^2}{s^2}$)	ΔV_{unopt}($\frac{km}{s}$)	ToF (Days)	Arr. V_∞($\frac{km}{s}$)
1	EVEJ	-3739	19.71	1.71	1162	6.64
2	EVEJ	-3739	19.71	1.53	1099	7.38
3	EVEEJ	-3781	20.75	2.42	2395	6.73
4	EVVEJ	-3739	19.71	2.02	1904	7.27
5	EVEEJ	-3739	15.94	2.65	2360	6.67
6	EEVEJ	-3852	27.93	2.37	2030	7.17
7	EVEEJ	-3710	28.73	2.82	2325	6.72
8	EVEEJ	-3724	21.75	2.78	2325	6.87
9	EEVEJ	-3824	28.97	2.70	2013	7.09
10	EVVEJ	-3710	15.42	2.62	1879	7.24
11	EVEEJ	-3724	22.03	2.77	2289	7.20
12	EVVEJ	-3710	15.42	2.65	1869	7.33

*Days since January 01, 2000 (J2000)

APPENDIX B: SEQUENCE FAMILY PLOTS

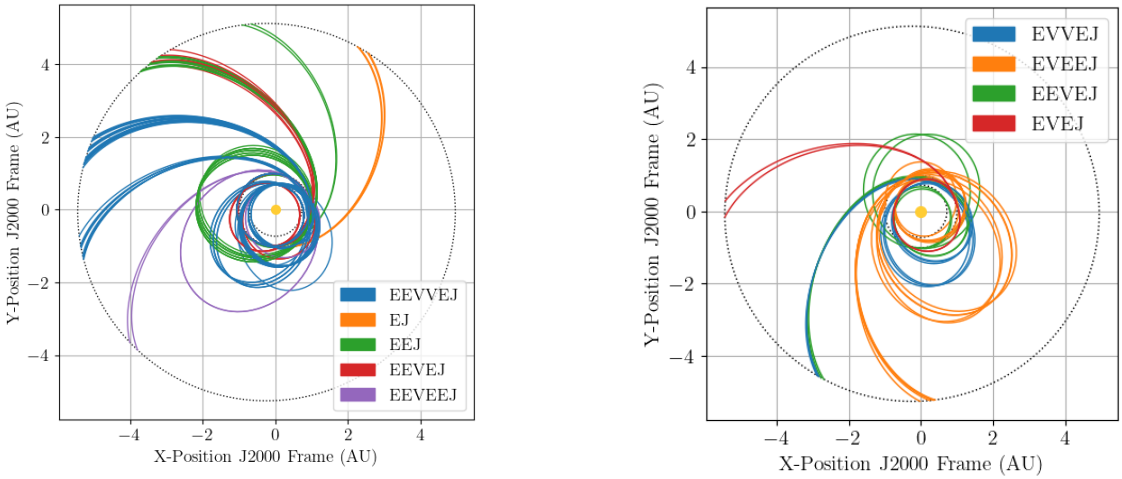

Figure 10. (Left) Top 50 results from Europa Clipper dataset, color sorted by sequence (Right) Top 12 results from Galileo dataset, color sorted by sequence

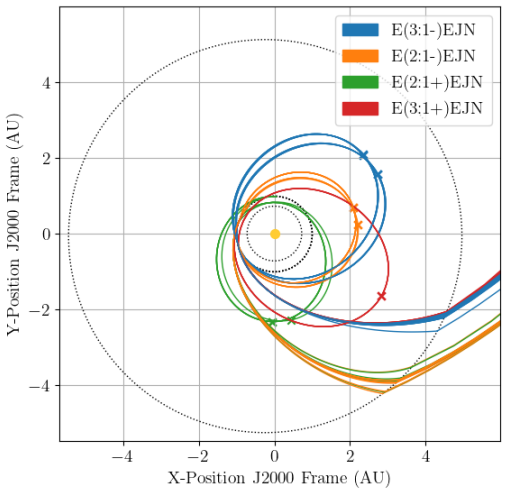

Figure 11. Top 25 results from Neptune Flyby dataset, color sorted by sequence

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] K. M. Hughes, "Gravity-Assist Trajectories to Venus, Mars, and The Ice Giants: Mission Design with Human and Robotic Applications," No. December, 2016.

[3] H. Li, S. Chen, D. Izzo, and H. Baoyin, "Deep Networks as Approximators of Optimal Transfers Solutions in Multitarget Missions," *arXiv preprint*, 2019.

[4] D. Izzo, C. I. Sprague, and D. V. Tailor, "Machine Learning and Evolutionary Techniques in Interplanetary Trajectory Design," *Springer Optimization and Its Applications*, 2019, 10.1007/978-3-030-10501-3_8.

[5] P. Nath and R. V. Ramanan, "Precise halo orbit design and optimal transfer to halo orbits from earth using differential evolution," *Advances in Space Research*, 2016, 10.1016/j.asr.2015.10.033.

[6] A. D. Olds, C. A. Kluever, and M. L. Cupples, "Interplanetary mission design using differential evolution," *Journal of Spacecraft and Rockets*, 2007, 10.2514/1.27242.

[7] Y. Zhuang and H. Huang, "Time-optimal trajectory planning for underactuated spacecraft using a hybrid particle swarm optimization algorithm," *Acta Astronautica*, 2014, 10.1016/j.actaastro.2013.06.023.

[8] M. B. Penas, *A Rapid Grid-Search Technique for KBO Exploration Trajectories.* PhD thesis, Delft University of Technology, 2019.

[9] D. Izzo, L. F. Simões, M. Märtens, G. C. De Croon, A. Heritier, and C. H. Yam, "Search for a grand tour of the Jupiter Galilean moons," *GECCO 2013 - Proceedings of the 2013 Genetic and Evolutionary Computation Conference*, 2013, pp. 1301–1308, 10.1145/2463372.2463524.

[10] C. B. Browne, E. Powley, D. Whitehouse, S. M. Lucas, P. I. Cowling, P. Rohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton, "A survey of Monte Carlo tree search methods," 2012, 10.1109/TCIAIG.2012.2186810.

[11] S. Wagner and B. Wie, "Hybrid algorithm for multiple gravity-assist and impulsive Delta-V maneuvers," *Journal of Guidance, Control, and Dynamics*, Vol. 38, No. 11, 2015, pp. 2096–2107, 10.2514/1.G000874.

[12] R. B. Adams and G. A. Richardson, "Using the Two-Burn Escape Maneuver for Fast Transfers in the Solar System and Beyond," *AIAA Joint Propulsion Conference*, Nashville, Tennessee, 2010.

[13] M. J. Brennan, "Preliminary Interplanetary Trajectory Design Tools using Ballistic and Powered Gravity Assists," 2015.

[14] H. D. Curtis, *Orbital Mechanics for Engineering Students*. 2013, 10.1016/C2011-0-69685-1.

[15] G. R. Hollenbeck, "New Flight Techniques for Outer Planet Missions," *Am Astron Soc/AIAA Astrodyn Spec Conf*, 1975.

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

[17] 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.

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

[19] J. A. Sims, P. A. Finlayson, E. A. Rinderle, M. A. Vavrina, and T. D. Kowalkowski, "Implementation of a low-thrust trajectory optimization algorithm for preliminary design," *Collection of Technical Papers - AIAA/AAS Astrodynamics Specialist Conference, 2006*, 2006, 10.2514/6.2006-6746.

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

[21] W. B. Hubbard, "Ice Giants Decadal Study," *Planetary Science Decadal Survey*, 2010.

[22] D. Landau, T. Lam, and N. Strange, "Broad search and optimization of solar electric propulsion trajectories to Uranus and Neptune," *Advances in the Astronautical Sciences*, Vol. 135, No. January, 2010, pp. 2093–2112.

[23] N. Strange, R. Russell, and B. Buffington, "Mapping the V-infinity globe," *Advances in the Astronautical Sciences*, Vol. 129 PART 1, 2008, pp. 423–446.