AAS 20-686

Paper Link

Broad Trajectory Searches Using Monte Carlo Tree Search with the Inclusion of ΔVEGA Trajectories

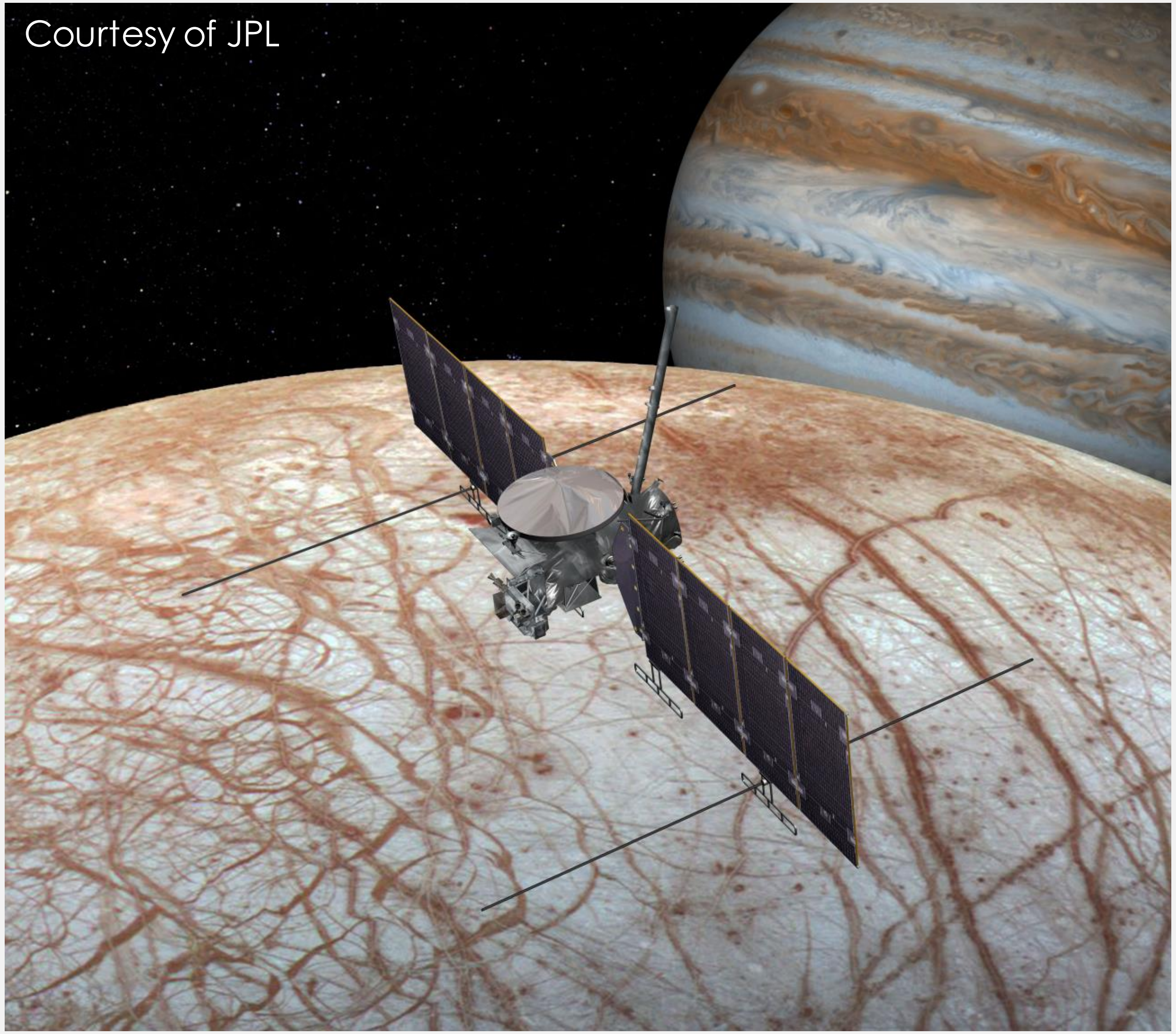
Authors: Burton Yale[*,†]
Rohan Patel[*,‡]
Jehosafat Cabrera[*,§]

Advisor: Navid Nakhjiri Ph.D[*,¶]

[*] 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

Introduction

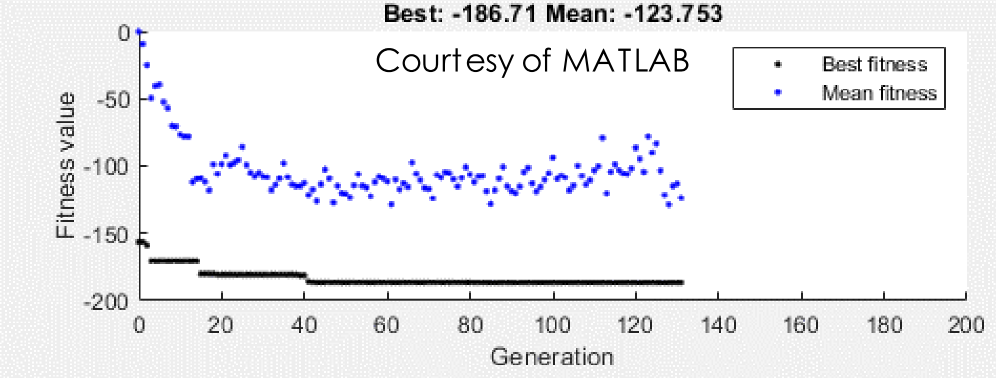
- Solution Space Complexity
  - With each additional flyby, another dimension is added to the solution space
- Broad Search Role
  - Reduce solution space to only areas of interest
  - **Solve for sequences** with rough timings to be converged in later steps



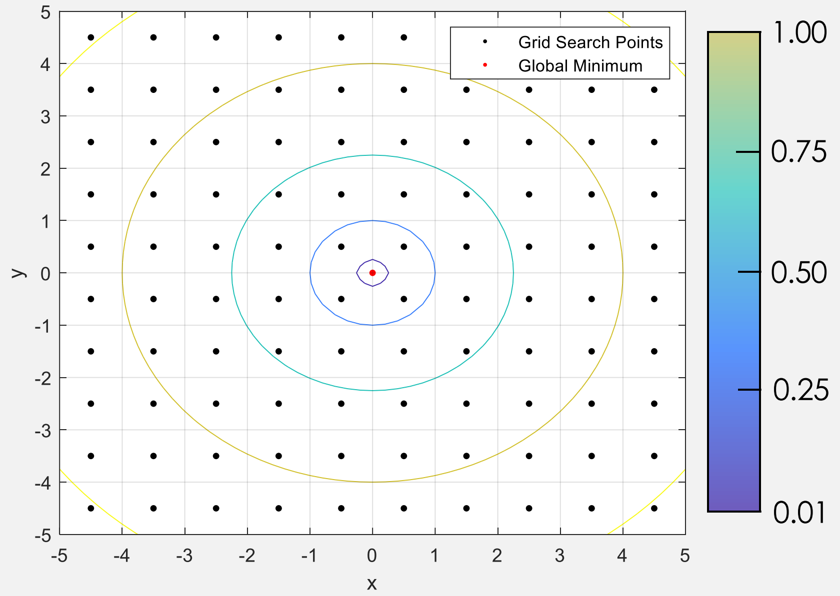
Broad searches are one of the first steps in trajectory design, by creating regions of interest

Previous Broad Search Algorithms

- Evolutionary Algorithms
  - Improves ability to search after each generation
  - Examples:
    - Particle Swarm Optimization
    - Differential Evolution
    - Genetic Algorithms
- Grid Search
  - Evenly Searches Solution Space
  - Examples:
    - Beam Search
    - Breadth-First Search
    - Depth-First Search
    - Lazy Race Tree Search



Genetic Algorithm Training



Grid Search Dispersion

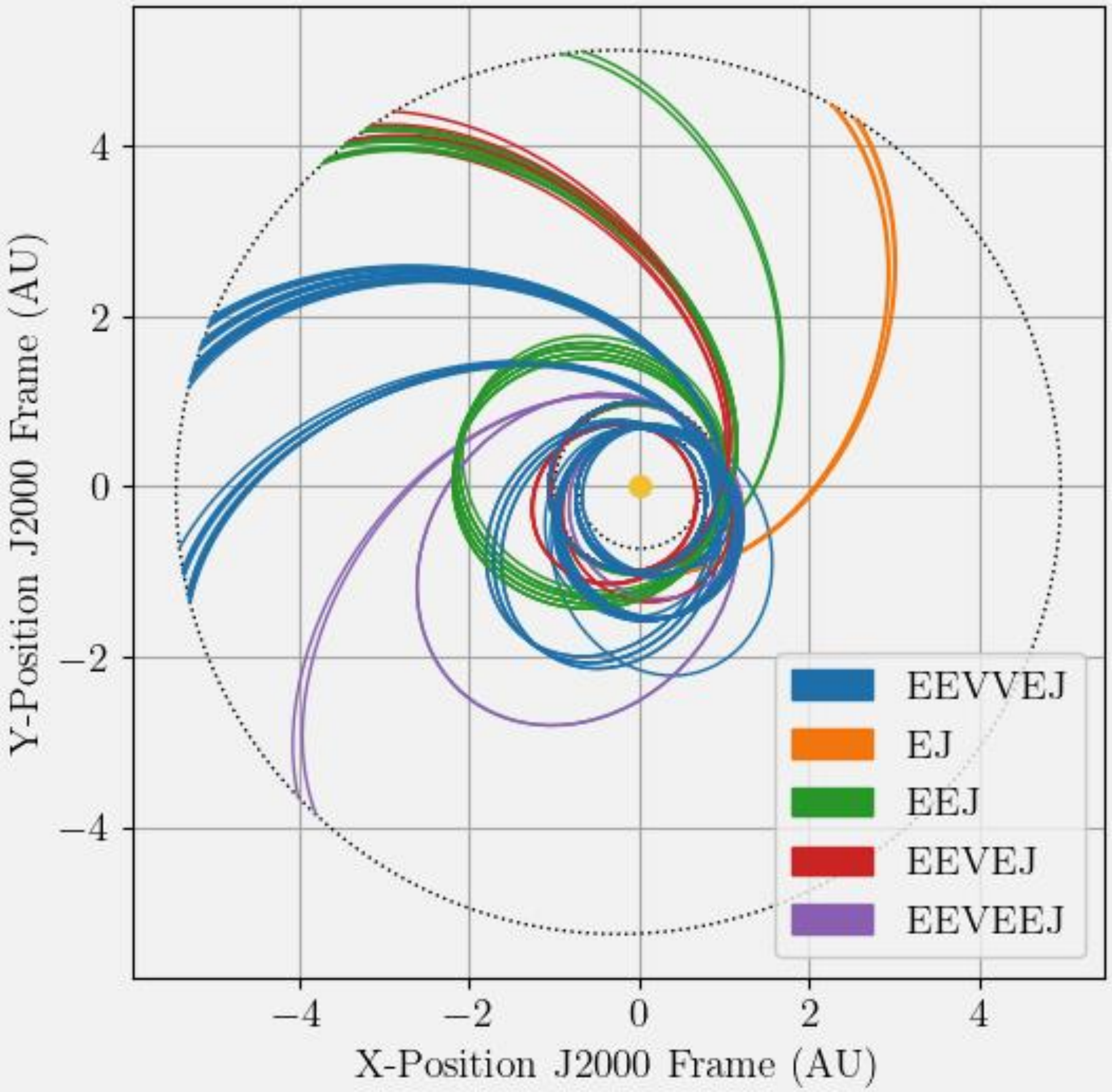
Grid searches are computationally expensive, but using heuristics reduces the cost

Objectives

- Create a tool to **find multi-flyby sequences** given solution space

- Rank solutions based off their viability for optimization

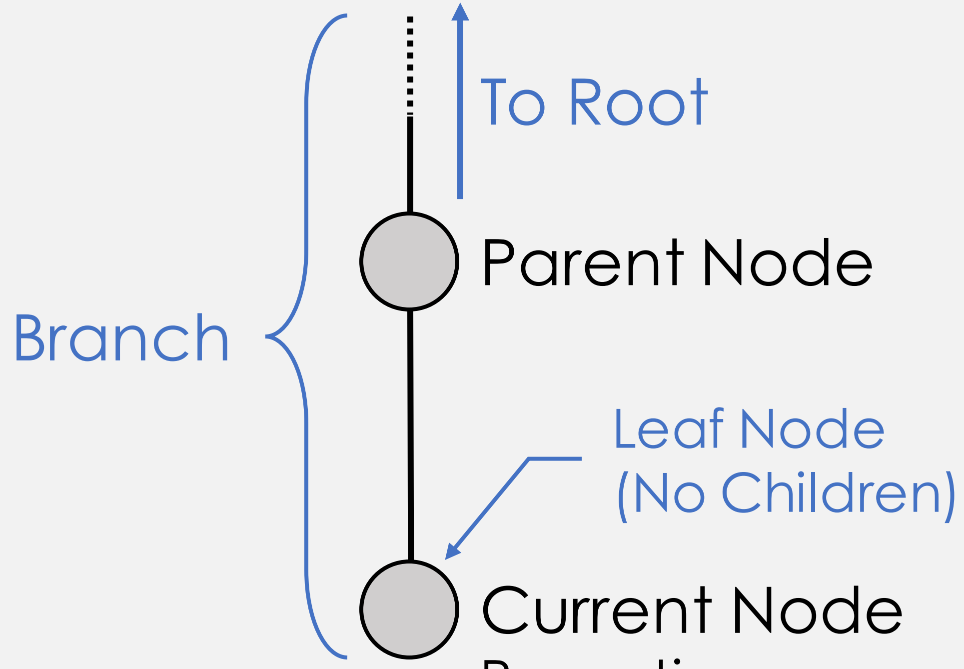
- Include the option for a ΔVEGA orbit in the solution space



The inclusion of ΔVEGA orbits provides a diverse set of possible trajectories to explore

Intro to **M**onte **C**arlo **T**ree **S**earches (MCTS)

- Used in competitive game AI (GO, Chess, etc.)
- Each node, ○, is one possible state
- Lines are transitions between states
- Uses **heuristics to narrow search options** to optimal candidates

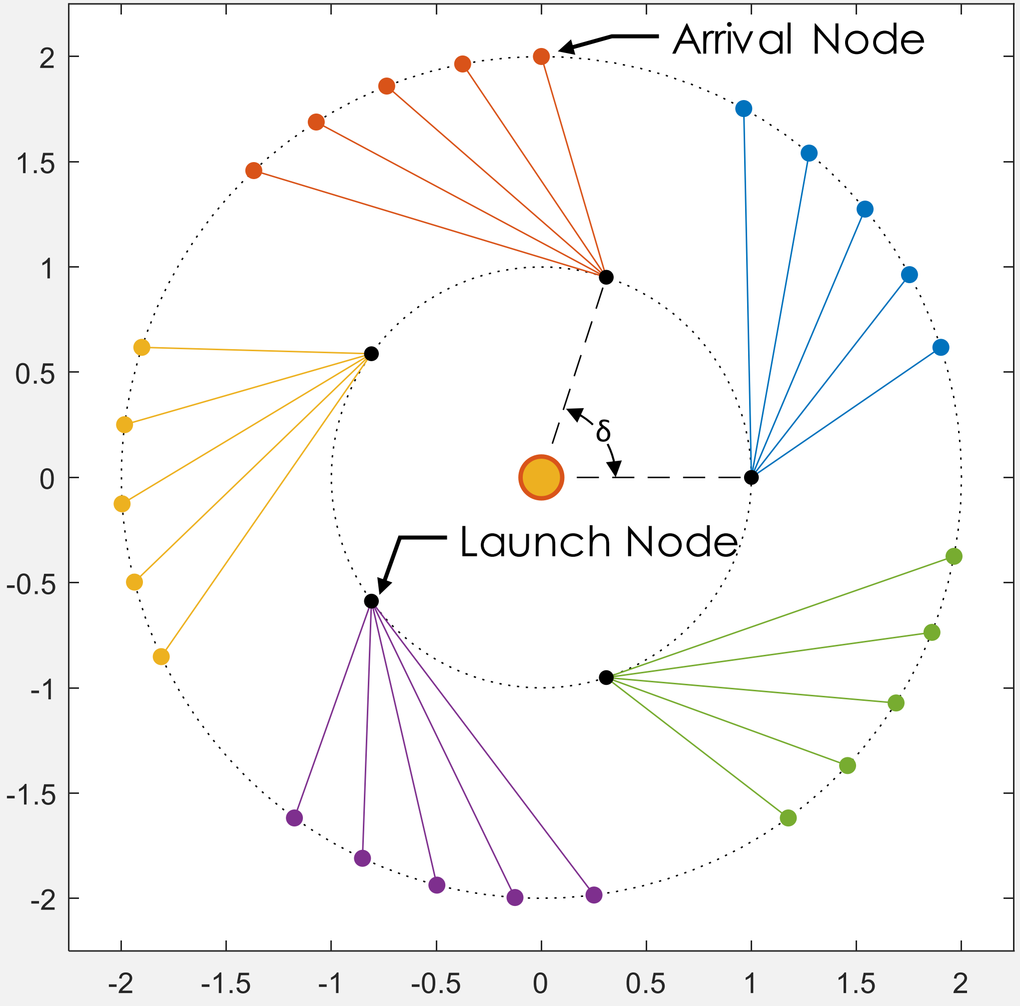


Properties:
- State
- Reward
- Visits
- Constraints met?

MCTS works best in environments with random behavior and minimal observability

Application to Broach Trajectory Searches

- Goal
  - Explore only the sequences that show promise while ignoring sequences that break constraints
- Angular Grid
  - Angular grids patterns better suit orbit conics than Cartesian grid patterns
- Node Properties
  - State: (Planetary NAIF ID, Encounter Epoch)
  - ΔV used to reach point in trajectory
- Constraints
  - Flyby Altitude/Bending Angle
  - ΔV Budget
  - Maximum C3
- Tree is **built using a loop of four steps**
  Selection → Expansion → Simulation → Backprop



Each node in a sequence is connected by a Lambert arc

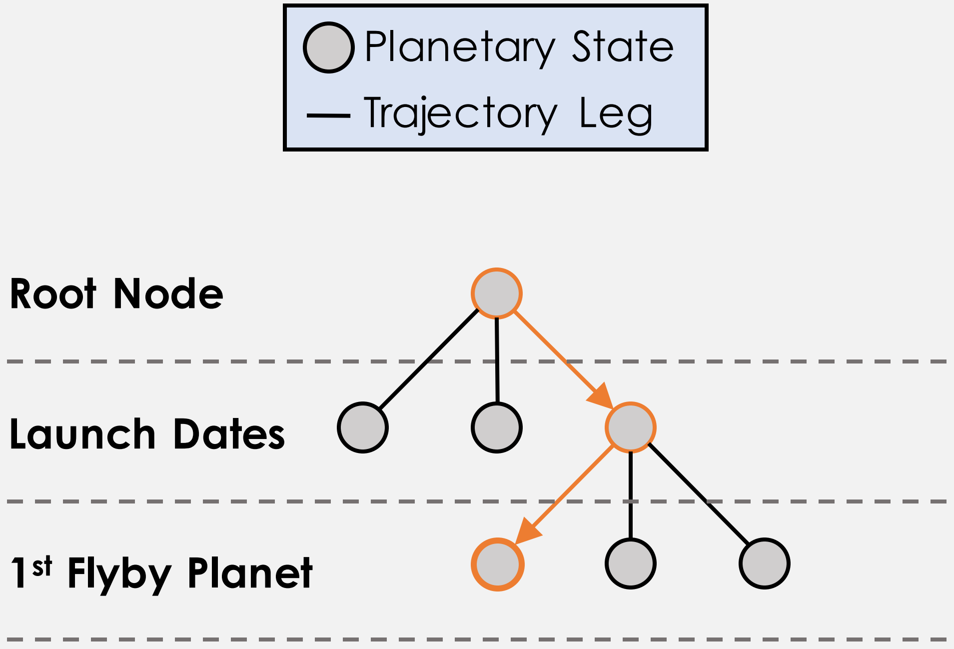
Application to Broach Trajectory Searches

- Paths down the tree to **find leaf a node**

- Estimates value of node through UCB1 function: [1]

$$X + C_p \sqrt{\ln n / N}$$

  - X: Future reward from child node
  - C_p: Exploration-Exploitation Parameter
  - n: Number of visits to current node
  - N: Number of visits to child node



Selection — Expansion — Simulation — Backprop

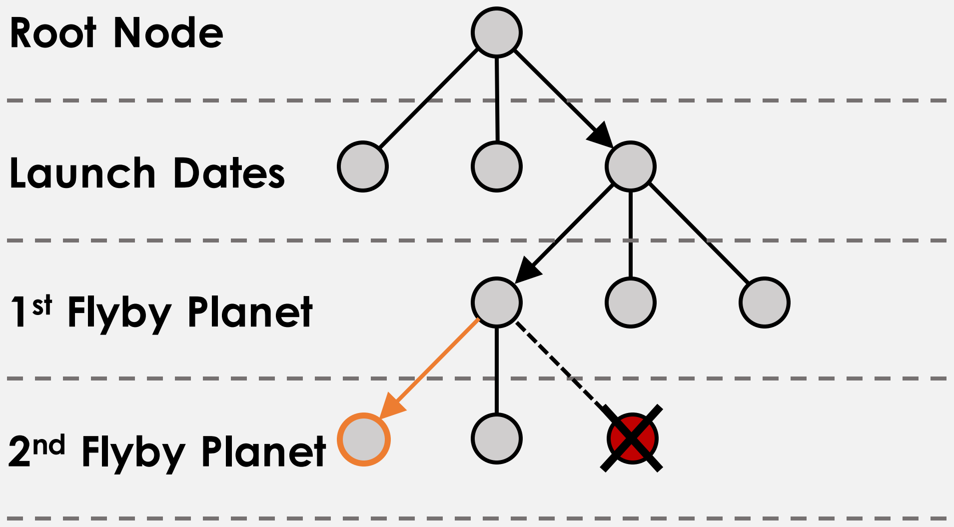
The C_p parameter was found to have the best results when set to 1/√2 [1]

Application to Broach Trajectory Searches

- Creates a **new layer of nodes** to explore
- Checks each child for feasibility
  - Calculates unoptimized ΔV using powered flyby assumption [2]
  - If node exceeds ΔV budget, it is made terminal,
- Selects lowest ΔV child by convention

Planetary State

— Trajectory Leg



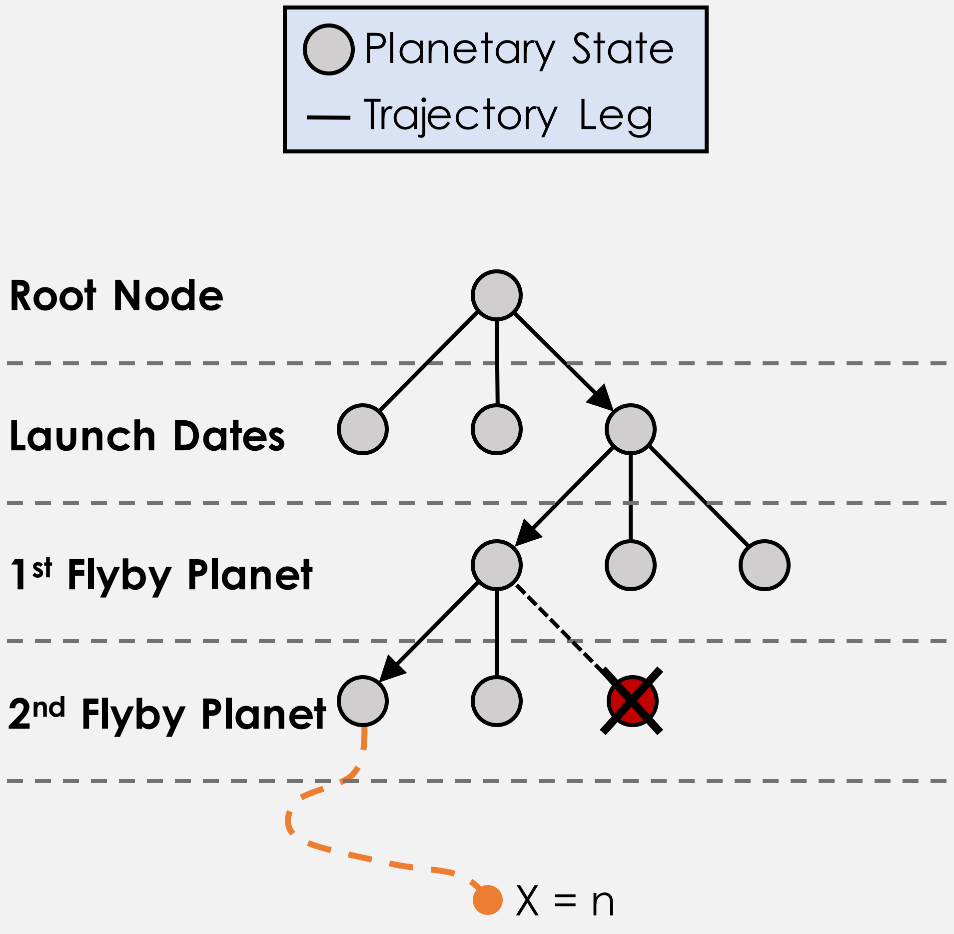
Selection — Expansion — Simulation — Backprop

Powered flyby approximations are used to mitigate the coarse nature of the grid search

Application to Broach Trajectory Searches

- Run Monte Carlo simulations from leaf node to determine future reward, *X*

- Takes **random steps** through possible decision and returns reward once terminal condition reached
  - Either the ΔV budget is exceeded or destination planet reached

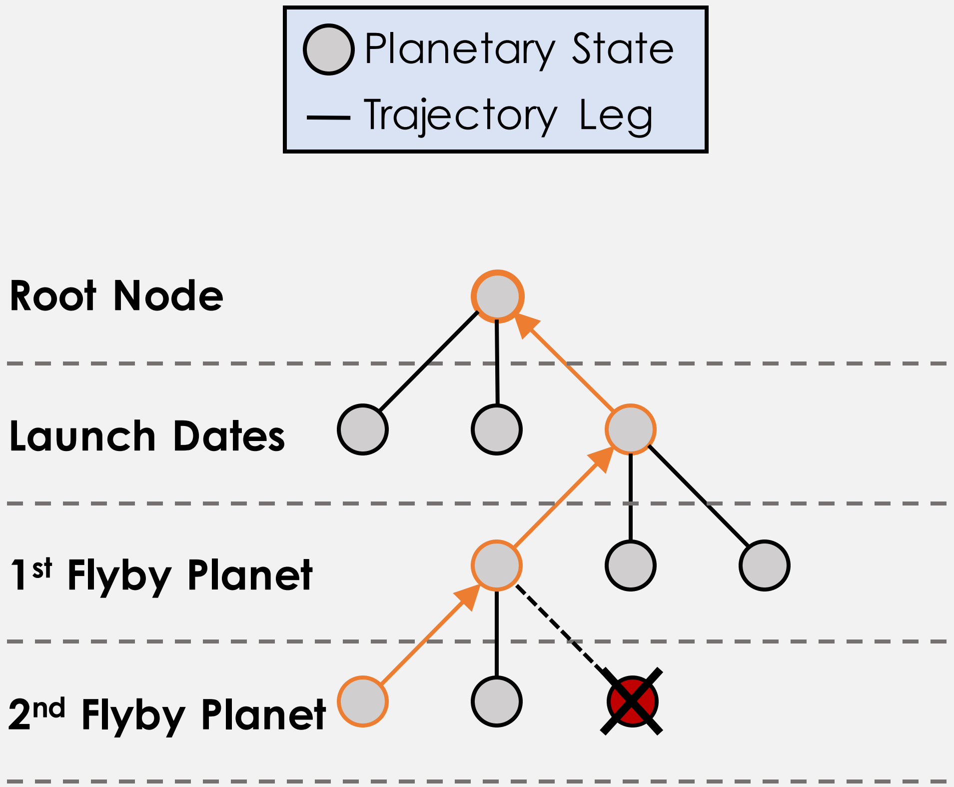


Selection — Expansion — Simulation — Backprop

A small bonus is provided to simulations that terminate early, corresponding to depth

Application to Broach Trajectory Searches

- Propagates simulated reward back through branch
- New rewards are weighted against previously received rewards
- **Creates a more accurate picture** of the branches as the tree builds

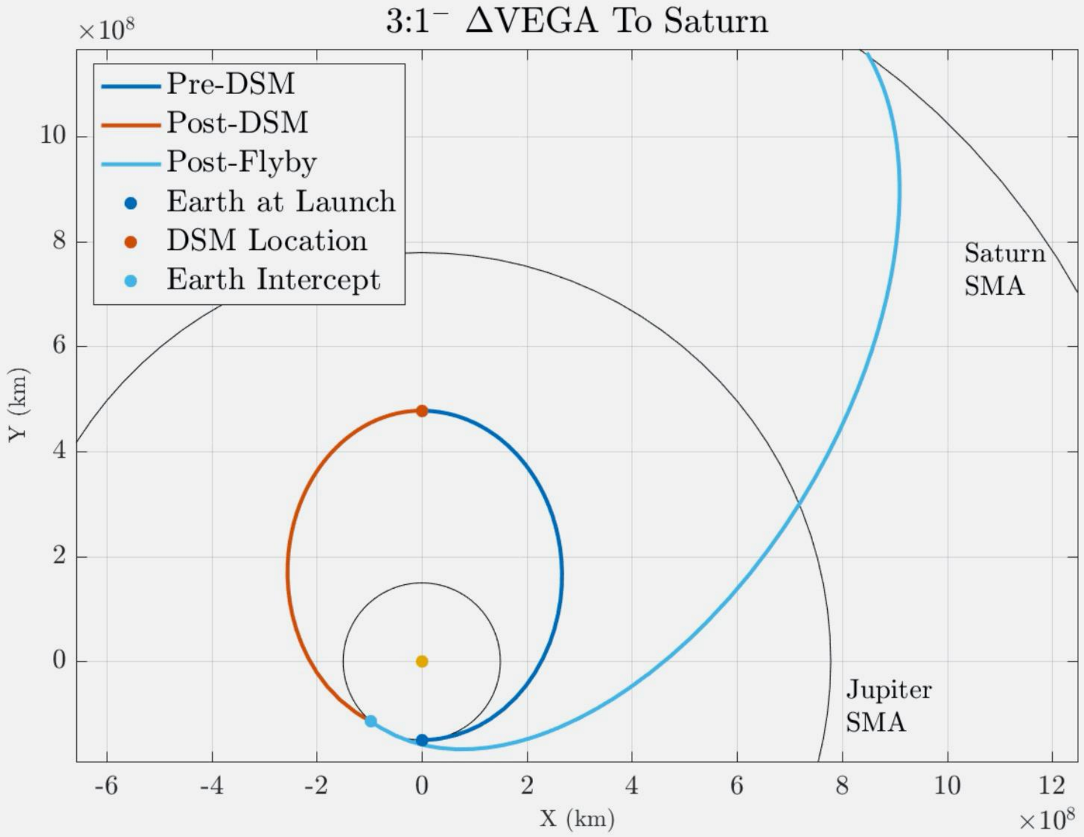


Selection —— Expansion —— Simulation —— Backprop

At the end of backpropagation, the loop continues until the iteration budget is gone

Implementation of ΔVEGA

- k:1$^{+/-}$ resonance of Earth (2, 3, and 4)
- Pre-DSM orbit properties determined from k
- Earth intercept point held fixed and a Lambert arc is used to determine DSM ΔV and EGA incoming relative velocity.
- Minimizer used to reduce normal (in-plane) component of ΔV by adjusting the pre-DSM orbit.[3]
- Solutions stored in lookup table and evenly spaced set of MCTS nodes correspond to each k and intercept true anomaly.

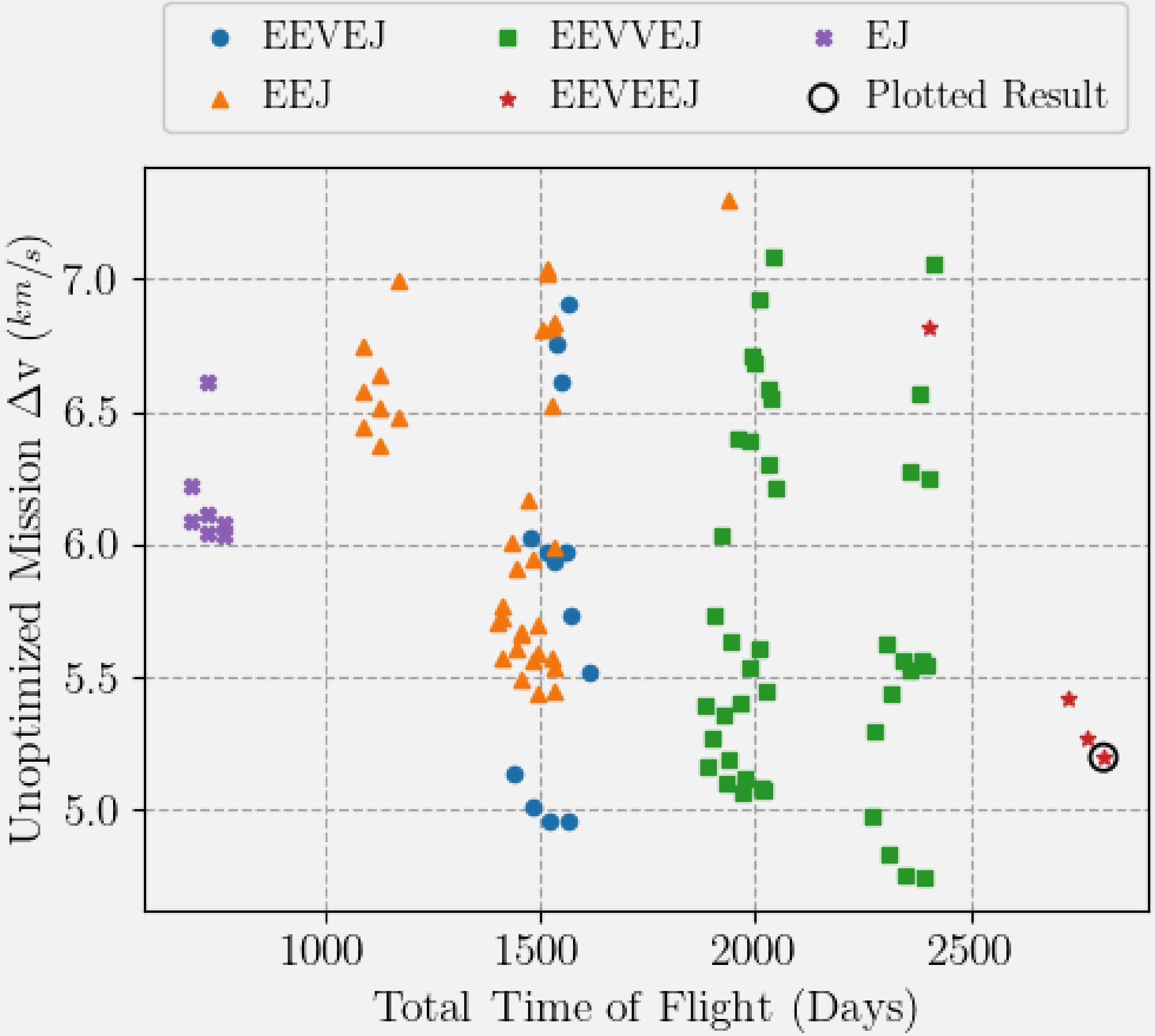


A lookup table method prevents the need of calculating the DSM during the tree search. This estimate serves as an initial condition for the optimization process.

Europa Clipper (1/2)

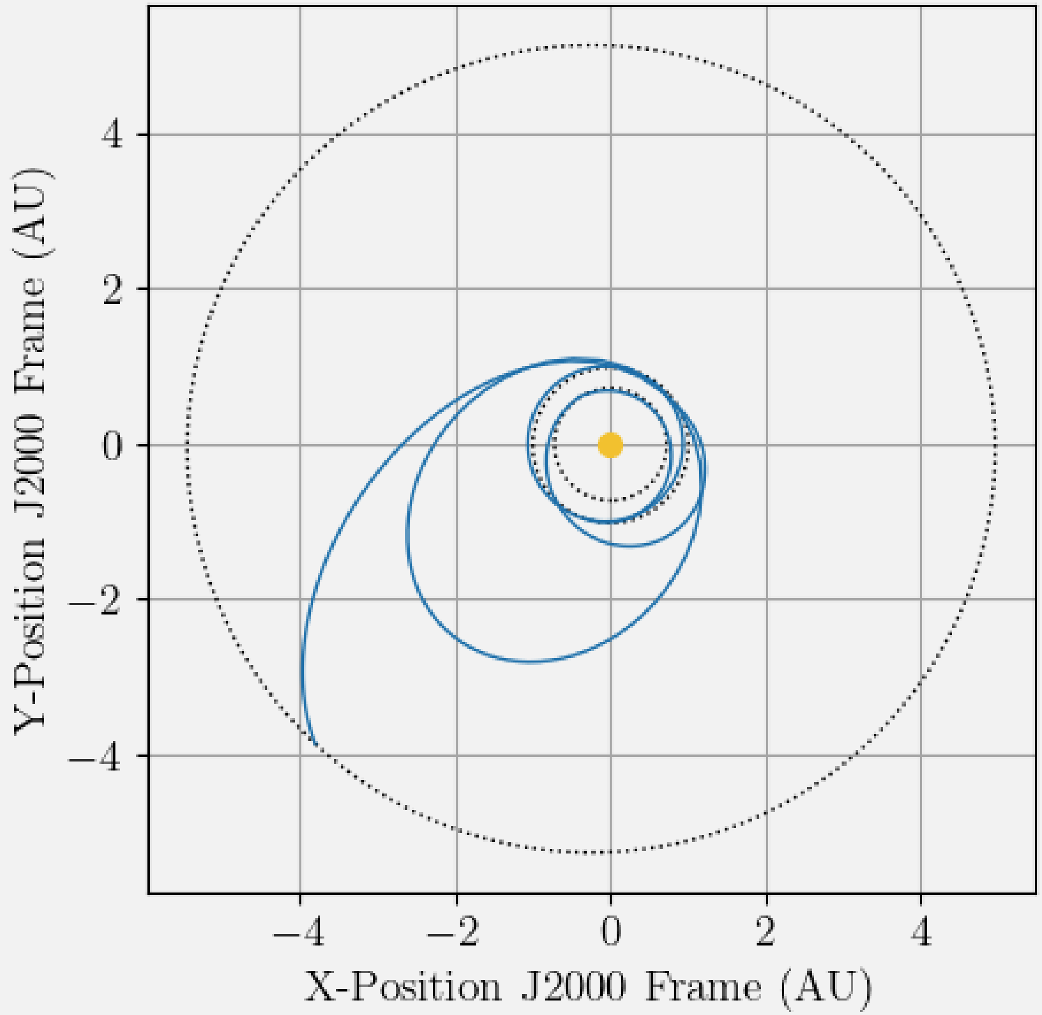
- Simulation Goal
  - Find EELV Europa Clipper trajectory (EEVEEJ) as described by Buffington[4]
  - Assess algorithm's ability to find long flyby sequences
- Tree Search Inputs

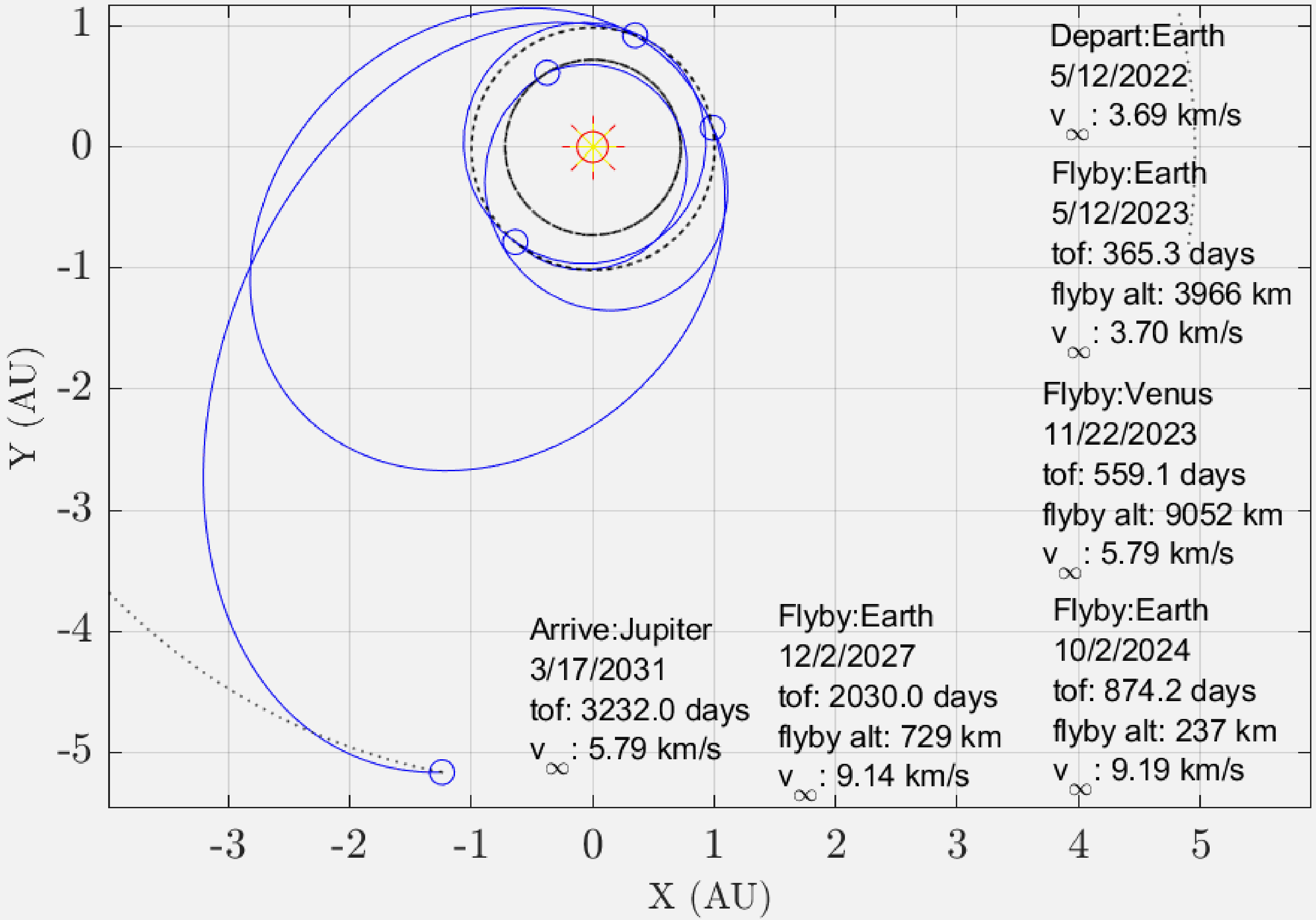
Input	Value
Arrival Planet	Jupiter
Launch Window	March 01 — September 01, 2022
Iterations	75,000
ΔV Budget	$10\ km/s$
Max C3	$10\ km^2/s^2$
Detail (d)	24



After the run completed, the tree search found 275 trajectories from 1.9B possibilities

Europa Clipper (2/2)



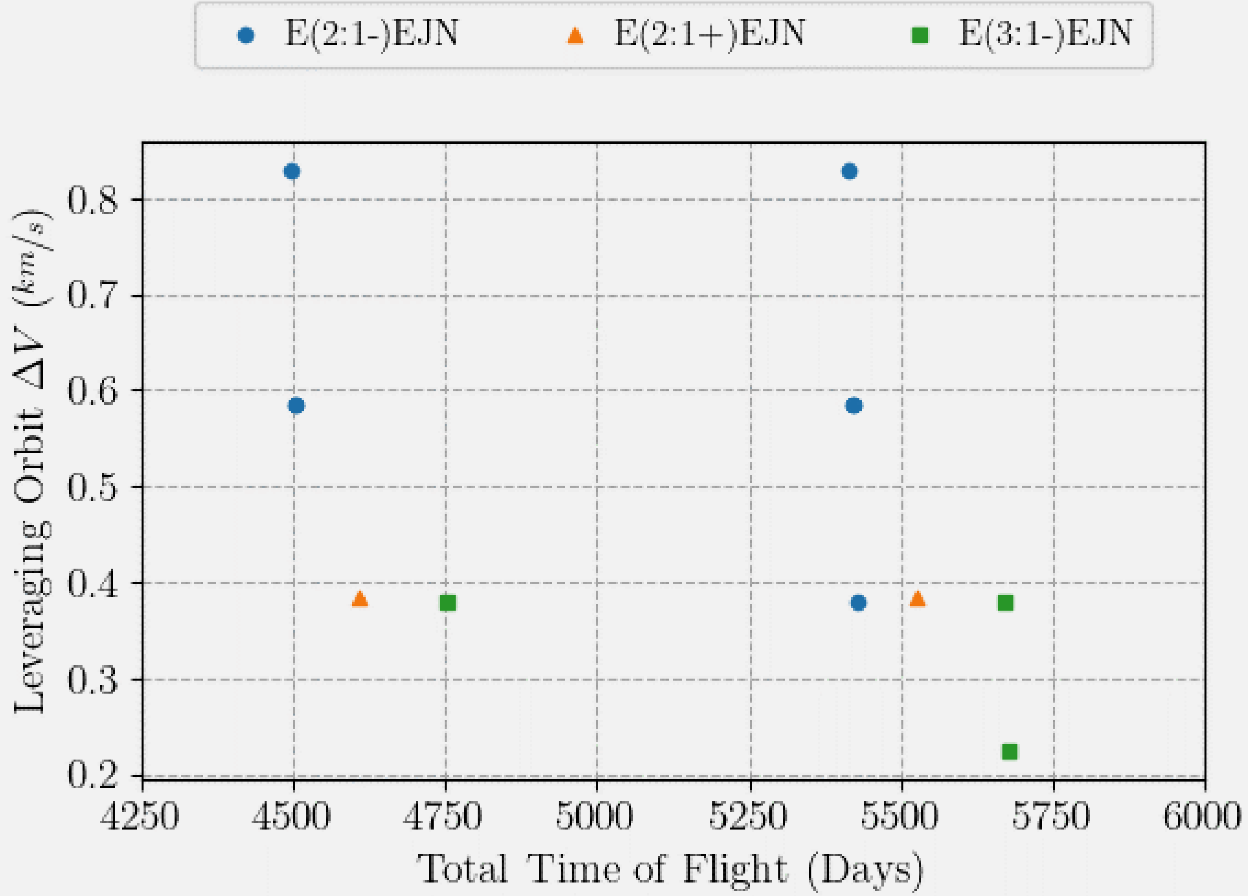


The tree search results (left) are within 30 days of their optimized counterparts (right) excluding Jupiter encounter. This confirms the algorithm's ability to find multi-flyby trajectories

Trajectories to Neptune (1/2)

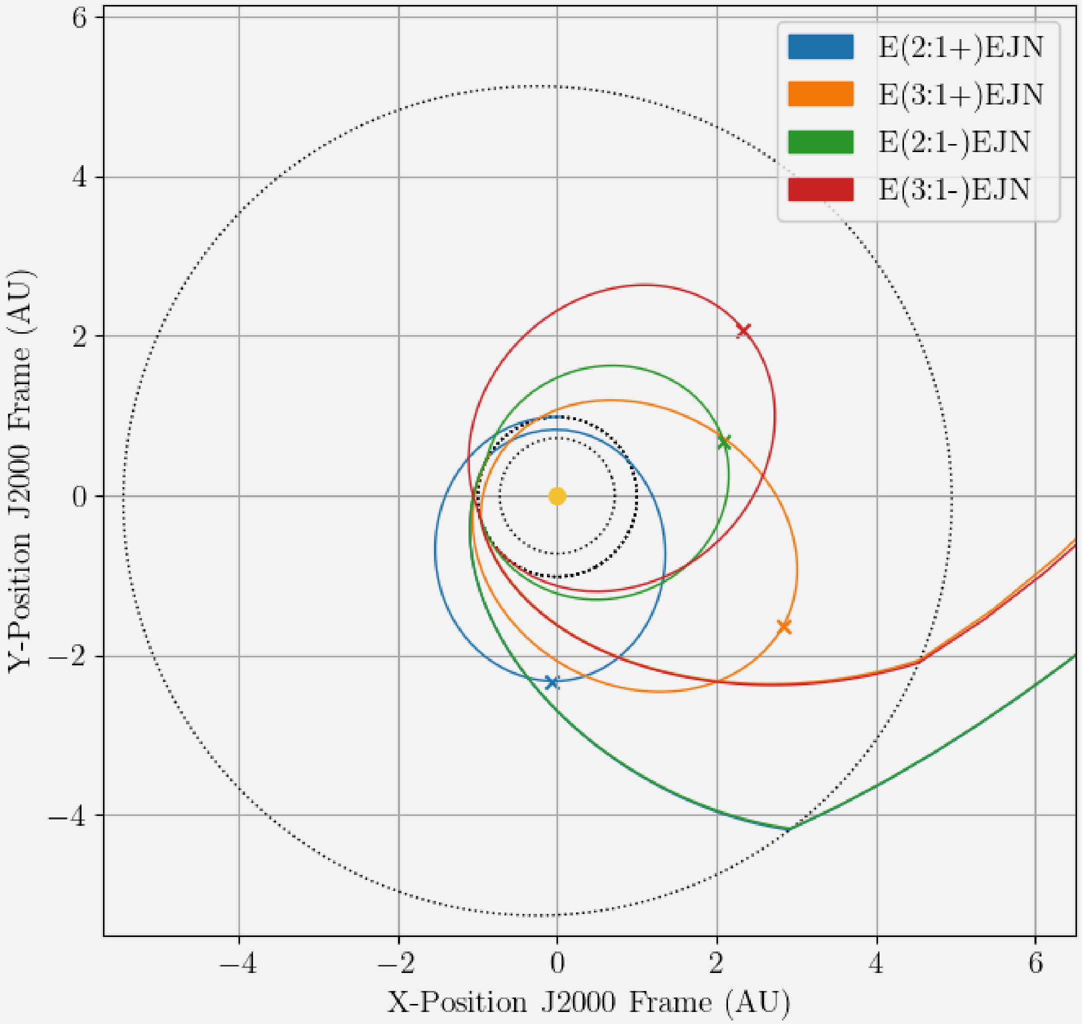
- Simulation Goal
  - Find trajectories to Neptune via a JGA and Earth orbit leveraging
  - Test ΔVEGA trajectories and the predicted required DSM ΔV
- Tree Search Inputs

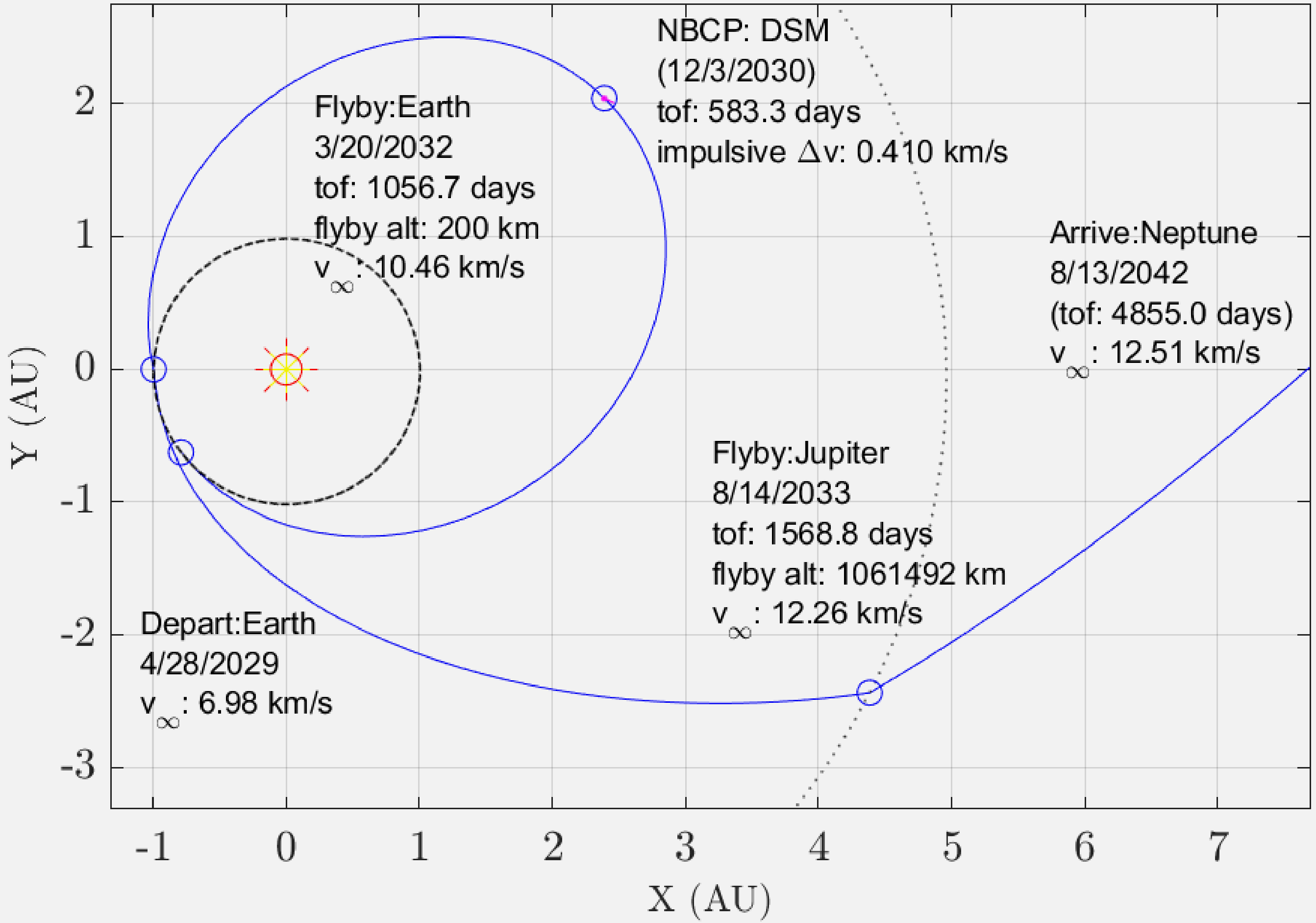
Input Name	Input Value
Arrival Planet	Neptune
Launch Window	Jan 01, 2029 — Jan 01, 2030
Iterations	15,000
ΔV Budget	$3\ ^{km}/_{s}$
Max C3	$60\ ^{km^2}/_{s^2}$
Detail (d)	16



Various ΔVEGA trajectory options exist for the EEJN search space.

Trajectories to Neptune (2/2)

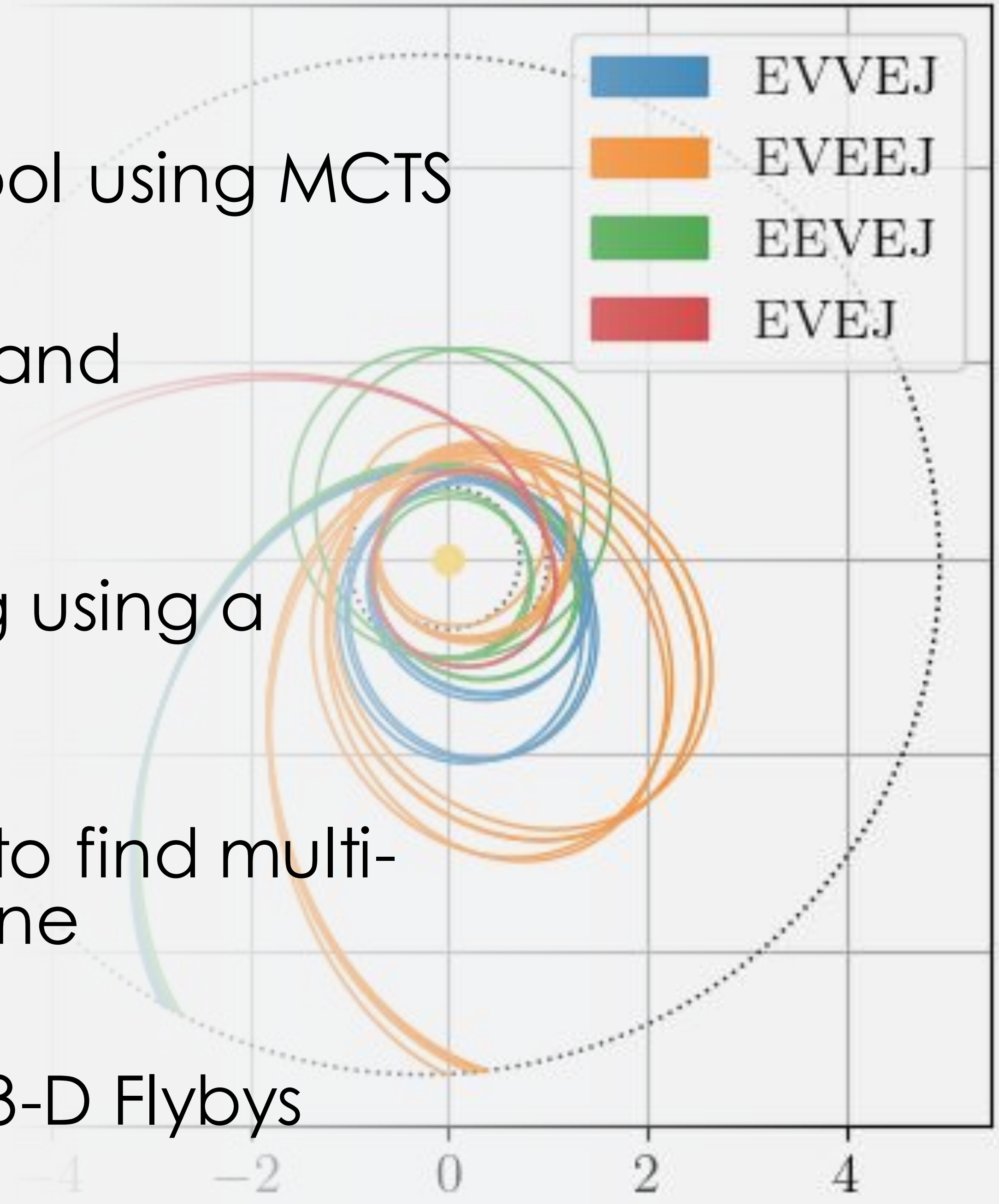




The MCTS results with Earth leveraging make for useful initial guesses for the optimizer, and the DSM ΔV estimate is <0.3 km/s of the predicted for all tested cases.

Conclusion

- Created a multi-flyby broad search tool using MCTS
- Good balance between exploration and exploitation of the search space
- Implemented ΔVEGA orbit leveraging using a lookup table solution
- Demonstrated algorithm's capability to find multi-flyby sequences to Jupiter and Neptune
- Future Work: V_∞ leveraging of Venus, 3-D Flybys



For any questions regarding the paper

Please join Virtual Room Trajectory Design and Optimization IV on August 12, 2020 at 11:20 AM EST.

Thank you

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] 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.

[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.