AAS 23-414

RAPID TRAJECTORY DESIGN IN MULTI-BODY SYSTEMS USING SAMPLING-BASED KINODYNAMIC PLANNING

Kristen L. Bruchko[*] and Natasha Bosanac[†]

In this paper, sampling-based kinodynamic planning is used to construct a roadmap that discretely summarizes a segment of the solution space in the Earth-Moon circular restricted three-body problem. The roadmap is a directed, weighted graph composed of nodes and edges that are constructed by randomly sampling configurations and trajectory segments that adhere to specified path constraints. Using graph search algorithms, the roadmap is searched to generate initial guesses that are corrected using collocation. This approach is demonstrated by constructing constrained transfers between planar and spatial periodic orbits near the Moon.

INTRODUCTION

Due to the chaotic nature of a multi-body gravitational environment, designing a trajectory that delivers a spacecraft to a specific target location subject to hardware, path, and operational constraints can be a challenging task. Current state-of-the-art approaches often rely on using dynamical systems theory for preliminary analysis of motions in a low-fidelity model such as the circular restricted three-body problem (CR3BP). In this model, a designer may select segments along fundamental solutions such as periodic orbits and their associated invariant manifolds to assemble a discontinuous initial guess with desired characteristics that is then corrected to produce a continuous trajectory. However, designing a preliminary spacecraft trajectory relies on a human-in-the-loop with expert knowledge of the environment. Furthermore, exploring the design space of potential trajectories that adhere to constraints can become intractable, especially if spacecraft requirements or parameters change frequently during the initial stages of mission development.

Similar challenges in rapidly constructing constrained trajectories and exploring the solution space exist in many other fields such as robotics, artificial intelligence, and control theory.[1] In these fields, path planning focuses on efficiently constructing collision-free paths from a source to a destination through the free space of the environment.[2] Foundational path planning techniques were originally used to solve problems for simple robots in static environments. However, recent path planning techniques are now capable of exploring complex environments where kinematic and dynamic constraints must also be satisfied; these are known as kinodynamic planning problems.[3]

Sampling-based path planning, such as probabilistic roadmap generation, relies on sampling to construct a graph, called the roadmap, that sufficiently summarizes the environment. The roadmap contains two key components: nodes, which are valid configurations an object may possess within

[*]Graduate Researcher, Colorado Center for Astrodynamics Research, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80303.
[†]Assistant Professor, Colorado Center for Astrodynamics Research, Smead Department of Aerospace Engineering Sciences, University of Colorado Boulder, Boulder, CO, 80303.

the environment, and edges, which are directed paths between nodes that can be weighted to reflect the difficulty of using the local path.[3] Once the roadmap is completed, the graph may be repeatedly searched to identify various paths connecting a specified pair of start and goal nodes that have been projected onto the roadmap. Sampling-based planners, such as probabilistic roadmap generation and rapidly-exploring random trees, have been demonstrated to rapidly solve high-dimensional problems in complex spaces with probabilistic completeness, i.e., the probability of finding a solution if one exists approaches unity as the number of nodes increases.[3]

Path planning methods have successfully been previously demonstrated in trajectory design problems by constructing graphs using known dynamical structures to represent a precomputed database of possible trajectories.[4,5] Dijkstra's search algorithm has then been demonstrated to search the graph for initial guesses for trajectories that adhere to required spacecraft parameters by Das-Stuart, Howell, and Folta as well as Trumbauer and Villac.[4,5] Starek et. al. have demonstrated that sampling-based kinodynamic planning can be used to automate guidance for real-time, propellant-optimized autonomous spacecraft.[6] Our ongoing work has built upon the foundation supplied by these contributions by using sampling-based kinodynamic planning to construct initial guesses for natural and maneuver-enabled transfers between Lyapunov and halo orbits using information from their stable and unstable manifolds or randomized motion.[7–9] This paper also builds upon our prior work by 1) constructing a generalized roadmap that sufficiently covers and connects the solution space near the Moon, 2) generating multiple paths that satisfy path constraints, and 3) applying this approach to explore spatial transfer design scenarios.

In this paper, sampling-based kinodynamic planning is used to generate transfers between libration point orbits in the Earth-Moon CR3BP. First, a generalized graph is constructed by randomly sampling nodes, which represent spacecraft states, and edges, which represent either impulsive maneuvers or natural trajectory arcs between nodes. The concept of a 1-neighborhood is used to guide the construction of these nodes and edges to improve the coverage and connectivity of the roadmap, both within and between neighborhoods. In the literature, neighborhoods that contain bundles of nodes and edges have been demonstrated to improve roadmap connectivity for motion planning queries by incorporating small locally intra-connected graphs that are later inter-connected,[1,10] similar to how local streets (edges) connect houses (nodes) within a neighborhood and highways connect neighborhoods. The goal of the resulting roadmap is to serve as a strongly connected graph that is a summarized, discrete representation of the continuous solution space.

The roadmap is augmented with application-specific information, and searched to generate sets of nodes and edges that supply initial guesses for trajectories. States along an initial and final orbit are projected onto the graph as new nodes and connected via edges. Additionally, if either orbit possesses stable or unstable manifolds, states and arcs along these manifold trajectories are added to the roadmap as nodes and natural edges, respectively, to improve the natural flow of the graph. Then, using the states along the initial and final orbit as the set of start and goal nodes for the search query, the roadmap is searched using Dijkstra's search algorithm to identify the optimal path with respect to a chosen heuristic. Additional unique paths are rapidly generated using Yen's algorithm, a k-best search algorithm that repeatedly searches various subgraphs of the roadmap.[11] For each path, the largest edge weights guide maneuver placement. These initial guesses and maneuver locations are then input to a collocation scheme to recover a set of continuous, maneuver-enabled trajectories with distinct maneuver costs and geometries.

BACKGROUND

Dynamical Model

In this paper, the circular restricted three body problem (CR3BP) is used to sufficiently capture the complexity of the dynamical environment in cislunar space while lowering the computational time for roadmap construction. This low-fidelity, autonomous dynamical model approximates the motion of a spacecraft under the gravitational influence of two primary bodies, i.e., the Earth and the Moon. The spacecraft is assumed to possess a comparatively negligible mass and the two larger primary bodies, P_1 and P_2, are assumed to have constant mass. These two primaries are also assumed to travel on circular orbits about their barycenter. A rotating reference frame is defined with the origin at the system's barycenter and axes $\hat{x}, \hat{y}, \hat{z}$: $\hat{x}$ is directed from P_1 to P_2, $\hat{z}$ is aligned with the system's orbital angular momentum vector, and $\hat{y}$ completes the right-handed orthogonal triad.[12] Then, mass, length, and time quantities are nondimensionalized by the characteristic quantities m^*, l^*, and t^*. Specifically, m^* is equal to the total mass of the system, t^* is defined such that the nondimensional mean motion of the primaries is unity, and l^* equals the assumed constant distance between the primaries. In the rotating frame, the nondimensional spacecraft state is then defined as $\boldsymbol{x} = [x, y, z, \dot{x}, \dot{y}, \dot{z}]^T$. Using these assumptions and definitions, the equations of motion governing the spacecraft are written in the rotating frame as[13]

$$\ddot{x} = 2\dot{y} + \frac{\partial U}{\partial x}, \qquad \ddot{y} = -2\dot{x} + \frac{\partial U}{\partial y}, \qquad \ddot{z} = \frac{\partial U}{\partial z} \tag{1}$$

where the mass ratio of the system, μ, represents the ratio of the mass of the smaller primary to the total mass of the system; $U = \frac{1}{2}(x^2 + y^2) + (1-\mu)/r_1 + \mu/r_2$; and the distances of the spacecraft from P_1 and P_2 are $r_1 = \sqrt{(x+\mu)^2 + y^2 + z^2}$ and $r_2 = \sqrt{(x-1+\mu)^2 + y^2 + z^2}$, respectively. Additionally, this dynamical model admits an integral of motion in the rotating frame, commonly referred to as the Jacobi constant and equal to[13]

$$C_J = 2U - \dot{x}^2 - \dot{y}^2 - \dot{z}^2 \tag{2}$$

This energy-like quantity supplies insight into the accessible regions of the system.

Fundamental Solutions

In the CR3BP, a variety of fundamental solutions exist, including five equilibrium points, continuous families of periodic and quasi-periodic orbits, and their associated hyperbolic invariant manifolds.[12] These natural transport mechanisms are often used as the basis for constructing initial guesses for complex trajectories. The roadmap generated in this paper to support trajectory design is also augmented by information from these fundamental solutions, where applicable.

Periodic orbits are paths that repeat in the rotating frame and exist in continuous families throughout the system. Periodic orbits generated in the CR3BP are often used in trajectory design because, in many cases, they indicate the existence of similar bounded motions in the rotating frame in higher-fidelity models. Additionally, a stability analysis of a periodic orbit reveals the behavior of nearby motion. Specifically, the monodromy matrix, i.e., the state transition matrix propagated for exactly one orbital period, is decomposed into reciprocal pairs of eigenvalues. A pair of real, reciprocal eigenvalues indicates the existence of stable and unstable hyperbolic invariant manifolds.[12]

A state along a stable or unstable manifold naturally approaches or departs the periodic orbit, respectively, as time tends to infinity.[12] These stable and unstable invariant manifolds govern natural

transport throughout the system and are, therefore, often used to design free or low-cost transfers between their associated periodic orbits. Stable and unstable manifolds are generated numerically. First, a periodic orbit is discretized into a sequence of states and a small perturbation is applied in the direction of the stable or unstable eigenspace. Each perturbed state is then numerically integrated backward or forward in time, respectively, to produce a trajectory along the manifold. This process is repeated for each discretized state along the periodic orbit and perturbations in each direction within the eigenspace, producing a numerical approximation of the stable or unstable manifold.

Numerically Correcting Trajectories

A two-point boundary value problem is solved to recover a continuous trajectory using a free variable and constraint vector formulation of collocation. Collocation is a numerical corrections technique for implicitly integrating differential equations of a dynamical system to recover a continuous trajectory.[14] The collocation approach with an additional mesh refinement process presented in this paper follows the procedure presented by Grebow and Pavlak as well as Pritchett.[14,15]

An initial guess, comprised of s discontinuous trajectory segments, is discretized into a mesh by dividing each segment into p arcs. This mesh includes $s+1$ boundary nodes between consecutive arcs and collocation nodes placed along each arc using an n^{th} order polynomial and Legendre-Gauss-Lobatto (LGL) node spacing. An LGL node spacing strategy is employed in this paper due to its higher order of accuracy and simplified design problem due to placing collocation nodes at the boundary nodes of each segment. A 7^{th} order polynomial has previously been demonstrated by Pritchett as well as Grebow and Pavlak to enable high accuracy trajectory generation in multi-body systems.[14,15] With a 7^{th} order LGL node spacing, 7 collocation nodes are placed along each arc at the roots of the derivative of the $(n-1)^{th}$ polynomial. The time τ along the j-th arc along the i-th segment is also normalized to $[-1,1]$ using the following transformation:

$$\tau = \frac{2}{\Delta t_i^j}(t - t_i^{j,n}) - 1 \tag{3}$$

where the integration time along the arc is $\Delta t_i^j = t_{i+1}^{j,1} - t_i^{j,n}$.

Along each arc, the collocation nodes are categorized into free nodes, placed at the odd-numbered nodes, and defect nodes, placed at the even-numbered nodes. The state components at the free nodes, $\boldsymbol{x}$, are used to construct the polynomial approximation of states the motion along an arc. This polynomial representation is then evaluated at the defect nodes, $\boldsymbol{p}$, and compared to the system dynamics.

To construct a free variable vector, all unique states at the free nodes as well as the integration time along each arc are designated as the design variables. The free variable vector along segment $i = [1, s]$ discretized into p arcs is defined as

$$\boldsymbol{X_i} = \begin{bmatrix} \boldsymbol{x}_i^{1,1^T} & \dots & \boldsymbol{x}_i^{1,n-2^T} & \Delta t_i^1 & \dots & \boldsymbol{x}_i^{p,1^T} & \dots & \boldsymbol{x}_i^{p,n^T} & \Delta t_i^p \end{bmatrix}^T \tag{4}$$

All design variables along each segment are then reshaped into a single column vector, defined as

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{X}_1 & \boldsymbol{X}_2 & \dots & \boldsymbol{X}_s \end{bmatrix}^T \tag{5}$$

which is the total free variable vector for the collocation problem.

A constraint vector is then composed of continuity constraints between trajectory segments and defect constraints at each defect node. The continuity constraint between the end of the i^{th} segment and the start of the $(i+1)^{th}$ segment is defined as

$$\boldsymbol{F}_{c_i} = \begin{cases} \boldsymbol{x}_i^{p_i,7} - \boldsymbol{x}_{i+1}^{1,1}, \text{if no maneuver} \\ \boldsymbol{r}_i^{p_i,7} - \boldsymbol{r}_{i+1}^{1,1}, \text{if maneuver} \end{cases} \tag{6}$$

where $\boldsymbol{r} = [x, y, z]^T$. Then, the constraint vector that enforces continuity between consecutive segments along the entire trajectory is defined as

$$\boldsymbol{F}_c = \begin{bmatrix} \boldsymbol{F}_{c_1}^T & \boldsymbol{F}_{c_2}^T & \dots & \boldsymbol{F}_{c_{s-1}}^T \end{bmatrix} \tag{7}$$

The defect constraints for the j-th arc along the i-th segment are defined as

$$\boldsymbol{F}_{d_i}^j = \begin{bmatrix} (\dot{\boldsymbol{p}}_i^{j,2^T} - \dot{\boldsymbol{x}}_i^{j,2^T})w_2, & (\dot{\boldsymbol{p}}_i^{j,4^T} - \dot{\boldsymbol{x}}_i^{j,4^T})w_4, & (\dot{\boldsymbol{p}}_i^{j,6^T} - \dot{\boldsymbol{x}}_i^{j,6^T})w_6 \end{bmatrix} \tag{8}$$

where w_i are the quadrature LGL node weights and $\dot{\boldsymbol{p}}$ is the derivative of the polynomial state with respect to normalized time. Then, the defect constraints for the i-th segment are written as $\boldsymbol{F}_{d_i} = [\boldsymbol{F}_{d_i}^1, \boldsymbol{F}_{d_i}^2, \dots, \boldsymbol{F}_{d_i}^p]^T$. Finally, the complete constraint vector is defined as

$$\boldsymbol{F}(\boldsymbol{X}) = \begin{bmatrix} \boldsymbol{F}_c & \boldsymbol{F}_{d_1} & \boldsymbol{F}_{d_2} & \dots & \boldsymbol{F}_{d_s} \end{bmatrix}^T \tag{9}$$

The discretized mesh is iteratively updated by applying Newton's method to adjust the location of the free nodes along each arc until the resulting polynomials satisfy the system dynamics at every defect node and the continuity constraints are met within a specified tolerance of 10^{-12}.

Last, a control with explicit propagation (CEP) mesh refinement method is used in this paper to update the number of arcs in the mesh to further increase the accuracy of the polynomials along the final solution trajectory, following the approach presented by Pritchett.[14] After the collocation problem is solved, given the pre-specified mesh, the dynamics at each defect node may be sufficiently satisfied by the polynomials. However, the trajectory may not be accurately approximated at other locations. First, subsequent arcs are tested to be combined into one arc. The initial boundary node on the i^{th} arc is propagated for time $t = \Delta t_i^j + \Delta t_i^{j+1}$. If the difference between the state at the end of this propagation time and the final boundary node on the subsequent arc is below a specified tolerance, set to 10^{-12}, the arcs are combined into one arc and the location of the collocation nodes are recomputed. The collocation problem is iteratively solved until the number of arcs does not change. Next, arcs are added by splitting arcs that do not meet the accuracy of the solution within the tolerance specified. The initial boundary node of each arc is propagated for the integration time associated with that arc. If the difference between the state at the end of this propagation time and the final boundary node on the arc is above a specified tolerance, set to 10^{-12}, the arc is split into two arcs of each integration time. Again, the collocation problem is then iteratively solved until the number of arcs does not change. Using the CEP mesh refinement method results in a higher accuracy along the trajectory that is not dependent on an initial user-defined mesh.[14]

Sampling-Based Kinodynamic Planning

In the fields of robotics, artificial intelligence, and control theory, path planning techniques have been used to construct a collision-free path between two configurations.[1] Path planning, also referred to as motion planning, algorithms that were originally designed for simple linear environments have since evolved to handle more challenging problems, such as planning paths with uncertainty, adhering to physical laws, and geometric constraints. Kinodynamic planning attempts to

solve a motion planning problem subject to kinematic and dynamic constraints, e.g., finding a path from a given start position and velocity to a goal position and velocity while avoiding obstacles and respecting constraints on velocity and acceleration.[16] Common path planning algorithms typically fall into three categories: potential functions, cell decompositions, or sampling-based algorithms.[3]

Sampling-based planning algorithms aim to solve the planning problem by generating samples which are collision-free, valid configurations in the environment, and by connecting the samples with collision-free, valid paths.[3] Two popular algorithms are probabilistic roadmap (PRM) generation and rapidly-exploring random tree (RRT). The PRM planner focuses on generating a graph, called the roadmap, that sufficiently summarizes the environment and can be used for multiple search queries. A RRT explores the environment randomly by generating a space-filling tree from a given start configuration until the tree reaches a desired goal configuration. As such, it is most often used for single query problems. These planners are popular for solving higher-dimensional problems while being probabilistically complete, meaning the planner will eventually return a solution if one exists and the number of samples drawn increases. While the main focus of this paper will rely heavily on traditional PRM methods, some steps, such as the construction of neighborhoods that will be defined in the next section, will be inspired by tree-based methods such as RRT.

Probabilistic Roadmap Generation

Path-planning with probabilistic roadmap generation is performed in two key phases: the learning phase and the query phase. During the learning phase, a directed, weighted graph, called the roadmap, containing nodes and edges is constructed. Nodes represent valid configurations in the environment and edges represent valid paths, or motion, between neighboring nodes. Valid configurations and paths must adhere to all system dynamics without intersecting any known obstacles. Nodes added to the graph are determined by the selected sampling scheme, such as uniform, random, or informed sampling, or a combination of techniques.[3] Nodes are added to the graph until either a desired number of valid nodes are contained within the graph or until a graph metric, such as environment coverage or density of nodes, is satisfied.[1,17]

After the desired number of nodes are added to the graph, edges are constructed between neighboring nodes. Neighbors may be identified using methods such as radius-based measures or the k-nearest neighbors.[3] These edges can either be directed, meaning the path can only be traversed in one direction, or undirected. Edges can also be weighted with a non negative edge weight to reflect either a desired characteristic of the environment or the likelihood of a specific edge being traversed.

Graph-based metrics may also be used to dictate the number of edges in the graph, such as constructing edges until the graph is either weakly or strongly connected. A weakly connected graph contains all nodes in the same component and there exists a path to every node starting from any other node, assuming each edge can be traversed in any direction. A strongly connected graph contains all nodes in the same connected component, but there exists a path between all nodes while still considering the direction of edges. Once the desired number of nodes and edges are added to the graph, or other graph-based metrics are satisfied, the graph is considered complete.

Next, the query phase occurs where the roadmap is searched for paths between user-specified start and goal configurations. To search the roadmap, the graph is queried using a graph search method, such as a depth-first or breadth-first search algorithm, for various paths between given start node(s) and goal node(s). A conceptual example of the PRM algorithm appears in Figure 1, where nodes are indicated by circles, obstacles are indicated by squares, and directed, weighted edges are indicated with a dashed arrow and the weight indicated with a non-negative number. During the

query, a search algorithm, Dijkstra's algorithm in this example, is used to compute the lowest cost path from the start node, indicated by the blue circle, to the goal node, indicated by the red circle. A completed roadmap may be queried as many times as needed to produce a variety of unique paths between the same start and goal node(s).

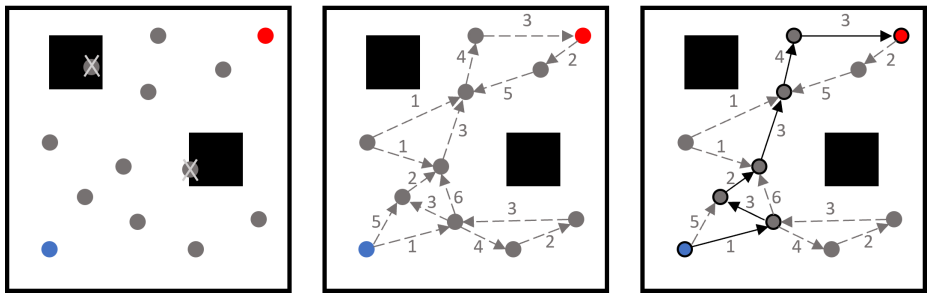

Figure 1. Example of probabilistic roadmap generation: a) sample valid nodes, b) connect nodes to create directed, weighted edges, and c) search the completed roadmap for an optimal solution path with respect to the sum of edge weights. Image reproduced from Bruchko and Bosanac 2022.[8]

Graph Search Methods

Dijkstra's algorithm is a graph search method to identify the shortest, or lowest cost path if a heuristic is selected, between two nodes, or two sets of nodes, in a weighted, directed graph. The algorithm searches the graph starting from the start node(s) by creating two lists: the priority list and the closed list. The priority list contains the nodes currently being explored at a single iteration and the closed list contains all nodes previously explored, ensuring an efficient algorithm that does not explore any node more than once. To start the priority list, all neighboring nodes from the start node(s) are added to the list, meaning any directed edge away from the start node(s) is explored first, and the start node(s) is added to the closed list. Additionally, the total cost, or weight, of each path from the start node(s) to the current neighboring node being searched is retained within the priority list. The total cost of all potential paths is then used to sort the priority list and find the neighbor with the lowest cost path away from a start node(s), indicating the next best node to explore. The next best node is then removed from the priority list and added to the closed list, while all of its neighboring nodes and their associated total weights from the start node are added to the priority list. The priority list is then sorted again to find the next best node to explore. This process is repeated until the goal node is contained within the priority list, indicating a solution has been found, or until all nodes have been explored. The result of this procedure is the optimal path based on the heuristic selected for the edge weights.

Additional sub-optimal paths may also be generated from a single roadmap. One approach involves searching subgraphs, defined as graphs containing a subset of the nodes and edges of a completed graph, repeatedly until the desired number of paths is returned or until all paths have been explored. This is commonly referred to as the k-shortest paths problem, where the paths become slightly longer or more expensive as the number of paths found increases.[18] Most algorithms created for searching for the k-shortest simple paths given a completed graph build upon Yen's algorithm; one of the first algorithms to search a directed, weighted graph for k loopless paths using a simple graph search algorithm, such as Dijkstra's algorithm, on multiple subgraphs.[11] Yen's algorithm searches the graph by creating two lists, A and B. List A contains the k-shortest paths while list B contains potential candidate paths to be added to list A. For example, Dijkstra's algorithm is used to search a completed graph for the optimal path, A_1, that contains three edges. Next, three

distinct subgraphs are created by removing one edge, selected from path A_1, from the completed graph. Then, each subgraph is searched using Dijkstra's algorithm to find paths B_1, B_2, and B_3; each of these paths are candidates for the next best path. The three candidate paths are then sorted and the lowest cost path becomes path A_2 and list B is emptied. After paths A_1 and A_2 are identified, additional candidate paths for $k > 3$ are also checked to determine if the path already exists in list A. This additional check ensures the top two paths are not repeated as the $k > 2$ shortest paths. This process of searching various subgraphs using Dijkstra's algorithm is then repeated until k paths, or all possible paths, are found.

In some cases, subsequent best paths that are composed of slightly different sequences of nodes and edges may closely resemble previous best paths. Searching for the k-shortest but sufficiently different paths may be useful in this scenario to support solution space exploration; in this paper, these paths are labeled as the k-shortest unique paths. One approach to calculating these unique paths uses a similarity measure that compares the number of shared edges to the number of unique edges for each new path to limit the number of shared edges between paths:[18] paths that share no edges have a similarity of 0, while paths that share every edge have a similarity of 1. To incorporate this uniqueness measure into Yen's algorithm, a third list C is created to contain all unique paths.[18] The first optimal path A_1 is also added to list C and labeled C_1. Next, as additional paths are found and saved within A, the similarity measure for each subsequent path in A is computed, e.g. $Sim(A_2, C_1)$. If the similarity between path A_2 and C_1 is below a specified threshold, path A_2 is saved as path C_2. Otherwise, path A_3 is the found and compared to C_1. This process is repeated until the last path added to list A is unique enough compared to the last path in list C. Once complete, this approach returns only the subset of paths generated via Yen's algorithm that are deemed sufficiently different according to a specified measure and threshold.

TECHNICAL APPROACH

In this paper, sampling-based kinodynamic planning is used to construct transfers between periodic orbits in the Earth-Moon CR3BP with minimal input from a trajectory designer. The technical approach for this paper is divided into three key phases:

1. Learning phase: a generalized roadmap is constructed using randomized nodes and edges.
2. Application phase: desired start and goal nodes, as well as any specific dynamical structures information, are added to the roadmap.
3. Query phase: the roadmap is searched to produce a variety of initial guesses for trajectories.

A conceptual overview of this approach is depicted in Figure 2 and each phase is described in more detail throughout the remainder of this section.

Phase 1: The Learning Phase

The learning phase constructs an efficient graph that supplies a discrete summary of a region of the continuous solution space in the CR3BP. In this roadmap, nodes are defined as nondimensional states in the rotating frame where a spacecraft could potentially be located at discrete time intervals. Edges represent two distinct types of motion in the CR3BP. Edges either 1) connect neighboring nodes with the same position vector but distinct velocities with one impulsive maneuver or 2) connect neighboring nodes with distinct states with a natural trajectory segment. Edges are also weighted to reflect the cost associated of traversing each edge, equal to the total norm of any impulsive maneuvers along an edge. This results in edges that either have an impulsive maneuver to

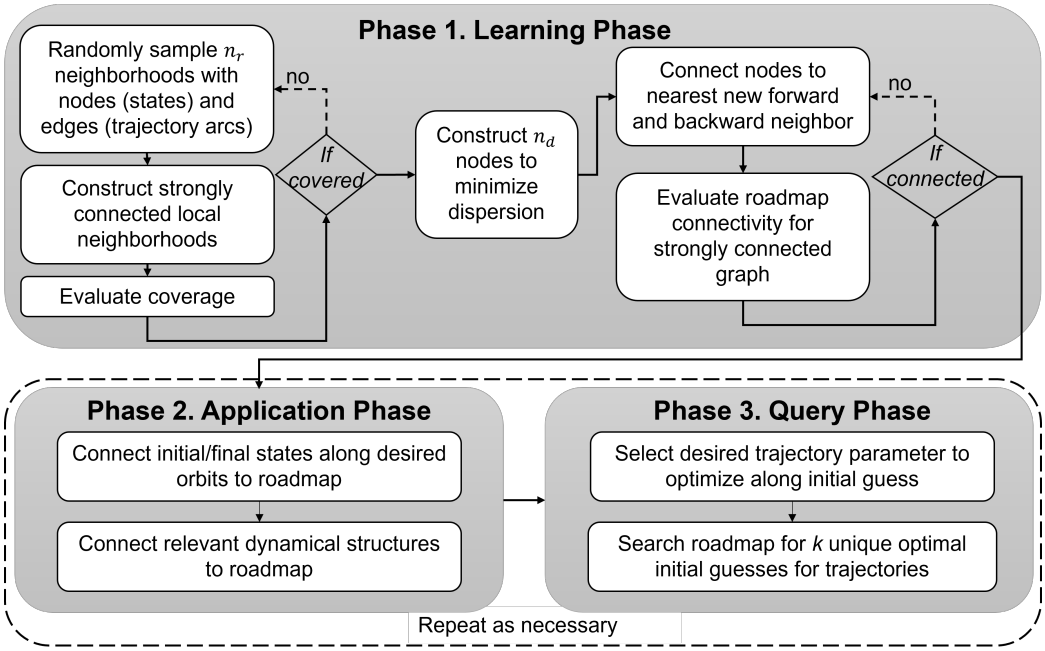

Figure 2. An overview of the key phases in roadmap generation used to construct initial guesses for transfers between two periodic orbits in the CR3BP.

change the velocity at a single location, resulting in a nonzero edge weight, or edges with a zero edge weight, that correspond to a natural arc. In this paper, these nodes and edges are added iteratively to limit the dependency on a human-in-the-loop: nodes are added until the solution space is sufficiently covered whereas edges are added until the graph is strongly connected. These terms and the process used to add these components to the roadmap are described within this section.

To construct the roadmap, the concept of a 1-neighborhood is employed. In this paper, a neighborhood is considered a strongly connected local graph that covers a specific region of the configuration space. A strongly connected graph guarantees that every node can reach every other node within the graph using the directed edges.[19] Accordingly, a spacecraft that possesses a state vector associated with a single node can reach any other node within its local neighborhood. Each neighborhood is constructed similar to a RRT: a centroid is selected and then subsequent nodes and edges are constructed forwards in time to explore the local solution space.

To define the centroid of a neighborhood, a valid position vector is randomly sampled within the desired region of the solution space. The configuration is only valid if the position vector does not intersect with any known obstacles in the configuration space. For this paper, valid configurations must lie within the allowable regions of motion at a given Jacobi constant, i.e., the zero velocity surfaces, and satisfy a 500km altitude constraint around the Moon.

At each centroid, k nodes are defined with the same position vector but k distinct velocity vectors. The value of k is selected to be proportional to the speed, corresponding to adding more nodes in sensitive regions. In this paper, $k = 10v$ is selected empirically but may adjusted as a tuning parameter of the roadmap generation process. Each of these k nodes is then defined to possess a state with a desired Jacobi constant and, therefore, the same speed, calculated using Eq. 2. Next, the velocity vector of a single node at the centroid of a neighborhood is defined using a randomly sampled unit vector. The velocity vectors of the remaining $k - 1$ nodes are then distributed nearly uniformly along a unit sphere (or circle for planar motion) to ensure the centroid is evenly connected

to the rest of its neighborhood. However, these velocity directions can only be evenly distributed within a unit sphere if the number of directions selected is also the number of vertices of a platonic solid, i.e., $4, 6, 8, 12, 20$. Thus, in order to distribute any number of velocity directions as uniformly as possible, the Fibonacci sphere is used.

The Fibonacci sphere supports generating a near-uniform sampling on a sphere by maximizing the smallest neighboring distances among the desired number of points.[20] This approach has been used in many applications such as numerical simulations and computer graphics due to its computationally efficiency and ease of implementation. The Fibonacci sphere is fundamentally related to the Fibonacci sequence and the golden ratio, $\Phi = (1 + \sqrt{5})/2$. To construct the Fibonacci sphere for a total of k^2 points, the 2D representation of the Fibonacci spiral is first computed. To construct the Fibonacci spiral, two angles, θ_i and ϕ_i, are computed for each integer number $i = \{1, k^2\}$ as

$$\theta_i = 2\pi \frac{i}{\Phi}, \phi_i = \cos^{-1}\left(1 - \frac{2(i + 0.5)}{N}\right) \tag{10}$$

These angles are mapped to a unit sphere using spherical coordinates to produce the i-th unit vector, expressed in the rotating frame as

$$\hat{v}_i = \{\cos(\theta_i)\sin(\phi_i), \sin(\theta_i)\sin(\phi_i), \cos(\phi_i)\} \tag{11}$$

This unit vector is multiplied by the speed calculated at the centroid to produce the velocity vector at a specified Jacobi constant to complete the state definition for each of the k nodes with the same centroid position vector. Finally, to ensure each center node is connected, a maneuver-enabled edge is constructed between each node with the same position vector to represent a spacecraft being able to instantaneously change velocity with an impulsive maneuver.

Once all k nodes at the center of a neighborhood are defined, they are propagated forward in time to create a natural edge in a unique direction. The propagation time t_f for each natural edge is selected to be proportional to the speed to ensure that nodes with a larger speed (and more likely to be more sensitive) explore a larger number of directions within solution space away from the center position vector; in this paper, $t_f = v$ is selected empirically but may be adjusted as a tunable parameter of the roadmap generation process. The final state along each of these natural edges is saved as an additional node, if it is valid. Each of these center nodes is connected to their additional nodes by a directed edge that possesses an edge weight of zero, corresponding to the natural flow of motion between these configurations. A neighborhood's radius is then equal to the minimum distance to all of the natural neighboring child nodes, i.e., the states associated with the nodes at the end of each of the k natural edges. The center k nodes with the same configuration, as well as their associated k neighboring child nodes connected by a natural edge, are labeled as the neighborhood's core nodes for the remainder of this paper.

The center k nodes as well as the child k nodes, i.e. all core nodes, are constructed first in each neighborhood. This process is conceptually demonstrated in Figure 3a), where the horizontal axis represents position and the vertical axis represents velocity. In this figure, nodes with the same position and distinct velocities are vertically stacked. The first centroid that is defined with a randomly sampled configuration and randomly selected velocity direction is depicted as a blue circle in the center of the figure. Then, 2 states with distinct velocity directions, depicted as red circles, are computed to create $k = 3$ nodes with the same position but distinct velocities. These 3 center nodes are then connected using a maneuver-enabled edge to change velocity, depicted as a red arrow. Finally, each of the center nodes are propagated to create a natural edge, depicted as a

blue arrow, where the final state along each edge is additionally saved as a new node, depicted as additional blue circles.

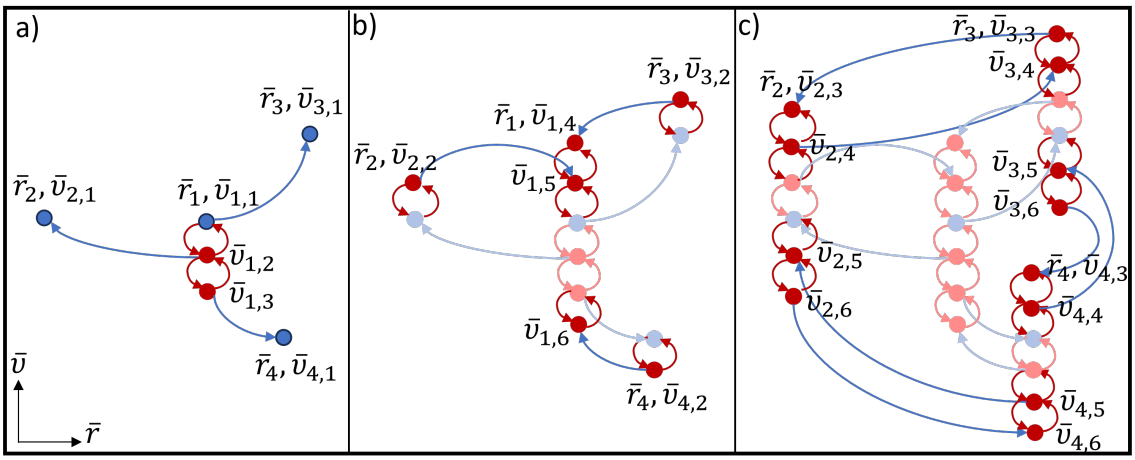

Figure 3. A conceptual diagram depicting the three steps in constructing a strongly connected neighborhood: a) core center and child nodes are constructed, b) core child nodes are connected back to their center core nodes, and c) core child nodes are connected to each other.

To ensure each neighborhood is a strongly connected local graph, all nodes within the neighborhood are connected through additional edges to every other node in the neighborhood. Each core child node is connected back to its associated center node using three edges. Before adding these edges, a single shooting corrections algorithm is used to compute a new arc that allows the velocities at the child node and the center node as well as the integration time to vary. Accordingly, impulsive maneuvers are placed at the beginning and end of the arc to match the velocity vector at each node. To construct an initial guess for the single-shooting scheme, the selected node is propagated naturally forward or backward in time. The elapsed time at the state along this arc that is closest in position space to the neighboring node seeds the integration time for the initial guess. If a maneuver-enabled edge is successfully computed via single-shooting, the impulsive maneuver magnitude is further reduced via local optimization implemented using sequential quadratic programming within the fmincon tool in MATLAB®.[21]

This trajectory segment is used to construct three new edges in the neighborhood: two single impulse edges to change the velocity at each node and one natural edge. The first maneuver-enabled edge is used to change the velocity from the child node to the initial state along the new trajectory segment. The second edge is a natural trajectory segment that is equal to the solution of the single shooting corrections algorithm, resulting in a free edge. The third edge is another maneuver-enabled edge that is used to change the velocity from the final state along the new trajectory segment to the velocity of the associated center node. This process is then repeated for each core child node in the neighborhood and conceptually demonstrated in Figure 3b) using the same color scheme as previously described, resulting in one additional node at each child node's position and k additional nodes at the center node's position. The new nodes and edges in this step are depicted in a darker blue and darker red color.

Last, to ensure there is a path between every pair of core nodes in a neighborhood, each core child node is connected to every other core child node in the neighborhood using the same process as described for connecting the child nodes to the center nodes. This final connection step is con-

ceptually depicted in Figure 3c) using the same color scheme as previously described, resulting in k additional nodes with the same position at each child node. Through this definition of a strongly connected neighborhood, a spacecraft located at any position vector associated with a node can reach any other node within the neighborhood by traversing the edges.

Neighborhoods are iteratively added to the roadmap until the desired region of the solution space is sufficiently covered. In this paper, coverage is defined by the number of neighborhoods that overlap with any other neighborhood in the graph. Accordingly, neighborhoods are iteratively constructed until all neighborhoods intersect with at least one other neighborhood to increase the likelihood of nodes associated with one neighborhood connecting to nodes within one other neighborhood; this number is selected empirically but may be adjusted as a tunable parameter of the roadmap generation process. By constructing neighborhoods and evaluating the coverage iteratively, the resulting graph is well-covered with minimal, redundant, information. Furthermore, this approach facilitates adaptive selection of the total number of nodes and edges within the graph.

Once all neighborhoods are added to the graph, additional nodes are added to the graph to minimize the dispersion. The dispersion of the graph is defined by the largest empty sphere in the configuration space, which indicates areas that are unreachable by a spacecraft.[22] Empty spheres, or under-sampled regions, in the configuration space are identified using a Delaunay triangulation constructed in the configuration space using the current unique position nodes in the graph. This method is commonly used in path planning problems to identifying areas of a solution space that are insufficiently approximated by a discrete set of nodes or edges.[23,24] A Delaunay triangulation is constructed by creating a set of tetrahedron that do not contain any nodes within their circumcircle. Once constructed, the centroid and volume of each tetrahedron is computed. The largest tetrahedra are then used to identify regions within the configuration space that indicate the largest empty regions within the graph. Starting with the largest tetrahedra, the centroid is a candidate for the configuration of a new node. If the configuration is valid, the states at the associated node is defined using the speed the produces the desired value of the Jacobi constant and the velocity direction is randomly sampled. Then, the average of the minimum distances to every node's nearest neighbor in configuration space is computed. Additional nodes are added to the graph until the average of all minimum distances stabilizes to within some tolerance, selected as 10^{-7} for this paper. Through this approach, the resulting graph better covers the configuration space without the need to select the number of additional nodes to add to the roadmap.

After all the neighborhoods are constructed and additional nodes are dispersed, additional edges are constructed until the graph is globally strongly connected using the same process as constructed new edges in neighborhoods. For every core node from all the neighborhoods in the graph, one neighbor forward in time and one neighbor backward in time is identified to construct three new edges leaving from and arriving to the current node, respectively. The selected core node is first naturally propagated for $t_f = \pm v$ using the same method used to create the natural edges within each neighborhood. Then, neighbors are identified using a distance measure that captures both the distance between two nodes in the configuration space as well as the angle between their velocity vectors; the goal is to locate nearby nodes with motion that flows in similar directions. The distance measured used to assess whether node j is a neighbor of node i is defined as

$$d_{i,j} = \|\boldsymbol{d}_i(t_f) - \boldsymbol{d}_j(t_0))\| + \theta \tag{12}$$

where $\boldsymbol{d}_i(t_f)$ is the position vector produced by propagating node i for time $t = \pm v$, $\boldsymbol{d}_j(t_0)$ is the position vector of the j-th node, and θ is the angle between their velocity vectors. Additionally,

because the maximum value of the angular difference between two velocity vectors is π, the distance vector to all neighboring nodes for node i, $D_i = [d_{1,2}, d_{1,3}, ..., d_{1,j}] \, \forall j$, is also normalized such that the minimum and maximum bounds are $[0, \pi]$.

To connect the core node with its identified neighbor, a single-shooting corrections algorithm is used to connect the positions of the core node with its neighboring node using three edges; the first edge will change the velocity at the current node, the second edge will change the state, and the third edge will change the velocity back to the neighboring node. This process is identical to connecting a child node within a neighborhood to another node within the same neighborhood described previously. Additionally, similar to the nodes, each edge is also subject to a 500km altitude constraint around the Moon. If any state lies below this altitude, the edge is not saved and a new neighbor is identified. Once again, if a maneuver-enabled edge is successfully computed via single-shooting, the impulsive maneuver magnitude is further reduced via local optimization implemented using sequential quadratic programming within the fmincon tool in MATLAB®.[21] Using the optimized solution, the maneuver magnitude serves as the edge weight for every edge that changes distinct velocities between identical positions.

After every node gains one new neighbor forward and backward in time, the connectivity of the graph is checked. If the graph is not strongly connected, each core node will again gain one neighbor forward and backward in time. This process is repeated until the graph is strongly connected, facilitating an adaptive number of edges and a strongly connected, yet efficient graph.

Phase 2: The Application Phase

Next, specific mission information relevant to the search query is projected onto the generalized roadmap. If the desired initial or final orbit possesses stable or unstable manifolds, states and small trajectory segments that lie on these manifolds may be added to the roadmap as nodes and natural edges, respectively, to improve the natural flow of the graph. These additional nodes and natural edges are then connected to the generalized roadmap's core nodes using the same distance measure to identify neighboring core nodes. First, a random state along a manifold trajectory is selected to add to the roadmap. If the node is valid, the state is propagated forward in time to create a natural edge. Then, if the final state along the trajectory segment is valid, it is also saved as a node to the graph and the edge is saved between the two nodes. Last, the initial node and final node along this new segment are connected to its k nearest backward and forward neighbors, respectively. Each of these nodes gains three edges per identified neighbor and uses the same equation $k = 10v$ to determine the number of centroid's in a neighborhood that determines how many neighbors to identify. This process is repeated until the desired amount of information from the available information about transport mechanisms is connected to the graph.

To complete the roadmap for a specific search query, states along the desired initial and final orbits are projected onto the roadmap as new nodes that serve as the set of source and goal nodes for the search queries, respectively. These new nodes are first connected to its nearest neighboring node along the same orbit using a natural edge, such that the initial and final orbit are free to transverse and the initial guess can connect to any location along the orbit. Next, each node is connected to neighboring core nodes already contained within the generalized roadmap and any nodes added from the associated fundamental solutions. Neighbors are identified using the same distance measure used in the generalized roadmap. Nodes added along the initial orbit are propagated forward in time to identify neighbors while nodes added along the final orbit are propagated backward in time to identify neighbors to ensure all initial guesses will depart along edges leaving the initial orbit and

finish along edges arriving onto the final orbit.

Phase 3: The Query Phase

After the generalized roadmap is constructed and any additional relevant mission information is connected to the roadmap, the query phase searches the graph for valid initial guesses for transfers between the desired periodic orbits. Dijkstra's search algorithm is used to find the lowest cost initial guess, a sequence of nodes and edges, that minimize the cumulative sum of edge weights across the path. The nodes along the initial orbit serve as the start nodes while nodes along the final orbit serve as the goal nodes for the search algorithm. Typically in PRM, if no path exists between the selected start and goal node(s), the search algorithm will terminate with no solution. However, since the graph is constructed to be strongly connected, at least one path is guaranteed to exist since at least one path exists between every pair of nodes in the graph.

Once the optimal path is found, additional unique paths between the same start and goal node sets are identified by applying Yen's algorithm. Specifically, Dijkstra's search algorithm is used in conjunction with the modified similarity measure to search multiple subgraphs of the roadmap. For this application, additional unique paths are identified by modifying a common similarity measure that is a ratio of the sum of the edge weights of the shared edges between two paths, P_i and P_j, and the path with the maximum cumulative edge weight, defined in this paper as:

$$Sim(P_i, P_j) = \frac{L(P_i \cap P_j)}{max\{L(P_i), L(P_j)\} + \sum_{n=1}^{d} \|\delta \bar{\boldsymbol{r}}\|} \tag{13}$$

where $L(\cdot)$ represents the cumulative weight of the total path and the summation term in the denominator represents the position difference between the d unique nodes in each path.[9] This summation term is added to encourage geometric differences between two paths. Multiple unique paths that are geometrically dissimilar may be identified by searching one completed graph repeatedly using this similarity measure in conjunction with Yen's algorithm.

All initial guesses identified are then corrected using collocation with the CEP mesh refinement process. The state of each node and the integration time along the subsequent edge in the initial guess form the segments used in the initial mesh, so the total time of flight may vary during corrections. If impulsive maneuvers are applied along the transfer between segments, only position constraints are included. The number of allowed maneuvers must be user-selected prior to corrections, although higher edge weights are used to indicate locations where a maneuver may be beneficial in recovering a nearby transfer.

RESULTS

Initial guesses for transfers between L_1 and L_2 periodic orbits in the Earth-Moon CR3BP are constructed using the sampling-based kinodynamic planning approach presented in the previous section. To demonstrate this approach, two roadmaps are constructed at a single energy level. First, a roadmap is generated at $C_J = 3.15$ for planar motion and used to recover two transfers between Lyapunov orbits. Next, a roadmap is generated at $C_J = 3.15$ for spatial motion. Maneuver-enabled transfers are generated between two halo orbits because heteroclinic connections that do not complete any revolutions around the Moon do not exist at this energy level.[25] The results for each of these examples are demonstrated in the remainder of this section.

Planar Transfers Between an L_1 and L_2 Lyapunov Orbit at $C_J = 3.15$

To demonstrate the process presented in the previous section, a generalized roadmap is constructed for planar motion at $C_J = 3.15$. The roadmap constructed for this example lies within the zero velocity curves at $C_J = 3.15$ and in the range of $x = [0.8, 1.2]$. All nodes and edges must fall within this region of the solution space to be considered valid, as well as adhere to a 500km Moon altitude constraint. Any constructed edge that exceeds either bound on the value of x is terminated at the boundary.

To begin constructing the generalized roadmap, neighborhoods are constructed. First, 10 neighborhoods are added to the graph at a time until the desired solution space is covered, meaning all neighborhoods intersect with one other neighborhood. The selected number of neighborhoods added at each iteration should balance minimizing the number of calls to the coverage check with minimizing the number of redundant neighborhoods constructed. If a smaller number of neighborhoods were added at a time, the coverage would need to be checked more frequently, while if a larger number of neighborhoods were added at a time, redundant neighborhoods may be constructed that are unnecessary for the graph. For this example, 50 neighborhoods were constructed before the specified region of the solution space is deemed to be covered. For each neighborhood, the number of velocity directions per center node position computed ranged from $k = 4 - 12$. Two neighborhoods of the first iteration are shown in Figure 4; blue circles depict the unique positions of all nodes of each neighborhoods, blue arcs represent natural trajectory edges between all nodes within a neighborhood, and the red circle depict the area of the configuration space covered by each neighborhood. Once all neighborhoods are constructed, the roadmap possesses 2424 nodes with 296 unique positions.

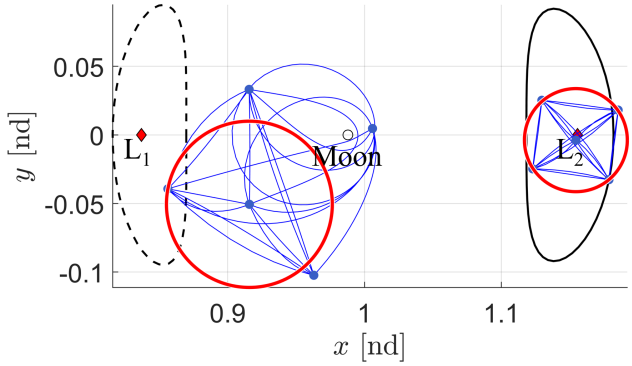

Figure 4. Two locally connected neighborhoods constructed at $C_J = 3.15$ that cover distinct regions of the configuration space indicated by the red circles.

Next, once all neighborhoods are constructed, additional nodes are added by computing a Delaunay triangulation in the configuration space using the 296 unique positions of the nodes currently within the graph. Nodes are added to the graph until the difference of the average distance of each node's closest neighbor between node addition is within 10^{-7}. This procedure results in 442 additional nodes being added to the graph. The distribution of nodes after the neighborhood construction procedure and after additional nodes were added via the Delaunay triangulation are depicted in Figure 5a) and 5b), respectively. Nodes constructed during the neighborhood construction process are depicted by blue circles while nodes added to minimize dispersion are depicted by magenta circles. There are still regions within the desired solution space that contain less nodes than other regions; addressing this limitation is an area of ongoing work.

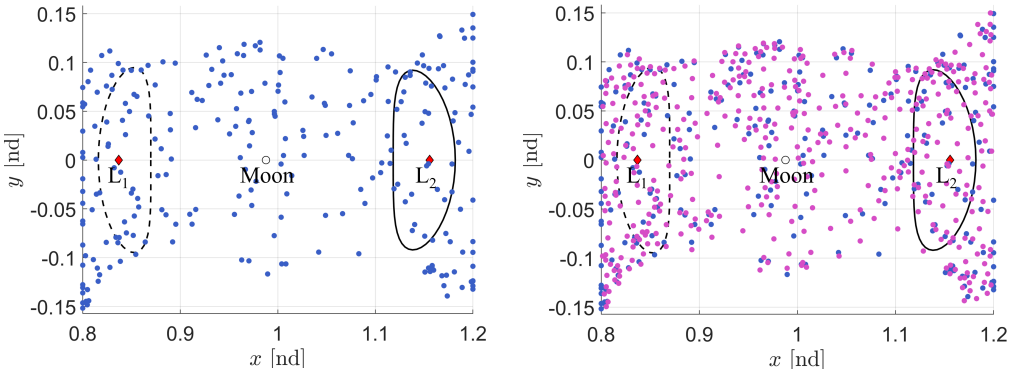

Figure 5. a) Distribution of nodes after all neighborhoods are added to the graph b) Distribution of nodes after additional nodes are added to minimize dispersion.

Last, to construct a strongly connected generalized roadmap, each core node gains one neighbor both forward and backward in time iteratively until the graph is strongly connected. For this example, each core node only requires one neighbor forward and backward in time until the connectivity criteria was met. The completed roadmap for this example appears in Figure 6. While this roadmap is constructed at $C_J = 3.15$, edges are not constrained to possess this energy level. Therefore, some edges may cross the zero velocity curves computed at $C_J = 3.15$.

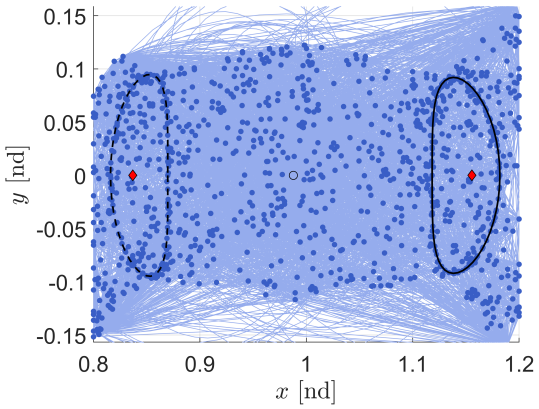

Figure 6. A completed roadmap generated to summarize the planar solution space in the lunar vicinity at $C_J = 3.15$.

After the generalized roadmap is constructed, fundamental transport mechanisms along with states along the initial and final orbit are added. 200 segments from the unstable and stable manifolds computed from the initial and final orbits, respectively, are added to the roadmap. Each segment consists of one node at the initial state of the trajectory arc, one node at the final state of the trajectory arc, and one natural edge between the two nodes. The node at the initial state is connected backward in time to its nearest $k = 10v$ neighbors while the node at the final state is connected forward in time to its nearest $k = 10v$ neighbors, to aid motion onto and off this trajectory segment. Then, 50 states along each the initial and final orbit, evenly discretized in time, are connected to their $k = 10v$ nearest neighbors forward and backward in time to guide the search algorithm to depart the initial orbit and arrive to the final orbit.

Once the roadmap is completed, Yen's algorithm that repeatedly calls Dijkstra's search algorithm

is used to search the graph for initial guesses that can depart the L_1 Lyapunov orbit and arrive onto the L_2 Lyapunov orbit. The tolerance for the similarity measure is set to $Sim(P_i, P_j) < 0.6$ to distinguish unique paths. These initial guesses correspond to paths that are a sequence of nodes and edges that represent a sequence of instantaneous impulsive maneuvers and natural trajectory arcs. The lowest cost initial guess (depicted in blue) and the second lowest cost unique initial guess (depicted in green), constructed from the completed roadmap in Figure 6, are displayed in Figure 7a). The initial and final Lyapunov orbits are plotted with dashed and solid black lines, respectively.

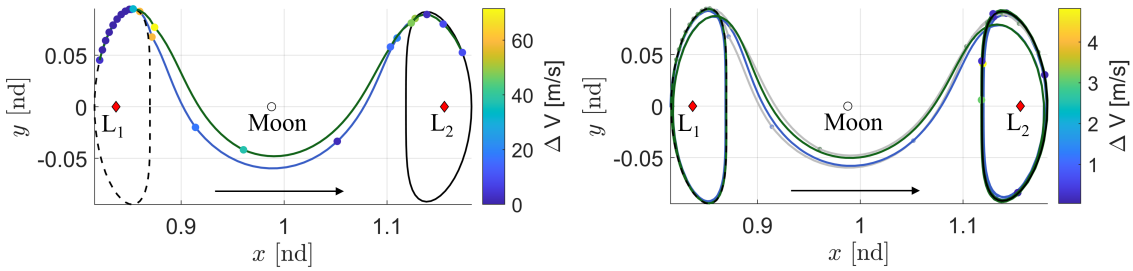

Figure 7. a) The optimal (blue) and second best (green) initial guesses constructed from the roadmap in Figure 6 and b) the corresponding two continuous transfers.

To recover a continuous trajectory, each initial guess is modified to use four impulsive maneuvers: one maneuver to depart from the L_1 orbit, one maneuver to arrive onto the L_2 orbit, and the last two maneuvers are selected at nodes along the initial guess with the two largest maneuver magnitudes. The corrected continuous transfers for the lowest cost and second lowest cost initial guesses are displayed in Figure 7b) in blue and green, respectively, while both initial guesses are depicted in grey. The optimal initial guess has a cumulative edge weight of $223.4m/s$ and the corrected solution has a total maneuver magnitude of $5.1m/s$. The second best initial guess has a cumulative edge weight of $225.0m/s$ and the corrected solution has a total maneuver magnitude of $3.7m/s$. While the second best initial guess has a lower initial guess maneuver magnitude, it corrects to a solution that is very close to the natural heteroclinic connection that exists at this energy level.[26] While the first two unique solutions identified by Yen's algorithm are close in geometry, Yen's algorithm identified the initial guesses as unique due to the distinct configurations of a few of the nodes and natural edges contained in the initial guesses. Improving the search algorithms for computational efficiency to identify additional geometrically distinct transfers, such as transfers with revolutions around the Moon, is an avenue of ongoing work.

Spatial Transfers Between an L_1 and L_2 Halo Orbit at $C_J = 3.15$

To demonstrate the roadmap construction process for spatial motion, a generalized roadmap is constructed at $C_J = 3.15$ to generate transfers between two northern halo orbits around L_1 and L_2. The region of interest and Moon altitude constraint for this roadmap is the same as the previous example. First, 10 neighborhoods are added to the graph at a time until the desired solution space is covered. For this spatial example, 130 neighborhoods are required until the graph is deemed covered. For each neighborhood, the number of centroids computed lie in the range of $k = [4 - 6]$. While the number of neighborhoods required to construct this spatial roadmap is much larger compared to the planar case, each neighborhood contains a smaller amount of core nodes. After all neighborhoods are constructed, the roadmap contains 5240 nodes with 653 unique position vectors.

Next, additional nodes are added by constructing a Delaunay triangulation in the configuration

space using the 653 unique position vectors of the nodes currently within the graph. Additional nodes are added until the difference of the average distance of each node's closest neighbor is within 10^{-7}. Using this threshold, 252 additional unique nodes are added to the graph. Last, each core node gains one neighbor both forward and backward in time until the graph is strongly connected. Each core node requires two neighbors forward and backward in time for this graph to be strongly connected. The completed roadmap is shown in Figure 8 using the same color scheme as Figure 6.

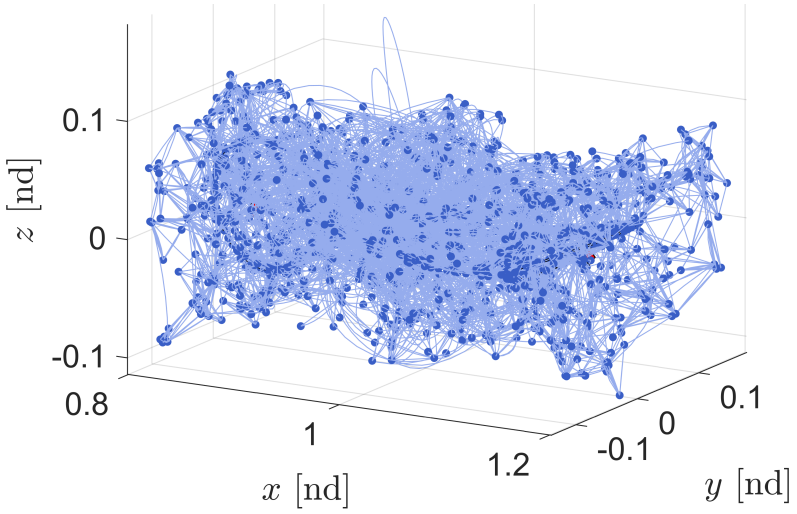

Figure 8. A completed roadmap generated to summarize the spatial solution space in the lunar vicinity at $C_J = 3.15$.

After the generalized roadmap is constructed, application specific information required for the search query is added in the same manner as the planar roadmap. First, 200 segments from the unstable and stable manifolds computed from the initial and final orbits are projected to the roadmap as new nodes and natural edges. Next, 50 states along each orbit are added as new nodes. Last, each of these new nodes are connected to the generalized roadmap either forward or backward in time depending on the original fundamental solution.

After the roadmap is improved with the application specific information, Yen's algorithm and Dijkstra's search algorithm are used to generate transfers that depart any location along the initial L_1 halo orbit and arrive onto any location along the L_2 halo orbit. The optimal initial guess (depicted in blue) and the second lowest cost initial guess (depicted in green) are displayed in Figure 9a) and the corrected continuous transfers are displayed in Figure 9b). The initial and final halo orbits are plotted with dashed and solid black lines, respectively. Each initial guess in then input into a collocation corrections algorithm to contains four maneuvers: one to depart the initial L_1 orbit, one to arrive onto the L_2 orbit, and the last two maneuvers are selected at the nodes along the initial guess with the two largest maneuver magnitudes. The initial guess for corrections is also appended to contain 3 revolutions along each the initial and final orbit. The optimal initial guess has a cumulative edge weight of $494.6m/s$ and the corrected solution has a total maneuver magnitude of $304.9m/s$. The second best initial guess has a cumulative edge weight of $545.2m/s$ and the corrected solution has a total maneuver magnitude of $538.3m/s$. While using the automatic maneuver placement of only two maneuvers along each initial guess, each corrected solution closely resembles the geometry of the initial guess. While the corrected transfers depart and arrive to different locations along each orbit than the initial guesses, additional constraints may be added to the corrections algorithm to constrain the corrected solution to depart and arrive specific locations along each orbit.

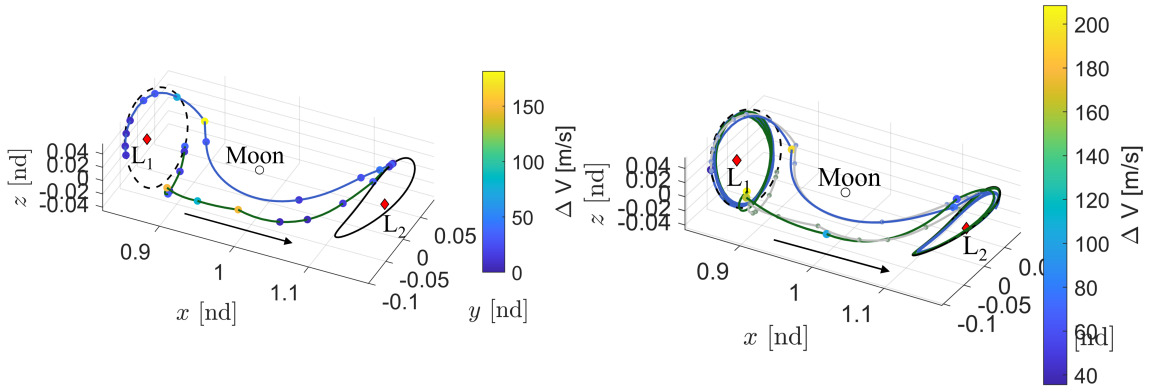

Figure 9. a) The optimal (blue) and second best (green) initial guesses constructed from the roadmap in Figure 8 and b) the corresponding two geometrically distinct continuous transfers.

CONCLUSION

In this paper, sampling-based kinodynamic planning is used construct a graph, called the roadmap, in the lunar vicinity in the Earth-Moon CR3BP in three key phases: the learning phase, the application phase, and the query phase. The roadmap is a directed, weighted graph that contains nodes, which represent valid spacecraft states, and edges, which represent valid trajectory motion or impulsive maneuvers. First, strong locally connected graphs called neighborhoods are constructed and iteratively added until the desired solution space is sufficiently covered. Then, maneuver-enabled edges are added by connecting neighboring nodes in distinct neighborhoods. These edges represent natural trajectory arcs between distinct positions associated with two nodes and single impulsive maneuvers between nodes with the same position but distinct velocities. The magnitude of the impulsive maneuver is saved as the associated edge weight, resulting in a zero edge weight for edges that represent natural trajectory arcs. The core nodes of all neighborhoods gain one neighbor forward and backward in time in distinct neighborhoods until the graph is strongly connected. Lastly, states along the initial and final orbit and segments along the associated invariant manifolds are connected to the graph.

Once the roadmap is completed, Yen's algorithm using Dijkstra's search algorithm is used to search the graph for the two lowest cost initial guesses that minimizes the cumulative sum of the total edge weights across the paths. All solution paths found represent initial guesses for transfers between the desired periodic orbits that are then input to a collocation corrections method with a CEP mesh refinement process to recover continuous solutions.

This approach was employed to recover geometrically distinct maneuver-enabled solutions between Lyapunov and halo orbits at $C_J = 3.15$ in the Earth-Moon CR3BP. The location of the maneuvers for each transfer were selected based on the highest edge weight along the initial guess. The results for this application motivate further exploration of sampling-based kinodynamic planning in the CR3BP to more efficiently summarize the solution space and decrease the computational time required to identify additional feasible solutions.

ACKNOWLEDGMENT

This research is being completed at the University of Colorado Boulder. It is supported by a NASA Space Technology Graduate Research Opportunity.

REFERENCES

[1] S. M. LaValle, *Planning Algorithms*. Cambridge, UK: Cambridge University Press, 1st ed., 2006.

[2] P. Svestka and M. H. Overmars, *Robot Motion Planning and Control*. Berlin: Springer, Berlin, Heidelberg, 1st ed., 1998.

[3] K. Lynch, G. Kantor, H. Choset, L. Kavraki, S. A. Hutchinson, W. Burgard, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, 1st ed., 2005.

[4] A. Das-Stuart, K. Howell, and D. Folta, "A Rapid Trajectory Design Strategy for Complex Environments Leveraging Attainable Regions and Low-Thrust Capabilities," Adelaide, Australia, 68th International Astronautical Congress, September 2017.

[5] E. Trumbauer and B. Villac, "Heuristic Search-Based Framework for Onboard Trajectory Redesign," *Journal of Guidance, Control and Dynamics*, Vol. 37, No. 1, January-February 2014, 10.2514/1.61236.

[6] J. A. Starek, E. Schmerling, G. D. Maher, B. W. Barbee, and M. Pavone, "Real-Time, Propellant-Optimized Spacecraft Motion Planning under Clohessy-Whilshire-Hill Dynamics," Big Sky, MT, IEEE Aerospace Conference, March 2016.

[7] K. Bruchko and N. Bosanac, "A Preliminary Exploration of Path Planning for Initial Guess Construction in Multi-Body Systems," AAS/AIAA Astrodynamics Specialist Conference, August 2021.

[8] K. Bruchko and N. Bosanac, "Designing Spatial Transfers in Multi-Body Systems Using Roadmap Generation," Charlotte, NC, AAS/AIAA Astrodynamics Specialist Conference, August 2022.

[9] K. Bruchko and N. Bosanac, "Adaptive Roadmap Generation for Trajectory Design in the Earth-Moon System," Austin, TX, AAS/AIAA Space Flight Mechanics Meeting, January 2023.

[10] R. Shome and L. E. Kavraki, "Asymptotically Optimal Kinodynamic Planning Using Bundles of Edges," 2021, pp. 9988–9994, 10.1109/ICRA48506.2021.9560836.

[11] J. Y. Yen., "Finding the K shortest loopless paths in a network," *Management Science*, Vol. 17, July 1971.

[12] W. S. Koon, M. W. Lo, J. E. Marsden, and S. D. Ross, *Dynamical Systems, The Three-Body Problem and Space Mission Design*. Marsden Books, 3rd ed., 2006.

[13] V. Szebehely, *Theory of Orbits: The Restricted Problem of Three Bodies*. New York, NY: Academic Press, 1967.

[14] R. Pritchett, *Strategies for Low-Thrust Transfer Design Based on Direct Collocation Techniques*. PhD thesis, Purdue University, IN, 2020.

[15] D. J. Grebow and T. A. Pavlak, "MCOLL: MONTE Collocation Trajectory Design Tool," Stevenson, WA, AAS/AIAA Astrodynamics Specialist Conference, August 2017.

[16] J. C. Bruce Donald, Patrick Xavier and J. Reif, "Kinodynamic motion planning.," *Journal of the ACM (JACM)*, Vol. 40, No. 5, 1993, pp. 1048–1066.

[17] J.-H. Park and T.-W. Yoon, "Maximizing the Coverage of Roadmap Graph for Optimal Motion Planning," *Hindawi*, Vol. 2018, November 2018, 10.1155/2018/9104720.

[18] B. Y. Huiping Liu, Cheqing Jin and A. Zhou, "Finding Top-k Shortest Paths with Diversity," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 30, No. 3, March 2018, 10.1109/TKDE.2017.2773492.

[19] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2 ed., 2001.

[20] B. Keinert, M. Innmann, M. Sänger, and M. Stamminger, "Spherical Fibonacci Mapping," *ACM Trans. Graph.*, Vol. 34, No. 6, 2015.

[21] Matlab, *Optimization Toolbox Version 9.3 (R2022a)*. Natick, Massachusetts: The MathWorks Inc.

[22] L. E. Kavraki and S. M. LaValle, *Motion Planning*. Springer Handbook of Robotics. Springer, Berlin, Heidelberg, 2008, 10.1007/978-3-540-30301-5_6.

[23] Y. Huang and K. Gupta, "A Delaunay Triangulation Based Node Connection Strategy for Probabilistic Roadmap Planners," New Orleans, LA, IEEE International Conference on Robotics and Automation, April 2004.

[24] W. C. T. Gene Eu Jan, Chi-Chia Sun and T.-H. Lin, "An O(nlogn) Shortest Path Algorithm Based on Delaunay Triangulation," *IEEE/ASME Transactions on Mechatronics*, Vol. 19, No. 2, April 2014, 10.1109/TMECH.2013.2252076.

[25] A. Haapala, *Trajectory Design in the Spatial Circular Restricted Three-Body Problem Exploiting Higher-Dimensional Poincaré Maps*. PhD thesis, Purdue University, IN, 2014.

[26] E. Canalias and J. J. Masdemont, "Homoclinic and Heteroclinic Transfer Trajectories between Lyapunov Orbits in the Sun-Earth and Earth-Moon Systems," 10 2007.