NETWORK METHODS FOR MULTIPLE GRAVITY-ASSIST MISSION DESIGN

by

James W. Moore

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2023

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. James Longuski, Chair

School of Aeronautics and Astronautics

Dr. Kathleen Howell

School of Aeronautics and Astronautics

Dr. William Crossley

School of Aeronautics and Astronautics

Dr. Carolin Frueh

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory Blaisdell

To my wife, Dori, and to our children

What a beautiful face
I have found in this place
that is circling all 'round the Sun.
What a beautiful dream
that could flash on the screen
in a blink of an eye and be gone from me.
Soft and sweet.
Let me hold it close and keep it here with me.

Neutral Milk Hotel, "In the Aeroplane Over the Sea"

ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. James Longuski for the thoughtful guidance he has offered and the limitless patience he has exhibited during my studies. I would also like to thank the members of my advisory committee, Profs. William Crossley, Carolin Frueh, and Kathleen Howell for their excellent instruction and challenging questions. The courses I took from my committee members are among the most enjoyable and useful in my college career and I often refer back to those class materials.

The members of Prof. Longuski's research group have provided helpful review and encouragement during my time in the group. I especially wish to thank Dr. Kyle Hughes and Dr. Alec Mudek for the many discussions that inspired much of this work. I am grateful for the interest and feedback provided over the years by Dr. Rachana Agrawal, Dr. Weston Buchanan, Tyler Hook, Rob Potter, Jeffrey Pekosh, Dr. Sarag Saikia, Dr. Nathan Strange, Dr. Paul Witsberger, and to the many other researchers whose work has inspired my own.

I also offer my gratitude to the friends and colleagues who have inspired me with their engineering excellence throughout my professional career. Many of the skills I employed in this research were learned from coworkers and honed on rewarding work projects.

Finally, I thank my family for their love and support. I would not have taken my first steps on this journey without the endless support of my parents Jim and Colleen and the encouragement of my siblings Tom and Annie. My deepest and most heartfelt gratitude goes to my wife Dori and to our children Mayzie, Zelly, Finn, and Cael. My graduate work has often come at the cost of time spent together or full participation in your pursuits. During my studies I have depended upon your understanding, encouragement, and patience more than I can possibly express. For better or worse, you now have my full attention.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

a	semi-major axis
α	pump angle
δ	flyby bending angle or turn angle
$\delta_{\max}$	maximum flyby bending angle
C_3	characteristic energy
D	date (of alignment, departure, or arrival)
e	eccentricity
E	eccentric anomaly, also the number of edges in a network
$\mathscr{E}$	specific mechanical energy
G	a data structure representing a graph or network
h	angular momentum
H	hyperbolic anomaly
i, j, k	iteration indices
L_m	spacecraft revolution on which leveraging maneuver occurs in a VILT
m	the total number of some countable object
M	spacecraft revolutions in a resonant VILT
$\mathscr{M}$	mean anomaly
μ	gravitational parameter
n	mean motion, also used to indicate the total number of some countable object
$n : m$	ratio of planet revolutions to spacecraft revolutions in resonance
N	planet revolutions in a resonant VILT
ν	true anomaly
p	semi-latus rectum
P	orbit period
Φ	phasing angle
r	radius distance
t	time
$t_{\max}$	maximum time of flight

θ	transfer angle (0 to 2π)
Θ	total angle swept out by orbiting body
v	velocity
V	the number of vertices in a network
$\mathbf{V}_{\mathrm{p}}$	heliocentric planet velocity vector
$\mathbf{V}_{\mathrm{sc}}$	heliocentric spacecraft velocity vector
$\mathbf{V}_{\infty}$	hyperbolic excess velocity vector
V_{∞}	hyperbolic excess velocity magnitude

Subscripts and Superscripts

x^{-}	quantity x before a gravity assist or maneuver
x^{+}	quantity x after a gravity assist or maneuver
$x^{'}$	rotated quantity x
x_0	initial value of quantity x
x_{a}	quantity x at arrival body
x_{align}	quantity x at alignment time
x_{arrive}	quantity x at arrival time
x_{d}	quantity x at departure body
x_{depart}	quantity x at departure time
x_{ga}	quantity x supplied by gravity assist
x_i	i^{th} value of quantity x
x_{max}	maximum of quantity x
x_{p}	quantity x for the planet or central body
x_{prop}	quantity x supplied by propulsion
x_{sc}	quantity x for the spacecraft

ABBREVIATIONS

ACO	Ant Colony Optimization
APS	All-Paths Search
BFS	Breadth-First Search
CR3BP	Circular Restricted Three Body Problem
DFS	Depth-First Search
DSM	Deep Space Maneuver
EMTG	Evolutionary Mission Trajectory Generator
GREMLINS	GRidded Ephemeris Map of Lambert INterplanetary Solutions
JPL	NASA Jet Propulsion Laboratory
KBO	Kuiper Belt Object
MGA	Multiple Gravity Assist
NASA	National Aeronautics and Space Administration
SLSQP	Sequential Least SQuares Programming
SOI	Sphere of Influence
STOUR	Satellite Tour design program
SPICE	Spacecraft Planet Instrument Camera-matrix Events
TOF	Time of Flight
TNO	Trans-Neptunian Object
VEEGA	Venus Earth Earth Gravity Assist
VILT	V-Infinity Leveraging Transfer

NOMENCLATURE

adjacent	Two network vertices that are connected by an edge are said to be adjacent.
connected	An undirected graph or network in which a path exists between every pair of vertices is said to be connected.
dense	Describes a network with a high edge to vertex ratio.
directed	An edge that operates in only one direction. Also describes a network with directed edges.
edge	A member of a network representing a link between vertices.
graph	The locus of ordered pairs (x, y) describing a relationship (a plot).
network	A collection of vertices connected by edges (as in graph theory).
node	The intersection of V_∞ contours on the Tisserand graph.
parallel	used to describe two or more edges connecting the same vertices.
path	The sequence of planets in a multi-gravity-assist trajectory. A path is identified by the initials of its flyby bodies where Earth departure is implied. For example, JSUN signifies Earth-Jupiter-Saturn-Uranus-Neptune.
route	A sequence of vertices in the Tisserand network identified by planet, V_∞, and inbound/outbound location of the encounter. For example, E10-I, J7-I, S8-O, U12-O, N14-O.
sparse	Describes a network with a low edge to vertex ratio.
tree	A connected network containing no cycles.
vertex	A member of a network that serves as an abstract representation of an object with various properties.
walk	A series of alternating adjacent vertices and edges in a network.

ABSTRACT

An innovative network model of the gravity assist problem enables the quick discovery and characterization of candidate trajectories from a small set of search criteria. The network elements encapsulate information about individual gravity assist encounters and connectivity. This organization of the astrodynamical information makes it possible to deploy well-established search methods to find sequences of flyby encounters with reduced human effort and in a fraction of the time previously required. The connectivity encoded in the model considers energy feasibility and scheduling constraints. Therefore, paths found using the network algorithms are feasible from both an energy and phasing perspective.

Current initial-guess methods only identify a sequence of planet names that may form a tour. Broad searches over launch date and launch V_∞ (sometimes requiring months of computation time) are currently required to identify realistic paths from each possible sequence. The network approach provides (in a shorter period of time) more detailed initial guesses that include the approximate V_∞ and date of each encounter. These initial guesses can directly generate a set of patched-conic trajectories or initialize existing grid-search tools. The technique can accept fidelity improvements and may be extended for use on other mission types.

A collection of potential gravity assist encounters serve as the network vertices. Keplerian models for connecting the gravity assists in energy and time translate into network edges. Network models of more sophisticated trajectory concepts such as resonant transfers and V_∞-leveraging extend the approach to include more complex paths.

General network traversal algorithms form the basis for gravity-assist trajectory searches. Problem-specific network filtering reduces network size and search times. A detailed discussion of algorithm complexity and problem size is also provided.

The new search technique successfully rediscovers known trajectories from historical gravity assist missions. The network method also identifies preliminary gravity-assist trajectories to the Trans-Neptunian Objects Haumea and Makemake.

1. INTRODUCTION

In February 1974, Mariner 10 made a close pass of Venus to bend the spacecraft trajectory in the direction of Mercury. This was the first time that a gravity assist from one planet was used to reach a second planet. In the 50 years since this event, gravity assists have enabled increasingly complex missions, including visits to every planet and tours of satellite systems.

To develop a multiple gravity-assist (MGA) mission, a trajectory designer must be able to identify promising flyby sequences from myriad combinations of planetary encounters that exist for brief periods of time. Once identified, candidate paths must be further evaluated with higher-fidelity analysis. A candidate path is typically a "first guess" for an iterative process. The present research identifies better first guesses with higher fidelity and in significantly shorter time than was previously possible. Moreover, the techniques developed will be applicable to more complex problems for which current manual methods are impractical.

The present work is an innovative application of network analysis techniques that simultaneously addresses the energy and time components of the gravity-assist problem. We will construct a network of possible flybys from energy matching methods analogous to the Tisserand graph. We next augment the Tisserand information with possible encounter times derived from ephemeris data. The result is a network of flybys that are connected in time. Once the network is constructed, gravity-assist paths can be found with established network algorithms.

The gravity-assist paths found in the discrete energy and time network can immediately be used to construct continuous solutions in the patched-conic framework. Alternatively, the network results can focus grid searches performed in other design tools.

1.1 Historical Context

The effect of planetary encounters on the orbits of comets and asteroids was understood by many of the astronomers and mathematicians of the 18^{th} and 19^{th} centuries, including d'Alembert, Laplace, Leverrier, and Tisserand [1]. In the first half of the 20^{th} century, the utility of this gravitational perturbation for modifying a spacecraft trajectory became in-

creasingly clear. Important contributions in this era include work by Hohmann, Kondratyuk, Tsander, and von Pirquet [1], [2].

At the dawn of the Space Age, work by Lawden, Grocco, and Ehricke [3] set the stage for the practical application of gravity assists [2]. The first intentional use of a gravity assist was the 1959 flyby of the Moon by the Soviet spacecraft Luna 3 enabling transmission of photographs of the far side of the Moon. Mariner 10 was the first mission to perform a gravity assist in order to reach another planet.

Minovitch [4] and Flandro [5]–[7] did pioneering work on the use of gravity assists to enable fast transfers to the outer planets. Burgeoning ability to engineer gravity assists and meet the associated navigation challenges enabled landmark exploration missions in the 1970s and 80s. Pioneer 11 and the Voyager missions included flybys of multiple planets. The Voyager 2 Grand Tour is perhaps the archetypal multi-gravity-assist (MGA) mission.

The last several decades have seen missions with increasingly complex MGA trajectories. The Galileo mission was redesigned to include a flyby of Venus and two flybys of Earth on the way to Jupiter after the original direct transfer became impossible following the Challenger accident. Cassini included two flybys of Venus and flybys of Earth and Jupiter on its trip to Saturn. Both Galileo and Cassini followed up their interplanetary trajectories with satellite tours. In the 21st century, the Rosetta, MESSENGER, Deep Impact, and Dawn missions have included flybys of inner solar system planets to reach Mercury, comets, and asteroids.

1.2 Problem Definition

A gravity assist is the intentional use of a secondary celestial body's gravitational field to alter the orbit of a spacecraft about a primary body. A spacecraft performs a gravity assist by passing close to a massive body without entering a closed orbit about that body. For this reason, the event is also called a *flyby* or a *swingby*.

During the gravity assist, the energy of the spacecraft relative to the flyby body is unchanged. However, the spacecraft energy relative to the primary can be altered significantly. This energy change comes with little to no propellant cost. The propellant saving benefit

has made the gravity assist an important and enabling technique for exploration of the solar system.

The focus of this research is the fundamental question of multi-gravity-assist trajectory design: what gravity assist trajectories are possible? The answer to the question requires understanding whether a candidate gravity assist is capable of advancing the spacecraft to the next planet and whether the planets are positioned to facilitate the transfer.

1.2.1 Pathfinding and Pathsolving

A popular paradigm for conceptualizing MGA trajectory design breaks the problem into two parts: pathfinding and pathsolving. This distinction was coined by Longuski. Pathfinding is the process of discovering a candidate sequence of gravity-assist bodies using orbital energy. Pathsolving is the problem of finding continuous trajectories along a path using an ephemeris model.

This division addresses the energy and time components of the problem separately. The pathfinding problem answers the question of whether a gravity assist at one planet can change the energy of the spacecraft orbit enough to reach another planet. The pathsolving problem answers the question of whether a series of gravity assists can actually be scheduled given the constantly changing alignment of the planets. The ephemerides in the pathsolving problem provide the time histories of planet motion. Each gravity assist must occur at a time when the planet phasing is correct.

Pathfinding involves discrete variables (e.g., planets, the number of encounters, and discrete V_∞ levels). Pathsolving involves continuous variables (e.g., time, position, and velocity). The problems are typically solved separately, either in two sequential steps or in an alternating, iterative approach.

The present work does not fit perfectly into the pathfinding/pathsolving paradigm. The goal of the preliminary exploration was to develop an automated and "exact" (or exhaustive) solution to the discrete pathfinding problem to generate initial guesses for an existing grid-based pathsolver (namely, STOUR [8]). However, the work eventually incorporated the scheduling question into the discrete path problem. So the initial guesses provided by the

technique presented here include not just what paths are feasible, but also, when they are possible. Finally, to assist in cataloging and selecting attractive initial guesses, the present work transforms the initial-guess paths into continuous patched-conic trajectories. Thus, this work addresses both the pathfinding and pathsolving parts of the trajectory problem. But the division between the energy and time components of the problem is not as clear as the sequential or iterative approaches used by other researchers.

1.3 Gravity-Assist Pathfinding Approaches

A preliminary question for designing a multiple gravity-assist mission is, what sequence of planets shall we visit? Perhaps more fundamentally, what sequences are possible? A simple enumeration of the possible sequences is generally impractical. The number of permutations of b potential flyby bodies in a path of length l is b^l. For example, suppose we wish to create a trajectory to Neptune using the planets Venus, Earth, Mars, Jupiter, Saturn, and Uranus. If we allow up to five gravity assists, we would need to evaluate 7,776 paths. If we allow seven flybys, we would need to evaluate nearly 280,000 paths. Beyond the sheer number of cases, many of these sequences would be impractical. For example, due to the anticipated flight time, we can reasonably rule out any path that visits one of the inner planets after a flyby of one of the outer planets.

Since satellite tours require many flybys, a similar brute force approach would generate an astronomical number of possibilities. A 25-flyby tour that only considers four moons as gravity-assist bodies could follow over 1×10^{15} possible paths.

These solar system and satellite tour examples show that, for a variety of reasons, we must be selective about the potential paths that receive a detailed evaluation. However, the majority of MGA search methods presented in the literature focus on the continuous, *pathsolving* problem. While providing sophisticated capabilities for evaluating potential trajectories, these methods leave the solution of the *pathfinding* problem to experience, intuition, stochastic methods, or brute force.

The Tisserand graph provides mission designers with a graphical technique for methodically selecting feasible paths. Strange and Longuski [9] provide an introduction to the

method and describe an early automation. Additional history and advancement is provided in Appendix A. The method is a great improvement over brute force, but its use must be guided by experience and intuition and graphical pathfinding can become intractable for long paths. An automated means of searching the Tisserand graph is clearly desireable.

1.3.1 Related Research in Automated Pathfinding

De la Torre Sangre *et al.* have concurrently developed a tree search of the Tisserand graph called Tisserand PathFinder [10]. The authors describe an automated Depth First Search (DFS) of the Tisserand graph capable of discovering some Tisserand-graph paths found manually by earlier researchers (including Strange and Longuski [9] and Hughes *et al.* [11]). The tree search finds energy-feasible paths but does not attempt to compute transfer times or address the scheduling problem. The method does not generate patched-conic trajectories along the paths. The limitation on flyby bending angle (Section 2.1.2) is observed but it is not addressed with resonance modeling (Chapter 4).

In another concurrent study, Bellome *et al.* [12] address the phasing problem in the context of a Tisserand graph search and produce trajectories to Jupiter and Saturn similar to Galileo and Cassini. The authors use a different approach to process the Tisserand graph information than the methods presented here. The technique appears to only consider a single (V_∞, pump angle) starting point on the Tisserand graph and does not generate an exhaustive list of feasible trajectories.

1.4 Gravity-Assist Pathsolving Approaches

Many researchers have developed or employed search methods for multiple gravity-assist trajectories. Most research has focused on the pathsolving problem. In the simplest analysis, the transfer between planetary encounters is purely ballistic. However, most existing tools also include more sophisticated transfer options such as resonant re-encounters, V-infinity leveraging transfers (VILTs), low-thrust arcs, and a limited number of Deep Space Maneuvers (DSMs). Some tools may also model escape from or insertion into orbit at the gravity-assist

bodies. These pathsolvers may include global optimization methods or may simply provide a global search capability which identifies all trajectories meeting the search conditions.

Among these pathsolvers, the method for deciding where and when to look for solutions varies by researcher and tool. Categorical parameters like the sequence of planets to be encountered (Venus-Earth-Earth-Jupiter, etc.) are either explicitly provided by a human designer, created by permutation, or randomly generated. In some cases, rules-of-thumb may guide the automatic generation of the search space. But, in general, these search methods use some variation on a "guess-and-check" technique that will explore regions of the search space where no solutions exist, unless directed by a knowledgeable designer.

1.4.1 Exhaustive Methods

The most popular approach for finding MGA trajectories is to perform an exhaustive evaluation of Lambert problems over a collection of potential planetary encounters (i.e., planet and date combinations). In some cases, the desired path is completely defined beforehand. The automated Satellite Tour design program (STOUR) [8], and the Explore pathsolving tool [13] follow this approach. Hughes [14] and Mudek *et al.* [15] conduct searches for trajectories to the Ice Giants using STOUR with complete paths provided by manual Tisserand graph analysis.

Alternatively, a pathsolver may automatically generate search paths by selecting from a list of predefined options for each encounter or permuting a list of flyby bodies. The Star [16] and GREMLINS [17] search tools employ this method. The permutations on the path are automatically created, but the sequence generation is purely mathematical and is not based on energy or scheduling considerations. Experience-based rules may also guide or constrain the path selection.

1.4.2 Stochastic Methods

A multiple gravity-assist trajectory search may also be treated as a hybrid optimal control problem [18], [19]. This approach typically uses a set of nested loops to iteratively solve the pathfinding and pathsolving components of the problem. For example, an outer loop might

solve the discrete problem of where we can go next, sometimes called the accessible region. Often, the discrete outer loop can be modeled as a *tree*, a special case of a graph in which every pair of vertices is connected by exactly one path. For a given accessible region, an inner loop solves the continuous problem of how we get to the destination. The outer loop solves an integer programming problem. The inner loop solves a continuous optimization problem within the parameters of the outer-loop solution.

Englander [20] and Ellison [21], working in the Evolutionary Mission Trajectory Generator (EMTG), address both the pathfinding and pathsolving problems using this nested technique. Englander performs the pathfinding with an outer-loop integer genetic algorithm starting from a random initial population of candidate sequences [20]. An inner loop uses a variety of evolutionary algorithms to do the pathsolving. The inner-loop solutions determine the best performing paths for the outer-loop genetic algorithm. Ellison, provides improvements to the techniques developed by Englander and also proposes a satellite tour search architecture inspired by Lantukh [13], [21].

A significant amount of study has also been focused on metaheuristic searches or global optimization algorithms which are frequently stochastic in nature. These techniques include Monotonic Basin Hopping [20], Genetic Algorithms [22]–[24], Ant Colony Optimization (ACO) [25], or other biological algorithms [19]. Izzo provides a comparison of some of these stochastic approaches for benchmark missions [26]. These methods are distinct from the present research and a full survey is beyond the scope of this document. The reader is referred to Ellison [21] and Lantukh [13] for excellent summaries of techniques and software tools.

1.5 The Tisserand Network Approach

This research introduces a new capability to judiciously generate gravity-assist paths that are likely to produce practical trajectories. Most existing gravity-assist trajectory design methods focus on providing excellent or, in some cases, optimal solutions by following a predesignated path or choosing among predefined options along the way. Their path selection

methods use mathematics (array generation, permutation, etc.) to select design parameters without insight from the astrodynamics that govern the possibilities.

In contrast, the *Tisserand network* approach developed in this dissertation provides a method to systematically target paths where solutions are likely to exist based on orbital energy and scheduling considerations. This method identifies many preliminary solutions with no a priori knowledge of the possible paths. The technique automatically finds gravity-assist paths using only the name of the target and constraints on the motion imposed by astrodynamics. The network can find paths that use powered flybys, resonant re-encounters, and V-infinity leveraging transfers.

The approach uses a discrete representation of the search space that focuses on the pathfinding problem. Therefore, results are intended as initial guess trajectories that can focus continuous pathsolving tools on the most promising trajectories. However, the Tisserand network also provides some basic pathsolving capability.

Other automated methods for generating path sequences [10], [12] may return results that are energy-feasible but not time-feasible or may not exhaustively explore the available options. For example, an automated pathfinder might only return the paths that can be achieved starting from a single Earth departure condition. A Tisserand-graph-based search might include results that are energy-feasible but cannot be executed during the desired mission time frame. The Tisserand network contains both energy- and time-feasible solutions and can be exhaustively searched within constraints provided by the mission designer.

1.5.1 Comparison of Gravity-Assist Trajectory Search Methods

The Tisserand network is a novel pathfinding aid and a network extension of the Tisserand graph. This new network model of the gravity-assist problem, the search algorithms that operate on it (to be described in Chapter 5), and related pathsolving and analysis capabilities are combined in a newly-created, multiple-gravity-assist search tool named **Wander**. This software package implements the concepts introduced throughout this dissertation. More details on **Wander** are provided in Chapter 6.

Table 1.1. Multiple Gravity-Assist Tool Summary

	PATHFINDERS			*PATHSOLVERS*				
	Wander	**TPF[1]**	**MTME[2]**	**STOUR[3]**	**Explore**	**Star**	**GREMLINS[4]**	**EMTG[5]**
Exhaustive Pathfinding	✓	✓						
Energy-Feasible Pathfinding	✓	✓	✓					
Time-Feasible Pathfinding	✓		✓					
Lambert-based Pathsolving	✓			✓	✓	✓	✓	✓
Resonance	✓			✓	✓	✓		✓
Powered Flybys	✓			✓	✓	✓	✓	✓
VILT[6]	✓			✓	✓	✓	✓	✓
Low Thrust				✓	✓	✓		✓
DSM[7]				✓	✓	✓	✓	✓
Escape and Insertion[8]						✓	✓	✓
Reference		[10]	[12]	[8]	[13]	[16]	[17]	[20]

Pathfinding - The task of identifying the sequence of gravity-assist bodies to be visited
Pathsolving - The task of solving trajectories along a specific flyby sequence
[1] Tisserand PathFinder
[2] A Modified Tisserand Map Exploration method without a published name
[3] Satellite Tour design program
[4] GRidded Ephemeris Map of Lambert INterplanetary Solutions
[5] Evolutionary Mission Trajectory Generator
[6] V-Infinity Leveraging Transfer
[7] Deep Space Maneuver
[8] Calculation of the ΔV for departure and/or capture at a body

Table 1.1 revisits the difference in emphasis between pathfinding and pathsolving tools described above. The table highlights some features of several multiple gravity-assist search tools that have detailed descriptions in the literature. The tools are roughly organized into groups that focus on pathfinding versus pathsolving. This distinction is apparent in the lack of dedicated pathfinding features on the upper right portion of the table. It is worth noting that the pathsolving tools will indeed arrive at feasible paths after detailed evaluation of the search space using their respective grid search, tree search, or evolutionary approaches. This result is categorically different from the pathfinding tools that identify paths without first solving them.

Each tool approaches the MGA problem with unique techniques and strengths. Table 1.1 represents the author's best effort to condense the many capabilities of each tool into a collection of comparable features. Even among common features, each tool exhibits a nuanced approach to different types of problems. The reader is encouraged to review the provided references for a full appreciation of the features and capabilities.

1.6 Other Related Research

While not focusing on multiple gravity-assist trajectory design, other researchers have employed similar discretization and graph-based techniques for spacecraft trajectory searches. These researchers apply graph or tree searches to different dynamical models.

Tsirogiannis works in the Circular Restricted Three Body Problem (CR3BP) and uses a graph with vertices that model various periodic orbits and edges that model impulsive maneuvers between the orbits [27]. Trumbauer and Villac also build and search graphs for impulsive maneuvers in the three body problem with an intended use for onboard maneuver planning [28]. Das-Stuart *et al.* apply Dijkstra's algorithm to trajectory searches in the Earth-Moon system [29]. The framework presented by these researchers uses a database of natural motion orbit families in the Circular Restricted Three Body Problem (CR3BP) model and also includes low thrust with both constant and variable specific impulse [30]. Stuart explores a framework for autonomously designing tours of multiple bodies using a network of potential encounter sequences. This framework is applied to a network of impulsive maneuvers in a Keplerian model for an orbital debris removal study and a network of low-thrust trajectories in a CR3BP model for a Trojan asteroid tour study [31], [32]. The researchers examine both an exhaustive tree search and ACO to find tours in these problems.

1.7 New Work in This Dissertation

This dissertation describes a unique approach to finding multiple gravity-assist trajectories. While tree-like traversals of a Tisserand graph have been mentioned by some authors [9], [10], an explicit and methodical construction of a broadly searchable network from fundamental gravity-assist concepts (as in Chapter 3) has not been presented in the literature.

Previous authors describing Tisserand-graph searches understandably apply limits or simplifications to bound the possible search branches. For example, automated searches proceed to a maximum depth [10] or track only best-case transfer times [9] and manual searches rely on rules of thumb [11], [15]. The present work strives to include all feasible variations in the general case while giving the trajectory designer the tools to prune those variations based on mission-specific constraints.

As mentioned above, the network methods described here include complex transfer models such as resonant re-encounters and V_∞-leveraging transfers in a pathfinding application. These capabilities have previously been confined to full-featured pathsolving tools. The time-feasibility of potential paths has not previously been addressed in the pathfinding step. The search results provide more detailed descriptions of the possible trajectories (V_∞ and encounter location) than can be provided with existing pathfinding methods.

The discrete and generalized approach to the VILT problem (Chapter 4) is unique. Other researchers frequently use an iterative Lambert solving method to find the VILT return orbit [17], [33]–[35]. The difference in constraints on the return orbit leads to a different solution method here.

The modification of the Depth-First Search (DFS) to employ gravity-assist, problem-specific constraints (Chapter 5) is a new capability. The use of the line graph of a tree or graph for weighting and searching has not appeared in context of gravity-assist searches. The utility of the transitive closure for predicting fruitful searches has also not been discussed in any similar application. The work in Chapter 4 is the first rigorous and versatile treatment of resonance in Tisserand-graph-based searches.

Finally, instead of temporarily traversing a Tisserand graph to produce a list of possible gravity-assist paths, the present work creates a product, a *Tisserand network*, that itself might be studied to understand the structure of the design space.

1.8 Structure of the Dissertation

Having introduced the problem and discussed solution approaches in Chapter 1, the next chapter reviews some foundational astrodynamics culminating in the Tisserand graph.

Chapter 3 develops the Tisserand network, a network analog to the Tisserand graph. The Tisserand network will be extended to include encounter date information, enabling it to address the phasing problem left unanswered by previous Tisserand analyses.

Chapter 4 will describe how important mission events can be modeled so that they may be included in network searches. Powered flybys, resonant transfers, and V_∞-leveraging transfers expand the types of trajectories that can be found with the network approach.

Chapter 5 describes the search algorithms that can be deployed on the network model. Fundamental graph traversal methods form the basis for more complex algorithms that uncover gravity assist paths through the network. The combinatorics of gravity-assist trajectory searches result in search spaces that grow quickly. Chapter 5 discusses this problem and multiple techniques for managing it.

Chapter 6 presents some search results and compares them to some actual historical missions. A detailed discussion of a Voyager 2 trajectory search provides a verification test for the network technique. Similar searches for Voyager 1, Galileo, and Cassini extend confidence in the method and test the models of Chapter 4.

Chapter 7 demonstrates the use of the Tisserand network in finding trajectories to Trans-Neptunian Objects (TNOs). The solar system model used by the network can be readily updated to include new bodies. In Chapter 7, the Tisserand network discovers trajectories to Haumea and Makemake. Chapter 8 suggests some future work and feature enhancements and summarizes the contribution of this research.

2. MATHEMATICAL BACKGROUND

This dissertation introduces a new multiple gravity-assist trajectory design device called a Tisserand network that is an extension of the successful Tisserand graph technique. In this chapter, we will lay the foundation for the Tisserand network by constructing the Tisserand graph from basic astrodynamics. The Tisserand network applies concepts from graph theory to the information in the Tisserand graph to create a searchable network of gravity-assist transfers. The necessary graph theory concepts will be introduced in this chapter as well.

2.1 Gravity Assist Fundamentals

Let us explore some fundamental relationships related to gravity assist. These concepts will be important for understanding the Tisserand graph and the advancements made in the present research. Throughout this dissertation, we will discuss orbits of a spacecraft about a massive primary body linked by gravity assists from less massive secondary bodies. For simplicity, we will consistently use the Sun as the primary body and one of the planets as the secondary body. The analysis and methods developed will be transferable to trajectories in a satellite system where the planet is the primary and moons are the secondaries.

2.1.1 V-infinity and Pump Angle

Figure 2.1 shows the relationship between the heliocentric spacecraft velocity, V_{sc}, and the heliocentric planet velocity, V_{p}, at a point beyond the gravitational influence of the planet. This relationship is parameterized by the V_∞ vector which represents the spacecraft velocity relative to the planet and an angle α. The hyperbolic excess velocity, V_∞, is the velocity of the spacecraft in excess of that needed to escape the planet's gravitational pull. The $\mathbf{V}_\infty$ vector is the velocity of the spacecraft relative to the planet.

$$\mathbf{V}_\infty = \mathbf{V}_{\text{sc}} - \mathbf{V}_{\text{p}} \,. \tag{2.1}$$

The V_∞ direction differs from the planet velocity direction by the pump angle, α. In this analysis, the pump angle is strictly defined to be between zero and 180 deg.

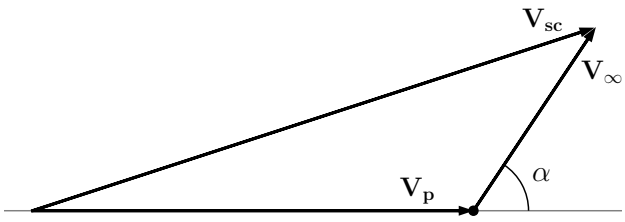


Figure 2.1. The velocity triangle depicts the relationship between the spacecraft velocity, V_{sc}, and the planet velocity, V_p in the heliocentric frame. The V_∞ vector represents the velocity of the spacecraft relative to the planet at the arrival or escape condition. The V_∞ direction and planet velocity direction differ by the pump angle, α.

Under some simplifying assumptions (circular-coplanar planet orbits), the V_∞ and α, uniquely identify the spacecraft heliocentric orbit through some basic equations of Keplerian motion [36]–[39]. Let us briefly review these equations to emphasize the importance of the (V_∞, α) parameterization.

From the velocity triangle, the spacecraft heliocentric velocity, V_{sc}, can be found by application of the law of cosines:

$$V_{sc} = \sqrt{V_p^2 + V_\infty^2 - 2V_p V_\infty \cos(\pi - \alpha)}\,. \tag{2.2}$$

We assume a zero-sphere-of-influence encounter so that the radius of the spacecraft from the Sun is equal to the radius of the planet from the Sun. For a circular planet orbit, we can compute the angular momentum, h, from

$$h = r_{sc} V_{sc} \cos\alpha = r_{planet} V_{sc} \cos\alpha\,, \tag{2.3}$$

where, r_{sc} is the radius of the spacecraft orbit which equals the radius of the planet orbit at the flyby epoch. The eccentricity, e is given (for all conics) by

$$e = \sqrt{1 + (2\mathscr{E}h^2)/\mu^2}\,, \tag{2.4}$$

where μ is the gravitational parameter of the central body (the Sun in this case). In Equation 2.1.1, $\mathscr{E}$ is the specific mechanical energy, which can be obtained from the energy integral as

$$\mathscr{E} = \frac{V_{\text{sc}}^2}{2} - \frac{\mu}{r_{\text{sc}}} \,. \tag{2.5}$$

$\mathscr{E}$ can also be used to find the semi-major axis, a:

$$a = -\frac{\mu}{2\mathscr{E}} \,. \tag{2.6}$$

Some additional orbit descriptors are worth noting. The period of the heliocentric orbit will be important in developing the Tisserand network:

$$P = \frac{2\pi}{\sqrt{\mu}} a^{3/2} \,. \tag{2.7}$$

We also have the apoapsis and periapsis radii given by

$$r_{\text{a}} = a(1+e) \,, \tag{2.8}$$

$$r_{\text{p}} = a(1-e) \,. \tag{2.9}$$

Finally, we can locate the encounter on the heliocentric orbit. The polar equation of a conic section,

$$r = \frac{p}{1 + e\cos\nu} \,, \tag{2.10}$$

can be rearranged to give

$$\cos\nu = \frac{1}{e}\left(\frac{p}{r} - 1\right) \,, \tag{2.11}$$

where ν is the true anomaly and p is the semi-latus rectum:

$$p = \frac{h^2}{\mu}. \tag{2.12}$$

Under the zero-sphere-of-influence assumption we treat the planet and spacecraft as if they are collocated in the heliocentric frame at the encounter. We set r in Equation 2.11 equal to the circular radius distance of the planet. Equation 2.11 can then be evaluated for the true anomaly, ν, of spacecraft at the encounter. The sign ambiguity of the inverse cosine can be resolved with knowledge of whether the spacecraft is approaching or departing periapsis.

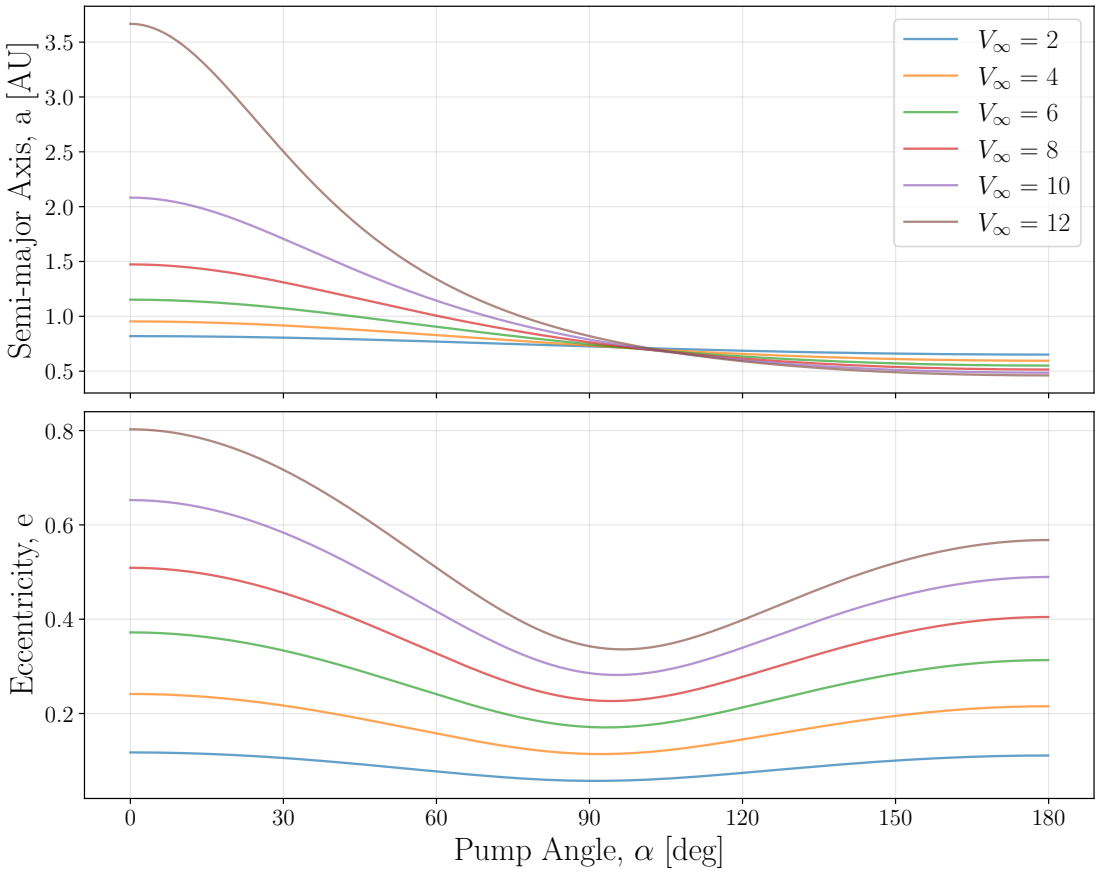


Figure 2.2. The V_∞ and pump angle at the encounter are related to the size and shape of the heliocentric orbit. Here, V_∞ levels are shown as separate lines and the pump angle varies from 0 to 180 deg on the bottom axis. The plots above assume Venus as the flyby body.

Equations 2.2 through 2.11 show that, starting from V_∞ and α, we can identify the size and shape of the heliocentric orbit and the location of the encounter associated with the flyby geometry in Figure 2.1. Since we are assuming co-planar orbits, the inclination, longitude of the ascending node, and argument of periapsis are undefined. Figure 2.2 visualizes this parameterization showing semi-major axis, a, and eccentricity, e, as functions of V_∞ and pump angle. The different lines represent discrete levels of V_∞ with a continuous variation of pump angle.

2.1.2 Bending Angle and Flyby Radius

We now must consider how the gravity assist affects the V_∞ and pump angle. These effects will form the basis for linking a series of gravity assists through the intermediate heliocentric orbits.

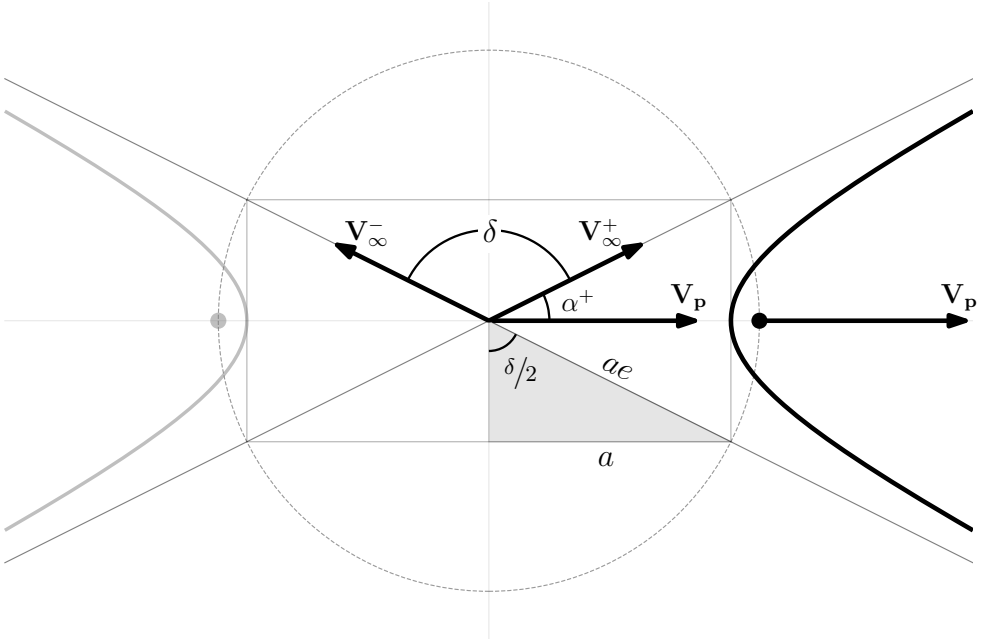


Figure 2.3. The gravity assist rotates the V_∞ vector by an angle δ. The magnitude of the V_∞ vector is unchanged under our simplifying assumptions.

In the planet-centered frame, the velocities along the inbound and outbound asymptotes of the flyby hyperbola are $\mathbf{V}_\infty^-$ and $\mathbf{V}_\infty^+$, respectively. From conservation of energy, we know that the velocity before and after the flyby have the same magnitude:

$$|\mathbf{V}_\infty^-| = |\mathbf{V}_\infty^+| = V_\infty \,. \tag{2.13}$$

For a point-mass planet, the gravitational acceleration on the approaching spacecraft is balanced by an equivalent deceleration on departure. The benefit of the flyby is derived from the change in direction of the V_∞ vector. The amount of direction change is called the bending angle, δ (Figure 2.3).

Let δ be the central angle between the asymptotes of the gravity assist hyperbola. From the geometry of the hyperbola we have

$$\sin\frac{\delta}{2} = \frac{a}{ae} = \frac{1}{e}\,. \tag{2.14}$$

Rearranging gives

$$\delta = 2\arcsin\frac{1}{e}\,. \tag{2.15}$$

Equation 2.9 can be rewritten as

$$e = 1 - \frac{r_\mathrm{p}}{a} \tag{2.16}$$

and we also know that

$$a = -\frac{\mu}{V_\infty^2}\,. \tag{2.17}$$

Collecting Equations 2.16 and 2.17 into Equation 2.15 gives

$$\delta = 2\arcsin\frac{1}{1+\frac{r_\mathrm{p}V_\infty^2}{\mu}} = 2\arcsin\frac{\mu}{\mu + r_\mathrm{p}V_\infty^2}\,. \tag{2.18}$$

For a fixed value of V_∞ at any given planet (μ fixed), the maximum possible bending angle occurs when r_p is minimized.

There are practical limitations of the minimum periapsis. Depending on the celestial body, we may encounter a minimum permissible flyby radius due to the planet surface, the atmosphere, or radiation limits. A lower limit on flyby radius creates an upper limit on the amount of trajectory turning that can be achieved in a flyby. The minimum flyby radius has important implications for reading the Tisserand graph and for designing the Tisserand network. Note: In early analyses, it may be beneficial to allow trajectory searches to exceed these physical limits in order to identify preliminary trajectories that can be later refined to meet constraints.

2.2 The Patched Conic Method

A patched conic analysis of a spacecraft trajectory assumes that the complete trajectory of a spacecraft in an n-body system can be approximated by a series of two-body (Keplerian) orbits. In this simplified analysis, we consider only one gravity field at a time. This is a

significant yet powerful assumption. Battin justifies the approximation in the context of planetary encounters with the following reasoning. The time period during which the planet dominates the spacecraft motion is small compared to the total mission duration. And, during this short time, the distance between the planet and the spacecraft is much smaller than the solar distance. So the Sun affects the motion of both bodies in the same way [40].

Prado compares the patched-conic approach to the circular restricted three-body problem (CR3BP) for flyby trajectories and finds that the two methods agree well in most cases [41]. The techniques typically agree on predictions of the energy gained during the flyby as well as the semi-major axis, eccentricity, and angular momentum before and after the encounter. The study supports the use of patched conics for initial studies even in low energy cases where subsequent analysis in a higher fidelity model is also recommended. Systems with a high mass ratio between the secondary and primary body (like the Earth-Moon system) may require more careful consideration [42].

As a spacecraft approaches a gravity-assist body, the gravitational influence of the secondary body (e.g. a planet) eventually dominates the gravitational influence of the primary body (e.g. the Sun). The spacecraft is said to have entered the secondary body's sphere of influence (SOI). The SOI is determined by considering the ratios of perturbing acceleration to primary acceleration in the three-body system. Tisserand showed that the surface where these ratios are equal has the radius

$$r_{\text{SOI}} = \left(\frac{m_{\text{planet}}}{m_{\text{Sun}}}\right)^{\frac{2}{5}} r_{\text{planet}} \,, \tag{2.19}$$

where m_{planet} and m_{Sun} are the masses of the planet and the Sun, respectively, and r_{planet} is the radial distance of the planet from the Sun [43].

The sphere of influence is a natural patch point in patched conic analysis. Frequently, the elliptical orbit about the Sun is patched to the hyperbolic orbit about the planet at the SOI. Among the planets, however, the great distances of r_{planet} in Equation 2.19 are dwarfed by the mass ratios and the exponential. The SOI for the terrestrial planets are less than one percent of the planetary orbit's semi-major axis. Even among the giant planets, the SOI is only a few percent of the semi-major axis with Jupiter having the highest ratio (about

six percent)[43]. Therefore, we employ a further simplification and assume that the sphere of influence of the planet has zero radius. Under this assumption the gravity assist occurs instantaneously. The position and velocity of the planet are constant during the gravity assist and the spacecraft and planet are co-located with respect to the Sun at the gravity-assist epoch. The gravitational force of the planet effectively imparts an impulsive ΔV to the spacecraft.

Patched conic analysis has been used for initial trajectory designs with great success and has proven to provide a good representation of the true motion. Each of the ballistic, high-thrust tools discussed in section 1.4.1 use the zero-SOI, patched-conic assumption for gravity assist searches. The deviation from reality cannot be ignored and transitioning from a patched-conic solution to a full ephemeris model can sometimes be non-trivial [44]. But the simplification allows trajectories to be analyzed in a fraction of the time that a full n-body, ephemeris-based analysis would require. As we will see, many thousands of Keplerian orbits are modeled in a matter of seconds in the present research.

2.3 The Tisserand Graph

Let us now consider how we might determine whether a gravity assist at one planet can be used to direct a spacecraft to another planet. A gravity assist conserves the energy of the spacecraft-planet system while increasing or decreasing the energy relative to the Sun. The Tisserand graph is a useful tool for understanding how this change in heliocentric energy can be used to connect multiple gravity assists.

2.3.1 Tisserand's Criterion

Tisserand's parameter is an approximation of the CR3BP Jacobi constant expressed in orbital elements:

$$C_{\mathrm{T}} = \frac{1}{a} + 2\sqrt{\frac{a(1-e^2)}{r_p^3}}\cos i\,, \tag{2.20}$$

where r_p is the orbital radius of the planet perturbing the motion [40], [45]. Like the Jacobi constant, the Tisserand parameter is an energy-like constraint on the possible motion in a three-body system.

An encounter with a massive body can significantly change the orbital elements of a spacecraft or comet orbiting the Sun. So much so that it may be difficult to determine if the orbit before and after the encounter belong to the same body. François Félix Tisserand derived the constant in Equation 2.20 while studying the affect of Jupiter on comet orbits [40], [43], [46], [47]. Consider two sets of orbital elements (a_1, e_1, i_1) and (a_2, e_2, i_2) from comet observations made at two different times. If the two sets of elements approximately satisfy *Tisserand's criterion*:

$$\frac{1}{a_1} + 2\sqrt{\frac{a_1(1-e_1^2)}{r_p^3}}\cos i_1 = \frac{1}{a_2} + 2\sqrt{\frac{a_2(1-e_2^2)}{r_p^3}}\cos i_2\,, \tag{2.21}$$

then they are assumed to belong to the same comet—before and after an encounter with a massive body.

The application of Tisserand's criterion is not restricted to comets. It also places a constraint on the possible spacecraft motion achievable after a gravity assist and has historically been used to confirm flyby calculations in patched-conic analyses [48]. Tisserand's parameter describes the possible orbits that might result from a flyby. Let us now consider the inverse problem. Given a heliocentric orbit (a, e, i), what flyby might it have resulted from? Could it have resulted from a flyby of different planets? If so, then we have more than one gravity assist related to the same heliocentric orbit. This line of thought is the basis for the Tisserand graph, a graphical method for associating heliocentric orbits with planetary flyby parameters.

2.3.2 A Graphical Analog to Tisserand's Parameter

The Tisserand graph is a proven tool for identifying energy-feasible, gravity-assist paths. The approach builds a multiple gravity assist trajectory by linking a series of gravity assists that match in heliocentric energy. Importantly, the Tisserand graph does not consider orbit

phasing and the paths identified with this technique might not be practical or might only occur in the very distant future.

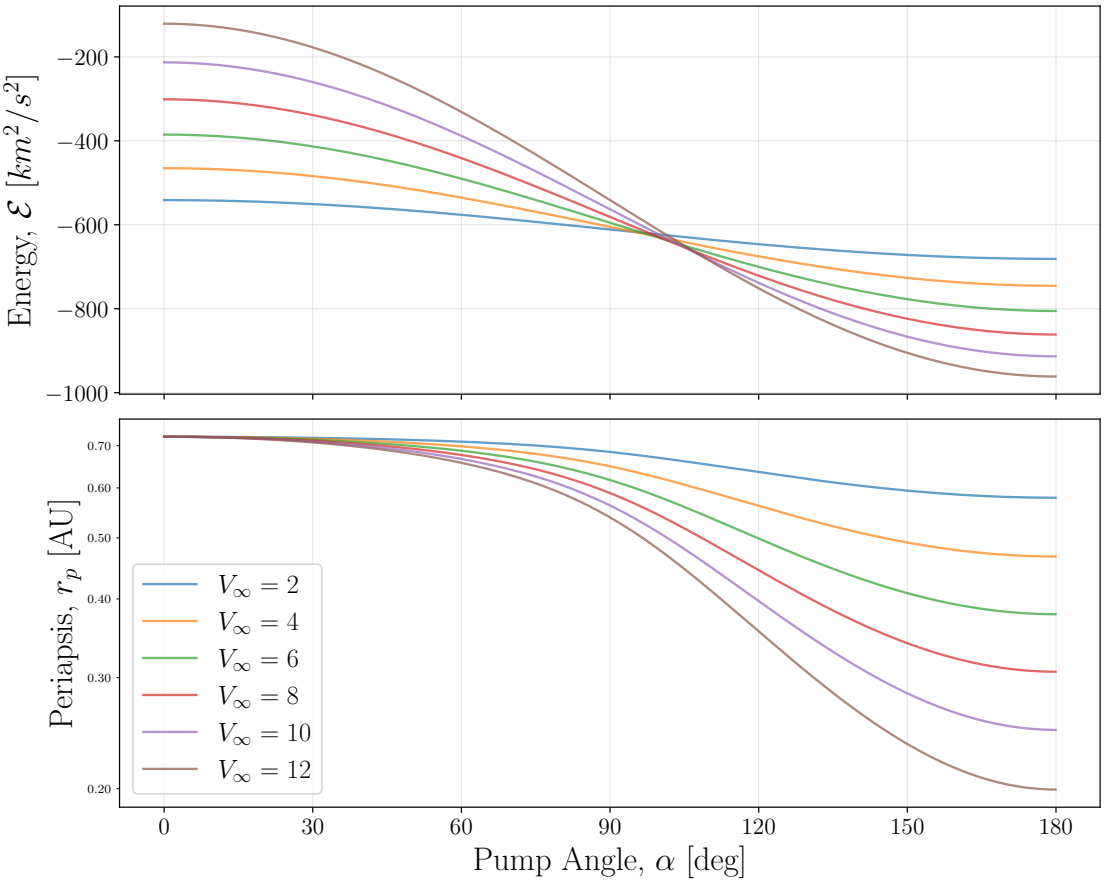


Figure 2.4. The V_∞ and pump angle at the encounter are related to the energy and periapsis of the heliocentric orbit. Here, V_∞ levels are shown as separate lines and the pump angle varies from 0 to 180 deg on the bottom axis. The plots above assume Venus as the flyby body.

To identify candidate gravity assists, we graphically present heliocentric orbit parameters (such as those derived from V_∞ and α in section 2.1.1) as a set of constant flyby V_∞ contours. Intersections of the V_∞ contours locate flybys with common heliocentric orbits. These orbits can be strung together to create a tour. The same principle can be applied to a series of satellite flybys around one of the planets. This type of analysis was first presented in the literature by Labunsky *et al.* [49]. The term "Tisserand graph" was coined by Longuski [9] to describe such a display after the related concept, Tisserand's criterion [9], [40], [43].

Consider again the flyby (V_∞, α) parameterization of the heliocentric orbits. Figure 2.2 showed the affect of V_∞ and pump angle on the size and shape of the orbit in terms of semi-major axis and eccentricity. Figure 2.4 gives an equally valid representation of the size and shape of the orbits as a function of these parameters. In this case, we see the trends in specific mechanical energy, $\mathscr{E}$, (an analog of orbit size) and periapsis radius, r_p, (which combines the effects of size and shape).

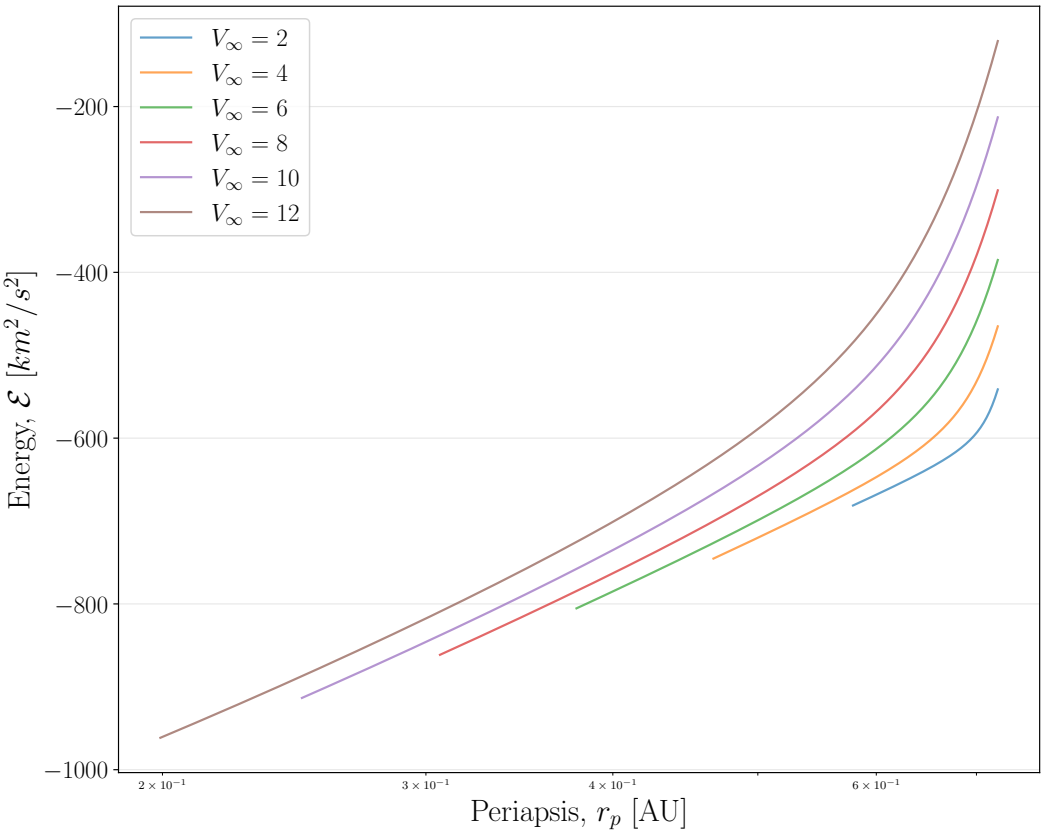


Figure 2.5. The Venus V_∞ contours combine the size and shape relationships into a single view. The pump angle varies from 0 deg at the upper-right to 180 deg at the lower-left.

Now let us combine these relationships into a single plot. Figure 2.5 shows contours of V_∞ at Venus in the $\mathscr{E} - r_\mathrm{p}$ plane. At the upper-right points on each contour, the pump angle is zero. The pump angle decreases as we move down the contour until it reaches 180 deg at the lower-left.

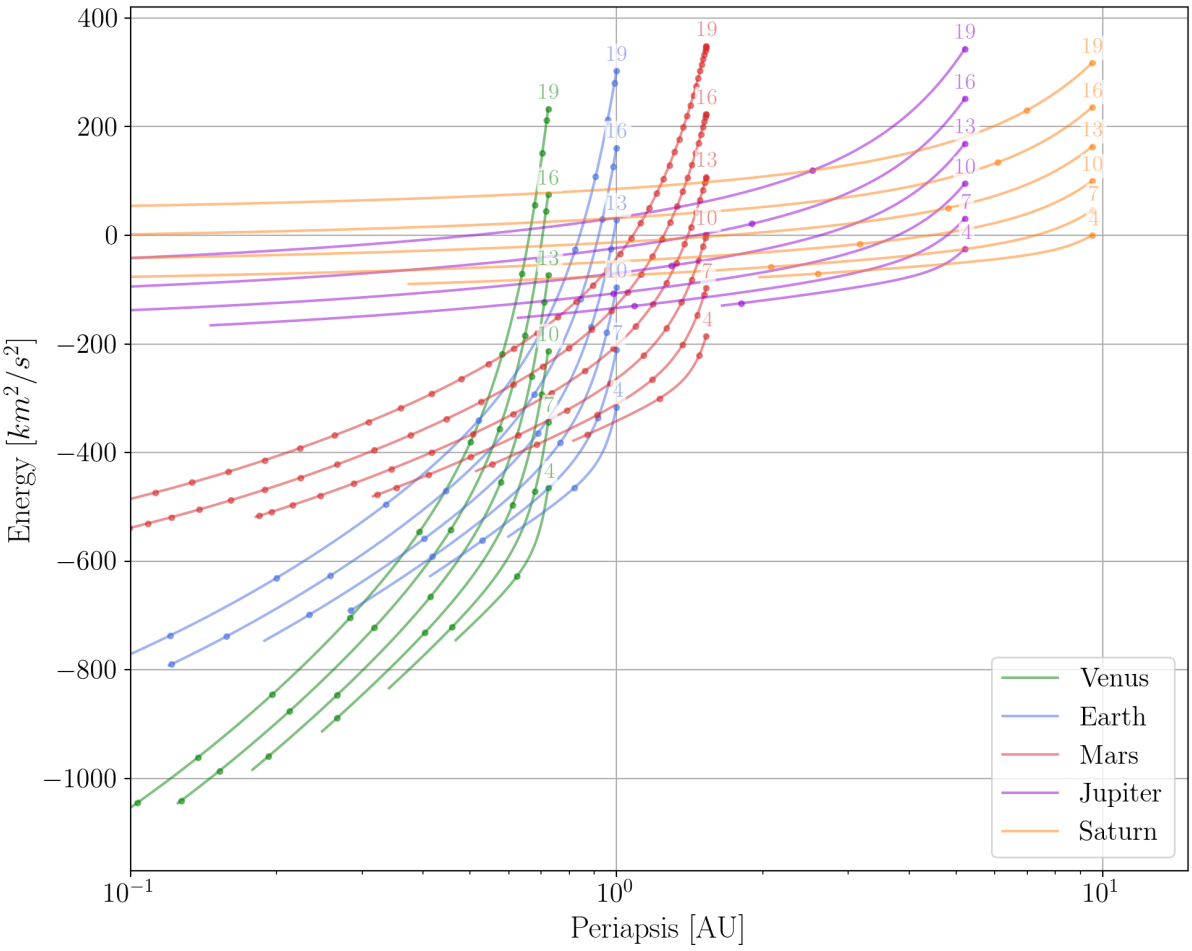


Figure 2.6. A sample Tisserand graph including Venus, Earth, Mars, Jupiter, and Saturn. The contours for each planet represent the locus of heliocentric orbit parameters for flybys at a given V_∞. Intersections of the contours identify pairs of flybys for two planets connected by the same heliocentric orbit.

Intuitively, when α is zero, the spacecraft velocity is aligned with the planet velocity. This corresponds to the highest energy orbit that can result from a flyby at the given V_∞. Conversely, when α is 180 deg, the spacecraft velocity is anti-parallel to the planet velocity, yielding the lowest energy orbit achievable from this flyby.

Finally, if we co-plot the V_∞ contours from Figure 2.5 for flybys at multiple planets, we have a Tisserand graph. Figure 2.6 shows a simple Tisserand graph. We refer to the intersection of two V_∞ contours as a *node*. Each node defines a single heliocentric orbit (given by the particular $\mathscr{E}$ and r_p in the plot space). Most importantly, this heliocentric

orbit is related to a flyby of the two planets (given by the V_∞ contour values and the pump angle at each planet).

We identify each node by the planets and V_∞ contours of the intersection. For example, at the intersection of the Venus 7 $\mathrm{km\,s^{-1}}$ contour with the Earth 10 $\mathrm{km\,s^{-1}}$ we have the V7E10 node. The pump angle is given by the distance along each contour where the node is found. The connections at the nodes are the basis for the graphical pathfinding method. A sequence of nodes through the Tisserand graph describes a series of flybys and the connecting heliocentric orbits. Appendix A includes a detailed example of how a gravity assist path may be read from the Tisserand graph.

Since the Tisserand analysis only identifies encounters that are viable from an energy perspective, a series of Lambert problems solved over a discrete range of times is typically required to provide information on launch opportunities [8], [17], [50].

Notwithstanding the success of the Tisserand graph, the graphical technique requires training, experience, and intuition to yield viable paths. If the number or order of flyby bodies is allowed to vary, a considerable amount of time can be required to simply enumerate the permutations. As described above, the phasing problem is typically solved separately in a subsequent step. The present work addresses these shortcomings.

2.4 Graph Theory

The present research establishes a framework for the analysis of gravity assist pathfinding borrowed from the field of graph theory—a branch of mathematics concerned with the connections between objects [51]–[54]. Graph theory has been applied to many classical problems in combinatorics, such as the Königsberg bridge problem, the traveling salesman problem, the vehicle routing problem, and the knapsack problem. Since the time that Euler solved the Königsberg bridge problem, graph theory has developed into a major field of discrete mathematics with its own powerful theories and algorithms.

2.4.1 Network Nomenclature

The graph theory field has its own expansive vocabulary and many terms have different meanings in other contexts. The reader is encouraged to consult the nomenclature section for clarity. Some fundamental concepts from graph theory have the same names as unrelated concepts in astrodynamics. To avoid confusion, we use the term *network* to refer to a series of connected objects (as in graph theory) and reserve the term *graph* (as in the graph of a function) for the already established Tisserand graph (as shown in Figure 2.6). Also, the term *node*, commonly used in graph theory for the objects that are connected in a graph, has already been used in the Tisserand-graph literature to refer to the intersection of V_∞ contours on the Tisserand graph. Therefore, we will use *vertex* to refer to the connected points in a network.

We will refer to the link between two vertices as an *edge*. Two vertices connected by an edge are said to be *adjacent*. Edges are sometimes also called *lines* in the graph-theory literature. This usage gives rise to the name *line graph*. A line graph is a type of inversion of a network that will be important in the development in Chapter 3. Since the term *line graph* is unambiguous we will use both *line* and *graph* in this context. Edges may or may not have an explicit direction. If the edges have a direction, then the network is said to be *directed*. In the literature, such a network is also called a *di-graph*. A pair of vertices may be connected by more that one edge. In this case, two or more edges connecting the same vertices are *parallel*. A network with parallel edges is sometimes called a *multi-graph*. An undirected graph in which a path exists between every pair of vertices is said to be *connected*. A connected network containing no cycles is called a *tree*.

We will use the symbol V to represent the number of vertices in a network and E to represent the number of edges. Context should prevent confusion with the symbols V_∞ (spacecraft velocity relative to the gravity assist planet) and E (eccentric anomaly). The number of edges in a directed network may be as high as $V(V-1)$. When parallel edges are permitted, there is no limit to the number of edges. Networks with a low number of edges relative to vertices are called *sparse* and those with a high number of edges are called *dense*. Sedgewick gives $E \approx V \log V$ as a rough threshold for sparse networks [54]. The

computational complexity of network algorithm is typically a function of V and E. So the density of a network can adversely affect the speed of execution. The networks in this research will typically be dense (many edges per vertex) but we will examine some filtering techniques that will reduce the density considerably.

A *walk* or *path* through a network is any alternating series of adjacent vertices and edges. A *simple path* is a path that does not revisit any vertex. The term *path* is also used in the Tisserand-graph literature to refer to a sequence of gravity assists at two or more planets (typically identified by the initial letter of each planet name). When path is used in the context of the Tisserand network we intend its meaning from graph theory (a sequence of vertices). However, the concept being described will be very similar to the Tisserand graph path (as in pathfinding and pathsolving). A small, but important, difference is that a Tisserand network path will include more information than just the names of the gravity assist bodies. When there is the possibility of confusion we will use the term *route* to refer to the more detailed gravity-assist sequence produced by the Tisserand network. The paths discussed in this dissertation will be simple paths. An example of a non-simple path in the Tisserand network might be a cycler orbit.

Edges may carry one or more numerical *weights* characterizing the connection between the two vertices. For example, the weight of an edge might be assigned to be the travel time between its two vertices. In this way, a path through the network will immediately reveal not just the waypoints along the path but the total travel time.

2.4.2 Summary of Fundamentals

This chapter develops the Tisserand graph from the basic dynamics of the gravity assist. The V_∞ and pump angle, α, describe the spacecraft flyby velocity vector and provide enough information to determine the related orbit about the primary body. We use discrete V_∞ levels and continuous pump angle values to construct a graphical representation of the energy-feasible transfers between bodies in the Tisserand graph. In Chapter 3, the graph theory concepts just introduced will be used to construct a network extension to the Tisserand graph.

3. NETWORK MODEL FOR THE GRAVITY-ASSIST PROBLEM

In this chapter we develop a network analog of the Tisserand graph. The network model exposes the connectivity inherent in the Tisserand graph to well-established search algorithms. This effort improves the effectiveness of the energy-based technique and establishes a foundation for further expansion. We will build on this foundation to add encounter time information to the network, enabling the energy and phasing problems to be solved simultaneously.

The network will be developed for gravity-assist trajectories between the planets. However, the technique can be applied with similar assumptions in the planetary satellite systems.

3.1 Solution Approach

A guiding principle in developing the Tisserand network and its associated models is a separation between the astrodynamics and graph theory algorithms. A common theme will be to pre-compute parameters describing the potential transfers. Many descriptive characteristics of the potential transfer arcs (V_∞, time of flight, etc.) can be rapidly computed from the information in the Tisserand graph without the need for iterative solution techniques. These parameters are typically integrated into the network itself (usually as a weighting for the network edges).

This integration allows multiple searches to be executed on a single, pre-configured network. But, more importantly, search speed is improved because the search algorithm does not need to perform astrodynamics calculations as it traverses the network. The network is not modified once the search begins. Software objects that model gravity-assist transfers do not need to be created or destroyed during the traversal. These qualities contribute to reduced computational overhead. Additionally, we will see in Chapter 5 that the pre-calculation of potential encounter dates provides an effective filter for simplifying the network prior to performing a search.

The search speed improvement comes with an additional cost in constructing and weighting the network. Accordingly, simple and approximate mathematics are emphasized and we

seek low to medium fidelity solutions. Some of the auxiliary models (V-infinity leveraging transfers, resonant flybys) to be discussed in Chapter 4 do involve some iterative solutions.

The pre-calculation of the astrodynamics problems was suggested by the great deal of information available in the Tisserand graph, but it is not a requirement for applying the network method. Indeed, an argument can be made that, depending on the search criteria, a good portion of the preliminary calculations performed to weight edges are for trajectory arcs that are eventually filtered out. A "lazy evaluation" strategy, in which expressions are formulated but not evaluated until they are needed, can mitigate the problem of unnecessary computation. Other researchers implementing this technique may consider the best place to perform these, or other, problem-specific calculations in light of their research goals.

The research presented here will generally attempt to cast a wide net when developing search results. For example, we may allow more spacecraft revolutions in resonance pairs than are typically considered. Depending on the V_∞ discretization chosen, we may want to allow fairly large tolerances in encounter dates.

We adopt this posture to give the network the opportunity to discover paths that may not be intuitive or expected. Rough approximations of trajectories can be refined later in higher fidelity tools. The cost of this approach is paid in additional search time. We will see, in Section 3.7, that some search results may need to be discarded if they are assessed to be impractical. Tolerances may easily be tightened if the extended time does not produce quality trajectory candidates.

3.1.1 Differences with Other Approaches

Some fundamental differences between the Tisserand network approach and other path-solving techniques such as Williams [8], Hughes [14], Landau *et al.* [16], and Mudek [17] could be a source of confusion. These key differences are highlighted below.

Frequently, a multiple gravity assist (MGA) trajectory will be found by searching over a grid of V_∞ and encounter dates, solving a series of Lambert problems between the grid points. An iterative procedure, such as C_3 matching [8] will adjust the encounters until the epochs and V_∞ agree. There is no C_3 matching of gravity assists in the Tisserand network.

By definition, $V_\infty^- = V_\infty^+$ since these are defined by the Tisserand network vertices (Tisserand graph nodes). The trade off is a time gap between arrival and departure at each gravity-assist body.

Many other trajectory search methods perform the astrodynamics calculations during the search iterations. Researchers employing this approach include Lantukh [13], Vasile *et al.* [19], Ellison [21], Stuart *et al.* [32], and Wu and Russell [35]. An example of this technique might perform a discrete outer loop search over trajectory options, and also perform a series of astrodynamical calculations in an inner loop to identify the next reachable encounter.

The Lambert problem is not fundamental to the Tisserand network approach. In the Tisserand network method, time appears in the problem through Kepler's equation (3.1, 3.7) in such form that the mean anomaly, $\mathscr{M}$, can be computed directly and not iteratively. Paths through the Tisserand network are found without any Lambert solving. Once paths are identified, a Lambert solver can be used to create patched-conic trajectories from the discontinuous network solutions. But in this case, there is not an outer-loop iteration over a series of Lambert problems to achieve a target trajectory as is typical of gravity assist grid searches.

3.1.2 Solution Sequence

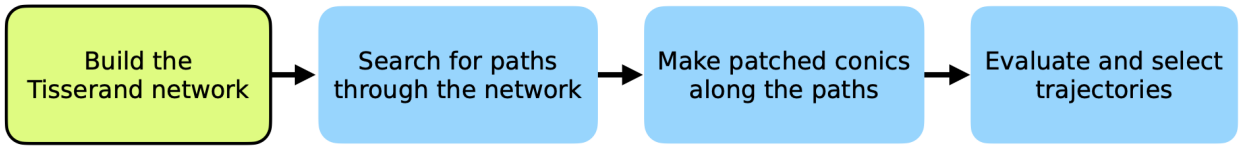


Figure 3.1. The process of finding gravity-assist trajectories with the Tisserand network includes four major steps. The first step constructs the network from simplified dynamical models.

In broad terms, the identification of gravity assist trajectories using the Tisserand network approach will include the four sequential steps shown in Figure 3.1. First, the Tisserand network is constructed and fully populated with the required astrodynamics knowledge. Next, we conduct a search using a network or graph-based traversal algorithm. The search

yields paths through the discrete network which are sufficient to initialize some existing grid search tools. However, as a third step, we can also directly construct patched-conic trajectories from the information in the search output. Finally, we evaluate the patched-conic trajectories to select candidates for higher fidelity analysis.

The pre-computing of the astrodynamical properties and the network searchability facilitate the automation of this process. In contrast, the previous state of the art for Tisserand-based trajectory searching required manual consultation of the Tisserand graph followed by repeated application of grid-search tools guided by intuition and experience.

Each of the tasks in Figure 3.1 will be presented in this chapter. Sections 3.2 through 3.4 develop the Tisserand network and how it is constructed from possible gravity assists. Section 3.5 introduces the basic search algorithm, but the entirety of Chapter 5 is also dedicated to this topic. Section 3.6 describes how patched-conic trajectories may be constructed from the search results. Finally, Section 3.7 introduces some methods for evaluating the patched conics to find the most promising trajectories.

The first task—building the Tisserand network—is the major focus of this chapter and a key contribution of the current work. To construct the network we will need to answer some fundamental questions:

- What trajectory elements will we use as our vertices?
- In what ways are these elements connected?
- How can we weight the connections to help answer trajectory design questions?

These questions will be answered in Sections 3.2 and 3.3. First, let us review the assumptions underlying the Tisserand graph.

3.1.3 Assumptions and Limitations

The following simplifying assumptions accompany the original form of the Tisserand graph and will be carried forward for the network approach:

- Ballistic trajectories

- Point-mass gravity
- Circular, coplanar planetary orbits
- Heliocentric orbits
- Patched, two-body dynamics
- Zero sphere of influence
- Practical constraints on minimum flyby radius

We will find that this level of fidelity is appropriate for pathfinding and preliminary trajectory generation. These assumptions are common among most, if not all, of the similar ballistic searches discussed in Section 1.4.1. The simple models will help, rather than hinder, the identification of many potential trajectories. Fidelity can be increased after preliminary paths are discovered.

While the network construction using the Tisserand graph assumes purely ballistic transfers, the evaluation of the search results will allow for powered flybys. Low-thrust transfers are not currently modeled. The network models will be extended to include two classes of impulsive maneuvers in Chapter 4. However, the lack of a formal Deep Space Maneuver (DSM) model may limit the quality of initial guesses that can be created for some mission architectures. Chapter 8 suggests a pathway to include a general DSM model.

The general architecture is extendable to satellite tour studies, but the assumptions may be more limiting. In planetary systems we might expect more elliptical orbits and transfers that require inclination changes. The perturbing effects of the nearby satellites and non-spherical gravity of the primary body may challenge the patched-conic and point-mass assumptions. These limitations do not necessarily prohibit the use of the Tisserand graph for generating initial guesses. Chapter 8 discusses a possible extension to address inclination changes.

3.2 Constructing a Network of Energy-Feasible Transfers

The series of connected contours and nodes that make up the Tisserand graph (Figure 2.6) immediately calls to mind a road map. Each node in the Tisserand graph identifies a

heliocentric orbit that is common to two encounters. To construct a network, we might be tempted to create vertices from the nodes and connect them with edges representing the contours. This procedure mimics the way we read a Tisserand graph and is the method employed by Strange and Longuski [9] and de la Torre Sangre *et al.* [10]. However, while each node represents a single heliocentric orbit, that orbit may contain up to eight transfer arcs. The elegance of the graph obscures the individual connecting transfers and makes it difficult to directly automate traversal. We will reorganize the information in the Tisserand graph to expose the individual transfers.

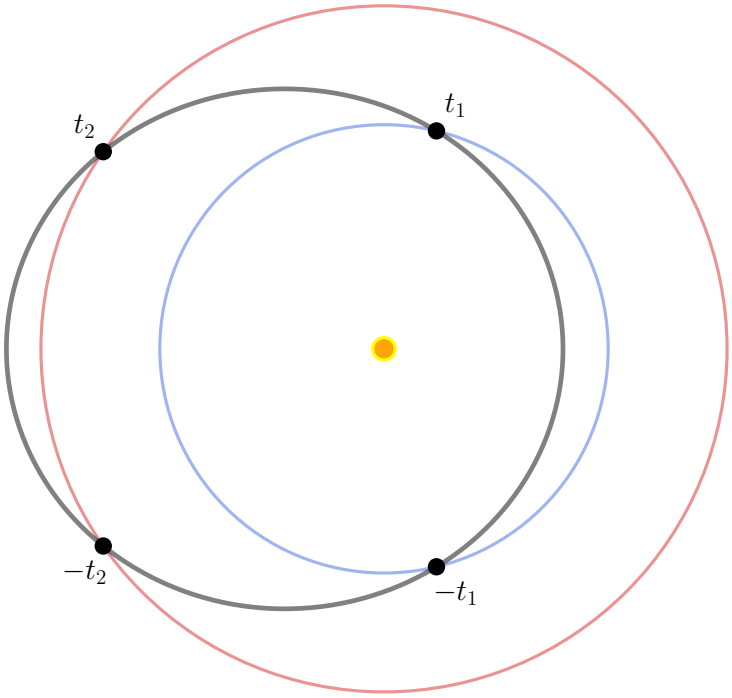


Figure 3.2. A particular heliocentric orbit (black) is unique to each node in the Tisserand graph. A circular or elliptical heliocentric spacecraft orbit intersects the orbits of the two planets in the node (red and blue) at four points. The points are labeled by the times relative to periapsis passage in the spacecraft orbit.

3.2.1 Choosing Vertices and Edges for the Network

Let us first consider the possible transfer arcs between two planets in circular coplanar orbits. For this discussion, we can ignore the positions of the planets at any particular time and consider only the geometry of their orbits. According to the assumptions above, each Tisserand-graph node represents a Keplerian, heliocentric orbit. The V_∞ and α at each planet are explicitly defined by the V_∞ contours that are intersecting at the given node.

Let us consider the general case of a closed heliocentric spacecraft orbit. As seen in Figure 3.2, an elliptical transfer orbit intersects the two circular, planetary orbits at four locations (two locations on each planetary orbit). A transfer arc on this heliocentric orbit can start at any of the four intersection points and end at either of the two points at the other planet's orbit. This produces eight possible arcs. Figure 3.3 shows this geometry for the four cases starting at the inner planet. There are also four cases starting at the outer planet that are not shown.

We label the intersection of the transfer orbit and the planetary orbit as "inbound" or "outbound" depending on whether a spacecraft on the transfer arc is approaching or departing from periapsis. In other words, an inbound encounter has a heliocentric true anomaly between π and 2π; an outbound encounter has a true anomaly between 0 and π. The eight possible transfer arcs connect the inbound or outbound encounter of one planet with the inbound or outbound encounter of the other planet.

Table 3.1. The Eight Possible Transfers at Each Tisserand Node

No.	From Body	From Location	To Body	To Location	Up/ Down	Transfer Angle
1	planet 1	Outbound	planet 2	Inbound	Up	$2\pi - \nu_2 - \nu_1$
2	planet 1	Outbound	planet 2	Outbound	Up	$\nu_2 - \nu_1$
3	planet 1	Inbound	planet 2	Inbound	Up	$2\pi - \nu_2 + \nu_1$
4	planet 1	Inbound	planet 2	Outbound	Up	$\nu_2 + \nu_1$
5	planet 2	Outbound	planet 1	Inbound	Down	$2\pi - \nu_2 - \nu_1$
6	planet 2	Outbound	planet 1	Outbound	Down	$2\pi - \nu_2 + \nu_1$
7	planet 2	Inbound	planet 1	Inbound	Down	$\nu_2 - \nu_1$
8	planet 2	Inbound	planet 1	Outbound	Down	$\nu_2 + \nu_1$

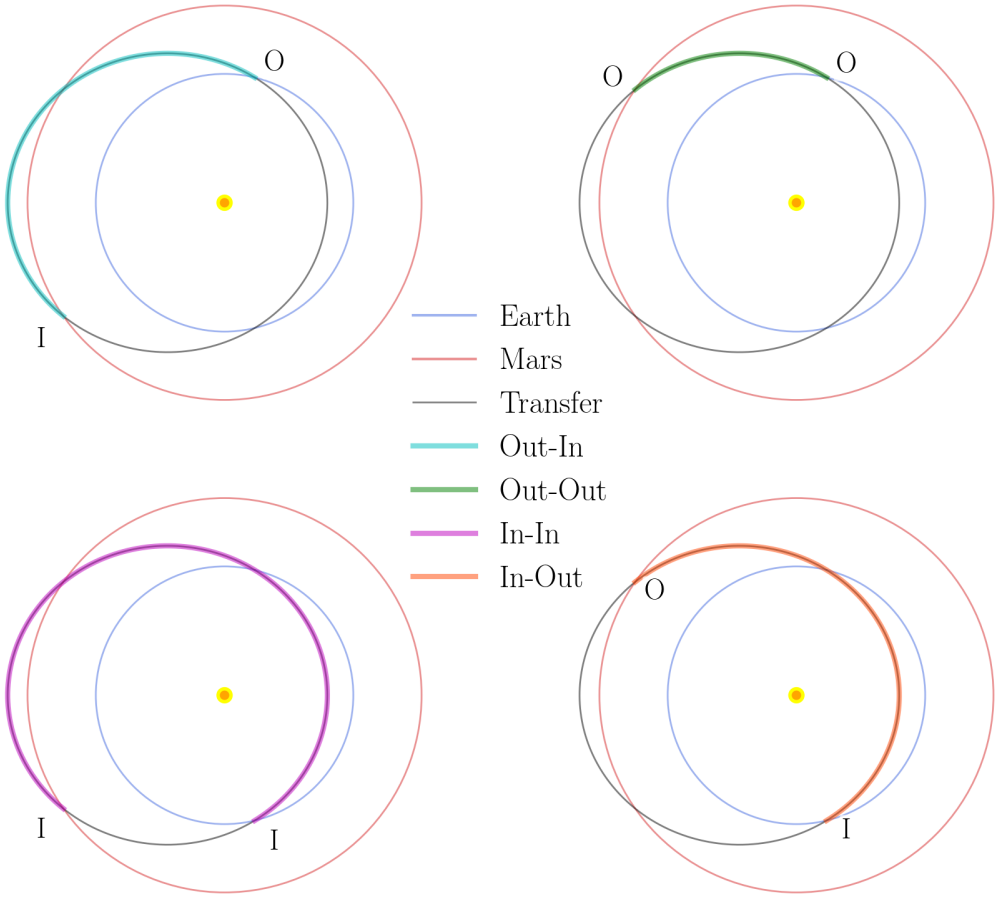


Figure 3.3. An example of the four possible upward transfer arcs between planets in circular, coplanar orbits. This transfer orbit is defined by the intersection of the Earth-10 and Mars-7 contours and crosses each planetary orbit at two points (denoted I for inbound and O for outbound). The two points on one orbit connect to the two points on the other orbit to create four unique arcs in the transfer orbit. These figures represent half of the possible transfers at the Earth-10/Mars-7 Tisserand graph node. Four additional arcs can be constructed for downward transfers (starting on the outer planet). The transfer time depends on the arc and the direction of travel.

We will also occasionally benefit from categorizing the transfers as "upward" when the departure planet is closer to the Sun than the arrival planet and "downward" when the opposite is true. The transfers are enumerated in Table 3.1 (where planet 2 is further from the Sun than planet 1). In Table 3.1 the ν_i are the true anomalies on the spacecraft orbit at

the intersections in Figure 3.2. If the heliocentric transfer orbit is parabolic or hyperbolic, then there are only four transfer arcs.

The shape of the transfer orbit (and therefore the location of the intersection points) is determined by the departure velocity in the heliocentric frame. However, in anticipation of applying the Tisserand graph, we will identify the transfer orbits by the associated V_∞ of the flyby at each planet. To link consecutive flybys, we must take care that the location of the flyby is consistent. For example, an arc that ends at an inbound encounter must connect to an arc that begins at an inbound encounter.

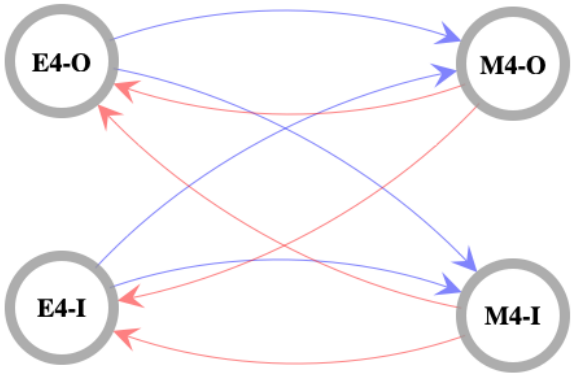


Figure 3.4. A single Tisserand graph node represents up to eight transfer arcs between the two planets at the V_∞ contour crossing. This situation can be modeled as the network above. A vertex is created for each planet, V_∞, and encounter location. The eight edges connecting the vertices represent the eight possible transfers between the four points on the heliocentric orbit.

Let us choose the various flyby encounters (the intersections in Figure 3.2) as the vertices of our network. Each Tisserand graph node becomes four vertices: an outbound and inbound encounter at both planets in the node. We identify each vertex by the planet, the V_∞, and the inbound or outbound location of the encounter. For example, an inbound flyby of Venus at a V_∞ of 7 $\mathrm{km\,s^{-1}}$ is denoted V7-I.

The vertices are connected by the eight possible transfer arcs on the common heliocentric orbit (four of which are shown in Figure 3.3). The up/down and inbound/outbound classifications uniquely identify the various transfer arcs. Each arc has different properties, so we must use a distinct network edge for every arc. Each Tisserand graph node can now

be expressed as a system of four interconnected vertices as shown in Figure 3.4. The four vertices represent the possible gravity assists at the node and the eight edges represent the eight different transfer arcs between the four vertices. To create a network representing the entire Tisserand graph, we simply repeat this network expansion at each node. We call the result a *Tisserand network*.

Of course, before expanding the Tisserand graph nodes into network components, we must first find them all. This is a non-trivial task but it can be performed with an iterative root-finding algorithm for every pair of contours in the Tisserand graph. For any two V_∞ contours (from two different planets) in Figure 2.6, we seek the point where the energy and periapsis are the same. At this contour intersection, the V_∞ and pump angle on the two contours will be different. But the combination of the V_∞ and pump angle with each planet's parameters will generate the same heliocentric orbit (as shown in Figure 3.2).

Let's visualize the network by creating a grid of the vertices with increasing V_∞ as we move from bottom to top and increasing distance from the Sun as we move from left to right. This is a categorical arrangement; the scale of "x" and "y" axes are not important. This visualization is shown in Figure 3.5. For illustration purposes, the downward edges are colored red and the upward edges are colored blue.

Each arrow in Figure 3.5 represents a unique heliocentric transfer between two planets. We can identify a path through the network (and through the solar system) by starting at any of the connected vertices and following the arrows to other vertices.

3.3 Weighting the Network Edges

The Tisserand network shown in Figure 3.5 is useful for performing searches that simply identify paths through the connected vertices. However, additional steps must be taken to determine the travel time for a given path.

To answer questions about the mission timing, we will assign a weight to each edge equal to the time of flight of the associated transfer arc. The time of flight depends on both the inbound/outbound endpoints and the up/down direction of travel. In graph theory terms,

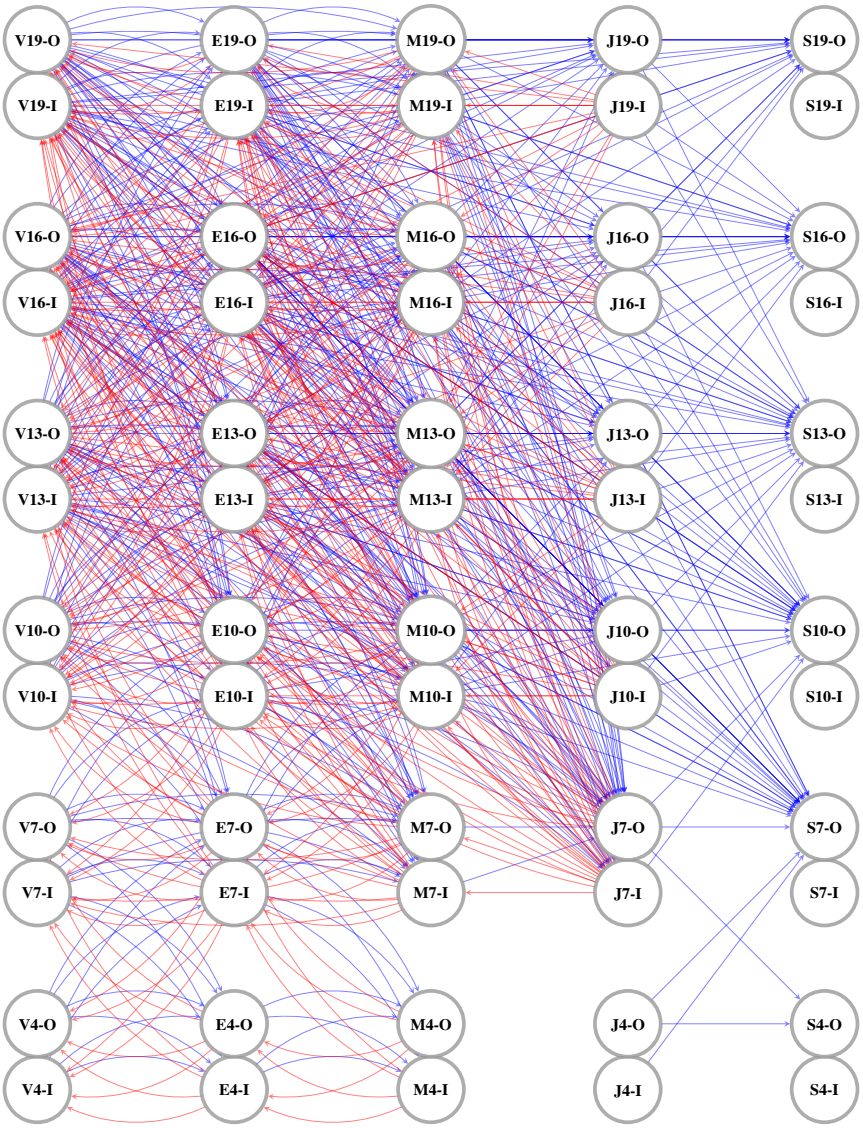


Figure 3.5. A visualization of the Tisserand network mirroring the Tisserand graph in Figure 2.6. The vertices represent flybys of a given planet at a particular V_∞ and location in the planetary orbit. The edges indicate flybys that can be connected and are analogous to the nodes of the Tisserand graph. There are several network edges for each Tisserand graph node. The red edges indicate a downward transfer. The blue edges indicate an upward transfer. In this example, downward edges from Saturn are removed.

our edges are "directed". The weight of the edge from A to B is different than the weight from B to A.

Table 3.2. Time of Flight for Possible Arcs

Upward	Downward	Time of Flight
I-I	O-O	$P + t_1 - t_2$
O-I	O-I	$P - t_1 - t_2$
I-O	I-O	$t_1 + t_2$
O-O	I-I	$t_2 - t_1$

3.3.1 Time of Flight for Simple Transfers

The circular-coplanar assumption assists in finding the time of flight on each possible arc. For simple transfers, the flight time is computed from the limited number of equations presented by Strange and Longuski[9] and reproduced in Table 3.2. These equations are derived in Appendix B. Note that some of the arcs listed in Table 3.2 do not exist for parabolic or hyperbolic transfers.

For circular or elliptical orbits, the time relative to periapsis passage can be computed through an application of Kepler's equation:

$$\mathscr{M} = E - e \sin E \,, \tag{3.1}$$

so that t is given by

$$t = \frac{\mathscr{M}}{n} = \frac{P}{2\pi}(E - e \sin E) \,. \tag{3.2}$$

Also, from the geometry of the eccentric anomaly, we have

$$r = a(1 - e \cos E) \,, \tag{3.3}$$

which, when solved for E, gives

$$E = \pm \arccos\left(\frac{1}{e} - \frac{r}{ae}\right) . \tag{3.4}$$

From the circular planet orbit assumption, we know that the intersections of the heliocentric orbit with the planet orbit in Figure 3.2 occurs at a radius distance, r_{planet}. So we have

$$E_{\text{encounter}} = \pm \arccos\left(\frac{1}{e} - \frac{r_{\text{planet}}}{ae}\right), \tag{3.5}$$

which can be used in Equation 3.2 to find $\pm t_1$ at $r_{\text{planet-1}}$ and $\pm t_2$ at $r_{\text{planet-2}}$. Referring to Figure 3.2, we are interested in both values of the inverse cosine which correspond to the inbound and outbound encounters.

Alternatively, we might first compute the true anomaly, ν, at r_{planet} in the spacecraft orbit and then use

$$\sin E = \frac{\sin\nu\sqrt{1-e^2}}{1+e\cos\nu} \qquad \cos E = \frac{e+\cos\nu}{1+e\cos\nu}, \tag{3.6}$$

to compute E with an *atan2* function.

To summarize, the (V_∞, α) at a given Tisserand graph node determines the heliocentric orbit (a, e). Knowing that the encounters occur at the radial distance of the planet, we may compute the eccentric anomaly (Equation 3.5) and then the possible encounter times (relative to periapsis) using Equation 3.2. These encounter times can be used to compute the time of flight of the eight arcs represented by the Tisserand graph node from Table 3.2. Note that because we have the positions and we are solving for time, we do not need to solve the transcendental Kepler equation iteratively.

A similar procedure may be performed for hyperbolic orbits. Kepler's equation is written as:

$$\mathscr{M} = e\sinh H - H, \tag{3.7}$$

and the mean motion for the hyperbolic orbit is defined by

$$n = \sqrt{\frac{\mu}{-a^3}}, \tag{3.8}$$

so that t is given by

$$t = \sqrt{\frac{-a^3}{\mu}}(e\sinh H - H). \tag{3.9}$$

We compute the hyperbolic anomaly from

$$\sinh H = \frac{\sin\nu\sqrt{e^2-1}}{1+e\cos\nu} \qquad \cosh H = \frac{e+\cos\nu}{1+e\cos\nu}\,, \tag{3.10}$$

where the true anomaly, ν, is found by rearranging

$$r = \frac{p}{1+e\cos\nu}\,, \tag{3.11}$$

to give

$$\nu = \pm\arccos\left(\frac{p}{er}-\frac{1}{e}\right), \tag{3.12}$$

and where p is the semi-parameter:

$$p = a(1-e^2)\,. \tag{3.13}$$

Alternatively, we may employ

$$r = a(1-e\cosh H)\,, \tag{3.14}$$

to solve for H at the intersection of the spacecraft and planet orbits. As before, the parameters above are all determined by the (V_∞, α) at the Tisserand graph node and the radius distance of the encounter (assumed to be the constant radius of the planet orbit). The times of flight may be computed from t in Equation 3.9 and Table 3.2.

3.3.2 Resonant Transfers

As discussed in Chapter 2, the minimum allowable flyby radius creates a maximum possible bending angle, δ. When linking flybys with a Tisserand graph, this constraint limits the V_∞ contours that are reachable from the current Tisserand node. In order to proceed along a given contour to another node, a repeat flyby is required to complete the trajectory turning.

When repeated gravity assists of a given planet are required, the time of flight calculations are more complicated. In this case, we must include integer multiples of the period of the post-flyby orbit until the spacecraft and planet re-encounter. The spacecraft has a *resonant*

orbit with the planet when the periods of the spacecraft and planet have a small integer ratio.

Resonant flybys create a significant complication for the network approach. Chapter 4 explores the methods developed to determine the desirable resonances, calculate the rendezvous times, and add this information to the Tisserand network. Presently, let us just consider how to detect when a resonance is required and how we must alter the network described above.

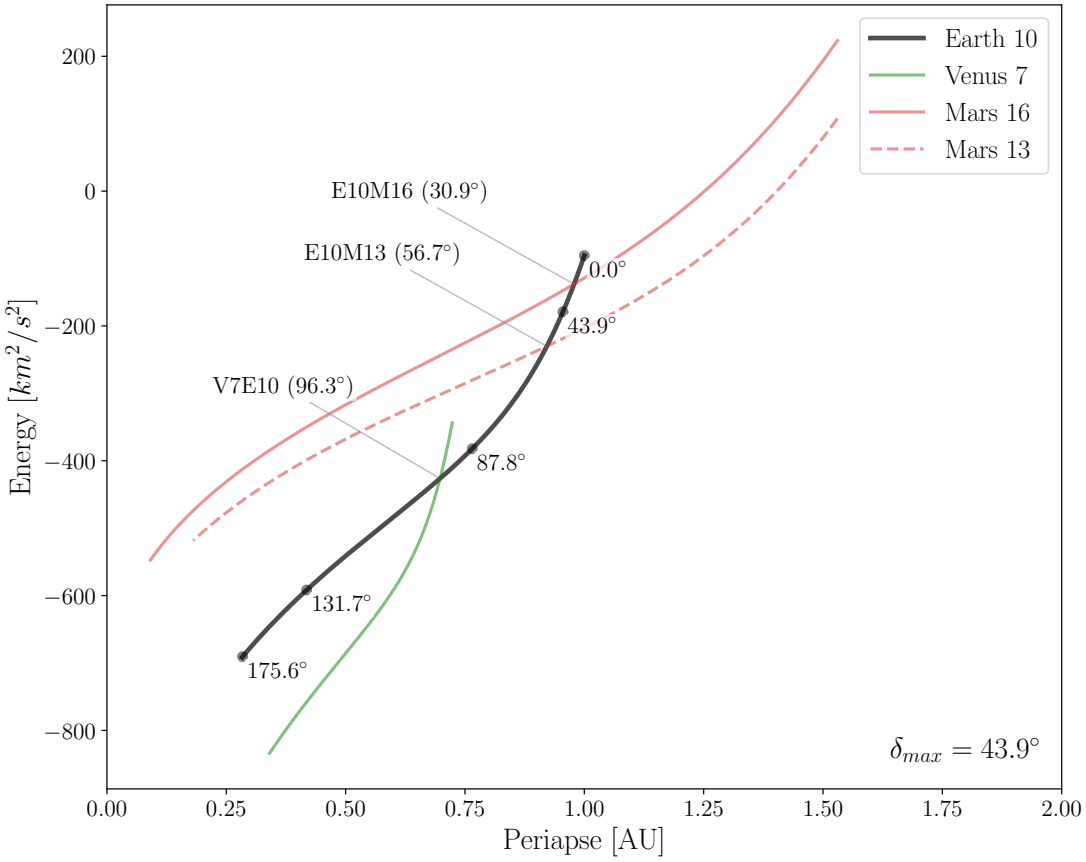


Figure 3.6. The maximum bending angle, $\delta_{\max}$, defines the maximum distance that can be traversed along a V_∞ contour in the Tisserand graph for a single gravity assist. Here, the Earth-10/Mars-16 node can be reached from the Earth-10/Mars-13 node but not from the Venus-7/Earth-10 node.

Figure 3.6 presents a highly simplified Tisserand graph including only four V_∞ contours. In the traditional Tisserand graph, the minimum flyby radius constraint is shown by reference

dots that represent the maximum distance one can travel along a contour before requiring a second pass of the planet in question. These dots are shown on the Earth 10 km s^{-1} V_∞ contour along with the pump angle values at those locations in the figure. For this contour, we have determined that the maximum change in α that can be achieved in one Earth gravity assist is 43.9°. This is the spacing between the dots on the Earth contour (shown for reference only). Suppose we want to reach the Mars 16 km s^{-1} V_∞ contour via the Earth contour (at the E10M16 node). If the Earth contour is first encountered from the Mars 13 km s^{-1} V_∞ contour (E10M13), the pump angle needs to be decreased from 56.7° to 30.9°. This δ of 25.8° can be achieved in a single Earth flyby. Conversely, if the Earth contour is first encountered from the Venus 7 km s^{-1} V_∞ contour (V7E10), the pump angle needs to be decreased from 96.3° to 30.9°. This 65.4 deg change would require two Earth flybys.

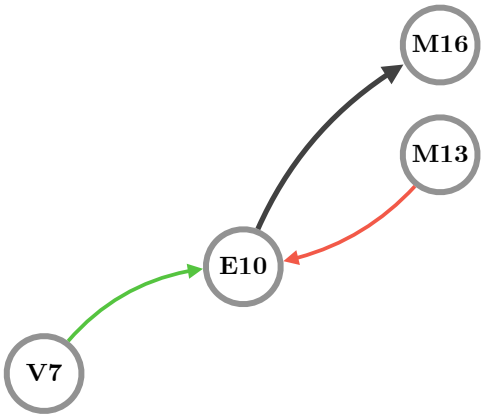


Figure 3.7. A schematic highlighting the resonance problem in the Tisserand network. Edge weights computed from Table 3.2 will not account for additional time required in the resonant orbit. In this example, the sum of the edge weights of the sequence (M13-E10-M16) corresponds to the total time of flight. The sum of the edge weights of the sequence (V7-E10-M16) does not.

Now consider the simplified Tisserand network in Figure 3.7. Suppose we wish to assign the time of flight as the weight of each edge. In this way, we can compute the total time of flight for a path through the network as the sum of the weights of the edges on that path. The network vertices contain enough information so that Table 3.2 can be used to calculate the weights for a single flyby. However, the sum of these weights does not correspond to the total

mission time of flight in the event that a resonance is required. In our example, sometimes the E10-M16 edge needs to include a resonance time (V7-E10-M16), and sometimes it does not (M13-E10-M16).

This nuance is easily resolved by a human reading a Tisserand graph using the reference dots. However, there is no equivalent way to measure distance along the V_∞ contour in the Tisserand network. An automated search algorithm would need to retain some history of the path that had already been traversed in order to determine the weight that must be added on the next step along the way.

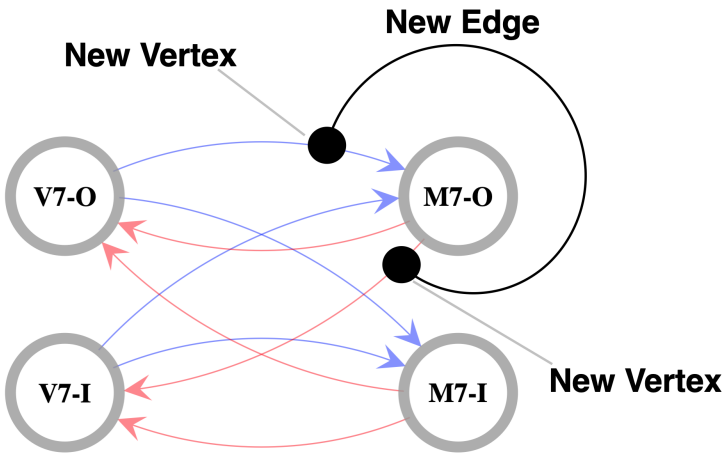


Figure 3.8. The network edges become the vertices of the new line graph. Line graph vertices that are adjacent to the same network vertex are connected by an edge in the line graph. The line graph vertices record the connectedness of the network.

3.3.3 The Line Graph

One solution to the maximum bending angle problem is to embed some history into the network itself. Doing so eliminates the need to write custom algorithms that modify the weight of the next step based on the previous step. We embed history in the network by constructing the *line graph* of our network. In graph theory, a line graph is constructed from a network by setting the edges (or "lines") from the original network as the vertices of a new

graph. The edges of the line graph are then the sets of original edges that were adjacent to common vertices in the network. The procedure is demonstrated in Figure 3.8. Since there are already more edges than vertices in our network, the dimensionality of the line graph can grow large quickly.

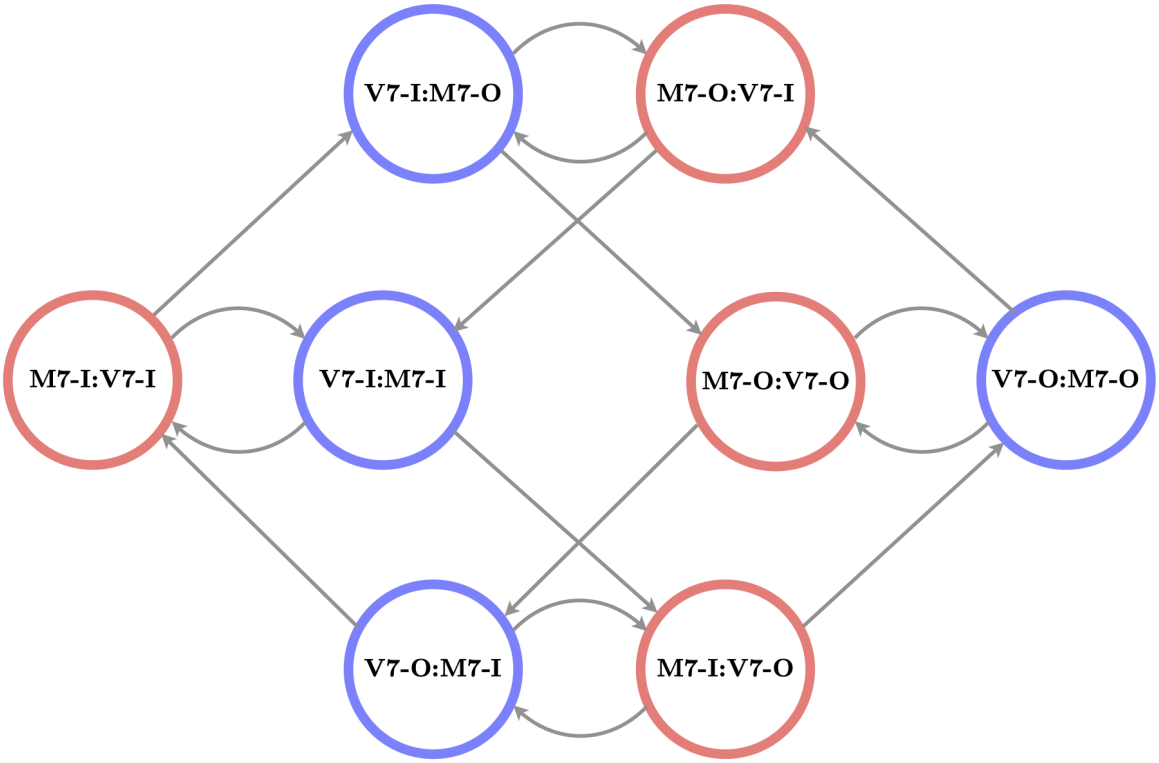


Figure 3.9. The line graph of the small portion of the Tisserand network (a single Tisserand graph node) from Figure 3.8 is visualized above. The red and blue vertices correspond to the red and blue edges in Figure 3.8. Each line graph vertex records how two vertices in the source network are connected. Each edge of the line graph records a relationship between three vertices in the source network.

The line graph of the full Tisserand network is difficult to visualize. The grid format devised for the earlier representations such as Figure 3.5 no longer has a geometrical meaning. Figure 3.9 represents the line graph of the portion of the network corresponding to a single Tisserand graph node shown in Figure 3.8. For our purposes, the key feature of the line graph is that the vertices record partial paths through the original network. We also note

the increase in the number of network elements. In Figure 3.9, the single Tisserand graph node Venus-7/Mars-7 now corresponds to eight line graph vertices and 16 line graph edges.

Suppose we have a sequence of connected vertices in our network E4-I : V7-I : M7-I. When converted to a line graph, this sequence becomes (E4-I, V7-I) : (V7-I, M7-I). We now have two vertices with information about three flybys. In Tisserand graph terms, we now know the distance to be traveled along the Venus-7 contour (namely, from E4 to M7). After constructing the line graph, we must weight the new edges before we can perform searches. However, we now have the information we need to determine whether the time of flight between the two line-graph vertices should include additional full revolutions of the transfer orbit and a second flyby of the middle planet. After completing the weighting procedure, we can use traditional search algorithms to search for paths in the line graph.

3.4 Adding Phasing the Network

The network described above may be automatically searched to identify all paths between any two nodes of the Tisserand graph [55]. The results of such a search would be the names of the planets in the gravity-assist sequence and the V_∞ of each flyby. Such a search automates the manual pathfinding traditionally done on the Tisserand graph [11], [14], [15], [17]. For very broad searches, the pathfinding step alone can require days of human labor performing the graph tracing to enumerate the energy-feasible gravity assist sequences as described in Appendix A.

We have also examined how some of the assumptions used to match the flybys to heliocentric orbits in the Tisserand graph are also sufficient to estimate the time of flight on the resulting transfer arcs. We still lack the capability to predict when the energy-feasible transfers in the Tisserand graph will align with solar system geometry to create an energy-and-time feasible transfer. The ability to identify when the connections in the network actually occur in time is a new and powerful feature that we will now explore. This modification will vastly expand the utility of the Tisserand graph approach and enable the simultaneous solution of the pathfinding and pathsolving problems.

3.4.1 Cataloging Opportunities

An opportunity to perform a gravity assist transfer exists only when the planets involved are properly aligned. The two-body assumptions used to develop the Tisserand graph provide us with the time of flight and transfer angle for any of the heliocentric transfers included in the network. Our assumptions also allow a simple solution to the phasing of the planets that enables the transfer.

Vickery and Horsewood investigated the possible windows for Jupiter flybys to enable missions to the outer planets [56]. Their work complements the Tisserand graph development by identifying time-feasible transfers. They develop a relationship between the Earth-Jupiter alignment date and the required Earth launch date that can be generalized for our problem.

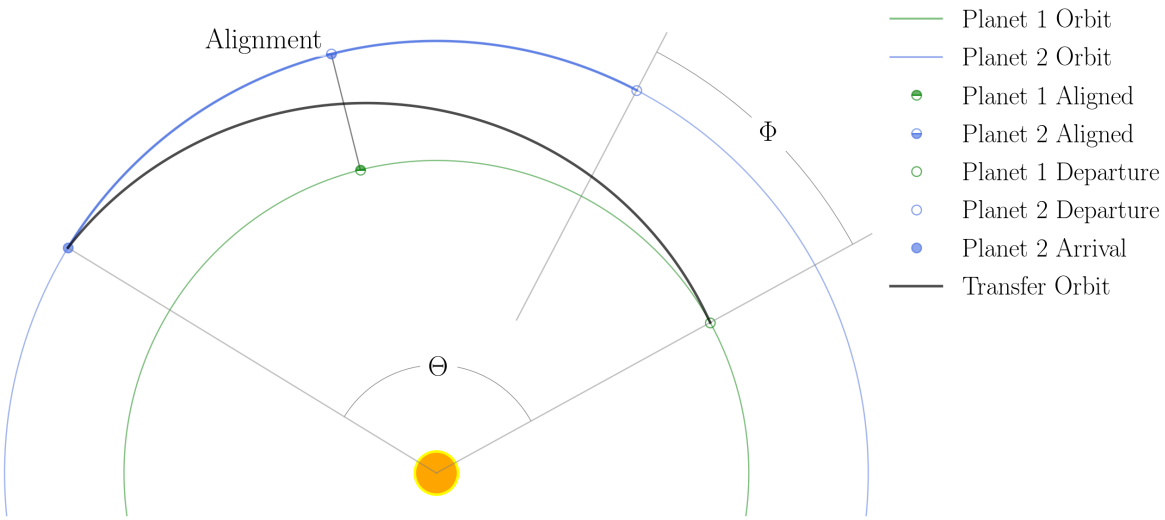


Figure 3.10. The basic geometry of the phasing problem. The alignment date may be used to reference phasing opportunities.

Inspired by the development in Vickery and Horsewood, we will now create a method for generating properly-phased transfer geometries using the date that the planets are aligned, D_{align}, as a key. Here, alignment means that the planets are in conjunction as observed from the Sun.

Referring to Figure 3.10, let Θ be the transfer angle between two planets at which flybys are to be performed. Also, let n_d and n_a be the mean motions of the departure and arrival planets, respectively. For the spacecraft to intercept the arrival planet, the phase angle, Φ, at departure must be the transfer angle, less the angle that will be swept out by the arrival planet during the spacecraft time of flight, t:

$$\Phi_{\text{depart}} = \Theta - n_a t \,. \tag{3.15}$$

The time required to close the launch phase angle (between the planets) depends on the difference in mean motions:

$$t_{\text{align}} = \Phi/(n_d - n_a) \,. \tag{3.16}$$

The date that the two planets are aligned is then:

$$D_{\text{align}} = D_{\text{depart}} + t_{\text{align}} \tag{3.17}$$

$$D_{\text{align}} = D_{\text{depart}} + \Phi_{\text{depart}}/(n_d - n_a) \,. \tag{3.18}$$

Rearranging gives

$$D_{\text{depart}} = D_{\text{align}} - \Phi_{\text{depart}}/(n_d - n_a) \,. \tag{3.19}$$

Substituting the phase angle from (3.15), the departure date, D_{depart}, is given by:

$$D_{\text{depart}} = D_{\text{align}} - \frac{\Theta - n_a t}{(n_d - n_a)} \,. \tag{3.20}$$

The arrival date is found by simply adding the time of flight to the launch date:

$$D_{\text{arrive}} = D_{\text{depart}} + t \,. \tag{3.21}$$

In (3.20), the mean motions are given by our two-body assumptions. Importantly, the transfer angle and time of flight are each functions of the V_∞ and the pump angle, α for a given gravity assist:

$$\begin{aligned} \Theta &= \Theta(V_\infty, \alpha) \\ t &= t(V_\infty, \alpha) \,. \end{aligned} \tag{3.22}$$

Coincidentally, V_∞ and pump angle are the independent variables that define the Tisserand contours. So the transfer angle and time of flight may be readily determined for each edge of the network. The transfer angle can be computed according to Table 3.1 and the time of flight is given by Table 3.2.

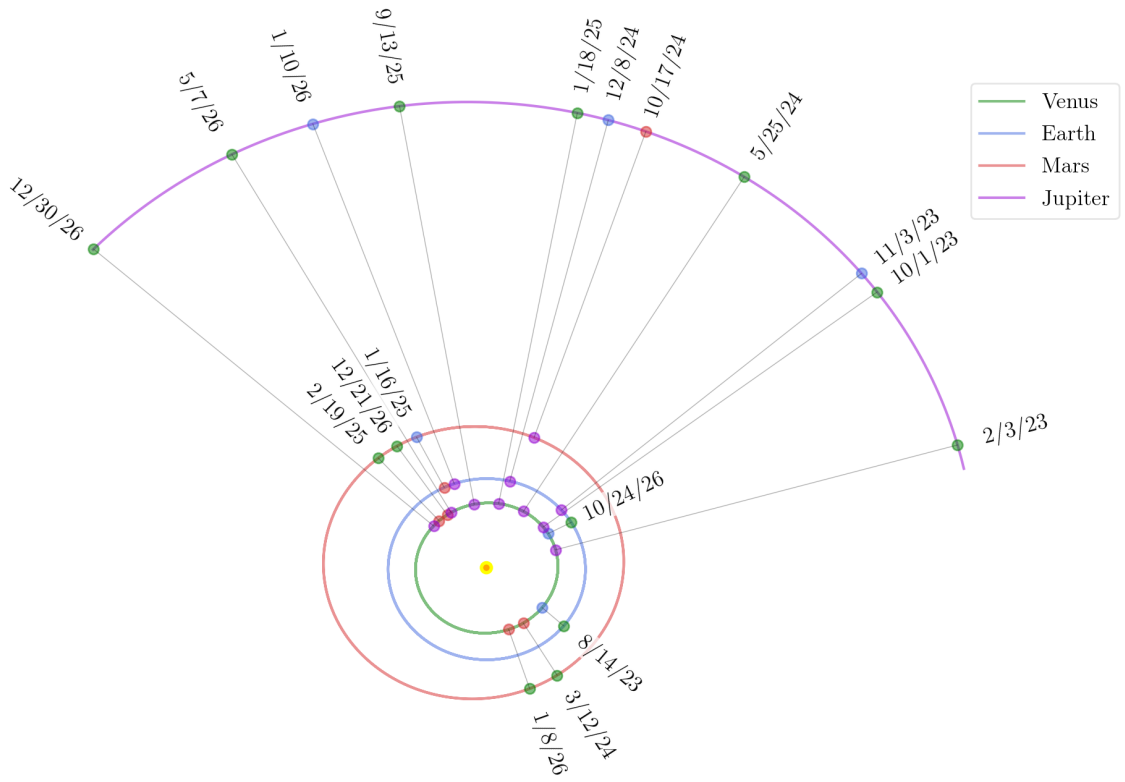


Figure 3.11. The alignment dates of Venus, Earth, Mars, and Jupiter for the calendar years 2023-2026 are visualized in the diagram. The colored dots show the location of each planet during an alignment. If two planets are aligned, the inner planet's color is overlaid on the outer planet, and vice-versa. Each alignment condition acts as the center for a family of transfers with different transfer angles.

For any particular Tisserand network edge, the only unknown in Equations 3.20 and 3.21 is the alignment date, D_{align}. By consulting or creating a catalog of alignment dates (for each

combination of planets in the network), the departure and arrival date(s) can be calculated for the possible transfers in the Tisserand graph. Additional consideration is required for transfers that require repeat flybys (Chapter 4).

A simple method for developing this alignment catalog is to step through a solar system ephemeris and compare the true longitude of each pair of planets. On a historical note, Flandro discovered the Grand Tour using a similar procedure [7]. The ephemeris may be of higher fidelity than the two-body trajectories used elsewhere in this analysis without causing any complication. Figure 3.11 gives an example of the results of such a procedure. The alignments for Venus, Earth, Mars, and Jupiter over the calendar years 2023 through 2026 are displayed in the figure. The set of planets and dates are kept small for illustration purposes. The radial lines in Figure 3.11 identify dates when a pair of planets are aligned. The contrasting color of the dot on a planet's orbit identifies which other planet is in alignment.

3.4.2 Adding Opportunities to the Network

The catalog of alignment dates developed above can be used to compute departure and arrival dates for each edge in the network. A single alignment date will generate many pairs of departure and arrival dates (one pair for each of the multiple transfer arcs at each Tisserand node). Figure 3.12 shows an example of the multiple transfers that can be generated from a single alignment date. The colored arcs represent upward transfers from Venus to Earth for a selection of Tisserand graph nodes. Each transfer arc in Figure 3.12 corresponds to a different combination of V_∞ levels at Venus and Earth. Only the Venus outbound to Earth inbound encounter points are shown. Similar figures could be drawn for both the upward and downward transfers at each encounter point combination, and each pair of planets (each node on the Tisserand graph).

Figure 3.12 considers only one alignment date. Some pairs of planets may be aligned multiple times during the time frame of interest. Each alignment date will generate a similar family of departure and arrival dates.

Figure 3.13 shows a polar view of a subset of the Tisserand network including only Earth, Jupiter, and Saturn for the period 2030 to 2040. The figure also only includes transfers that

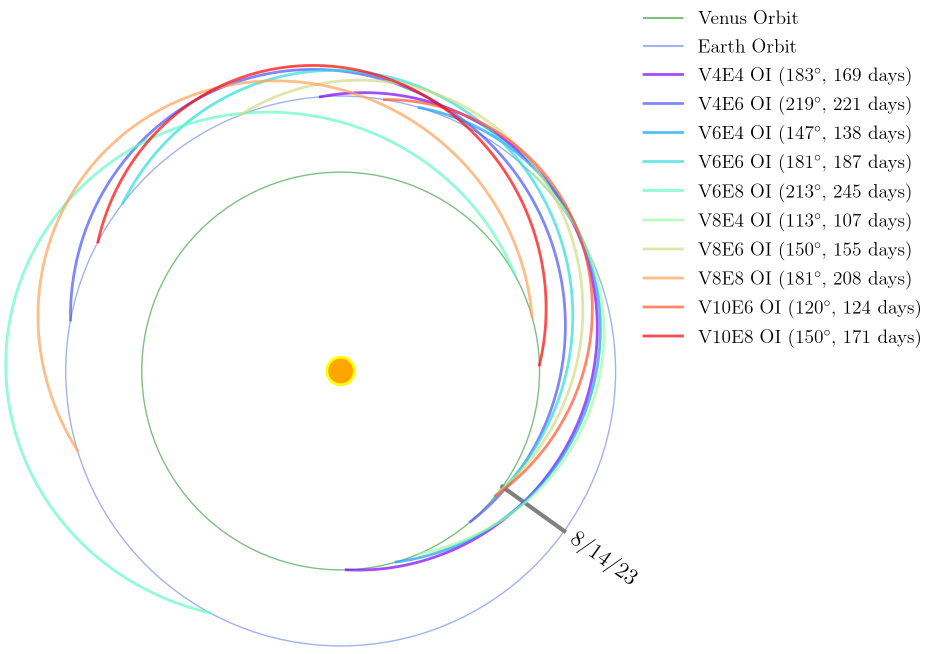


Figure 3.12. A single alignment date generates many discrete transfers. The figure displays a few of the possible transfers near the August 14, 2023 alignment of Venus and Earth. The different transfer arcs correspond to different V_∞ levels at each planet. The figure includes only upward outbound to inbound transfers from Venus to Earth. The variations on the inbound/outbound encounters constitute even more potential transfers.

require less than 5 years of flight time. Even with these limitations, multiple transfers between each pair of planets are visible. These are created from multiple V_∞ combinations (similar to Figure 3.12) at multiple alignment dates.

The edges in the Tisserand network visualized in Figure 3.5 represent the existence of an energy-feasible transfer between the associated vertices. We have just seen that these transfers will repeat at intervals centered around the alignment dates of the two planets. To include these multiple opportunities in the network, we duplicate the edges for each opportunity. In graph theory terms, we create *parallel* edges. Each parallel edge can be encoded with the proper departure and arrival date so that the correct encounter dates can be constructed after paths are discovered by the search.

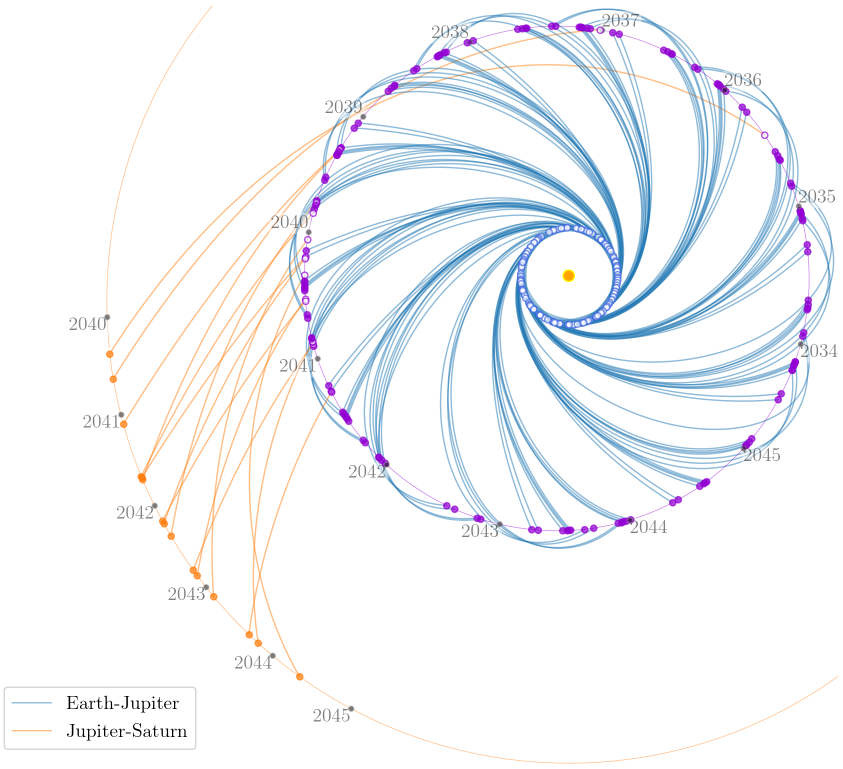


Figure 3.13. A small subset of the 2-planet transfers in the Tisserand network. The figure visualizes the Earth-Jupiter and Jupiter-Saturn network edges with time of flight less than 5 years in the years 2030 to 2040.

Figure 3.14 provides a visualization of the Tisserand network from Figure 3.5 with the parallel edges added for the specific date opportunities between 2030 and 2040. The increase in the number of edges is considerable. Chapter 5 will discuss the effect of edge count on search algorithm performance. However, the addition of the parallel edges allows the Tisserand network to find paths that are both energy-feasible and phasing-feasible.

In the present work, we do not consider the additional edges that might be added if we were to allow the spacecraft to complete some number of complete heliocentric orbits before the encounter with the next planet (except for resonant cases). In theory, the transfer angle in the analysis starting with Equation 3.15 could be re-defined as $\Theta' = 2k\pi + \Theta$. This adjustment would increase the number of edges in the network and potentially increase the number of search solutions. In effect, this change would allow a heliocentric loiter between

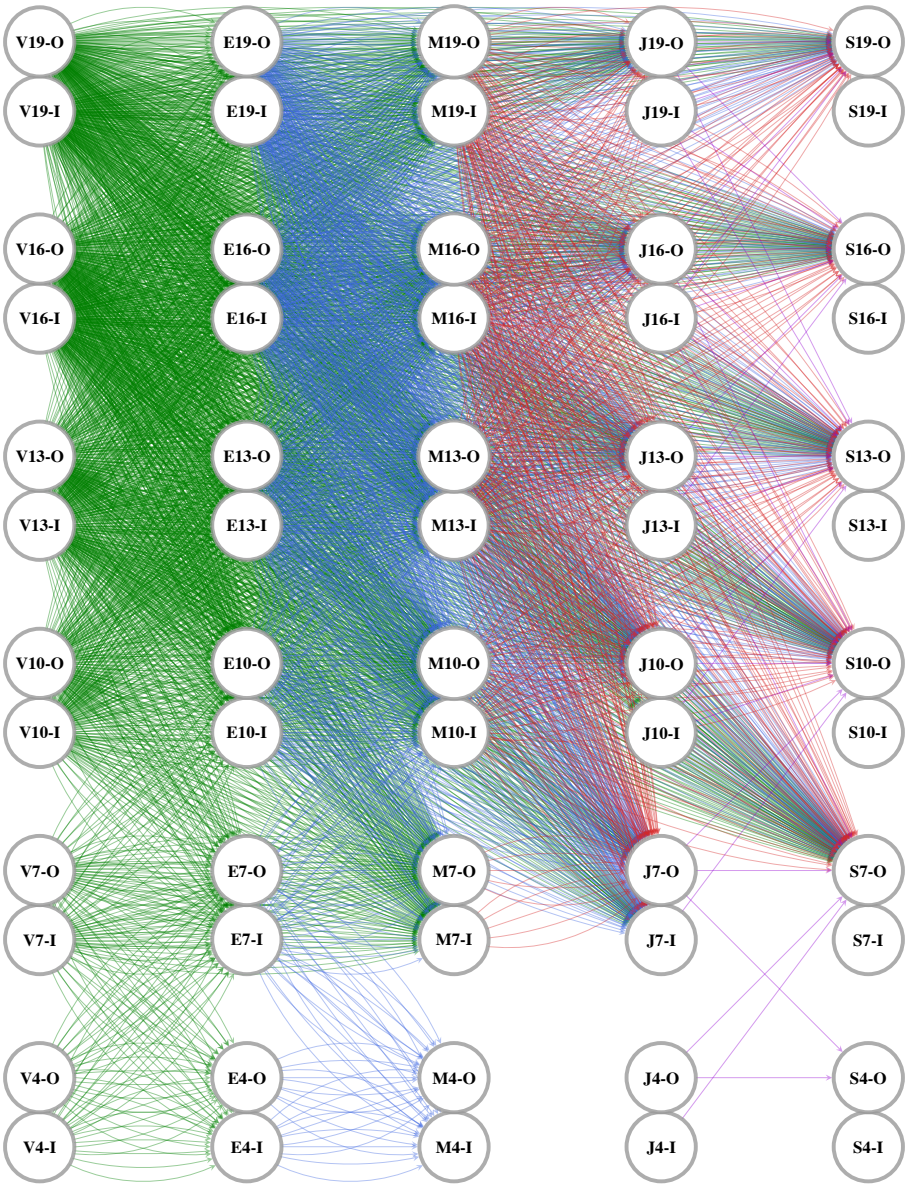


Figure 3.14. A visualization of the Tisserand network mirroring Figures 2.6 and 3.5 for alignment dates between 2030 and 2040. In this image, the color of the edge represents the source planet.

flybys which could provide some scheduling benefit. Presumably, this feature would only be applied in the inner solar system where the heliocentric periods are shorter.

The reader should note the arcs in Figure 3.13 represent only point-to-point transfers between two planets. If we wish to find a gravity assists path that includes multiple flybys, we must find a sequence of two-planet legs such that the arrival date of the first leg matches the departure date for the second leg, and so on. Clearly, in Figure 3.13 many legs do not connect. We also require that the V_∞ of the arrival match the V_∞ of departure on the next leg. The V_∞ connections are shown in Figure 3.14. Our discrete method implicitly requires that the V_∞ match exactly.

3.4.3 Identifying Viable Connections

In order to find paths through the network, we next determine the three-vertex sequences that are feasible. For the sequence A-B-C to be viable, the arrival date for leg A-B must be just prior to the departure date for leg B-C. For example, in Figure 3.13 there appear to be some trajectories that arrive at Jupiter in the early 2040s that might connect to trajectories that depart for Saturn in the same time period. Ideally the arrival and departure at vertex B would be identical. But since we are using a discrete set of V_∞ values in the construction of the network, this is unlikely to be the case.

For a viable connection, we require:

$$\begin{aligned} \mathrm{V}_{\infty,\mathrm{B}}^{-} &= \mathrm{V}_{\infty,\mathrm{B}}^{+} \\ IO_{\mathrm{B}}^{-} &= IO_{\mathrm{B}}^{+} \\ t_B^{-} - t_B^{+} &\leq \varepsilon \,. \end{aligned} \tag{3.23}$$

The first two requirements are satisfied by the way we construct the Tisserand network. We can satisfy the third requirement with a filter. These requirements serve a similar function to accessible regions in other trajectory search frameworks. For simplicity, we neglect cases near the extreme ends of the V_∞ contours where the turning capability would be sufficient to increase the pump angle past 180 deg or decrease it past 0 deg. These cases would allow a reversal in sign of the heliocentric flight path angle and would permit a change from inbound to outbound or outbound to inbound.

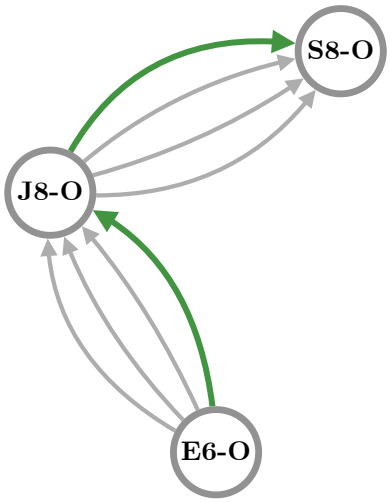


Figure 3.15. A schematic of several line graph edges with date filtering. Each three-vertex set represents an edge in the line graph. The gray connections have been filtered out of the network because the arrival and departure date at the J8-O network vertex exceeds a tolerance.

To identify paths that can be further refined in higher fidelity models, we accept arrival and departure dates that are within a certain tolerance. Sequences of edges that exceed this tolerance can be removed from the network. Returning to Figure 3.13, we might remove any Earth-Jupiter edges that arrive at Jupiter before 2040 or after 2042.

Since this analysis is concerned with the connections between edges, it will need to be performed in the line graph of the network. Figure 3.15 is a schematic of the date filtering on some potential Tisserand network edges in Figure 3.13. These network edges are equivalent to vertices in the line graph. In this example, there are several line-graph vertices (shown as arrows) representing E6-O—J8-O and J8-0—S8-0. The discrete nature of the Tisserand network guarantees that the gravity assist at Jupiter matches identically in V_∞ (8 $\mathrm{km\,s^{-1}}$) and encounter location (outbound). However, we must apply a filter to remove any line graph edges where the arrival and departure dates at Jupiter are so far apart that the connection is impractical. In the schematic, only one edge (in green) is preserved in the filtered Tisserand network.

The end product of this procedure is a network model of the possible gravity-assist sequences that are roughly feasible in both orbital energy and phasing conditions. The network can now be searched for multi-gravity-assist paths.

3.5 Searching the Network

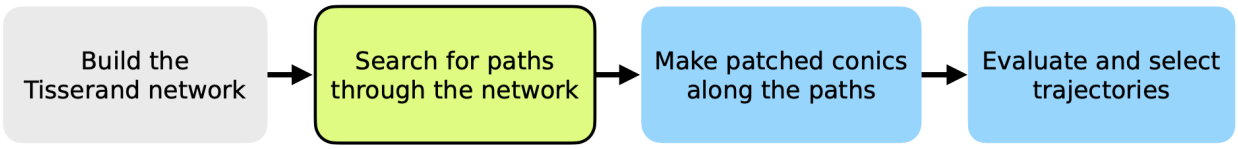


Figure 3.16. The process of finding gravity-assist trajectories with the Tisserand network includes four major steps. The second step applies network search algorithms to find gravity-assist paths.

Figure 3.16 provides an updated look at the Tisserand network trajectory search process. Now that we have constructed the network, we can search for possible paths. The network described in the previous sections may be organized in a data structure that records the vertices and weighted edges so that search algorithms may be deployed. Chapter 5 explores some search methods in detail. The graph theory and combinatorial optimization literature includes several well-established search algorithms that are suitable for different goals.

The gravity-assist paths in the present work come from an "all-paths" search. The goal of such a search is to identify all paths through the network that leave at the source vertex and end at the target vertex. For this research, the source will always be an Earth vertex, but this is not a requirement of the technique.

Trajectory searches using graph methods sometimes employ a "shortest-path" search [29]. This technique will find the path between the source and target vertices that uses the fewest vertices. In a weighted network, the shortest-path search will find the path with the lowest total weight. A shortest-path method such as Dijkstra's algorithm is generally much faster than an all-paths search. With clever algorithm design, one is not required to first find all paths in order tell which is shortest.

If properly configured, a shortest-path search in the Tisserand network will return a path with the lowest cumulative time of flight over the edges making up the path. However, because the Tisserand network paths are discontinuous, the total edge weight is not a reliable predictor of the actual total mission duration for a similar patched-conic trajectory. A k-shortest paths search might be more useful for time-of-flight comparisons of the possible routes in relative terms only. However, for the present research we will rely on all-paths techniques.

The target vertex for the all-paths searches discussed here will always be at a planet other than the source vertex. This is not a requirement of the Tisserand network method. The Tisserand network could be used to find round-trip paths or cycler trajectories. However, the searches presented in this research will mainly be concerned with one-way trips to the outer solar system.

Algorithm 1: Outer Loop Network Search

```
Data: TN: a Tisserand network
      destination: a destination body
Algorithm NetworkSearch(TN, destination):
    paths ← [ ]
    foreach v1 at Earth do
        foreach v2 at destination do
            newpaths ← AllPaths(TN, v1, v2)  ▷ Find paths between these vertices
            paths.insert(newpaths)
        end
    end
end
```

The `AllPaths` algorithm presented in Chapter 5 finds paths between pairs of vertices. The vertices in the Tisserand network have specific departure and arrival V_∞ levels. To perform broad searches we will consider multiple V_∞ levels for both departure and arrival. To achieve this we nest the two-point, all-paths search in loops that iterate on the desired departure and arrival options as demonstrated in Algorithm 1. The `AllPaths` search algorithm at the center of Algorithm 1 is self-contained so the outer loops are good targets for parallel computation.

The search will benefit from limiting the endpoints in the outer loops. The *transitive closure* of the network identifies which pairs of vertices are connected by at least one path. This topic will be examined in more detail in Chapter 5 but clearly, we need only search between endpoints where we know some path exists. If practical limitations on the launch and capture V_∞ are known, then these vertices can also be removed from the possible endpoints.

3.6 Examining Search Results

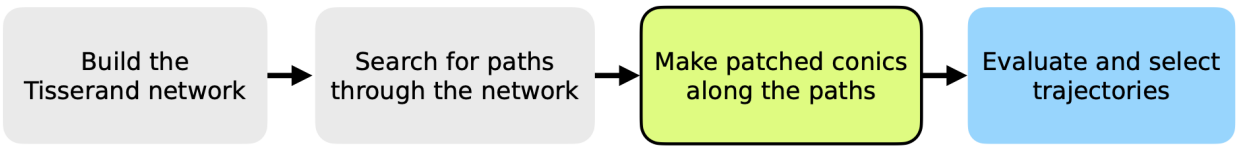


Figure 3.17. The process of finding gravity-assist trajectories with the Tisserand network includes four major steps. The third step creates patched-conic trajectories from the search results.

Figure 3.17 recalls our progress through the Tisserand network trajectory search process. Having identified some paths through the network with a search algorithm, we now explore the results.

The all-paths search described by Algorithm 1 will identify all sequences of network vertices between the departure and destination planets that can be connected through intermediate edges and vertices. Depending on the network data structure implementation, each discovered path may simply be a list of data structure addresses. These addresses can be mapped to the Tisserand network vertices and edges they represent to create a list of vertices for each path.

For demonstration purposes, suppose we have built a network including Earth, Jupiter, Saturn, and Uranus and executed a search for paths from Earth to Uranus. A grid representation of the Tisserand network and the search results are shown in Figure 3.18. Table 3.3 summarizes the paths discovered.

A network route is a list of network vertices found by the search, for example [E11-O, J12-O, S9-O, U12-O]. This list defines the planet, the V_∞, and the location in the heliocentric

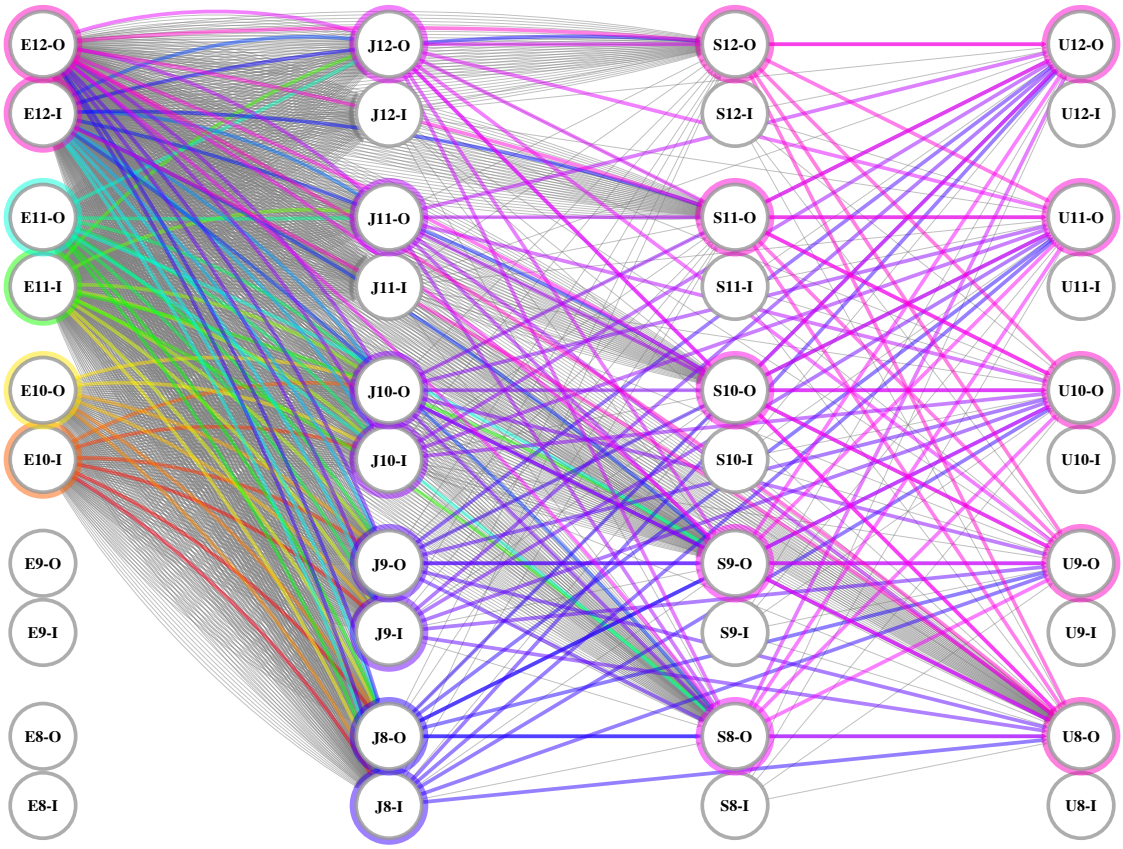


Figure 3.18. The routes found by an all-paths search for Uranus are highlighted in a grid representation of a Tisserand network. Colored edges are used in the solution paths. Gray edges are not part of a successful path. This visualization is provided to illustrate the many network connections throughout the search space and is not intended for tracing solutions. For color viewing or inspection of the vertex labels, the reader is referred to the electronic version of this document.

orbit of each encounter. Table 3.3 lists some randomly selected example routes for each path. There may be multiple variants of each route corresponding to different dates for one or more encounters. These are listed as date variants in Table 3.3.

The particular edges that were used to connect the vertices are also included in each route discovered by the search. These edges identify the departure and arrival dates of each leg (E11-O to J12-O, J12-O to S9-O, etc.). The information contained in this sequence of flybys and dates is sufficient to plot each two-body trajectory leg. Figure 3.19 shows a polar

Table 3.3. Example Uranus Search Summary

Path	Network Routes	Date Variants	Route Examples
JSU	156	346	E11-I, J10-O, S9-O, U12-O E12-I, J11-O, S8-O, U11-O E11-O, J12-O, S9-O, U10-O E12-O, J12-O, S9-O, U11-O E12-I, J11-O, S9-O, U11-O
JU	205	429	E11-I, J10-O, U12-O E10-I, J9-O, U12-O E12-I, J12-O, U9-O E10-O, J10-I, U11-O E11-O, J8-I, U10-O
SU	66	381	E12-O, S9-O, U9-O E12-I, S9-O, U11-O E12-O, S9-O, U9-O E12-I, S11-O, U11-O E12-O, S11-O, U12-O
U	2	44	E12-I, U8-O E12-O, U8-O
Total	429	1200	

view of the trajectories in the search results. In this visualization, each trajectory segment is a three-dimensional Lambert solution between the SPICE [57] ephemeris positions of the two planets on the network departure and arrival dates. The routes have been grouped into the families (or paths) JSU, JU, SU, and U (direct) based on the sequence of planets visited.

Like the Tisserand graph, the Tisserand network includes discrete values of V_∞ for the flyby of each planet. Excluding, for the moment, powered flybys, the exit V_∞ at each vertex matches the entrance V_∞ identically. Furthermore, as discussed in Section 3.2.1, there are discrete times of flight available at each node of the Tisserand graph. A consequence of this discrete architecture is that the flight segment between two vertices in the network will not necessarily align in time with the subsequent segment. For a three-planet sequence, the V_∞ in and out of the middle planet will match, but the encounter date typically will not. This

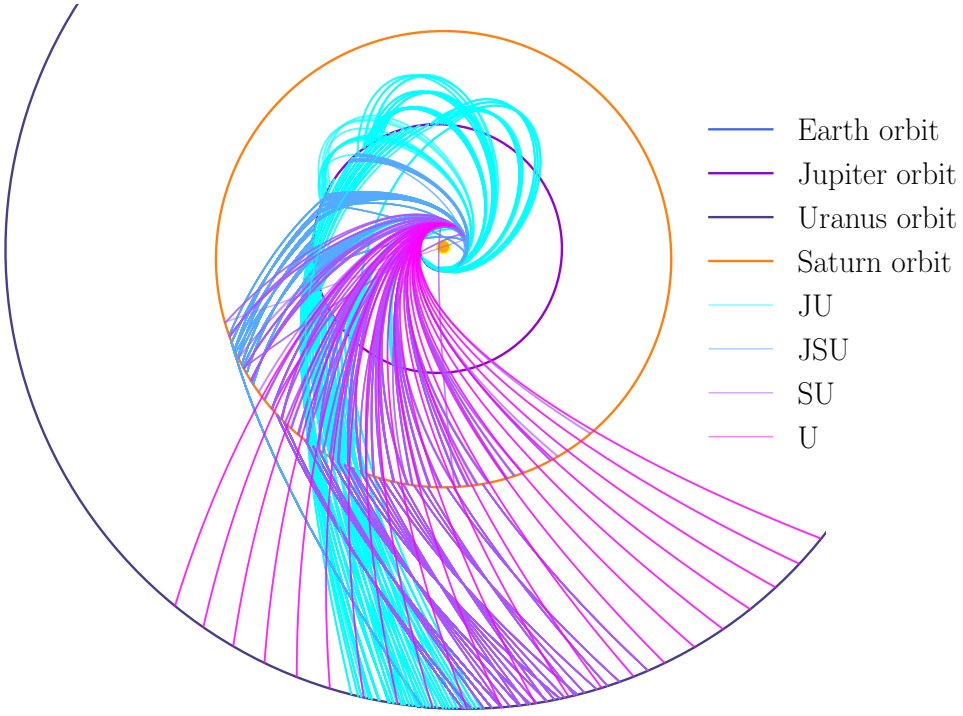


Figure 3.19. A polar view of the demonstration all-paths search for Uranus paths shows many potential solutions. Results can be grouped into traditional path categories (JU, JSU, SU, and U) based on the sequence of planets visited. Each path includes discontinuities arising from the discrete dates and V_∞ levels. Each trajectory segment is a three-dimensional Lambert solution between two planet locations taken from a SPICE ephemeris on the dates from the Tisserand network.

condition is readily visible in Figure 3.20 and will be discussed in more detail in Section 3.6.2.

Therefore, a path through the Tisserand network might be considered a "broken conic" solution rather than a patched conic. Depending on the tolerance for date mismatches chosen while filtering the network, the broken-conic solutions may be close enough to a closed trajectory to serve as an initial guess for higher fidelity pathsolver. But even in the case of large encounter date mismatches, the network solutions provide a sketch of a family of patched conic solutions.

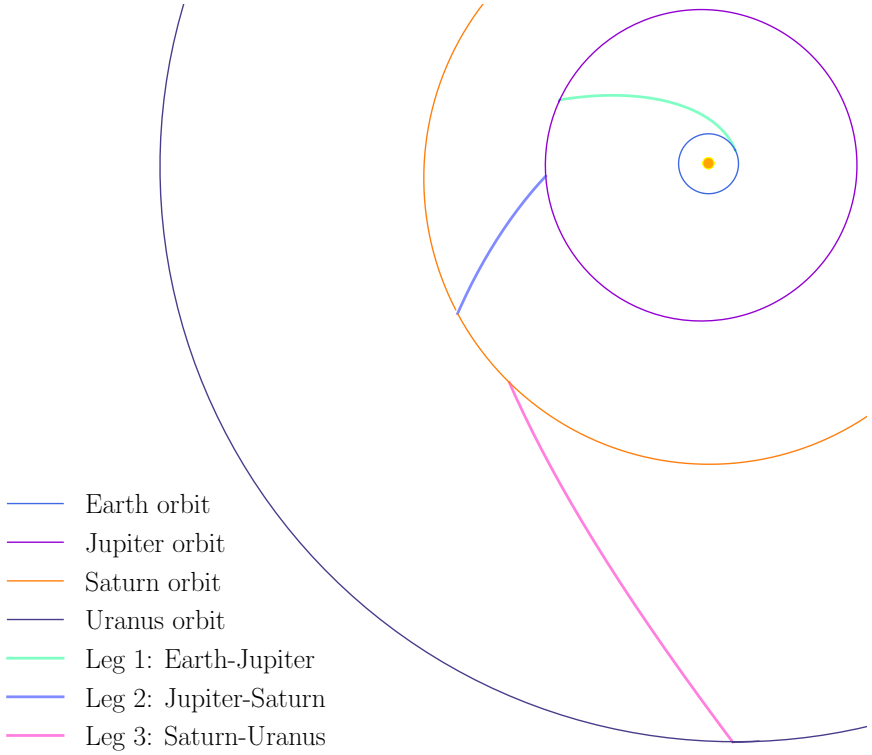


Figure 3.20. The vertices in a Tisserand network path identify transfer arcs between two planets that may not share an endpoint with the adjacent transfers.

3.6.1 Review of the Contribution Thus Far

At this point, the output of the search exceeds the original intent of the research. That goal was to provide automated, initial guesses for a patched-conic trajectory solver such as STOUR.

A manual analysis of the Tisserand graph can solve the pathfinding problem. For our demonstration search, this technique would have produced a path like J-S-U (Jupiter, Saturn, Uranus). With additional commitment, a more detailed path such as J12-S9-U12 could be found manually. This path could then be used to configure a patched-conic solver over a grid of launch and encounter dates determined independently through intuition or experience.

The Tisserand network results additionally supply the approximate dates of the encounters and their locations in the heliocentric orbit. These details can further constrain the

breadth of searches in grid solvers to areas where we have higher confidence in the existence of trajectories. The results presented above are suitable first guesses for established grid search tools. However, we can take further steps to create patched-conic trajectories within the Tisserand network framework.

3.6.2 Patching the Broken Conics

The discontinuities in network paths complicate efforts to characterize and evaluate paths against one another. Consider how we might compute the total time of flight for the network path shown in Figure 3.20. One possibility might be to sum the durations of the individual legs (Earth to Jupiter, Jupiter to Saturn, Saturn to Uranus). This sum provides an approximate lower limit on the total time of flight. Another method could be to compute the time between the Earth departure date and the Uranus arrival date. These methods would result in different times of flight, and neither fully characterizes the variety of flight times for trajectories that can be derived from the basic path.

This ambiguity limits the conclusions that may be drawn from optimal search methods that rely on the cumulative weight of the path (e.g., Dijkstra's algorithm). Section 5.1.5 includes more discussion on this problem and a possible resolution is provided as part of the future work in Chapter 8.

Let us now consider some means of constructing patched-conic trajectories from the broken trajectories identified by the network search. This additional step will permit a more meaningful comparison between the paths identified by the network search. Recall that Figure 3.20 shows the disconnected transfer arcs that make up a single network solution from Earth to Uranus.

The routes returned by the network search are a series of transfers that match identically in V_∞. In general, however, the ending date of one leg will not match the starting date of the next leg. In Figure 3.20, the sample network path shows gravity assists linking Earth to Jupiter, Jupiter to Saturn, and Saturn to Uranus. At each encounter $V_\infty^- = V_\infty^+$. However, the figure clearly shows that the departure and arrival at each planet would occur on different dates.

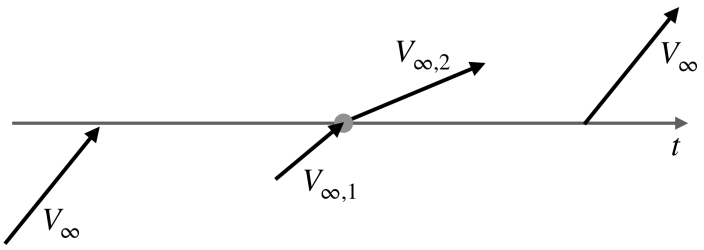


Figure 3.21. At each encounter in a Tisserand network path, the arrival and departure $\mathbf{V}_\infty$ vectors are identical but they encounter the planet at different times (left and right). When the trajectories are patched with a common encounter time (center), the $\mathbf{V}_\infty$ vectors are no longer necessarily equal.

Each discontinuous gravity-assist path from the network search (Figure 3.19) approximates a set of closed patched-conic solutions. Consider a single discontinuous path discovered by the network search (Figure 3.20). For the intermediate planets, there will be two encounter dates: one arriving from the earlier leg and one departing on the subsequent leg. We can construct the patched-conic trajectories by forcing the encounter to occur on a common date. Figure 3.21 displays this problem schematically. For simplicity, let's assume that we select the midpoint of the two dates at each encounter. The V_∞ of the encounter is no longer the value at the associated Tisserand network vertex. This action resolves a difference in time of the encounter but creates a difference in V_∞ at the encounter.

We may now require a propulsive ΔV to account for the difference in V_∞. The positions of the planets at each encounter in the path may be determined from an ephemeris such as the SPICE library using the selected dates. The known positions and encounter times define a Lambert problem between each gravity assist that can be solved with a choice of algorithms [58]. For the patched-conic problem, the velocities at the endpoints of the Lambert solution provide the departure V_∞^+ and arrival V_∞^- for the gravity assists at the endpoints of each trajectory segment. So starting from a series of arbitrary encounter dates near the Tisserand network solution, we can compute the $\mathbf{V}_\infty^-$ and $\mathbf{V}_\infty^+$ at each encounter.

In the general case, the magnitude of the inbound and outbound $\mathbf{V}_\infty$ are no longer equal. The Lambert-solved $\mathbf{V}_\infty^-$ and $\mathbf{V}_\infty^+$ correspond to different hyperbolic orbits about the planet

that must be patched together by a propulsive maneuver. The most efficient location for this maneuver is at the periapsis of the flyby hyperbola.

The following iterative procedure patches the inbound and outbound hyperbolas by finding a common periapsis radius and computing the required velocity change [24], [59].

The semi-major axes of the inbound and outbound hyperbolas are found by rearranging the *vis-via* equation and evaluating at $r = r_\infty$:

$$a_{\text{in}} = -\frac{\mu_{\text{planet}}}{V_{\infty,\text{in}}^2} \tag{3.24}$$

$$a_{\text{out}} = -\frac{\mu_{\text{planet}}}{V_{\infty,\text{out}}^2}\,, \tag{3.25}$$

where μ_{planet} is the gravitational parameter of the flyby body. We require that the periapsis radii of the inbound and outbound hyperbolas be equal:

$$r_{\text{p}} = a_{\text{in}}(1 - e_{\text{in}}) = a_{\text{out}}(1 - e_{\text{out}})\,, \tag{3.26}$$

where the eccentricities, e_{in} and e_{out}, that result in a matching periapsis distance are unknown.

The bending angle between the $\mathbf{V}_\infty$ vectors is found from their dot product:

$$\cos\delta = \frac{\mathbf{V}_\infty^- \cdot \mathbf{V}_\infty^+}{|\mathbf{V}_\infty^-||\mathbf{V}_\infty^+|}\,. \tag{3.27}$$

The total bending angle is also the sum of the a half angles from the inbound and outbound hyperbolas:

$$\delta = \frac{\delta_{\text{in}}}{2} + \frac{\delta_{\text{out}}}{2} \tag{3.28}$$

$$\delta = \arcsin\left(\frac{1}{e_{\text{in}}}\right) + \arcsin\left(\frac{1}{e_{\text{out}}}\right). \tag{3.29}$$

To construct an iterative solution, we first isolate e_{in} in Equation 3.26

$$e_{\text{in}} = \frac{a_{\text{out}}}{a_{\text{in}}}(e_{\text{out}} - 1) + 1\,. \tag{3.30}$$

By substituting e_{in} into Equation 3.29 and rearranging, we arrive at

$$f = \left(\frac{a_{\text{out}}}{a_{\text{in}}}(e_{\text{out}} - 1)\right)\sin\left(\delta - \arcsin\left(\frac{1}{e_{\text{out}}}\right)\right) - 1 = 0\,, \tag{3.31}$$

which is a function of the single independent variable, e_{out}. All other parameters are determined by the $\mathbf{V}_\infty$ derived from the Lambert solutions for the arriving and departing heliocentric legs. Equation 3.31 can be solved with a root-finder and the resulting common periapsis radius is then found using Equation 3.26. Finally, the required ΔV is the difference in velocity at periapsis on the two hyperbolas:

$$\Delta V = \left| \sqrt{V_{\infty,\text{in}}^2 + \frac{2\mu_{\text{planet}}}{r_{\text{p}}}} - \sqrt{V_{\infty,\text{out}}^2 + \frac{2\mu_{\text{planet}}}{r_{\text{p}}}} \right| . \tag{3.32}$$

A less-precise, but non-iterative method of computing the required ΔV from the inbound and outbound $\mathbf{V}_\infty$ vectors may be suitable for some applications and is included in Appendix B.

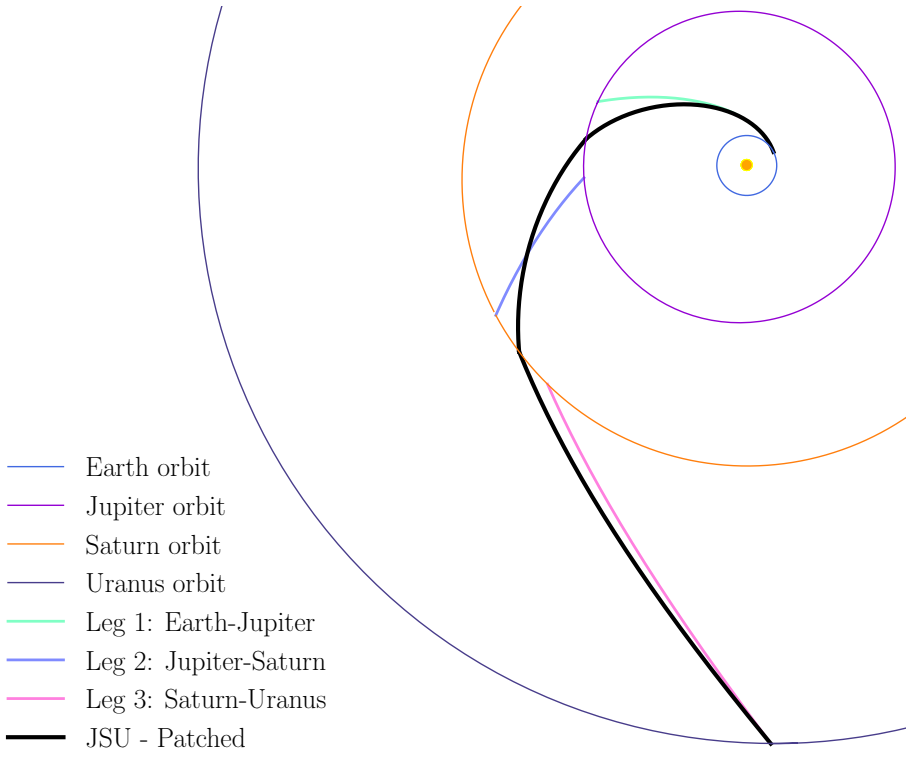


Figure 3.22. The black line indicates a patched-conic trajectory derived from a Tisserand network path. The midpoints of the split encounter dates from Figure 3.20 are chosen as the common encounter date at each planet. Forcing the common encounter date creates a V_∞ mismatch at the gravity assist.

The creation of the patched-conic trajectories from the Tisserand network search results can be summarized as follows. The pair of dates at each encounter in the Tisserand network path are replaced with a single encounter date. This sets up a series of $n-1$ Lambert problems for the n-planet path. The Lambert problem solutions provide the pre- and post-encounter $\mathbf{V}_\infty$. Equations 3.24 through 3.32 use the $\mathbf{V}_\infty$ vectors to compute the ΔV required to close the V_∞ gap. Figure 3.22 shows a patched conic resulting from this procedure. In this case, the midpoint of the encounter dates from Figure 3.20 were chosen as the patch points.

For many choices of encounter dates, the propulsive ΔV may be impractically large. Some methods for finding the encounter dates such that the patched trajectories are nearly ballistic are explored in the following sections.

Choosing the Encounter Dates

To directly compare Tisserand network paths, we would like to choose a set of encounter dates so that the resulting patched conic fully characterizes a particular Tisserand network solution. Unfortunately, the $\mathbf{V}_\infty^-$ and $\mathbf{V}_\infty^+$, and therefore the ΔV vary considerably depending on the selected encounter date. Referring to Figure 3.21, there is no single encounter date that is more representative of the scheduling options at any gravity assist. Similarly, the patched conic found in Figure 3.22 is not the only one that could be generated from the network path. We must choose a family of patched-conic trajectories that are representative—or derived from—each network solution.

A possible method for constructing this family is to simply choose m evenly-spaced dates within the gap between the split dates from the network path. With m encounter dates for each gravity assist, we can construct a full-factored set of patched-conic trajectories. This procedure will produce m^n trajectories, where n is the number of flybys. However, this approach creates a large number of trajectories, many of which differ by only one encounter date at one of the gravity assists. Each patched trajectory requires $n-1$ Lambert solutions resulting in a total of $(n-1)m^n$ Lambert problems. To efficiently compute a set of trajectories

that represent the network solution, we might construct a method that avoids repeating already-solved Lambert problems.

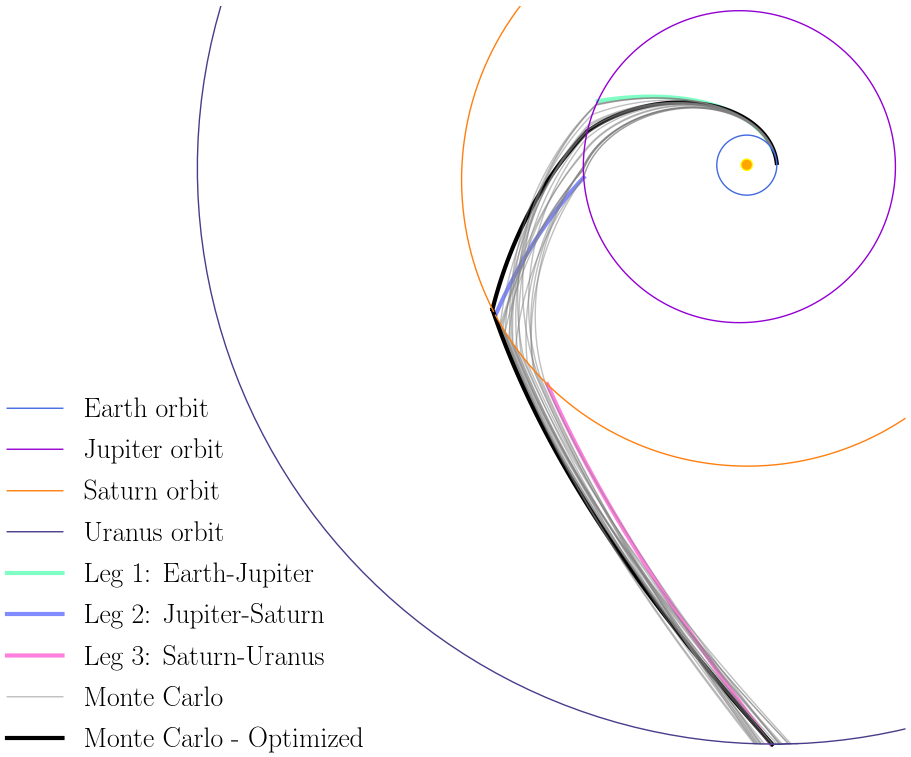


Figure 3.23. The gray lines are random patched-conic trajectories derived from a single Tisserand network route. A random encounter date is chosen between each of the split encounter dates from Figure 3.20 and the gravity assist is required to occur on this date. Twenty variations on the encounter dates are shown in this example. The black line is the common result of optimizing each variation to minimize ΔV.

However, a simpler method that efficiently covers the full region bracketed by the Tisserand network path is to create just m patched-conic trajectories where the m encounter dates for each gravity assist are chosen randomly from within the Tisserand network encounter window. This Monte Carlo approach requires the solution of just $(n-1)m$ Lambert problems.

For a given alignment date, the variation in arrival and departure date at a particular planet is determined by the discrete V_∞ levels chosen for the Tisserand graph on which the network is based. Therefore, we may want to expand the allowable encounter dates beyond

the window strictly defined by the Tisserand network path so that similar trajectories are also included. Figure 3.23 displays an example set of 20 patched-conic trajectories created using this method. Here we have selected the intermediate gravity-assist dates from strictly within the Tisserand network date windows. But, we have allowed the launch and arrival dates to vary by 30 days and one year, respectively.

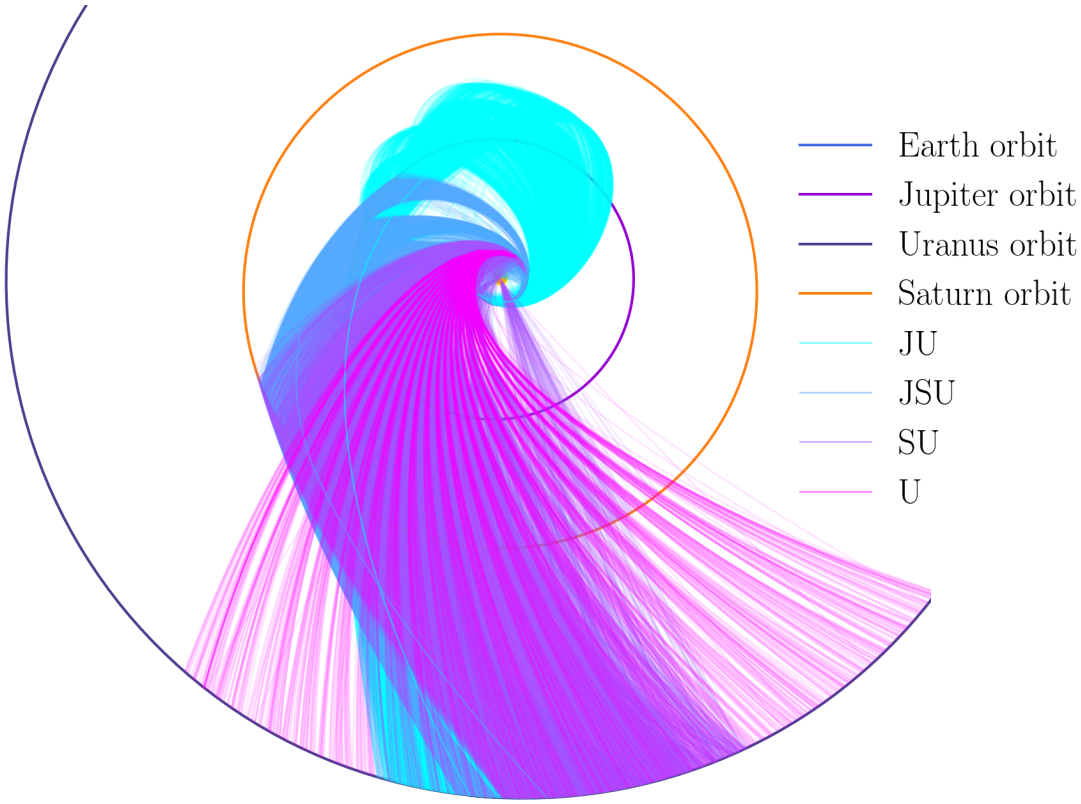


Figure 3.24. The network solutions serve as guidelines for closed patched-conic trajectories. Each trajectory segment is a three-dimensional Lambert solution between two planet locations randomly selected within windows informed by the Tisserand network routes. Compare with Figure 3.19.

The procedure just discussed creates n patched-conic trajectories in the neighborhood of each Tisserand network route. In Figure 3.24, each route from Figure 3.19 has been used to generate 20 random trajectories.

The Monte Carlo approach for generating patched-conic trajectories from the Tisserand network solutions is quick but the randomized dates can result in extremely large and impractical Delta-Vs. While these cases are easily filtered-out, an alternative method for generating

patched-conic trajectories would be a targeted grid search and C_3-matching procedure similar to those used in STOUR [8] or GREMLINS [17]. The grid search can be confined to small time and V_∞ windows based on the Tisserand network solution. The implementation of this idea is left for future researchers.

3.6.3 Optimizing Solutions

The randomly chosen encounter dates in the previous section will result in some encounters with high propulsive ΔV. The following procedure can be used to minimize the propulsive ΔV.

For a path with n planets, we select a vector containing the n encounter dates. We solve the Lambert problem in each leg of the path and compute the propulsive ΔV at each gravity assist according to Equation 3.32. The total mission ΔV is the sum of the propulsive ΔV at each encounter.

Now let us allow each encounter date to vary within bounds informed by the network search and recompute the total mission ΔV. We can repeat this process until the ΔV is minimized. The process above can be formalized as an optimization problem. Let us represent the encounter dates (including the launch and arrival) as a vector, $\mathbf{x}$, of n Julian dates. Then we have the minimization problem:

$$\begin{aligned}
\underset{\mathbf{x} \in \mathbb{R}^n}{\text{minimize}} \quad & J = \Sigma_{i=1}^{n-2} \Delta V_{\text{prop}} && \text{(3.33a)} \\
\text{subject to} \quad & x_i < x_{i+1}, && \text{(3.33b)} \\
& |x_i - \Delta x_{\max}| \leq 0, && \text{(3.33c)} \\
& \Delta V \leq \Delta V_{\max}, && \text{(3.33d)} \\
& C_3 \leq C_{3,\max} && \text{(3.33e)}
\end{aligned}$$

where ΔV_{prop} is computed from Equation 3.32. The bounds on how widely the encounter date may vary (Equation 3.33c) and the constraint on ΔV (Equation 3.33d) are optional. If the launch date, x_0, is allowed to vary then we may also wish to impose a limit on allowable launch energy (3.33e). The launch energy can be recast as a launch ΔV if a specific Earth

parking orbit is assumed. The objective function may be expanded to include a capture ΔV if needed. The bound in Equation 3.33b is simply protection against a gravity assist occurring before its predecessor. Each iteration will require the solution of $n-1$ Lambert problems.

Returning to Figure 3.23, the thick black line is the result of solving the ΔV minimization problem for each of the 20 Monte Carlo variations in gray. In this example, each Monte Carlo trajectory optimized to the same trajectory. This result suggests a hybrid optimization problem may be successful at producing nearly ballistic solutions. The Tisserand network search identifies discrete locations in the problem space where solutions are likely to exist. The continuous optimization problem in Problem 3.33 then identifies the optimal trajectories in each discrete configuration. The continuous optimization problem may be initialized with a few randomized patched-conic trajectories to prevent against finding only locally optimal trajectories. Alternatively, Monotonic Basin Hopping could be applied in problem 3.33 using a single initial patched-conic trajectory.

3.7 Evaluating Solutions

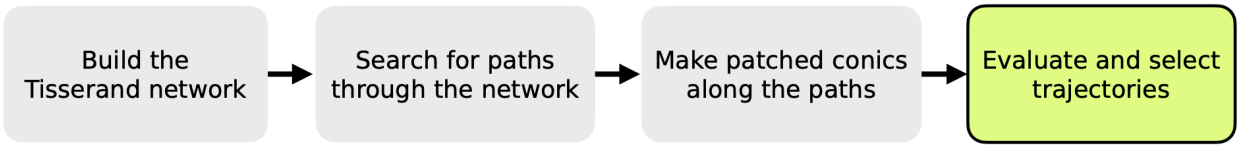


Figure 3.25. The process of finding gravity-assist trajectories with the Tisserand network includes four major steps. The final step reduces the set of patched-conic trajectories to the most interesting candidates.

Figure 3.25 shows the completion of the Tisserand network trajectory search process. Each Tisserand network route may be used to generate many patched-conic trajectories. Here we will briefly review some methods for selecting the most interesting trajectories for detailed study.

For a real mission with multiple trajectory attributes to be optimized, the Monte Carlo trajectories can be evaluated to establish a Pareto frontier. Lacking a specific mission goal

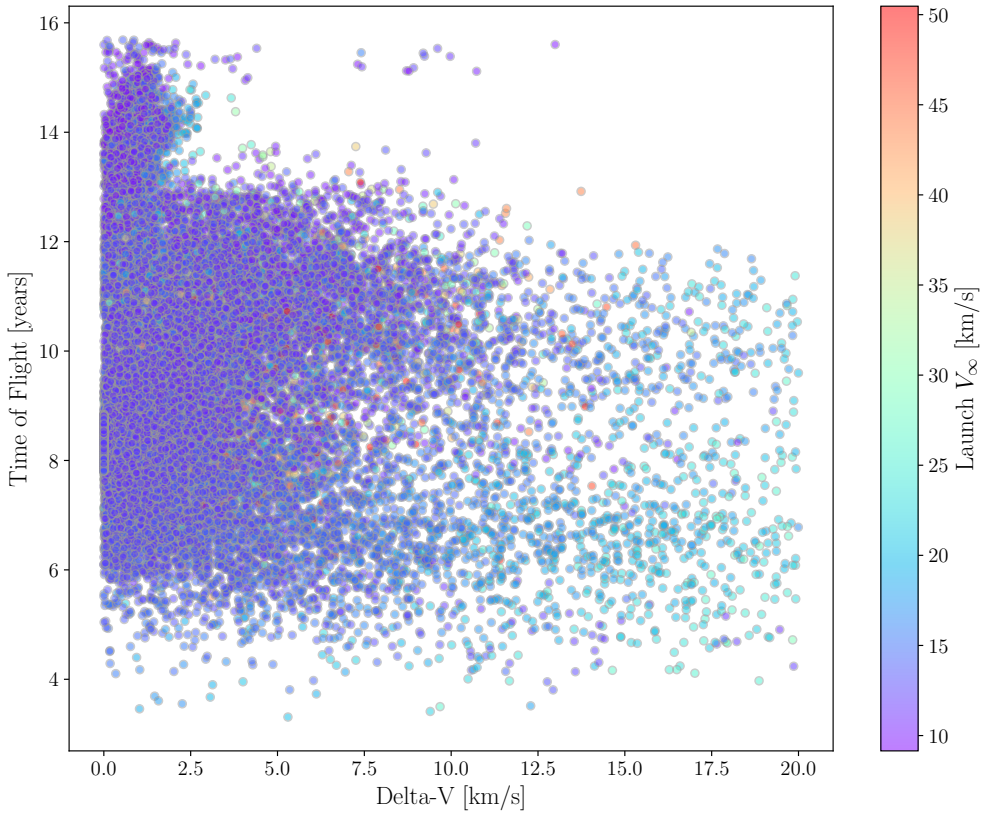


Figure 3.26. Each point represents a patched-conic trajectory derived from one of the Tisserand network search solutions. Points are colored by the Launch V_∞. The raw Monte Carlo results include some cases with unreasonably high V_∞ and ΔV.

for this demonstration, we will review the Monte Carlo trajectories graphically to illustrate the breadth of trajectories resulting from the search.

A scatter plot with color mapping is a useful tool for isolating interesting solutions. Each patched-conic trajectory has more decision attributes than we can simultaneously observe. This plotting method allows us to compare three attributes at a time. The best graphical representation will depend on the specific question to be answered. Here we will review a few arrangements that are generally helpful.

We begin by plotting the total mission time of flight vs the required mission ΔV. Figure 3.26 shows this type of plot for the example Uranus search. Each point in the plot represents an individual patched-conic trajectory derived from one of the solution routes from the

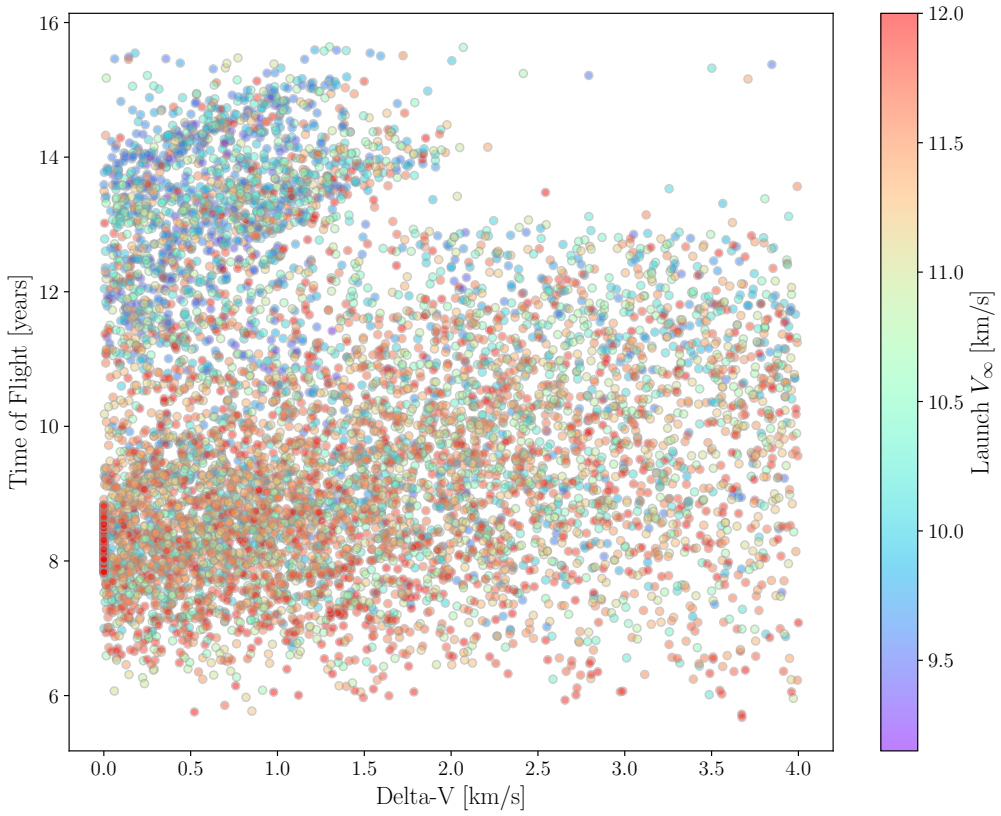


Figure 3.27. Each point represents a patched-conic trajectory derived from one of the Tisserand network search solutions. Points are colored by the Launch V_∞.

Tisserand network search. In this example, 20 patched-conic trajectories were created for each solution in Figures 3.18 and 3.19. Here, we have also colored the points according to the launch V_∞ (the departure V_∞ at Earth). Since the Monte Carlo technique chooses encounter dates without regard for the V_∞ before and after the encounter, we can expect some cases with a large ΔV required to patch the randomized conic segments. Indeed, in Figure 3.26 we find some cases requiring very large launch V_∞ and total mission ΔV that are unreasonably high for current spacecraft capabilities.

Some pre-processing of the Monte Carlo trajectories can filter out the impractical results. Without any particular mission architecture in mind, we might be generous with our filtering and retain trajectories with relatively large propulsion requirements, In Figure 3.27 we have filtered out any trajectories with launch V_∞ greater than 12 $\mathrm{km\,s^{-1}}$ or total mission ΔV

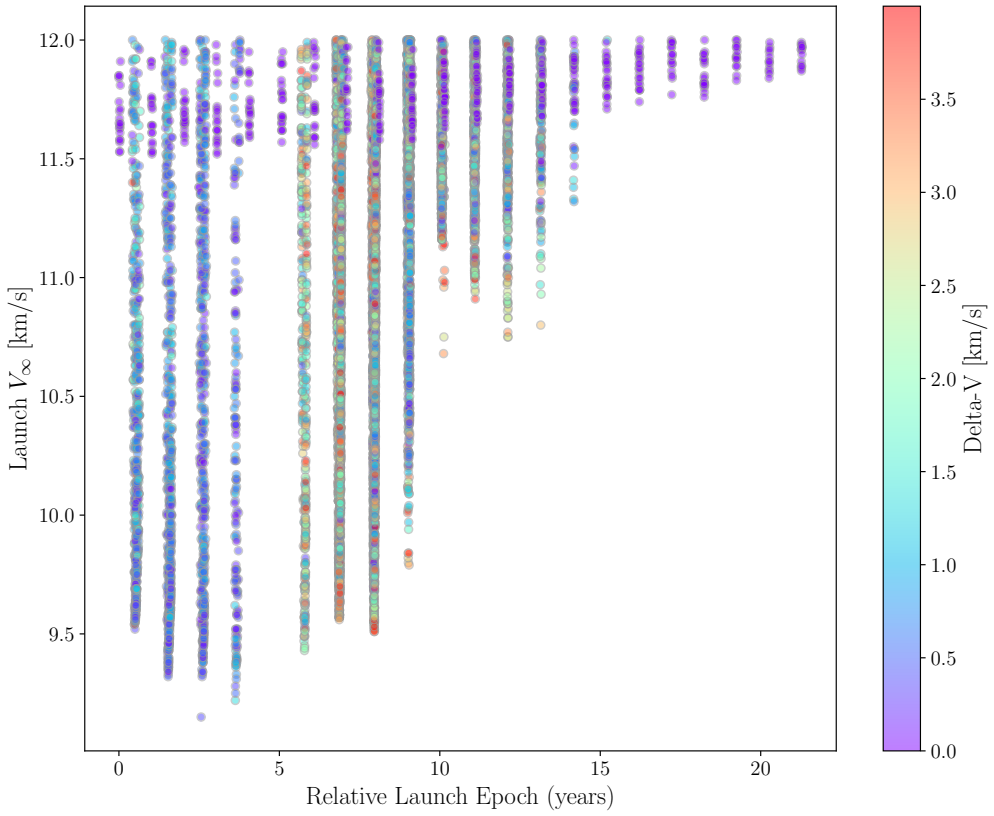


Figure 3.28. Each point represents the launch V_∞ and launch epoch of a patched-conic trajectory derived from one of the Tisserand network search solutions. Points are colored by the total mission ΔV.

greater than 4 km s^{-1}. With the trajectories organized in this way, we can begin to identify attractive options. For example, we might be interested in the ballistic or nearly-ballistic trajectories at the leftmost edge. Additionally, we might seek shorter mission timelines, these trajectories will be closer to the bottom of the plot. Finally, lower launch ΔV requirements will be attractive. So, we will likely want to pursue more detailed analysis of points in the lower-left corner of the plot with colors on the "cool" end of the spectrum. The numerical ranges to consider will depend on actual mission goals and spacecraft capabilities.

Figure 3.28 presents different aspects of the same trajectories. Here the points locate the launch V_∞ and launch epoch of each trajectory. These trajectories are presented for illustration purposes so the launch dates are given as relative values. The colors in Figure 3.28 represent the total mission ΔV. When comparing trajectories in Figure 3.28 we likely

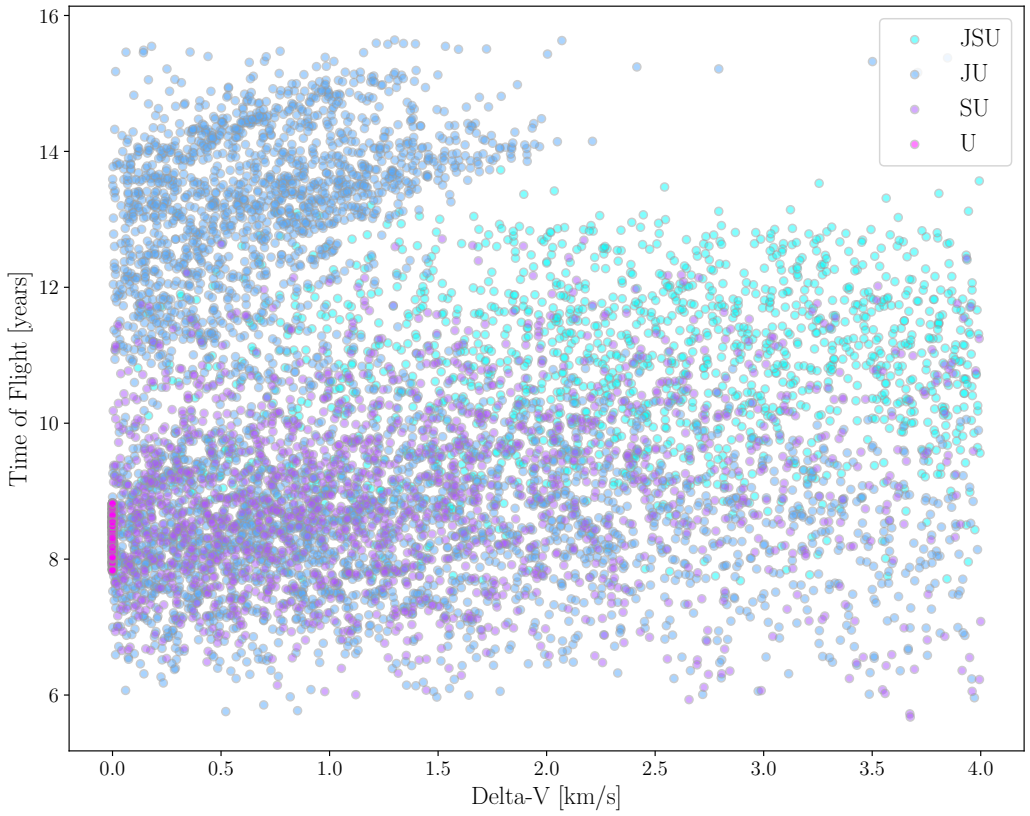


Figure 3.29. Each point represents a patched-conic trajectory derived from one of the Tisserand network search solutions. Points are colored by the gravity-assist path used.

would again prefer low launch V_∞ and low ΔV. This view, however, provides some additional insight into the timing and periodicity of the available mission options.

We also benefit by sorting the trajectories categorically by gravity-assist path as shown in Figure 3.29. In this case we immediately see that the vertical column of ballistic trajectories with low ΔV and short time of flight belong to the direct Uranus path. By definition, these trajectories have a ΔV of exactly zero. Revisiting Figures 3.27 and 3.28, we observe that these direct Uranus trajectories require a high launch V_∞ relative to the other paths.

Let's now examine the effect of the ΔV-minimization scheme in Problem 3.33. Let us first apply the minimization logic to each of the Monte Carlo patched-conic trajectories created from each Tisserand network search solution. Figure 3.30 reproduces Figure 3.27, this time comparing each of the random patched-conic trajectories (in blue) against its optimized

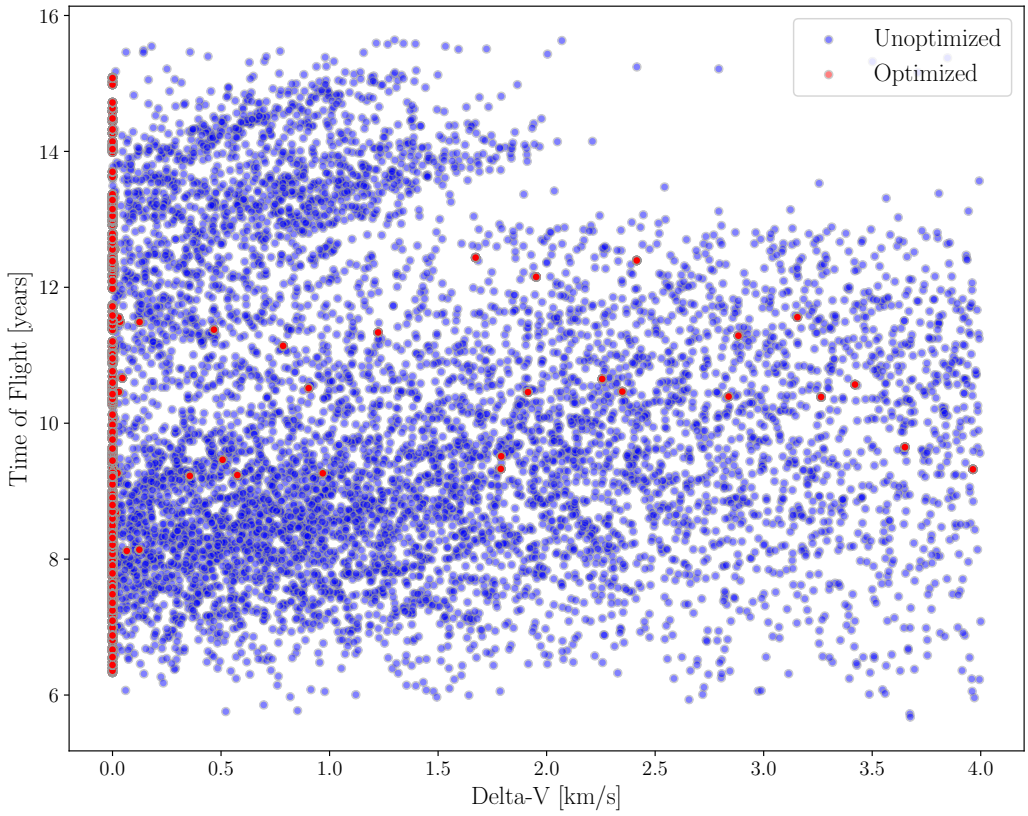


Figure 3.30. Each point represents a patched-conic trajectory derived from one of the Tisserand network search solutions. The blue points represent unoptimized, trajectories with encounter dates randomly selected within bounds provided by the Tisserand network. The red points are result from a ΔV-minimization procedure performed on each of the blue trajectories.

counterpart (in red). As seen in Figure 3.23, many of the Monte Carlo initial trajectories optimize to the same solution.

We observe in Figure 3.30 that the minimization scheme was fairly successful are reducing each randomized trajectory to a nearly-ballistic solution. Figure 3.31 displays the ΔV of each trajectory against the launch epoch. Here we see that the system was able to find nearly-ballistic patched-conic trajectories in each launch window.

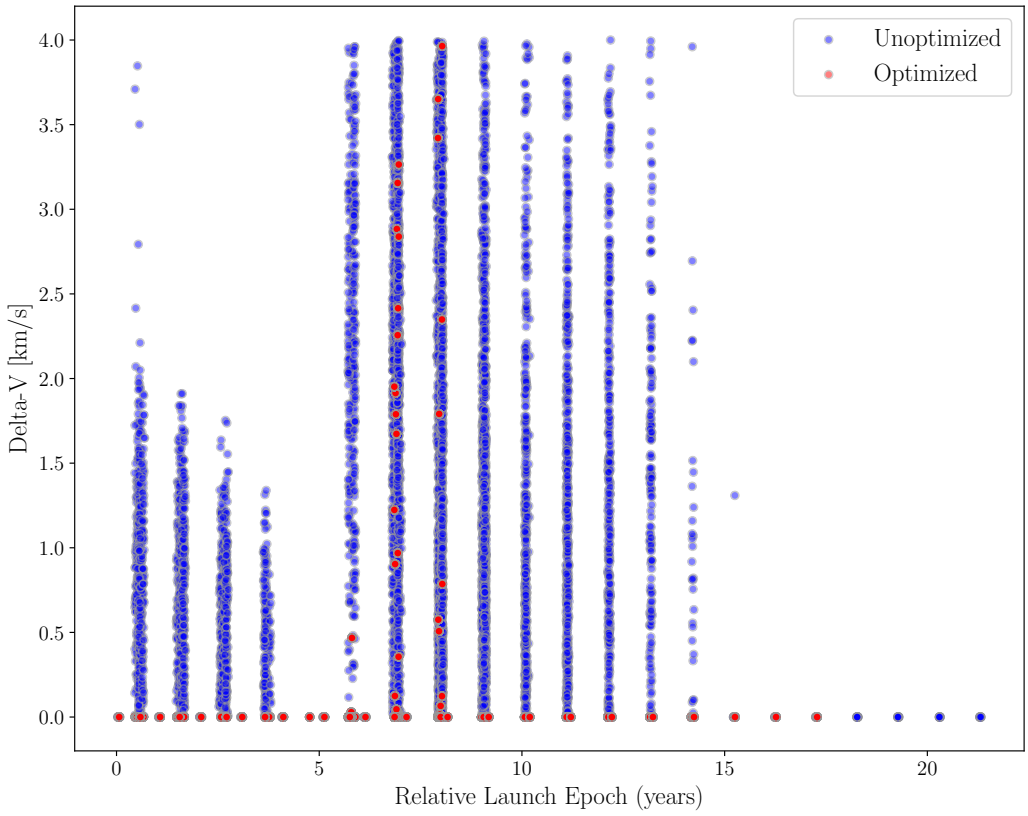


Figure 3.31. Each point represents a patched-conic trajectory derived from one of the Tisserand network search solutions. The blue points represent unoptimized, trajectories with encounter dates randomly selected within bounds provided by the Tisserand network. The red points are result from a ΔV-minimization procedure performed on each of the blue trajectories.

3.8 Tisserand Network Summary

This chapter has introduced a new method for discovering multi-gravity-assist trajectories. The Tisserand network technique addresses both the pathfinding and pathsolving aspects of the traditional MGA search. The network organizes the problem of finding gravity-assist trajectories in a manner that can be fully automated.

A Tisserand network is constructed using a few parameters: the bodies available for gravity assist, and boundaries on the time frame of interest. The two-body dynamics of the gravitational system and the scheduling information are then encapsulated in the vertices and edges of the network using the principles described in Sections 3.2 through 3.4. Once

the network is built, the name of the target planet is all that is needed to perform the search and generate the many trajectory solutions presented in the previous section.

A network search will discover paths through the network. Because of the discrete V_∞ values, search paths will connect precisely in energy but only approximately in time. A Monte Carlo approach to selecting common encounter times in the vicinity of the network results is just one method of creating patched-conic trajectories from the broken-conic search output.

The random nature of the Monte Carlo approach will generate some patched-conic trajectories that are impractical. These trajectories can be easily filtered out of the collection. If desired, the random patched-conic trajectories can be improved with a simple optimization procedure to find the most beneficial encounter dates.

4. NETWORK MODELS FOR MISSION TECHNIQUES

To deploy the Tisserand network for preliminary design of space missions we must also develop some network-ready models of important mission techniques. The common assumptions and/or inputs to the design of these methods need to be re-evaluated in light of the network design.

4.1 Powered Flybys

A powered flyby is a gravity assist with additional acceleration provided by thrust from the spacecraft. In keeping with our other assumptions, we will treat such a maneuver as impulsive.

We model the powered flyby as a discontinuity in V_∞. On the Tisserand graph a powered flyby would appear to be a vertical jump from one V_∞ contour to another contour of the same planet. A mission designer using the Tisserand graph for pathfinding might trace along one contour and allow a step up or down to an adjacent contour if it facilitates a transfer to another Tisserand node along the desired path.

In the Tisserand network we may simply add edges connecting the vertices of a single planet. For example, we might include an edge connecting the V6-O vertex and the V7-0 vertex. This would imply that the vehicle is capable of providing a 1 $\mathrm{km\,s^{-1}}$ boost during the gravity assist. The mission designer can add edges between any vertices that are within the vehicle ΔV capability.

Powered flybys will be weighted with a time of flight of zero. These edges will not directly contribute to the mission duration. The subsequent edges that the powered flyby permits will, of course, have an effect on the flight time. The ΔV of the powered flyby can be tracked as an additional edge weight. This allows the calculation of a total mission ΔV by summing the ΔV weights of the edges on a candidate path.

After making all such permissible connections and weight assignments, a network search can proceed normally. A powered flyby edge will be handled by the network algorithms in the same was as the Tisserand graph node edges. The difference between the types of

transfers is encapsulated in the time of flight and ΔV weights. In Chapter 5 we will show that the ΔV weight can be used to actively prune excessively costly paths during the search.

4.2 Resonant Flybys

We wish to provide a means for the Tisserand network to discover trajectories that include resonant flybys. A gravity assist path with a resonance will include multiple consecutive passes of a single planet. The Venus-Earth-Earth Gravity Assist (VEEGA) used on the Galileo mission is an example. Including resonance greatly expands the potential combinations of gravity assists that might be employed to achieve trajectory goals.

A spacecraft has a resonant orbit with a planet when the orbit periods can be expressed as an integer ratio, $n:m$. Here, n is the number of revolutions completed by the planet and m is the number of revolutions completed by the spacecraft between consecutive encounters. For example, a 3:2 resonance means the planet completes three revolutions in the time it take the spacecraft to complete two. The Galileo mission included a 2:1 VEEGA in which the spacecraft performed two gravity assists at Earth separated by two years. The present work only considers full revolution (or even-$n\boldsymbol{\pi}$) resonant transfers.

From the definition of the resonance, the time to complete each set of integer revolutions is equivalent:

$$nP_{\mathrm{p}} = mP_{\mathrm{sc}}\,. \tag{4.1}$$

We can also express this relationship as

$$\frac{n}{m} = \frac{P_{\mathrm{sc}}}{P_{\mathrm{p}}}\,. \tag{4.2}$$

4.2.1 The Resonance Problem

As discussed in Chapter 2, practical limits on the minimum flyby radius translate to maximum limits on the velocity turning, δ, that can be achieved with a gravity assist. The trajectory turning can be thought of as the angular difference between the pump angle of the inbound $\mathbf{V}_\infty^-$ vector and that of the outbound $\mathbf{V}_\infty^+$ vector as shown in Figure 4.1. With

regard to the Tisserand graph, this limit on turn angle translates to a maximum distance that we can traverse along a V_∞ contour to move from one node to the next.

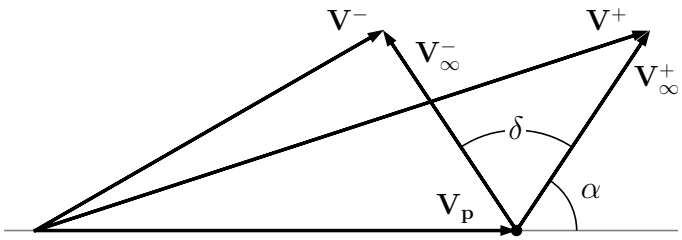


Figure 4.1. The bending angle, δ measures the angular difference between the inbound and outbound V_∞ vectors. The angle is determined by the radius of the flyby.

The development in this section can most easily be illustrated with a concrete example. Suppose we would like to advance from a heliocentric orbit defined at the V7E10 node of the Tisserand graph (a = 1.0 AU, e = 0.33) to an orbit defined at the E10J7 node (a = 3.3 AU, e = 0.70). Figure 4.2 provides an example using a minimal Tisserand graph for this particular problem. Similar problems need to be solved at a number of places while constructing the Tisserand network. As seen in Figure 4.2, the required δ to transform the heliocentric orbit is 66.7 deg (96.3 − 29.6). However, the maximum turning possible is only 43.9 deg.

To extend the distance along the Tisserand graph V_∞ contour, or to extend the vertices reachable in the Tisserand network, we can perform an additional (or several more) gravity assists at the current planet. Since the gravity assist does not alter the V_∞ magnitude, this second flyby will occur at the same V_∞ as the first but with a different pump angle.

The bending angle limit is also the impetus for creating the line graph of the Tisserand network (see Chapter 3). During the weighting procedure, the line graph can tell us whether a situation like the one shown in Figure 4.2 exists. If so, the time required to complete the resonance will need to be included in the time of flight.

If required, an additional gravity assist also introduces a phasing problem. The spacecraft must re-encounter the planet at sometime in the future. Since the heliocentric spacecraft orbit has a different period than the planet orbit, we must wait for an integer number of both periods.

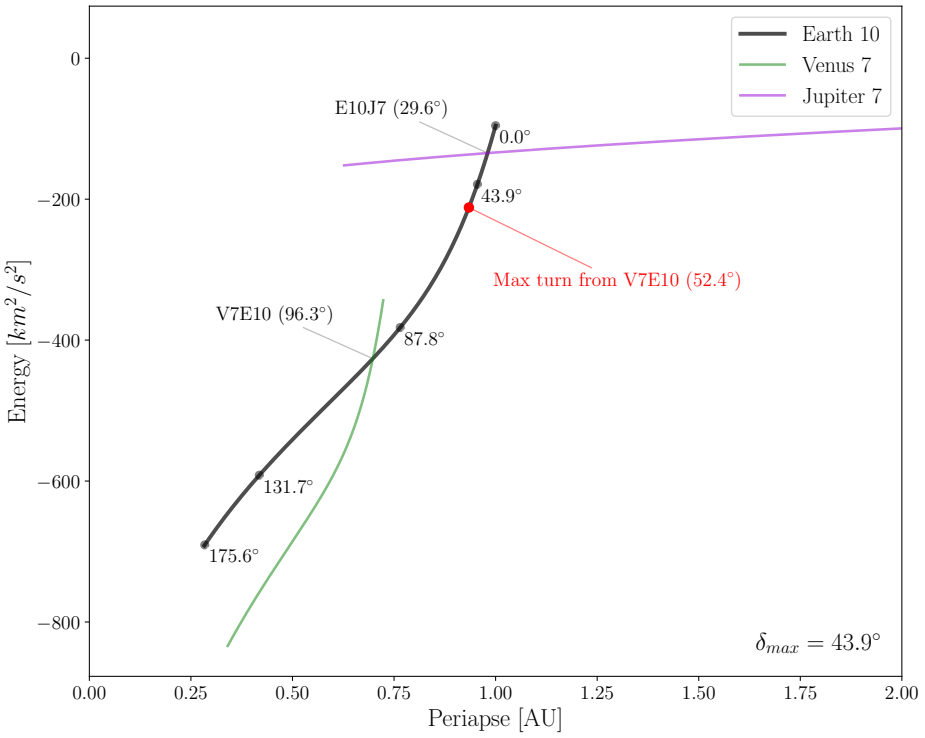


Figure 4.2. This simplified Tisserand graph shows how the maximum bending angle, $\delta_{\max}$, limits the reachable nodes. The heliocentric orbit at the V7E10 node cannot be transformed into the orbit at the E10J7 node by a single Earth gravity assist. The black dots along the Earth contour serve as α reference markers with a spacing equal to $\delta_{\max}$. The maximum change from the V7E10 pump angle is shown as the red dot.

4.2.2 A Network Solution for Resonance

To include resonant gravity assists in the Tisserand network, we need a method to compute the time of flight for the resonant sequence. In Chapter 2 we outlined the relationship between the pump angle, α, and the period of the heliocentric orbit, P, in Equations 2.2 through 2.7. We can summarize that relationship in Figure 4.3. Along a given V_∞ contour there is a unique heliocentric orbit period for each possible pump angle.

In Tisserand graph terms, we will need to take multiple steps along a single V_∞ contour (see Figure 4.2). The size of the steps we take is determined by the bending angle, δ, which itself is a function of the periapsis radius of the gravity assist hyperbola. We may choose any step size (up to $\delta_{\max}$). However, the period of the resulting heliocentric orbit is determined

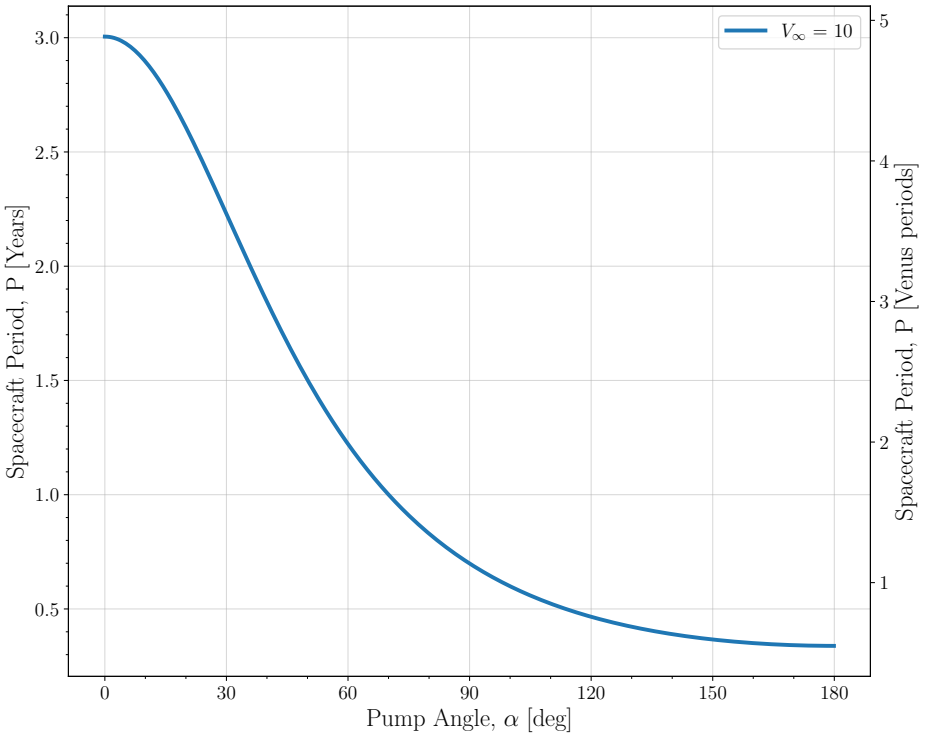


Figure 4.3. The period of the heliocentric orbit and the pump angle form a monotonic relationship. The period of the spacecraft orbit can be expressed in time units (left axis) or in multiples of the flyby body orbit period (right axis).

.

by our choice. A careless selection of δ will result in an orbit period with a large integer ratio to the planet period. In other words, we will need to complete a large number of revolutions before we arrive back at the original encounter location to find that the planet has also returned. For our purposes, we may assume that the multiple encounters occur at the same inertial location.

Resonance and Pump Angle

We seek an algorithm that will generate the pump angles that will result in convenient resonances. Because we are casting a wide net, we will want several options to choose from. We don't know, at the network building stage, which resonances will yield convenient times for subsequent gravity assists at different planets during the network searching stage.

We first generate a set of resonances to consider. Let us start by defining a maximum number of spacecraft revolutions that we are willing to allow, $m_{\max}$, and a maximum time of flight for the resonance to resolve, $t_{\max}$. The maximum number of planet revolutions is given by

$$n_{\max} = \left\lceil \frac{t_{\max}}{P_{\mathrm{p}}} \right\rceil , \tag{4.3}$$

where P_{p} is the orbital period of the planet and the ceiling operation, $\lceil\ \rceil$, rounds up to the next integer.

So, let us create two arrays of integers:

$$n_i = 1, 2, \ldots, n_{\max} \tag{4.4}$$

$$m_j = 1, 2, \ldots, m_{\max} . \tag{4.5}$$

These may be collected into pairs

$$\mathscr{R}_k = (n_i, m_j) \text{ for } i = 1, \ldots, n_{\max}, \text{ for } j = 1, \ldots, m_{\max} , \tag{4.6}$$

where $k = n \times m$. The set of pairs, $\mathscr{R}$, will have some equivalent ratios. For example, $(1, 1)$ and $(2, 2)$ or $(1, 2)$ and $(2, 4)$. The set $\mathscr{R}$ can be reduced to a unique set of ratios:

$$R = \mathscr{R}_p(0) \backslash GCD(\mathscr{R}_p) \ : \ \mathscr{R}_p(1) \backslash GCD(\mathscr{R}_p) \text{ for } p = 1, \ldots, k , \tag{4.7}$$

where the backslash operator $\backslash$ means integer division and $GCD(\mathscr{R}_p)$ is the greatest common denominator of the elements of $\mathscr{R}_p$. The result of these operations is a set of unique resonances, $R = [1\,{:}\,2,\ 1\,{:}\,3,\ 2\,{:}\,3,\ \ldots\ ]$, that satisfy the maximum time of flight constraint, $t_{\max}$ and the maximum number of spacecraft revolutions, $m_{\max}$.

Next we must translate each $n\,{:}\,m$ resonance pair in R into a pump angle. Since the planet period is consistent for all resonances, we can most easily express the time to complete the resonance as n planet periods. From Equation 4.1 or 4.2, we have:

$$P_{\mathrm{sc}} = \frac{n}{m} P_{\mathrm{p}} . \tag{4.8}$$

So that we may compute a spacecraft period, P_{sc}, for each $n\,{:}\,m$ resonance pair in R.

From here we can relate P_{sc} to α by reversing Equations 2.2 through 2.7 or through a graphical or numerical analysis of Figure 4.3 (which must be repeated on similar curves for each V_∞ and planet). Kepler's third law relates the period of the spacecraft orbit to the semi-major axis:

$$P_{sc} = \frac{2\pi}{\sqrt{\mu}} a_{sc}^{3/2}, \tag{4.9}$$

which can be solved for semi-major axis to give

$$a_{sc} = \mu^{1/3} \left(\frac{P_{sc}}{2\pi} \right)^{2/3}, \tag{4.10}$$

where μ is the gravitational parameter of the central (primary) body. The *vis-viva* equation gives the velocity of the spacecraft in the heliocentric orbit at the location of the planet:

$$V_{sc}^2 = \mu \left(\frac{2}{r_{planet}} - \frac{1}{a_{sc}} \right). \tag{4.11}$$

The spacecraft velocity is related to the V_∞ and pump angle through the Law of Cosines:

$$V_{sc}^2 = V_p^2 + V_\infty^2 + 2V_p V_\infty \cos\alpha, \tag{4.12}$$

which can be solved for the pump angle to give:

$$\cos\alpha = \frac{V_{sc}^2 - V_p^2 - V_\infty^2}{2V_p V_\infty}. \tag{4.13}$$

The sequence of Equations 4.8, 4.10, 4.11, and 4.13 provides a flyby pump angle for each resonance pair in R.

Table 4.1. Resonance to Pump Angle Mapping (Earth 10 km s^{-1})

Resonance	Spacecraft Period (years)	Pump Angle (deg)
1:1	1.0	99.7
3:2	1.5	79.3
2:1	2.0	67.5
5:2	2.5	59.1
3:1	3.0	52.7
4:1	4.0	43.1
5:1	5.0	35.7

The procedure described above provides a mapping between our candidate resonances and the pump angles that yield them. To demonstrate, suppose we allow no more that two resonant revolutions ($m_{\max} = 2$) at Earth with $V_\infty = 10$ km s^{-1} and we also require that any single resonant transfer take no more than 5 years ($t_{\max} = 5$). Under these requirements, the procedure outlined in Equations 4.3 through 4.8 yields the resonance options in Table 4.1. The pump angles that generate these resonances are highlighted in Figure 4.4. Two of the resonances (2 : 1 and 5 : 2) are far enough along the contour such that the second gravity assist (at the completion of the resonance) will be sufficient to complete the desired trajectory turning.

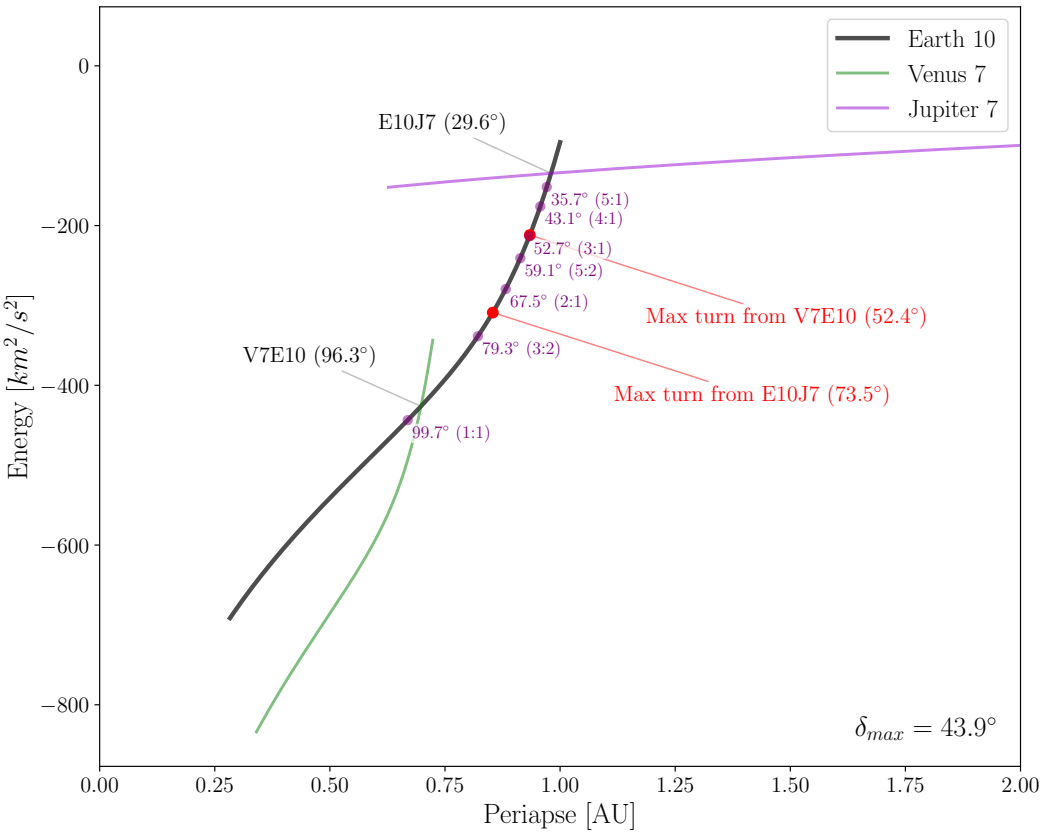


Figure 4.4. This simplified Tisserand graph shows some possible resonance options between the V7E10 and E10J7. The 2 : 1 and 5 : 2 resonances can be reached from the V7E10 node on the first gravity assist. From these resonances the E10J7 node can then be reached with the second gravity assist.

The resonance integers are kept simple in the example above for demonstration purposes. We may wish to increase the $m_{\max}$ and $t_{\max}$ when building our network. This will create more resonance options, which in turn yield more possible transfer times. We don't know, ahead of time, which transfer times might ultimately be convenient for the phasing problem. We should also keep in mind that longer times of flight and multiple solar orbits will weaken the validity of the two-body dynamics assumption.

Resonance Network Implementation

We now consider how to include the analysis above in the Tisserand network. The resonance options for each V_∞ contour (such as those shown in Table 4.1) can be computed with only the information used to construct the Tisserand graph and the $m_{\max}$ and $t_{\max}$ parameters. Therefore, these calculations can be performed as the Tisserand network is constructed.

The next step is to assemble the various sequences of resonant flybys that might be used to complete the connection from one Tisserand graph node to the next. Our first intuition might be to choose a resonance that is sufficient to reach the goal node in one intermediate step. However, we must keep our options open at this step since we do not yet know the time of flight that will be best for solving the phasing problem.

So, let us assemble the possible resonant sequences from Table 4.1 and let us now limit the cumulative time of flight on the resonance sequence to 8 years. Keep in mind, this process must be repeated at all V_∞ contours. The result for the Earth 10 $\mathrm{km\,s^{-1}}$ contour is shown in Table 4.2. The *Resonance Time* column in Table 4.2 includes only the time completing the intermediate orbits and not the entry or exit transfers. The intermediate orbits are full revolutions. So the resonance duration can easily be computed by summing the planet revolutions in the resonance sequence and multiplying by the planet period.

The entry transfer (i.e., Venus to Earth) and exit transfer (i.e., Earth to Jupiter) are partial arcs on the corresponding heliocentric orbits. These times depend on the inbound-

Table 4.2. Feasible Resonance Sequences (Earth 10 km s^{-1})

Resonance Sequence	Pump Angles (deg)	Resonance Time (years)
3:2	96.3, 79.3, 35.4	3
3:2, 2:1	96.3, 79.3, 67.5, 23.6	5
3:2, 2:1, 3:1	96.3, 79.3, 67.5, 52.7, 8.8	8
3:2, 5:2	96.3, 79.3, 59.1, 15.2	8
3:2, 3:1	96.3, 79.3, 52.7, 8.8	6
3:2, 4:1	96.3, 79.3, 43.1, 0.3	7
3:2, 5:1	96.3, 79.3, 35.7, 0.3	8
2:1	96.3, 67.5, 23.6	2
2:1, 5:2	96.3, 67.5, 59.1, 15.2	7
2:1, 3:1	96.3, 67.5, 52.7, 8.8	5
2:1, 4:1	96.3, 67.5, 43.1, 0.3	6
2:1, 5:1	96.3, 67.5, 35.7, 0.3	7
5:2	96.3, 59.1, 15.2	5
5:2, 3:1	96.3, 59.1, 52.7, 8.8	8
3:1	96.3, 52.7, 8.8	3
3:1, 4:1	96.3, 52.7, 43.1, 0.3	7
3:1, 5:1	96.3, 52.7, 35.7, 0.3	8

outbound encounter points according to Table 3.2. The total time for a transfer from planet A to B to C (e.g., Venus to Earth to Jupiter) is then:

$$t_{\mathrm{AC}} = t_{\mathrm{AB}} + t_{\mathrm{B_{res}}} + t_{\mathrm{BC}}\,, \tag{4.14}$$

where t_{AB} and t_{BC} are given by Table 3.2 and $t_{\mathrm{B_{res}}}$ is given by the procedure in this section (culminating in a table like Table 4.2).

The final step is to weight the network edges. As discussed in Chapter 3, we weight our network edges with the time of flight of the transfer. In general, the resonance sequences will have different durations (as seen in Table 4.2). Equation 4.14 must be evaluated at every row of Table 4.2 and this process must be repeated for each V_∞ and gravity-assist body.

For each three-vertex sequence (A-B-C) we will have many potential flight times corresponding to the resonances we have chosen to consider. Therefore, we will need to add parallel edges to the network and weight them accordingly. Parallel edges connect the same

vertices of a network but differ in some other property. In our application, the parallel edges will be weighted with different times.

For the broadest capability, we can choose to add an additional edge for every possible resonance sequence. Alternatively, we might choose to add a parallel edge only for those resonance sequences with unique durations.

4.3 V-Infinity Leveraging Transfers

A V-infinity leveraging transfer or, VILT, (sometimes, simply V_∞-leveraging) uses a small propulsive maneuver to alter the encounter V_∞ of a subsequent gravity assist [33], [60]–[62] . As shown in Chapter 2, the trajectory bending angle, δ, is a function of the V_∞ of the gravity assist (Equation 2.18). The effect of the propulsive ΔV will be magnified by the gravity assist ΔV though the change in V_∞. A maneuver that increases eccentricity will increase V_∞; a maneuver that decreases eccentricity will decrease V_∞. The term V_∞-leveraging was coined by Longuski to describe the magnifying affect of the V_∞ change on the subsequent gravity assist [8]. When applied after an Earth launch in order to obtain a direct gravity assist from Earth, this technique is also called a Delta-V Earth Gravity Assist or ΔV-EGA. The VILT is a generalization of the ΔV-EGA technique.

V-infinity leveraging has been applied on deep space missions such as NEAR and Cassini [63]–[66] to reduce total post-launch ΔV. Here, we will develop a network-ready model of the VILT problem so that our Tisserand network can consider this technique as a possible transfer option.

4.3.1 V-Infinity Leveraging Fundamentals

Some important concepts and nomenclature from the early work on VILTs [33], [61] will be reviewed below to lay the foundation for a network model. In this work, we only consider coplanar VILTs in which the leveraging maneuver ΔV is tangent to the orbit and occurs at apoapsis or periapsis. We also only consider same-body transfers in which the gravity assists before and after the maneuver occur at the same planet.

A V-infinity leveraging transfer can be categorized as *interior* or *exterior* based on whether the leveraging maneuver occurs inside or outside of the planet orbit. In an exterior VILT, the maneuver occurs at apoapsis of the spacecraft heliocentric orbit and in an interior VILT the maneuver occurs at periapsis. The spacecraft orbit prior to the leveraging maneuver is called the *nominal* orbit. The orbit after the maneuver is called the *return* orbit. Figure 4.5 is a schematic of an exterior VILT.

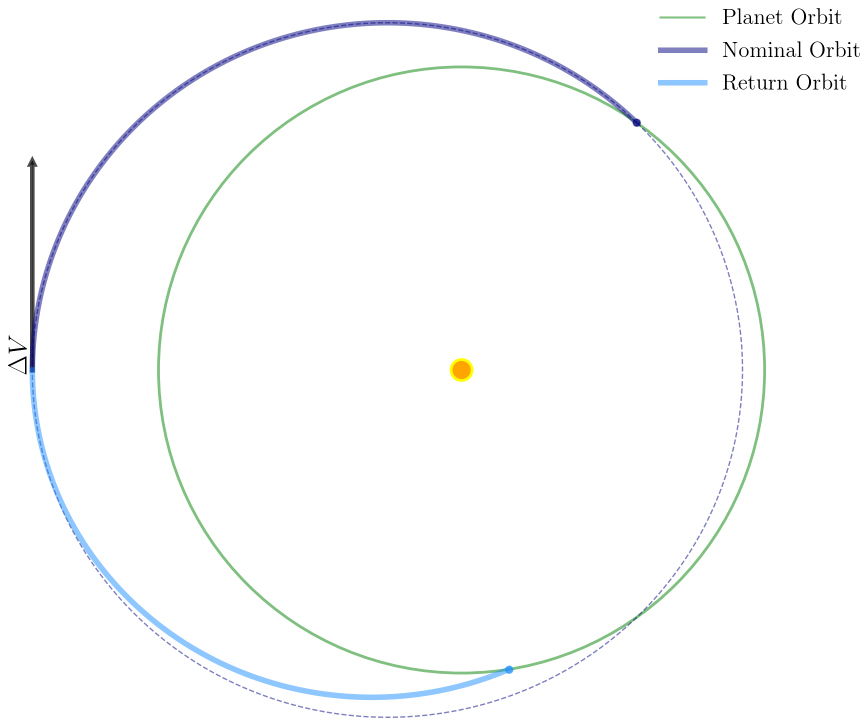


Figure 4.5. The schematic of an outbound-to-inbound V-Infinity Leveraging Transfer shows the nominal orbit and the return orbit affected by the leveraging maneuver. The V_∞ at the re-encounter with the planet is modified by the leveraging maneuver.

Since we are considering same-body transfers, we will have a synchronization problem similar to the resonance problem in Section 4.2. The spacecraft must cross the planet orbit at a time when the planet is also there. We will not require that the transfer take place in a single revolution of either body. Therefore, we will need to account for the resulting resonance. We will use the notation $N : M(L_m)$ where N is the number of planet revolutions,

M is the number of spacecraft revolutions, and L_m is the spacecraft revolution on which the maneuver is performed [34]. This convention is used here to mirror the $n:m$ notation used for resonances. The reader should note that Sims *et al.* [33] and other authors use the notation $K:L(M)$ for the same concept.

The initial and return encounters can occur at an inbound or outbound location. This means that there are four combinations of transfers (II, IO, OI, OO) for any set of nominal and return orbits. This is true for both the exterior and interior VILTs.

A typical approach to designing a VILT is to select the nominal orbit, $N:M(L_m)$ configuration, and departure and arrival locations, and then iterate on the size of the ΔV until the time for the spacecraft and planet to travel to the re-encounter point are equal (as in Sims *et al.* [33]). However, the approach may vary depending on the discrete/continuous or fixed/free variables in each implementation. Mudek [17], Strange [34], Wu and Russell [35], and Campagnola and Russell [67] provide solutions for a variety of design scenarios.

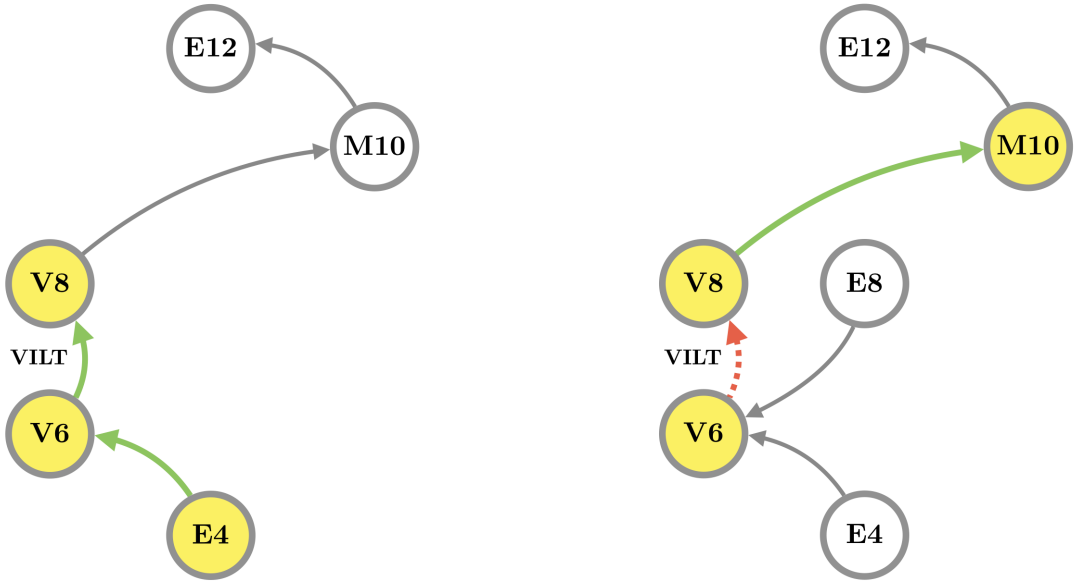
4.3.2 VILT Network Implementation

The V-infinity leveraging transfer will be treated similarly to other transfers in the network. We model the transfer as an edge connecting two Tisserand network vertices. The weight of the edge will be the time to complete the transfer. So we seek a method to measure the time required for an arbitrary VILT.

Some key features of the Tisserand network prevent us from using the existing solution methods. Most fundamentally, we have an additional constraint that the departure and arrival V_∞ be among the discrete values in our network. Traditional methods need only require that the flight time of the planet and spacecraft are equal. Additionally, we are interested in solving the problem for a variety of $N:M(L_m)$ sequences. Changing the $N:M(L_m)$ of a VILT will affect the flight time and our searches will benefit from a variety of potential flight time options. Finally, we will need to solve many versions of this problem by considering all of the inbound/outbound combinations for multiple planets and V_∞ levels.

An additional complication relative to early work on VILTs [33] is that the location of the initial departure is not fixed. The early ΔV-EGA literature assumes that the leveraging

maneuver follows an Earth launch. The departure V_∞ is tangent to the Earth orbit at apoapsis (interior) or periapsis (exterior) of the transfer orbit and the leveraging maneuver occurs at the opposite apsis.



(a) VILT on second edge.

(b) VILT on first edge.

Figure 4.6. Weighting a sequence containing a VILT is only possible when the VILT occurs on the second edge. Figure 4.6a highlights a three-vertex sequence where the first encounter of the VILT is determined by the initial vertex. Figure 4.6b highlights a sequence where the first encounter of the VILT cannot be determined because of ambiguity in the prior vertex.

In our version of the problem, the departure conditions are partially determined by the arrival condition from the previous gravity assist in the path. In general, the V_∞ will be non-tangent. The true anomaly of the initial encounter is determined by the V_∞, the inbound or outbound approach from the previous gravity assist, and the amount of trajectory turning, δ, according to Equation 2.11. The re-encounter must occur at one of the discrete network V_∞ levels but it may occur at the inbound or outbound encounter location.

To address these complications we return to the line graph of the Tisserand network. Recall that an edge of the line graph gives us information about a three gravity-assist sequence in the network. Consider the specific example shown in Figure 4.6. Here we see two schematics of three-vertex sequences (line graph edges) highlighted in yellow.

In Figure 4.6a, the VILT occurs on the second edge of the sequence (V6-V8). The pre-flyby pump angle for the VILT is determined by the first edge of the sequence (E4-V6). Since the three-vertex sequence (E4-V6-V8) is found in the line graph, we have enough information to compute the duration of the VILT. We could weight edge (E4-V6) with the time of flight from Table 3.2 and weight edge (V6-V8) with the duration of the VILT.

In contrast, consider Figure 4.6b where the VILT occurs on the first edge of the sequence (V6-V8). In this case, the line graph sequence (V6-V8-M10) tells us nothing about the pre-flyby pump angle at V6 (the beginning of the VILT). The time on the VILT nominal orbit is affected by the true anomaly of the initial encounter. The initial true anomaly depends on the bending angle, δ, and the pre-flyby pump angle, α^-. But α^- depends on the previous gravity assist, in this case, E4 or E8. Since this line graph sequence (V6-V8-M10) does not provide this information, we cannot compute the VILT duration for weighting purposes.

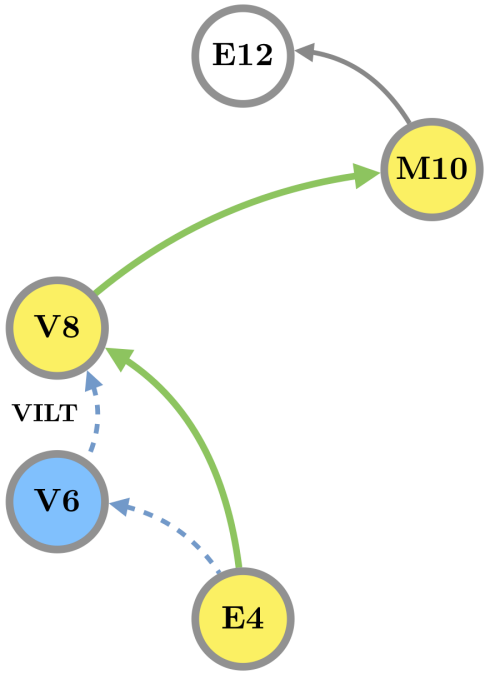


Figure 4.7. Creating a parallel edge connecting the vertices surrounding a VILT avoids the ambiguity in flight time calculation. The intermediate vertex is recorded as a waypoint so that the complete path can be reconstructed from the edge sequence.

We resolve this problem as follows. Instead of adding an edge between the two vertices at the VILT planet (which can only sometimes be weighted), we add an edge between the endpoints of the three-vertex sequence. Figure 4.7 provides an example. We assign the total time of the normal transfer and the VILT to the new edge (E4-V8). We also record the vertex V6 as a waypoint associated with this new edge. Additionally, we may add an edge connecting E8 and V8 by way of a VILT at V6 (as shown in Figure 4.6b). The new VILT edges (E4-V8) and (E8-V8) will have different total flight times as their weights.

Discrete V-Infinity Leveraging Transfers

Let us consider a VILT between two discrete V_∞ levels at the same planet. We start by choosing two V_∞ levels, $V_{\infty,1}$ and $V_{\infty,2}$, from among the discreet Tisserand graph contours for any one planet. Let $V_{\infty,1}$ be the V_∞ at the departure of the VILT and let $V_{\infty,2}$ be the V_∞ at the re-encounter (after the leveraging maneuver). Similarly, let α_1 and α_2 be the pump angles at the departure and re-encounter, respectively.

We will need to compare the times that the planet and spacecraft take to travel between the encounter points (including any complete revolutions). The planet travel time is given by:

$$t_{\text{planet}} = \left(N + \frac{\theta_{\text{planet}}}{2\pi} \right) P_{\text{planet}} \,, \tag{4.15}$$

where P_{planet} is the planet orbit period and θ is the simple transfer angle between the encounters. The cumulative angle swept out by the planet during the entire VILT duration can be found using the spacecraft flight time, t_{sc}, to be discussed below. That sweep angle is

$$\Theta = n t_{\text{sc}} = \frac{2\pi}{P_{\text{planet}}} t_{\text{sc}} \,, \tag{4.16}$$

and

$$\theta_{\text{planet}} = \Theta \,(\text{mod } 2\pi) \,. \tag{4.17}$$

We can compute the time required for the spacecraft to complete the full transfer by summing the time from the departure to the leveraging maneuver, t_{pre}, with the time from the maneuver to the re-encounter point, t_{post}:

$$t_{\text{sc}} = t_{\text{pre}} + t_{\text{post}} \, . \tag{4.18}$$

We must include any complete revolutions that occur before the maneuver. The pre-maneuver time is given by

$$t_{\text{pre}} = (L_m - 1)P_{\text{nominal}} + t_{\text{out}} \, , \tag{4.19}$$

where P_{nominal} is the period of the nominal orbit and t_{out} is the time spent on the initial arc between the departure point and the maneuver apsis. If the maneuver occurs on the first spacecraft orbit ($M = 1$) then t_{out} is the total time before the maneuver, t_{pre}. Similarly, the post-maneuver time must include any complete revolutions on the return orbit:

$$t_{\text{post}} = (M - L_m)P_{\text{return}} + t_{\text{in}} \, , \tag{4.20}$$

where P_{return} is the period of the return orbit and t_{in} is the time spent on the final arc between the maneuver apsis and the re-encounter point. The partial arc flight times, t_{out} and t_{in}, are computed from:

$$\begin{aligned} t_{\text{out}} &= t_{\text{an}} - t_1 \\ t_{\text{in}} &= t_2 - t_{\text{ar}} \end{aligned} \tag{4.21}$$

where t_1 is the departure time, t_2 is the arrival time, and t_{an} and t_{ar} are the times of the apsis passage on the nominal and return orbits, respectively.

As discussed in Chapter 2, the size and shape of the heliocentric orbit is fixed by the V_∞ and pump angle at the encounter. So $V_{\infty,1}$ and α_1 determine P_{nominal} as well as the possible intersection points with the circular planet orbit according to Equation 2.11. If we select the inbound or outbound location for the departure, then the true anomaly of the departure, ν_1, is also fixed. Now, the times t_1 and t_{an} may be found using Kepler's equation 3.1 on the nominal orbit.

We need a way to define the return orbit. We are not free to simply choose a return orbit that synchronizes with the planet. We must re-encounter the planet orbit at $V_{\infty,2}$. We can use the Tisserand graph relationships from Chapter 2 and the common apsis distance to link the two orbits. We must choose a $V_{\infty,2}$ and α_2 that provide a heliocentric orbit with the same apsis radius as $V_{\infty,1}$ and α_1.

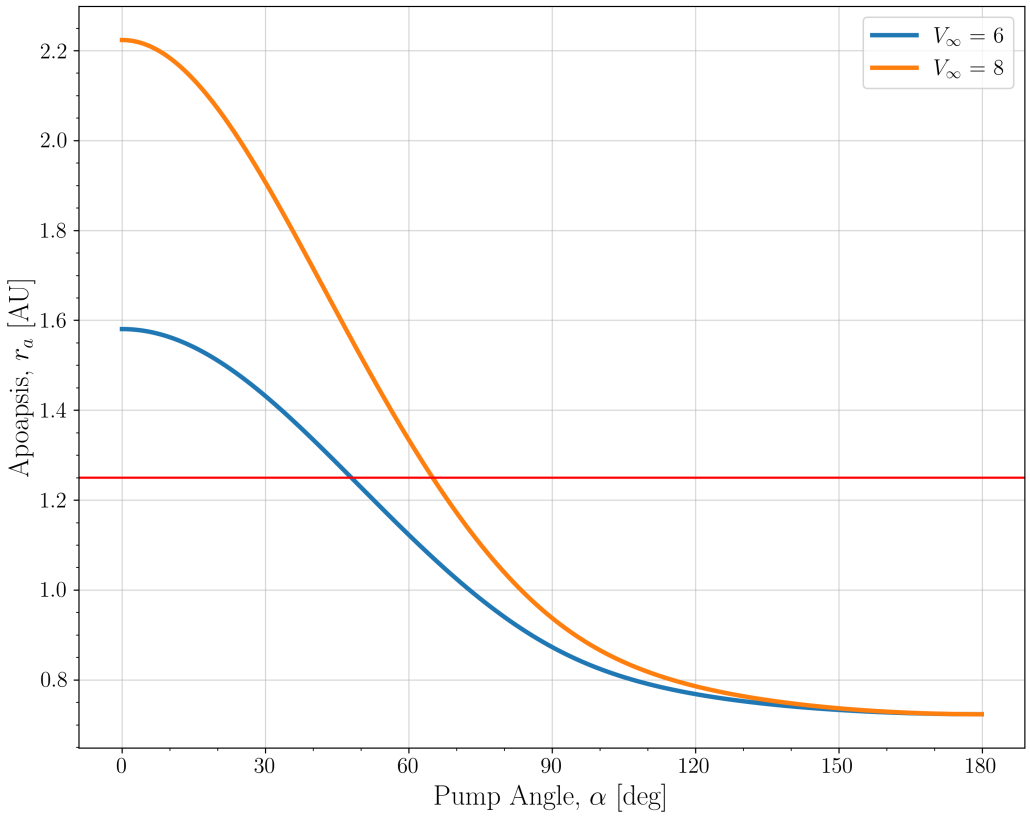


Figure 4.8. The figure gives an example of how Tisserand graph relationships can be used to connect discrete V_∞ levels through a VILT. A 6 km s^{-1} departure from Venus with a pump angle of 48 deg yields a heliocentric apoapsis of 1.25 AU. The return orbit must have the same apoapsis. To return to Venus at 8 km s^{-1} the pump angle must be approximately 65 deg. The difference in apoapsis velocity between the two orbits is the required ΔV.

Figure 4.8 shows a specific example. Suppose we have chosen $V_{\infty,1} = 6\,\mathrm{km\,s^{-1}}$ and $V_{\infty,2} = 8\,\mathrm{km\,s^{-1}}$ for an exterior VILT. If, in addition, $\alpha_1 = 48\,\mathrm{deg}$, then the apoapsis is 1.25 AU. This apoapsis corresponds to a return pump angle of $\alpha_2 = 65\,\mathrm{deg}$.

Therefore, α_1 is determined once the amount of turning, δ, at the VILT departure is chosen. And once α_1 is determined, α_2 can be found for our discrete $V_{\infty,2}$. The return orbit is now defined by α_2 and $V_{\infty,2}$, giving P_{return}. With a choice of inbound or outbound location on the re-encounter, the true anomaly, ν_2, is given by Equation 2.11. The times t_2 and t_{ar} may be found using Kepler's equation 3.1 on the return orbit. The transfer angle of the spacecraft is simply:

$$\theta_{\text{sc}} = \nu_2 - \nu_1 \,. \tag{4.22}$$

This procedure supplies all the information needed to complete the calculation of the spacecraft travel time during the VILT (Equations 4.18 through 4.21). Additionally, the required ΔV may be computed from the difference in velocity at the apsis in the nominal and return orbits. If we consider the planet, the V_∞ levels, the inbound/outbound locations, and the $N : M(L_m)$ as parameters, then α_1 is a free variable that can be tuned to modify ν_2 at the re-encounter orbit crossing and the spacecraft time of flight. In some cases, the tuning will be able to adjust these parameters so that the spacecraft and planet are in the same place at the same time at the end of the prospective VILT.

VILT Optimization and Scanning

Now we can formulate an optimization problem to find VILTs that re-encounter the planet. Previous researchers have minimized the difference in time traveled by the spacecraft and the planet—leaving the V_∞ of the re-encounter as a free variable. Because of our discrete V_∞ requirements, we minimize the difference in transfer angle during the execution of the candidate VILT. Recall, our goal is to find a set of candidate VILTs within our Tisserand network and use the durations of those VILTs to weight new edges between our network vertices.

Our parameters are the planets, the V_∞ levels, the inbound/outbound locations, and the $N : M(L_m)$ sequence. The planets and V_∞ levels are given by the contours of the Tisserand graph. The inbound/outbound locations are limited to II, IO, OI, and OO. We can create a scanning algorithm to identify viable VILTs for each permutation of the parameters. At the heart of the algorithm is a minimization procedure.

We first generate a list of candidate $N : M(L_m)$ sequences. The N and M values can be generated using the resonance procedure in Equations 4.4 through 4.7. The L_m values vary from 1 to M for each $N : M(L_m)$ pair.

For each parameter permutation, we wish to minimize the difference between the locations of the spacecraft and the planet when the spacecraft crosses the planet orbit on the return leg. More formally, we attempt to solve the optimization problem:

$$\begin{aligned}
\underset{\alpha}{\text{minimize}} \quad & \theta_{\text{miss}} = |\theta_{\text{sc}} - \theta_{\text{planet}}| && \text{(4.23a)} \\
\text{subject to} \quad & |\alpha - \alpha_0| \leq \delta, && \text{(4.23b)} \\
& |t_{\text{sc}} - t_{\text{planet}}| \leq \varepsilon, && \text{(4.23c)} \\
& \Delta V \leq \Delta V_{\text{max}} && \text{(4.23d)}
\end{aligned}$$

The ΔV constraint avoids impractical spacecraft performance solutions. Impractical flight durations are controlled by the revolution and flight time limits in the $N : M(L_m)$ selection process. The constraint on time helps to ensure that spacecraft and planet are not in the same location on different revolutions. The constraint on α recognizes the maximum bending angle relative to the arrival α_0 associated with the Tisserand graph node (or the previous vertex in the Tisserand network). The bounded, constrained optimization problem can be solved with an appropriate solution method such as SLSQP [68].

We arrive at a nested optimization problem where the error in the return condition will be minimized for each $N : M(L_m)$, for each inbound/outbound scenario, for each V_∞, for each planet in the network. The scanning logic is provided in Algorithm 2.

Figure 4.9 visualizes two solutions of the VILT algorithm at Venus between 6 $\text{km}\,\text{s}^{-1}$ and 8 $\text{km}\,\text{s}^{-1}$. The two solutions used different $N : M(L_m)$ sequences ($2:2(1)$ and $3:2(2)$). Both solutions assume outbound to inbound endpoints and are preceded by an Earth 4 $\text{km}\,\text{s}^{-1}$ gravity assist. For each transfer option, the large open circle symbol identifies the desired location of the planet at the end of the VILT and the small filled circle represents the actual location. The optimization procedure iterates on the departure pump angle to modify the apoapsis until the planet and spacecraft re-encounter at the desired V_∞. Alternate

Algorithm 2: VILT Scanning Algorithm

Data: *network*: A Tisserand network
ε: a tolerance on miss distance
Result: A nested data structure containing the viable VILTs

Algorithm `VILTscan`(*network*, ε)**:**

```
VILTS ←[ ]
foreach planet p_i in network do
    D ←[ ]
    foreach V_∞^- at p_i do
        C ← [ ]
        foreach V_∞^+ at p_i ≠ V_∞^- do
            B ← [ ]
            foreach io in [II, IO, OI, OO] do
                A ← [ ]
                foreach N : M(L_m) do
                    err ← MinimizeError(p_i, V_∞^-, V_∞^+, io, N : M(L_m))
                    if err ≤ ε then
                        A.insert(N : M(L_m))          ▷ retain this N : M(L_m)
                    end
                end
                B.insert(A)
            end
            C.insert(B)
        end
        D.insert(C)
    end
    VILTS.insert(D)
end
```

Function `MinimizeError`($p_i, V_\infty^-, V_\infty^+, io, N : M(L_m)$)**:**

Data: p_i: the planet
V_∞: pre- and post-V_∞ contours
io: endpoints
$N : M(L_m)$: resonance sequence
Result: The minimized position error at re-encounter
Apply optimization solver (SLSQP or similar) to Problem 4.23
err ← `Optimizer`
return *err*

solutions for the Venus 6 to 8 km s^{-1} VILT exist for different $N : M(L_m)$ sequences, different inbound/outbound endpoints, and different preceding gravity assists. These multiple solutions correspond to the nested loops in Algorithm 2.

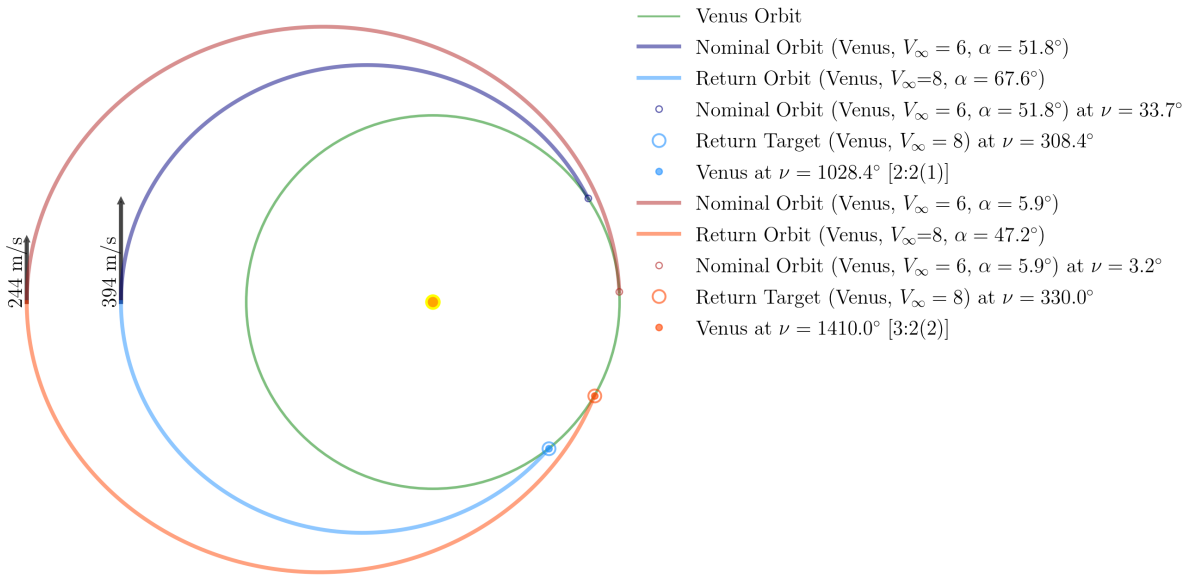


Figure 4.9. Solutions for a 6 km s^{-1} to 8 km s^{-1} VILT at Venus are shown above. The figure displays the nominal and return orbits for two $N : M(L_m)$ sequences with OI endpoints. Motion is counter-clockwise. The large and small circles in the lower right corner of the return orbits show the locations of the planet and spacecraft at the re-encounter. The solution algorithm ensures that the departure and arrival V_∞ are the precise discrete values from the Tisserand network.

Table 4.3 lists sample output from Algorithm 2. The table includes a subset of the potential VILTs connecting the Earth-4 vertices to some Venus vertices with a 6 km s^{-1} Venus gravity assist as a waypoint. This solution set limits the planet revolutions to three, the spacecraft revolutions to two, and the leveraging maneuver ΔV to no more than 600 m s^{-1}.

Table 4.3. Example VILT Scan Results

Source	Waypoint	Target	Endpoints	(N,M,L_m)	TOF (d)	ΔV (m/s)	θ_{miss} (deg)
E4	V6	V7	II	(1, 1, 1)	438	113	1e-06
				(2, 2, 1)	668	183	2e-07
				(2, 2, 2)	668	186	6e-09
				(3, 2, 1)	887	113	3e-06
				(3, 2, 2)	888	113	4e-07
			IO	(1, 1, 1)	329	410	2e-08
				(2, 2, 1)	561	445	2e-07
				(2, 2, 2)	564	463	4e-06
				(3, 2, 1)	721	179	8e-09
				(3, 2, 2)	721	181	8e-08
			OI	(1, 1, 1)	434	114	1e-07
				(2, 2, 1)	625	187	3e-07
				(2, 2, 2)	624	190	2e-07
				(3, 2, 1)	885	113	2e-06
				(3, 2, 2)	884	113	2e-06
			OO	(1, 2, 1)	444	472	2e-05
				(1, 2, 2)	444	506	5e-08
				(3, 2, 1)	680	182	3e-07
				(3, 2, 2)	680	184	8e-08
		V8	II	(1, 1, 1)	432	243	2e-06
				(2, 2, 1)	663	387	3e-07
				(2, 2, 2)	663	398	5e-07
				(3, 2, 2)	882	243	2e-05
			IO	(3, 2, 1)	724	375	4e-06
				(3, 2, 2)	726	385	7e-08
			OI	(1, 1, 1)	429	243	9e-07
				(2, 2, 1)	621	394	9e-07
				(2, 2, 2)	619	406	1e-07
				(3, 2, 2)	878	244	4e-07
			OO	(3, 2, 1)	685	381	1e-07
				(3, 2, 2)	685	391	1e-06
		V9	II	(1, 1, 1)	427	389	3e-05
				(3, 2, 2)	877	390	1e-06
			IO	(3, 2, 1)	728	586	2e-06
			OI	(1, 1, 1)	425	389	3e-05
				(3, 2, 2)	874	390	2e-06
			OO	(3, 2, 1)	690	594	5e-07

Each of the first five rows of Table 4.3 represent a parallel edge that can be added to the network between the vertices E4-I and V7-I. The next five rows are new edges connecting E4-I and V7-0, and so on. In general, the time of flight (TOF) and ΔV weights will vary on each new edge. The full set of results from Algorithm 2 would be much larger. In the extreme case, the *Source*, *Waypoint*, and *Target* columns would include each V_∞ contour in the Tisserand graph.

4.4 Network Model Summary

This chapter develops three models of common gravity-assist mission components for inclusion in the network architecture. In all cases, the output of the model identifies which Tisserand network vertices can be connected and how they must be weighted. The powered flyby fits naturally into the network approach. The main feature of the network powered-flyby model is the requirement for discrete increases in V_∞. The other two models are significantly more complicated.

A resonance model allows the Tisserand network to consider gravity-assist paths that require multiple consecutive flybys of the same planet. The key component of this model is an algorithm that identifies which pump angles will create resonances with manageable flight times. A method for sequencing multiple resonances creates several possible transfer times that may be used as parallel network edges. A variety of flight times increases the options available to the network for solving the scheduling problem. The resonance model can be extended to include half-revolution or odd-$\boldsymbol{\pi}$ transfers to add additional search capability.

Similarly, a VILT model adds many additional possible connections between gravity assists. The VILT targeting procedure is unique because of the need to return to the planet at a discrete V_∞. The line graph plays a key role in determining how a VILT can be integrated into the network vertex and edge structure. The VILT model presented in this chapter may be extended to different-body VILTs using similar techniques. This model may also provide a starting point for a general deep-space maneuver model.

5. NETWORK SEARCHES

A key benefit of the network model is the ability to apply standard search algorithms to identify gravity assist paths. The search algorithm chosen depends on the goal of the analysis.

In this research, we are mainly concerned with generating candidate initial-guess trajectories with little a priori knowledge of which gravity-assist bodies to visit or when flyby opportunities might occur. For example, we may wish to discover all Tisserand network paths that exist between Earth and a target planet in a given time frame. This type of search is sometimes called an All-Paths Search (APS). In contrast, many network problems are interested in finding the "shortest path". The APS is a more computationally expensive search than a shortest-path search.

Once we have generated all the possible paths through the network using an all-paths search, we can generate families of patched-conic trajectories seeded from those search results (Section 3.6). We can apply filtering at various steps in the process to discard search results or patched-conic trajectories that are not competitive for our mission goals.

This chapter takes a closer look at some network search algorithms for finding gravity-assist trajectories using the Tisserand network. Factors that affect the search time of these algorithms are also introduced and several methods to improve search performance are discussed.

5.1 Search Algorithms

A search for all paths through a network can be implemented with a Depth-First Search (DFS)[54]. In its most basic form, the DFS technique is a way to traverse a network and ensure that all vertices have been visited. The algorithm discovers a *spanning tree* as it proceeds through the network. With some modification, we can also compile a list of all the possible paths between a source vertex and a target vertex.

5.1.1 Depth-First Search

Let us first examine the depth-first search since it is foundational to the other methods to be discussed. Two important characteristics of the DFS are that it visits all vertices in a network and that it visits each vertex only once. The first characteristic means that we may use it to exhaustively search for paths, the second means that we will need to make modifications in order to find more than one path.

Algorithm 3: Basic Recursive Depth First Search

```
Data: A: an adjacency structure
      vertex: a starting vertex
      visited: a list of visited vertices
DFS(A, vertex, visited)                          ▷ Begin by calling DFS at vertex
Function DFS(A, vertex, visited):
    visited.insert(vertex)
    adj ← GetAdjacent(A, vertex)
    foreach adj_i in adj do
        if adj_i in visited then
            return                               ▷ Already visited this vertex
        else
            DFS(A, adj_i, visited)               ▷ Call DFS again
        end
    end
end
Function GetAdjacent(A, vertex):
    Data: A: an adjacency structure
          vertex: a vertex
    Result: A list of vertices adjacent to the input vertex
return adj
```

The procedure, shown in Algorithm 3, begins at a selected starting vertex and methodically visits each adjacent vertex. If an unvisited vertex is found, then the DFS immediately proceeds down to the next level (hence depth-first) by starting a new DFS with the unvisited vertex. This process continues until no further steps can be taken down the branch. At that point, the algorithm steps back up a level and continues with the next adjacent vertex. The algorithm lends itself to recursive implementations. Each vertex is marked as *visited* when

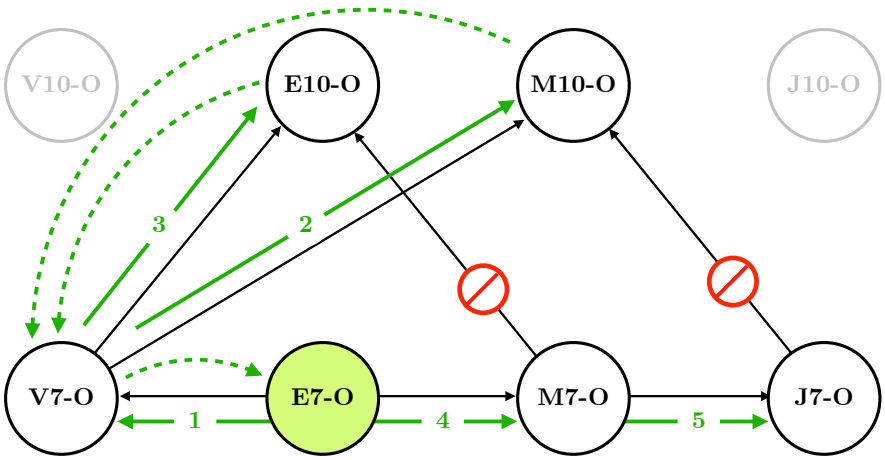
it serves as the source of the DFS. Vertices cannot be revisited in the basic DFS traversal. This prevents cycles or endless loops.

Figure 5.1 illustrates the DFS sequence in the context of the Tisserand network. We consider a small demonstration network consisting of a few V_∞ levels at Venus, Earth, Mars, and Jupiter in Figure 5.1a. Here, we suppose that, for some combination of flyby radii and date constraints, only the edges shown in black remain in the network. For simplicity, we only include the outbound encounter points.

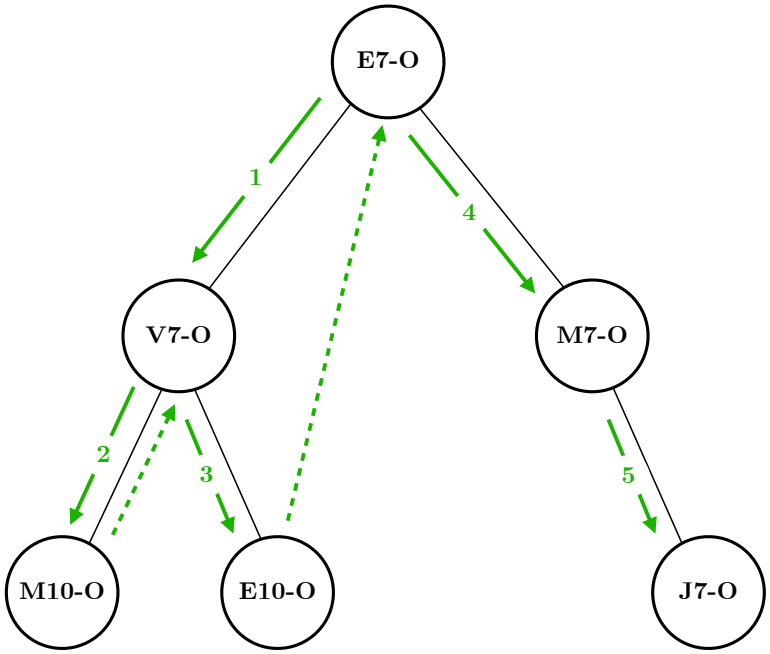
Let us trace the DFS algorithm through Figure 5.1a to see how each vertex is visited. The green arrows represent the progress of the algorithm with solid lines representing forward (or downward) advancement and dashed lines representing the backtracking that occurs at the end of a branch. Figure 5.1b shows the tree that the DFS discovers as it traverses the network.

The traversal begins at vertex E7-O. The vertex has two adjacent vertices (V7-O and M7-O). Which adjacent vertex is visited first depends on how the network is represented in memory. In this example, the DFS proceeds to visit V7-O where it marks the vertex as visited and begins a new search (we denote this recursion as DFS-V7). At V7-O, DFS-V7 discovers two more adjacent vertices (M10-O and E10-O). The algorithm visits each of these vertices in sequence. Neither M10-O nor E10-O have outbound edges to any adjacent vertices. So DFS-V7 marks those vertices as visited but does not make any more recursive calls. DFS-V7 has now completed visiting all neighbors of V7-O and it returns control to the original DFS. This completes the first branch of the tree in Figure 5.1b.

The main DFS algorithm (at E7-O) now proceeds to the next adjacent vertex (M7-O), marks it as visited, and starts a new recursion (DFS-M7). The DFS-M7 search finds two adjacent vertices (E10-O and J7-O). Importantly, since E10-O is marked as visited, the DFS-M7 search does not revisit E10-O and moves on to start another recursion at J7-O (DFS-J7). The DFS-J7 recursion finds only one outbound, adjacent vertex (M10-O) which has already been visited. Control is handed back to DFS-M7 and then the main DFS at E7. All vertices adjacent to E7-O have now been visited so the algorithm exits. The second branch of the tree in Figure 5.1b is now complete and the DFS has visited every connected vertex in the network exactly once.



(a) The network being searched with DFS.



(b) The DFS Search Tree.

Figure 5.1. The depth-first search traverses the Tisserand network by recursively visiting each unvisited, adjacent vertex. Black arrows represent edges in the network. Green arrows show the sequence visited by the algorithm. The DFS descends as far as possible through the adjacent vertices until there are no unvisited vertices on the current branch. At that point the algorithm returns to the previous level and continues the procedure.

The depth-first search algorithm requires some model of the network to perform the traversal. Typically, this is provided by an adjacency list structure or an adjacency matrix. An adjacency list is simply a list of all the vertices that are adjacent to a given vertex. The adjacency list structure includes an adjacency list for each vertex. Alternatively, an adjacency matrix is formed with rows and columns corresponding to each vertex. Element (i, j) is set to one if V_i is adjacent to V_j and set to zero if not.

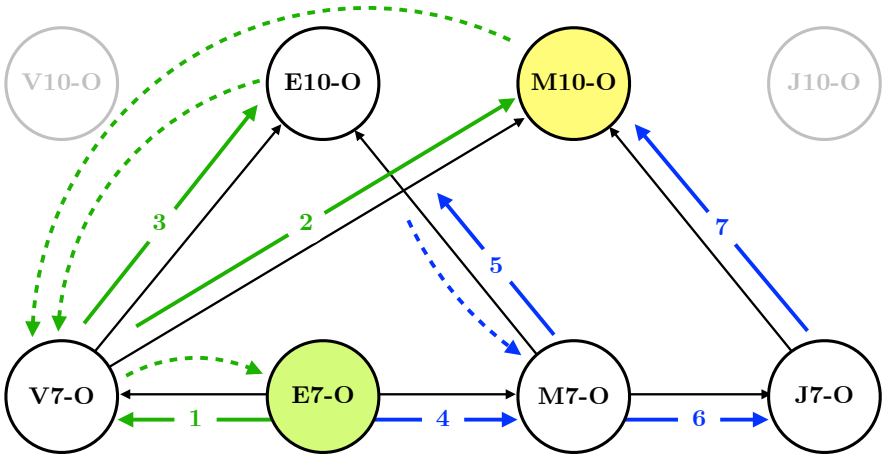
The DFS can be implemented recursively as shown in Algorithm 3. The auxiliary function `GetAdjacent` returns all vertices that are immediately adjacent to the input vertex using the adjacency lists or matrix, A. The algorithm also requires a single vertex to use as the starting point. The DFS has a time complexity of $O(V + E)$ if adjacency lists are used and $O(V^2)$ if an adjacency matrix is used [54].

5.1.2 All-Paths Search

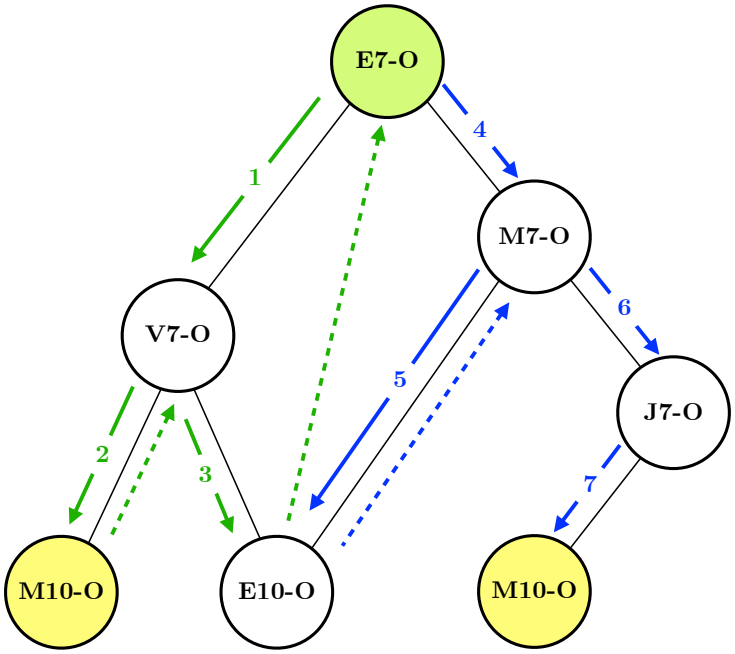
Algorithm 3 simply traverses the adjacent vertices in the network until all vertices have been visited. To adapt the DFS into an all-paths search, we must supply a target vertex in addition to the starting vertex. We also keep track of the history of vertices as we proceed. When we visit each vertex, we check to see if it is the target vertex. If so, then the steps taken to reach the vertex are recorded and the procedure continues. Importantly, as we exit the search centered at any vertex, we must remove it from the list of visited vertices so that it can be rediscovered along another path. Pseudocode for the `AllPaths` algorithm is provided in Algorithm 4.

Figure 5.2 shows how Figure 5.1 is adapted to perform an all-paths search. Figure 5.2a shows the same underlying network used in Figure 5.1a. Here we assume we are looking for paths from E7-O to M10-O. Figure 5.2b shows two paths (green and blue) found by the search.

We will trace the APS algorithm through Figure 5.2a to note the differences with the DFS search. We again start at vertex E7-O. The APS visits V7-O where it marks the vertex as visited and appends it to the current path. After checking that V7-O is not the target, the APS begins a new search. From V7-O, the APS visits vertex M10-O. This time the APS



(a) The network being searched with APS.



(b) The APS results.

Figure 5.2. The all-paths search traverses the Tisserand network by recursively visiting each adjacent vertex. Black arrows represent edges in the network. Green and blue arrows show the sequences visited by the algorithm yielding two distinct paths. The APS follows a DFS procedure but keeps a running list of vertices visited prior to the current vertex.

identifies M10-O as the target. M10-O is appended to the current path and that path (E7-O, V7-O, M10-O) is appended to the list of successful paths.

The new vertex M10-O is now removed from the current path and marked as **unvisited**. The APS proceeds to E10-O which, as before, is a dead end. The search backtracks to the original point (E7-O) removing E10-O and V7-O from the current path and marking them unvisited as well.

The search has now returned to its original state at E7-O with an empty current path and begins a new search at the remaining adjacent vertex (M7-O). Vertex M7-O is marked visited and added to the current path. The APS identifies E10-O as an unvisited, adjacent vertex. This is a key difference from the previous example where E10-O remained in the visited list (see Figure 5.1a). Vertex E10-O has now been visited twice. However, since E10-O is a dead end, the search returns to M7-O and unmarks E10-O again.

The APS next proceeds from M7-O to J7-O, adding it to the current path and labeling it visited. From J7-O the search proceeds to M10-O which is again identified as the target vertex. The current path (E7-O, M7-O, J7-O, M10-O) is added to the list of all paths. From here the search retreats through J7-0 and M7-O to E7-O, removing those vertices from the current path and marking them unvisited. Since there are no further adjacent vertices, the search is complete. The two search branches identifying the two successful paths are shown in green and blue in Figure 5.2b.

The act of marking the vertices as unvisited has two important consequences. First, it allows a vertex to be revisited later along a different path. Second, it increases the number of recursions. The time complexity can now be as bad as $O(V!)$. This worst case applies to a "complete" graph (one in which all vertices are connected to all others). In contrast, the original DFS algorithm has a time complexity of $O(V + E)$ or $O(V^2)$ depending on the implementation.

The All-Paths Search provided in Algorithm 4 includes an optional maximum path length, $D_{\max}$. This limit can protect against excessively long paths by preventing recursions past a maximum depth.

The APS starts and ends with specific vertices from the Tisserand network. The network vertices define not just a starting or ending body, but also the V_∞ and encounter location (E4-0, E6-I, etc.). Therefore, we will need to repeat the search for each vertex combination

of interest. Algorithm 1 from Chapter 3 provides the outer loop iteration to ensure that the search finds all paths between the two bodies.

Algorithm 4: Depth First Search for All Simple Paths

Data: A: An adjacency structure, *vertex*: a starting vertex
visited: a list of visited vertices, *target*: a target vertex
currentpath: current running path, *allpaths*: a list of successful paths
d: the current depth $D_{\max}$: the maximum allowable depth

Result: All simple paths between the starting vertex and the target

$visited \leftarrow [\,]$; $currentpath \leftarrow [\,]$; $allpaths \leftarrow [\,]$; $d \leftarrow 0$
`AllPaths(`A, *vertex, visited, target, currentpath, allpaths,* d, D_{max}`)`

Function `AllPaths(`A, *vertex, visited, target, currentpath, allpaths,* d, D_{max}`):`

```
adj ← GetAdjacent(A, vertex)
foreach adj_i in adj do
    if adj_i in visited then
        return                                   ▷ Already visited this vertex
    end
    d = d + 1
    visited.insert(adj_i)
    currentpath.insert(adj_i)
    if adj_i = target then
        paths.insert(currentpath)                ▷ Target found
        visited.remove(adj_i)                    ▷ Enable revisit
        currentpath.remove(adj_i)
        d = d − 1
        return
    else
        if d ≤ D_max then
            AllPaths(A, vertex, visited, target, currentpath, allpaths, d, D_max)
        else
            return
        end
    end
    visited.remove(adj_i)
    currentpath.remove(adj_i)
end
return allpaths
```

5.1.3 Bounded All-Paths Search

For our pathfinding problem, we sometimes know that there are practical limitations that make some paths infeasible. For example, there could be engineering or science constraints on the allowed time to arrive at the target planet. We can improve our search performance if we abort the current trial path in the depth-first search once we realize that we have exceeded this limit. This technique is generically called *branch-and-bound*. The `BoundedAllPaths` search in Algorithm 5 is a problem-specific variant of the DFS-based all-paths search that exploits this insight.

The `BoundedAllPaths` algorithm parallels the APS method but adds a set of auxiliary variables that accumulate the value of key weighting parameters as each edge is added to the path. Before proceeding to the recursive step, the cumulative weights of the next step are projected and compared against limits. If the resulting cumulative weights are within the allowable limits then the recursion proceeds and the weights are updated. If not, then the branch that would have started at the next vertex is effectively *pruned* from the search tree, providing an execution time savings. Pseudocode for this evaluation is included in Algorithm 6.

An example may help explain the usefulness of the modification to the basic search. Important values that accumulate over the course of a mission are the total time of flight and the total ΔV expended. Suppose we have a constraint of 15 years on the total mission duration related to the lifetime of some spacecraft component. While performing an APS, the `BoundedAllPaths` algorithm will use the flight times (assigned as weights of the network edges) to accumulate the total time of flight for each potential journey. Let us further suppose that after two recursion steps, the current mission duration is 10 years. When evaluating the adjacent edges, the `BoundedAllPaths` algorithm will now proceed only along edges with a time of flight less than five years.

Algorithm 5: Weight-limited Depth First Search

Data: A: An adjacency structure, *vertex*: a starting vertex
visited: a list of visited vertices, *target*: a target vertex
currentpath: current running path, *allpaths*: a list of successful paths
d: the current depth, $D_{\max}$: the maximum allowable depth
W: weighted parameters to be tracked, L: weight limits

Result: All simple paths between the starting vertex and the target

visited $\leftarrow$ [ ]; *currentpath* $\leftarrow$ [ ]; *allpaths* $\leftarrow$ [ ]; $d \leftarrow 0$
`BoundedAllPaths(`*A, vertex, visited, target, currentpath, allpaths, d,* D_{max}`)`

Function `BoundedAllPaths(`*A, vertex, visited, target, currentpath, allpaths, d,* D_{max}`):`

```
adj ← GetAdjacent(A, vertex)
foreach adj_i in adj do
    if adj_i in visited then
        return                                   ▷ Already visited this vertex
    end
    d = d + 1
    visited.insert(adj_i)
    currentpath.insert(adj_i)
    if adj_i = target then
        paths.insert(currentpath)                ▷ Target found
        visited.remove(adj_i)
        currentpath.remove(adj_i)
        d = d − 1
        return
    else
        W_e, proceed ← CheckLimits(vertex, adj_i, W, L, D_max)
        if proceed = True then
            W ← W_e
            BoundedAllPaths(A, vertex, visited, target, currentpath, allpaths, d,
              D_max)
        else
            return
        end
    end
    visited.remove(adj_i)
    currentpath.remove(adj_i)
    W ← W − W_e
end
return allpaths
```

Algorithm 6: Weight-limited Evaluator for Depth First Search

Function `CheckLimits(`*vertex, adj*$_i$*, W, L,* D_{max}`):`
- **Data:** *vertex*: a vertex
  - adj_i: an adjacent vertex
  - W: weights to be monitored
  - L: weights limits
  - $D_{\max}$: the maximum allowable depth
- **Result:** A boolean telling whether to proceed to adj_i
- $proceed \leftarrow True$
- **if** $d > D_{max}$ **then**
  - $proceed \leftarrow False$
- **end**
- **foreach** w_i *in* W **do**
  - $w_i \leftarrow w_i +$ `EdgeWeight(`*vertex,* adj_i`)` ▷ Accumulate weight
  - **if** $w_i > L_i$ **then**
    - $proceed \leftarrow False$ ▷ Cumulative weight exceeded
  - **end**
- **end**

return W, *proceed*

Function `EdgeWeight(`u, v`):`
- **Data:** u, v: two vertices
- **Result:** The weight of the edge (u, v)

return *weight*

Complete paths that violate a limit could, of course, be filtered out after the search when the total weight of the path is known. Some Tisserand graph nodes represent very high energy heliocentric orbits with long flight times, especially when inbound arrivals are considered. So evaluating the cumulative weights "as we go" speeds up the search by avoiding proceeding too far down an infeasible path.

5.1.4 Trace Search

Suppose we wish to search the network for all paths that follow a certain pattern. For example, we might wish to identify all paths in a given Tisserand network that use the sequence Earth-Venus-Jupiter-Saturn. The `trace` search in Algorithm 7 was developed for this specific problem.

A trace can be implemented with a depth-first search and a *queue*: a first-in-first-out data structure. At each recursion level we consult the queue for the next planet that we desire to see in the path. The adjacent vertices are checked to see if the vertex planet matches the planet at the front of the queue. The DFS proceeds only if the vertex is at the desired planet. The planet in the front of the queue is removed when the algorithm descends to the next level and the search continues with the following planet.

A use case for the trace is to look for expected paths that were not discovered in the all-paths search. If we perform a trace in an unfiltered network and the expected path is found, then we must have filtered out an edge or vertex required to construct the path. If the path is not found by the trace search then we know filtering is not to blame. In this case, it is possible that the discretization may be too coarse to find the expected path.

Algorithm 7: Trace Search

Data: A: An adjacency structure, *vertex*: a starting vertex
visited: a list of visited vertices, *pattern*: a target pattern
currentpath: current running path, *paths*: a list of successful paths

Result: All simple paths between the starting vertex and the target that follow the desired pattern

visited ← [ ]; *currentpath* ← [ ]; *paths* ← [ ]; *deque* ← pattern

`Trace(`*A, vertex, visited, pattern, paths*`)`

Function `Trace(`*A, vertex, visited, pattern, paths*`):`

- *adj* ← `GetAdjacent(`*A, vertex*`)`
- *next* ← *deque.pop* ▷ Get planet off the deque
- **foreach** adj_i *in adj* **do**
  - **if** adj_i *in visited* **then**
    - **return** ▷ Already visited this vertex
  - **end**
  - *visited.insert*(adj_i)
  - *currentpath.insert*(adj_i)
  - **if** *deque is empty* **then**
    - *paths.insert*(*currentpath*) ▷ Pattern complete
    - *visited.remove*(adj_i)
    - *currentpath.remove*(adj_i)
    - *deque.push*(*next*) ▷ Push planet back on deque
    - **return**
  - **else**
    - **if** $adj_i = next$ **then**
      - `Trace(`*A, vertex, visited, pattern, paths*`)` ▷ Call `Trace` again
    - **else**
      - **return**
    - **end**
  - **end**
  - *visited.remove*(adj_i)
  - *currentpath.remove*(adj_i)
  - *deque.push*(*next*)
- **end**

return *paths*, *deque*

5.1.5 Other Notable Algorithms

The other fundamental vertex visiting algorithm is the breadth-first search (BFS). In the BFS, all the vertices at the current level are visited before proceeding down to the next level. Both DFS and BFS are examples of a more general priority-first search. In these searches, a priority rule determines the next vertex to be considered. In the DFS, the most recently discovered vertices have priority. In the BFS, the oldest known vertices have priority [54].

The BFS is essentially a graph traversal algorithm like the DFS. A breadth-first search naturally finds the shortest path between two vertices in an unweighted network. Here, the shortest path is the path with the fewest edges. The BFS first visits all vertices that are one edge away from the starting point. It then visits all vertices that are two edges away. Accordingly, the first time BFS encounters the target vertex, it has found the shortest path.

Dijkstra's algorithm [69] is a priority-first search for finding the shortest path in weighted graphs (networks). In this search, priority is given to the neighboring vertex with the smallest weight (i.e., travel time). The shortest path to that neighbor is then set to the lesser of its current best and the cumulative distance along the current path. Using this method, the shortest path can be found through the network without needing to visit all vertices to determine the cumulative weight.

While not explored in depth here, other researchers have applied Dijkstra's algorithm to the general trajectory search problem [29], [32]. The algorithm is an attractive choice for trajectory designers who frequently wish to find paths that minimize the flight time or ΔV—parameters that accumulate along the trajectory. Early phases of the present research employed Dijkstra's algorithm for shortest-path searches [55].

However, because of the discrete nature of the network (discussed in Section 3.6), the total time of flight of a raw network path is ambiguous. Furthermore, unless powered flybys are explicitly included in the network (Section 4.1), the cumulative ΔV of any raw network path is zero. For these key parameters, the cumulative weights along any path through the network do not reliably predict the true cost of the trajectory (in either a patched-conic or ephemeris model). We must first construct a family of closed, patched-conic trajectories—based on each network path—to assess the cumulative weights.

For this reason, Dijkstra's algorithm has been replaced in this research with a two-step process. The first step uses an all-paths search to identify the possible network paths (within constraints). The second step constructs patched-conic trajectories from those network paths. These trajectories can be more accurately evaluated and compared. Chapter 8 includes recommended future work that may improve the utility of Dijkstra's algorithm for Tisserand network searches.

5.2 Limiting the Search

The complexity of the search algorithms will typically be a function of the number of vertices and edges in the network. In general, there are more edges than vertices in the networks for the gravity-assist problem. Algorithm 1 will add a $V_E V_t$ multiplier to the time complexity of the core `AllPaths` algorithm. Here V_E is the number of network vertices for Earth and V_t is the number of network vertices for the target planet.

In the Tisserand network, the number of vertices is determined by the number of planets considered and the density of the V_∞ contours in the associated Tisserand graph. More planets and more V_∞ levels will lead to more Tisserand graph nodes (V_∞ contour intersections). The Tisserand network then contains up to four vertices per Tisserand graph node as demonstrated in Section 3.2.1. There are roughly eight network edges for each Tisserand graph node (fewer for hyperbolic heliocentric orbits).

Figure 5.3 demonstrates the growth of the Tisserand network. The figure shows the number of vertices and edges in a reference Tisserand network as the discretization of the V_∞ levels is changed. The figure is for trend evaluation only, the values on the y-axes will vary depending on the particular planets, V_∞ contours, and dates used in the network construction. The reference network used in Figure 5.3 includes all of the solar system planets except for Mercury. The V_∞ contours span 3 $\mathrm{km\,s^{-1}}$ to 16 $\mathrm{km\,s^{-1}}$ and are discretized at various step sizes as shown along the bottom axis.

The top plot in Figure 5.3 shows the increase in the number of vertices in the network as the spacing between the V_∞ levels is decreased. This is directly related to the increase in the number of contour intersections (Tisserand graph nodes) as the number of contours increases.

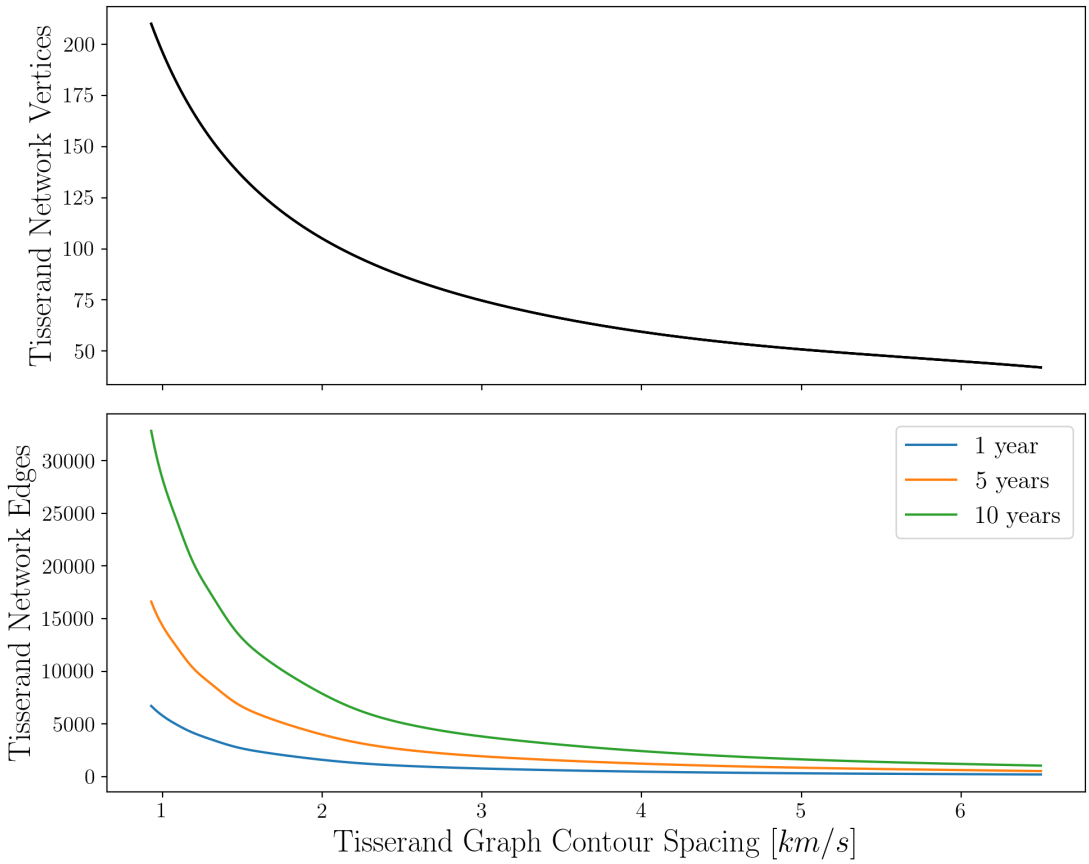


Figure 5.3. The size of the network increases as more V_∞ contours and more alignment dates are included. The number of vertices (top) and the number of edges (bottom) both increase as the spacing between discrete contours is decreased. The number of possible transfer dates only affects the edge count. The bottom plot includes the trend in edge count as the length of time allowed for planetary alignments is increased.

The network vertices are related to the energy problem in the traditional Tisserand graph and are not sensitive to the time frame for which the network is constructed. For this reason, the total size of the Tisserand network is dominated by the number of edges.

Using the same reference Tisserand network, the length of time over which planet alignments were searched was increased from one year to five years to ten years. See Section 3.4.1 for the method of finding planetary alignments. The number of edges will grow quickly when multiple alignment dates are considered. The inner planets, with shorter periods, will align frequently during a given time period. For each alignment epoch between two planets,

the number of edges between the vertices associated with those planets will double. The increasing trend in edge count as the encounter opportunities increase is readily observed in the three lines in the bottom plot of Figure 5.3.

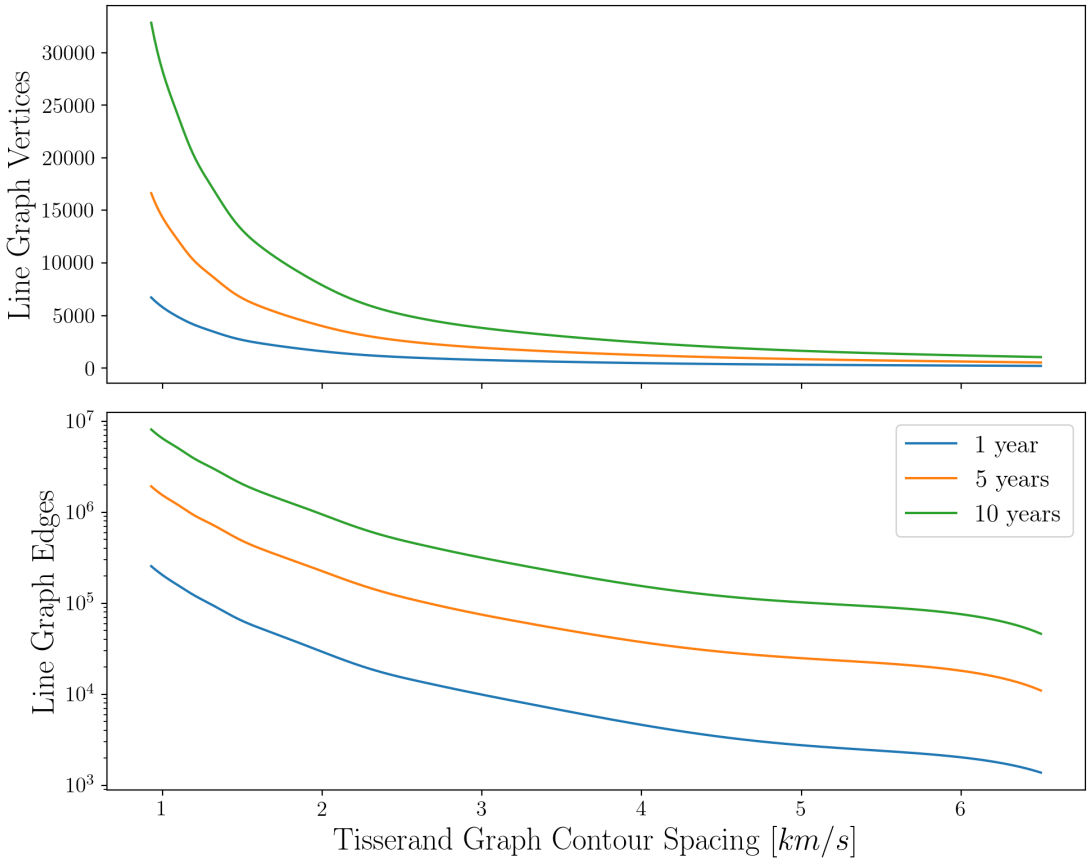


Figure 5.4. The size of the line graph increases as more V_∞ contours and more alignment dates are considered. The lower plot uses a log scale. The line graph size increases by orders of magnitude as the spacing between V_∞ contours is reduced and as the length of time allowed for planetary alignments is increased.

Finally, the transformation from the simple Tisserand network to the line graph of the network creates a vertex for each edge and an edge for each set of adjacent vertices in the original graph. This will magnify the growth trends just discussed. Figure 5.4 demonstrates this effect.

The upper plot of Figure 5.4 presents the trend in the total number of vertices in the line graph of the reference Tisserand network. This plot displays the same trend as the bottom

plot of Figure 5.3 because the line graph vertices are the edges of the original network. The bottom plot displays the trend in number of line graph edges over the same variation in V_∞ contour spacing and allowed alignment time (note the log scale on the lower plot). Figure 5.4 shows that the number of line graph elements can increase by several orders of magnitude (into the millions) for some choices of network design.

The DFS algorithms has a time complexity of $O(V + E)$ or $O(V^2)$ depending on the implementation. However, the complexity of the `AllPaths` algorithm can be as bad as $O(V!)$. This worst-case complexity assumes a complete graph; one in which every vertex is connected to every other vertex. The networks we construct will be much more sparse than a complete graph. However, we can expect performance to be slow for well-connected networks. The lower plot in Figure 5.4 shows the potential for a large number of edges (a rough indication of how well the vertices are connected).

With the caveat on completeness, $O(V!)$ algorithms are "inefficient" in computational terms and can be expected to scale poorly. The quick growth in the edge count, E, for reasonable search problems can lead to long computation times. Our searches will be performed in the line graph, which shows the potential for a large number of edges. We should therefore examine ways to reduce the edge count to improve performance.

5.2.1 Network Design Choices

We may improve search performance by eliminating vertices and edges where possible. The analysis in the previous section has shown that reducing the number of V_∞ contours and the time period over which alignments are assessed will reduce the number of elements in the network. Careful consideration of some parameter choices prior to building the Tisserand network can moderate the size.

There is no requirement that each planet use the same array of V_∞ contours. Removing unnecessary vertices will have a large effect on network size by eliminating all edges that connect those vertices. For some mission concepts, we may wish to avoid high V_∞ contours at Earth because the departure C_3 is unreasonable. For other concepts, we may wish to keep these higher V_∞ Earth vertices for use in VILTs.

We have seen that shorter alignment time frames reduce network size by limiting the number of parallel edges between vertices (representing different transfer windows). The resonance and VILT models in Chapter 4 also introduce parallel edges. Limiting the vertices where resonant transfers are considered (as well as limiting the number of the possible resonances) will reduce the number of parallel edges between vertices. Similarly, constraining the bodies and V_∞ levels where V-infinity Leveraging Transfers (VILTs) are permitted will reduce parallel edges.

Finally, for one-way missions, we will likely want to avoid adding edges that originate from the destination planet. Requiring that the destination only have edges directed *to* the planet will prevent a situation where the destination planet is found and then departed and then found again at a different vertex. If we remove the edges originating at the destination, then once the destination is reached, there is no where else to go. Figure 5.5 displays the preliminary Tisserand network used for the demonstration in Chapter 3. The network does not include edges from any Uranus vertex (rightmost column) to any other planet.

5.2.2 Pre-Search Filtering

There is an additional opportunity to reduce the size of the network after it is built but before we begin searching. The most important of these filters is one that removes line graph edges with excessive encounter date disagreements. This procedure was described in Section 3.4.3. The filtering principle is to compare the arrival date of one network edge with the departure date of an adjacent edge. If the encounter dates are not within some tolerance, then the entire sequence can be removed. This type of comparison can be readily performed in the line graph. An iteration over each edge of the line graph is required. The tolerance can be a raw number or some function of the gravity-assist parameters in the line graph edge, for example: a fraction of the time of flight or a fraction of the period of the encounter body.

Experimentation with various tolerance levels has shown that even very loose tolerances on encounter time are effective at filtering out roughly 99-percent of the line graph edges.

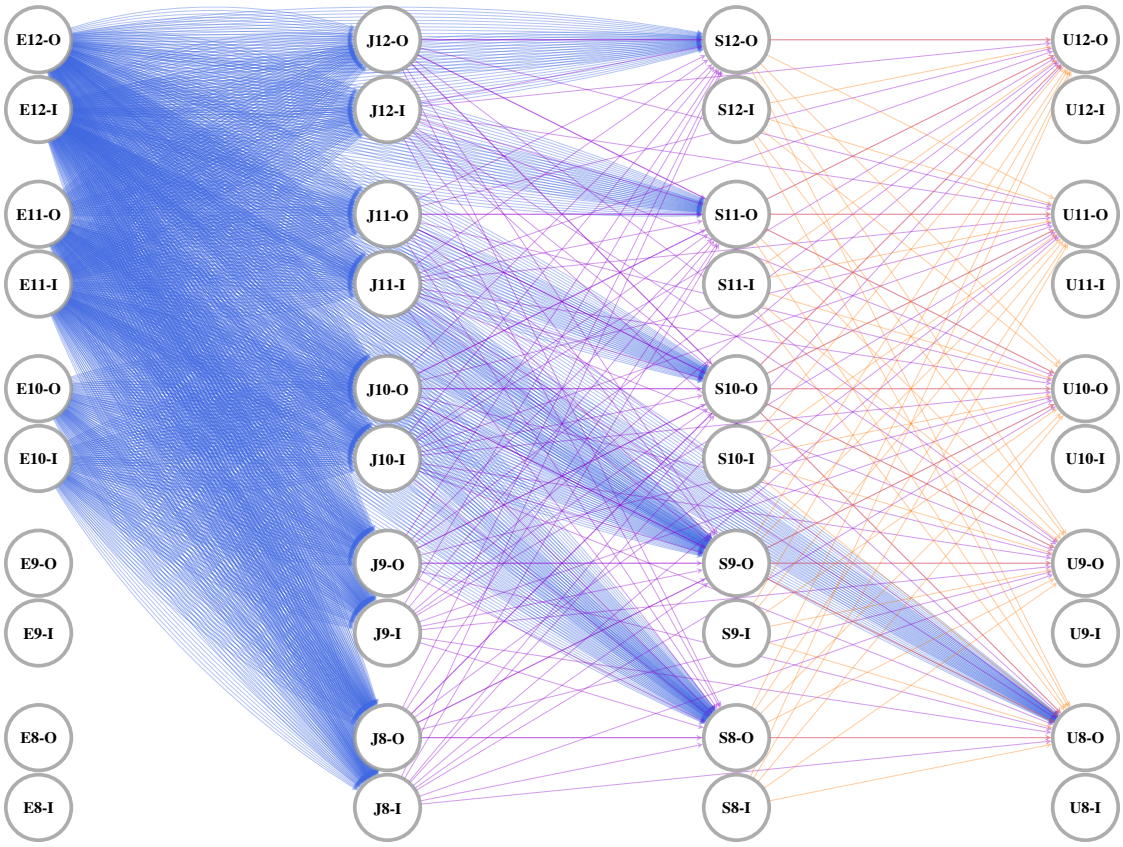


Figure 5.5. This unfiltered Tisserand network includes many parallel edges corresponding to the multiple dates on which a transfer is possible. Edges are colored by the departure planet. Departing edges from Uranus are not included during construction. This figure is provided to illustrate the network connections and is not intended for tracing solutions. For color viewing or inspection of the vertex labels, the reader is referred to the electronic version of this document.

This is perhaps not surprising. Referring back to Figure 3.13, we see that most of the two-planet trajectory legs are not connectable.

Figure 5.6 provides a more rigorous examination of the encounter-time filtering. The 1-year family of reference networks used in Figures 5.3 and 5.4 was filtered to remove edges with encounter-date discrepancies greater than 10-percent of the planet's orbital period. This filter still retains very large time discrepancies for the outer planets. Figure 5.6 compares the number of line graph edges before and after the filter application. The procedure is effective

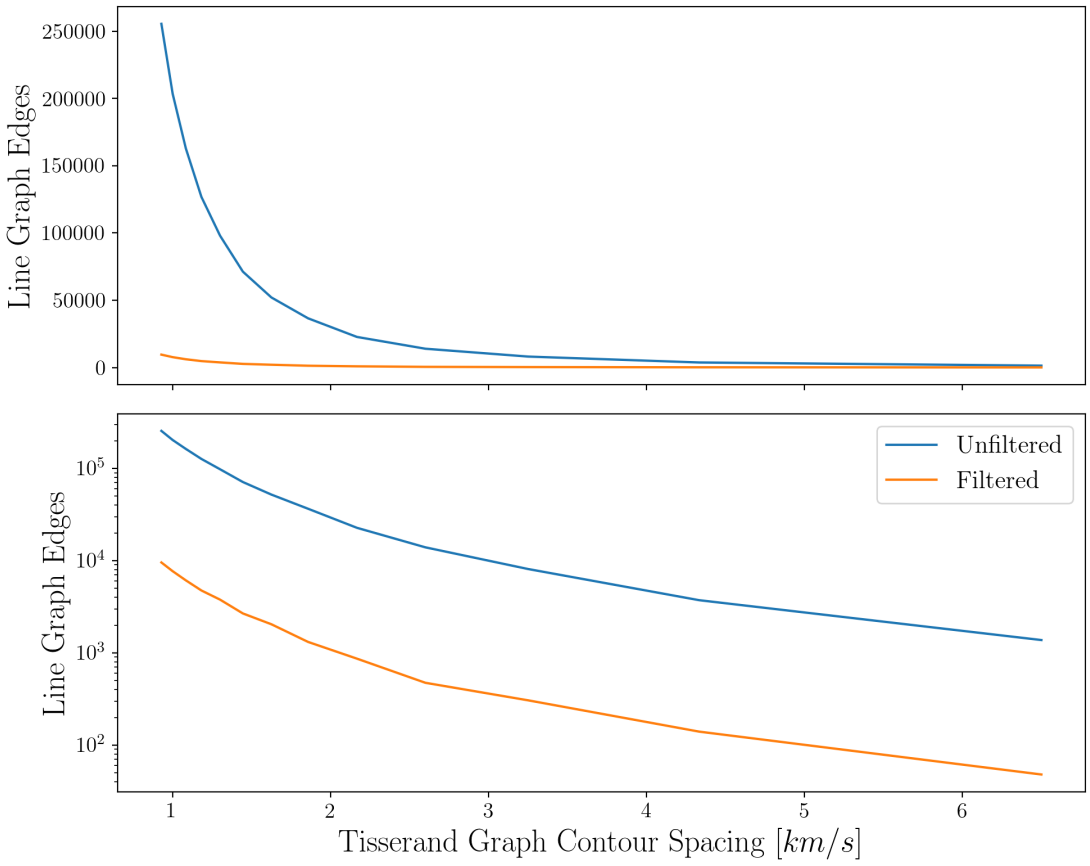


Figure 5.6. Encounter-date filtering can mitigate the growth in network size as more V_∞ contours are included. Here, a loose constraint of 10-percent orbit period consistently reduces the number of edges by roughly two orders of magnitude for the 1-year family of networks in Figures 5.3 and 5.4. The upper plot uses a normal scale and the lower plot uses a log scale.

at reducing the number of edges by roughly two orders of magnitude across the range of V_∞ spacing. This result is in line with the anecdotal experience showing a 99-percent reduction in edges.

We can also iterate through the line graph to find any edges with a non-permissible time of flight. For example, some individual legs may require flight times that exceed the total allowable mission time of flight. High-energy heliocentric orbits that encounter one of the outer planets on an inbound approach frequently have a very large semi-major axis and flight times measured in centuries. Such edges should be removed to prevent needless consideration

during the search algorithm execution. This logic can also be used to remove intermediate hyperbolic, heliocentric orbits that have an infinite time of flight.

Regardless of which resonances were included as edges in the Tisserand network (Chapter 4), we may wish to additionally limit the number of consecutive flybys that can occur at any planet. Edges requiring an excessive number of repeat visits can be filtered from the network.

Similarly, using the line graph, we can remove consecutive powered flybys of the same planet. In the Tisserand network, a powered flyby is modeled as an instantaneous jump between vertices with different V_∞ levels at the same planet. Two consecutive powered flyby vertices are equivalent to one powered flyby with the total change in V_∞.

In most cases, it is necessary to first build a Tisserand network including all of the potential edges and then construct the line graph of that network. The tests needed to determine if an edge should be removed frequently need the multi-transfer information only available from the line graph.

Figure 5.7 shows the result of applying these filters to the demonstration Tisserand network in Figure 5.5. The number of edges has been reduced considerably.

5.2.3 Search Parameters

We may also improve search performance by controlling the parameters of the search. Most obviously, we can influence the execution time of Algorithm 1 by limiting the number of starting and ending vertices in the nested loops. For example, we should limit the departure vertices to those that are compatible with the launch C_3. Capture conditions on the arrival planet may similarly limit the target vertices considered.

Mission timing requirements can also reduce the breadth of the search. The departure date, D_{depart}, is encoded with each network edge for encounter date comparisons. Some departure dates may be acceptable for an Earth gravity assist but may be too early or too late to use as an Earth launch date. We can enforce a launch window to mask some vertices from consideration as the starting point of the all-paths search.

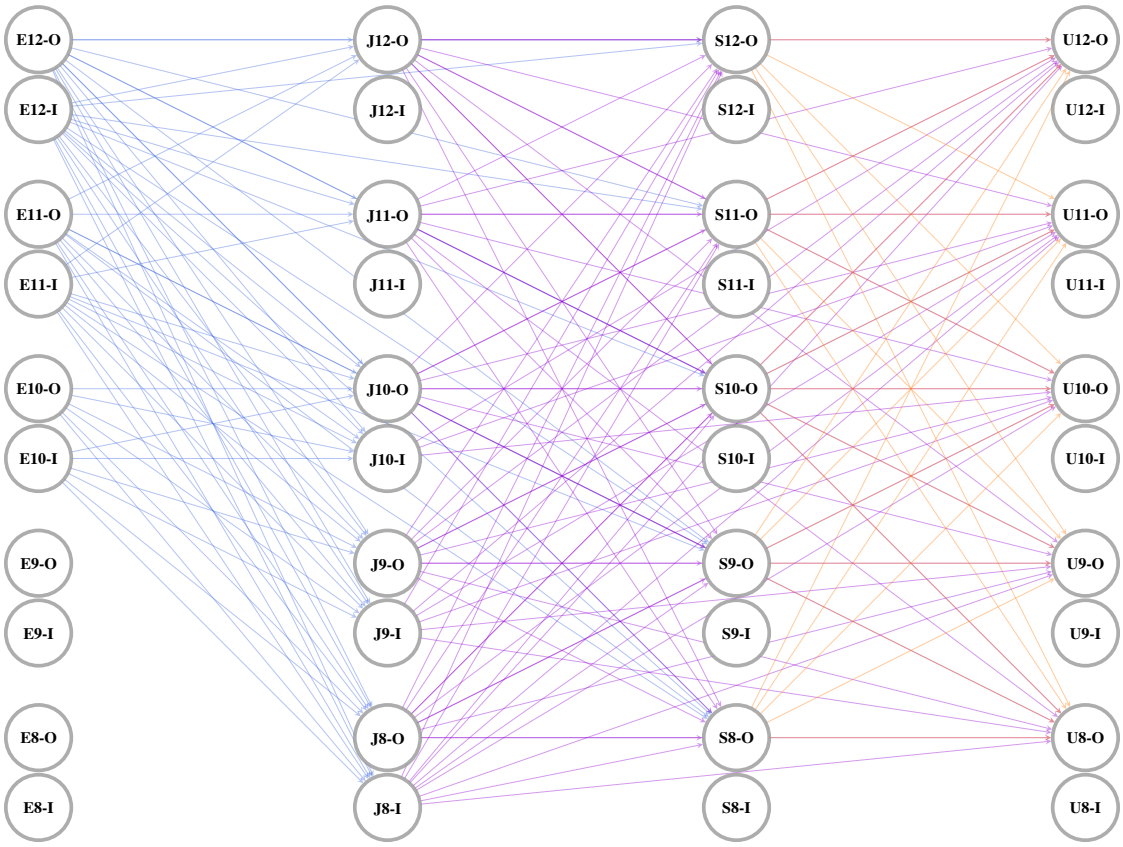


Figure 5.7. The Tisserand network in Figure 5.5 has been filtered to remove non-connecting edges, multiple flybys, and unacceptably long flight times. This figure is provided to illustrate the network connections and is not intended for tracing solutions. For color viewing or inspection of the vertex labels, the reader is referred to the electronic version of this document.

All-paths search implementations commonly allow a limit on the number of vertices to include in a path. This is, effectively, a limit on the allowable depth of a depth-first search. For the Tisserand network, this parameter limits the total number of gravity assists that are allowed in the path. A reasonable maximum number of flybys will speed the search and also provide protection against paths with very long flight times.

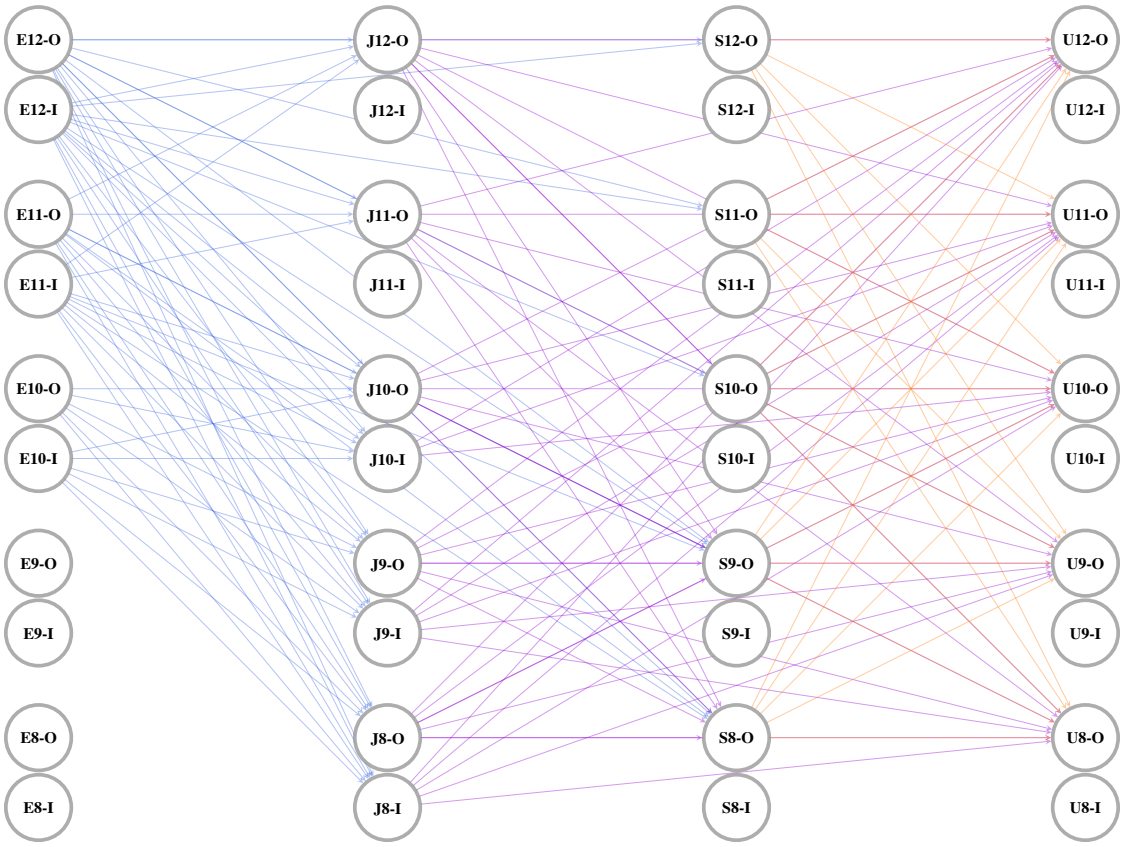


Figure 5.8. The Tisserand network in Figure 5.7 has been filtered to include only edges between vertices that appeared in a search through an undated Tisserand network. This figure is provided to illustrate the network connections and is not intended for tracing solutions. For color viewing or inspection of the vertex labels, the reader is referred to the electronic version of this document.

5.2.4 Pre-Searching with an Energy Network

Another technique that can simplify the network prior to searching is to perform a preliminary search of an undated Tisserand network (such as the network in Figure 3.5). We first retain an energy-only version of the Tisserand network (prior to adding the many parallel edges for the scheduling problem). This version of the network will necessarily have many fewer edges. An all-paths search through this network will yield all paths that are feasible from an energy perspective. This is typically the end goal of a traditional Tisserand graph search.

Since any useful path must satisfy both the energy and scheduling requirements, we can use the energy-only, all-paths search to filter any energy-infeasible vertices and edges in the full Tisserand network. Any vertex that does not appear in a path from the energy-only search is not useful and can be removed from the full network. The energy-only Tisserand network may only consist of a few hundred components and can be searched quickly. Experience with these searches has shown that the speed improvement is frequently worth the cost of performing two searches. However, in many cases, the overhead cost of the energy search is not justified when we also apply the transitive closure technique (Section 5.2.5) to the main search.

Figure 5.8 shows the Tisserand network from Figure 5.7 after this additional energy filter has been applied. An Earth to Uranus search was performed in an energy-only network. The results of that search identify a set of energy-feasible routes through the network. We obtain Figure 5.8 by removing any edges whose adjacent vertices did not appear in one of the routes from the energy-only search. Visually, the energy-only network appears very similar to Figure 5.7. However, close comparison will show that several additional edges have been removed (most noticeably between Jupiter and Saturn).

5.2.5 Transitive Closure

A directed network's transitive closure identifies all vertices that can be reached from each vertex in the network. The transitive closure may be computed with a DFS-based algorithm with $O(V(V+E))$ time complexity for sparse networks or $O(V^3)$ complexity for dense networks. For sparse networks, we simply perform a DFS ($O(V+E)$) at each of the V vertices. For dense networks, Warshall's algorithm is also easy to implement [54].

Let us first make an auxiliary network by copying all the vertices from our Tisserand network and leaving out all the edges. We then perform a DFS in the original Tisserand network. If we find that some path exists from vertex, u to vertex v, then we add an edge directly from u to v in the auxiliary network. This auxiliary network is the transitive closure of the original network.

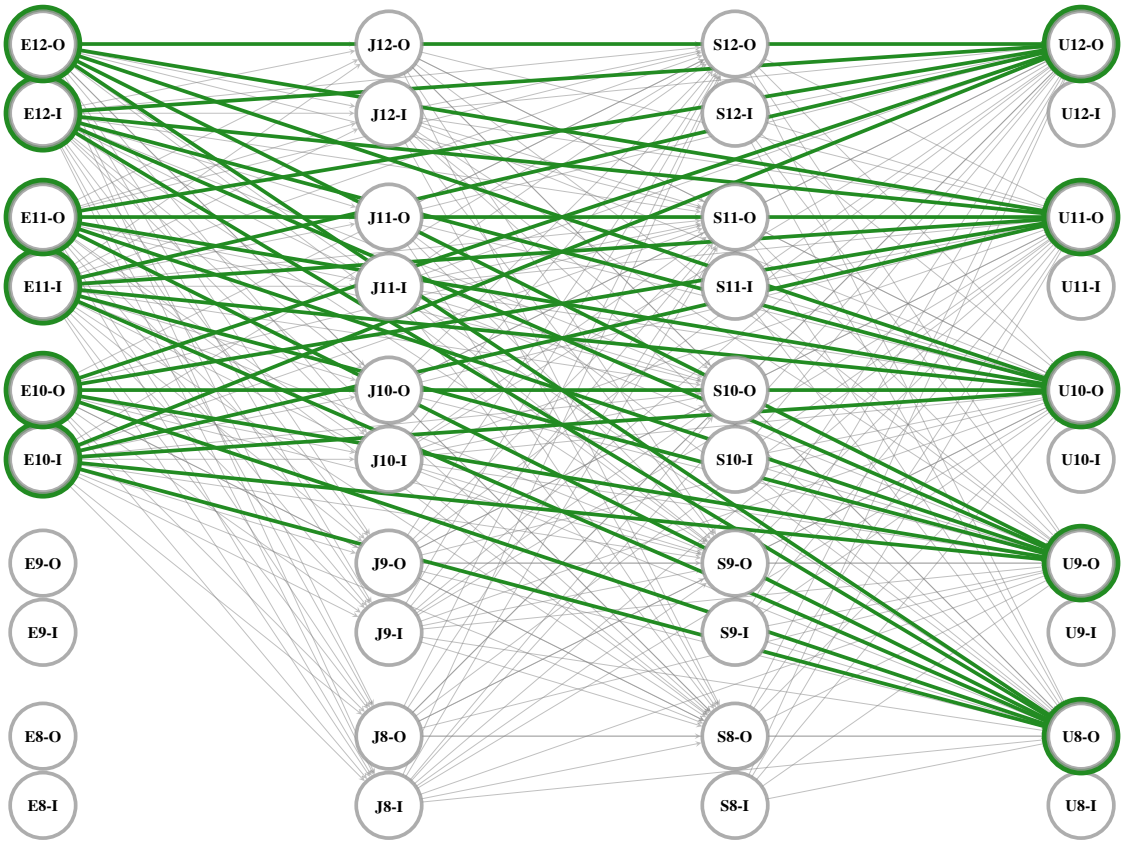


Figure 5.9. The transitive closure of a network directly connects any two vertices that can be connected by a path through the original network. The transitive closure of the filtered sample network from Figure 5.7 is shown here. The edges between Earth and Uranus are highlighted. These are the only endpoints that need to be used in Algorithm 1.

We need only compute the transitive closure once for any given network. Once computed, we have a tool that can tell us, for any starting vertex, which other vertices are reachable. This information may be used to limit the end points, v_1 and v_2, in Algorithm 1. There is no benefit in searching between endpoints that we know are not connected through the intermediate vertices in the network.

Figure 5.9 shows the transitive closure of the sample Tisserand network from Chapter 3. The gray and green lines directly connect vertices that can be connected by at least one path in the base Tisserand network. Figure 5.9 is provided for illustration only. The actual

endpoints used in our searches will be the corresponding line graph vertices which do not lend themselves to an intuitive visualization.

The green lines in Figure 5.9 highlight the connections between Earth and Uranus. By using the highlighted vertices in Algorithm 1, we are guaranteed to search between all the endpoints where paths from Earth to Uranus exists and only those endpoints where paths exist. We still must perform the `AllPaths` search to find the actual paths. The transitive closure only tells us that some path exists.

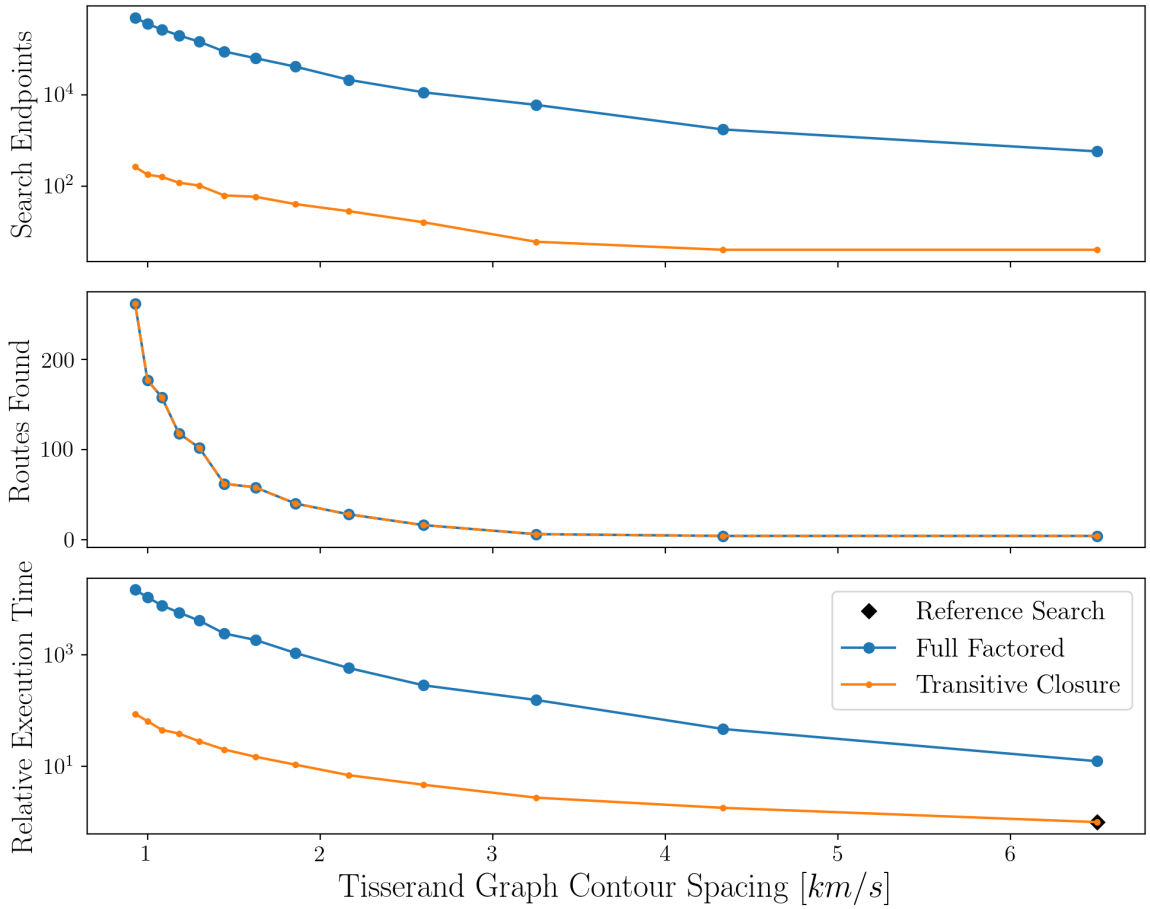


Figure 5.10. The transitive closure improves the efficiency of the search. Here, the transitive closure reduces the number of endpoints to be searched for the 1-year family of networks in Figures 5.3 and 5.4. The upper plot shows the reduction in endpoints. The bottom plot shows the resulting reduction in relative execution time. The least dense network (marked with a diamond) requires an execution time of one. The middle plot shows that both methods return the same number of solutions.

Figure 5.10 shows the trend in the number of endpoints searched in Algorithm 1 for the 1-year family of reference networks. Similar trends are observed for the other families. The top plot shows the number of starting and ending vertices used as endpoints in the APS for both a full-factored permutation and the transitive closure. Both techniques yield the same number of solution paths (middle plot) but the full-factored method examines more branches and requires more time.

The full-factored approach includes all combinations of Earth vertices and target vertices in the demonstration network. The transitive closure method uses only those Earth and target vertices that are connected in the transitive closure of the network. The transitive closure method consistently examines orders of magnitude fewer endpoints.

The bottom plot in Figure 5.10 shows the resulting improvement in execution time. Here, the quickest and least dense network (marked with a diamond in the figure) is searched in one unit of time. As the network becomes more dense, the figure shows the execution time growing more slowly for the transitive closure method. For the full-factored method, the most dense network requires about 1000 times the execution time of the least dense network. For the transitive closure method, the most dense network only requires about 100 times the execution time of the least dense network.

If we directly compare the two endpoint generation methods, we find that, for the sparsest network, the full-factored method requires about ten times the execution time of the transitive closure method. But for the most dense network, the full-factored approach requires about 100 times the execution time.

While not examined in this work, an additional savings can be expected by using a transitive closure test as an internal bound in the APS. Specifically, we might check the transitive closure for a connection between the next prospective vertex and the set of vertices associated with the target planet. If no connection exists then we can avoid proceeding down that branch.

5.3 Network Search Summary

The depth-first search serves as a basis for the network searching algorithms presented in this chapter. The `AllPaths` algorithm is a standard routine used to find all possible paths between two vertices. Two additional DFS-based algorithms `BoundedAllPaths` and `Trace` were developed to answer specific needs in this research.

The `AllPaths` and derivative algorithms, while popular, are not efficient and can be expected to scale poorly. Moreover, the networks for multiple gravity assist searching can grow in size exponentially. The large numbers of edges that can result from reasonably designed searches motivate the use of some network simplifications. These techniques are effective in decreasing search time without compromising results.

6. VALIDATING THE METHOD ON HISTORICAL MISSIONS

We choose some historical gravity-assist missions to test whether the network technique can discover trajectories that are known to exist. Each mission adds a new design technique and tests a different model. We will compare the network search results to approximations of the actual mission trajectories.

6.1 Wander Software

The algorithms and models described in the previous chapters are implemented in a new design tool named **Wander**. Because the Tisserand network includes the Tisserand graph and other astrodynamical information, **Wander** can produce search results with a very simple set of inputs. The information needed to build the Tisserand network includes the possible gravity-assist planets and their V_∞ levels, the date range for planetary alignments, and a tolerance on encounter time accuracy. With the Tisserand network defined, the only remaining input for a gravity-assist trajectory search is the target planet.

Table 6.1. Wander Required Search Parameters

Parameter	**Definition**	**Example**
Bodies	Available gravity-assist bodies	Venus, Earth, Jupiter
V_∞ range	V_∞ spacing for each body	4:1:8 $\mathrm{km\,s^{-1}}$
Start Date	Earliest date for planetary alignment	Jan. 1, 2023
End Date	Latest date for planetary alignment	Dec. 31, 2043
Date Tolerance	Allowable mismatch in encounter	5% time of flight 10% gravity assist body period 5° misalignment

Table 6.1 summarizes the parameters that define the Tisserand network. Once a network is built it can be stored and recalled for later use. In the simplest case, a mission designer can find thousands of patched-conic trajectory options by providing only two inputs: a destination planet and a Tisserand network in which to search. The Tisserand network can be treated as a model of the search space containing all the necessary two-body dynamical information.

Several optional parameters can be provided to control the performance of the search algorithm. These options are summarized in Table 6.2.

Table 6.2. Wander Optional Search Parameters

Parameter	**Definition**	**Example**
Max Flyby Count	Maximum total gravity assists	6
Max Repeat Visits	Maximum consecutive flybys of same planet	2
Max Time of Flight	Maximum time to target planet	15 y
Max ΔV	Maximum total mission ΔV	4 $\mathrm{km\,s^{-1}}$
Encounter Windows	Acceptable date ranges for planet encounters	2030 - 2035
Launch Window Close	Latest date for departure ΔV	Dec. 31, 2030

Importantly, and in contrast to other search tools, the path to be considered does not need to be defined. Only the endpoints (i.e. Earth and the target planet) are required. The network approach is designed to automatically consider all possible intermediate sequences.

Wander also manages the search results. The nested search results described by Algorithm 1 need to be unpacked and translated from a list of abstract data-structure addresses into gravity-assist paths that can be interpreted by a mission designer. The post-processing collects the sequences of vertices and edges, and organizes them into paths (sequences of planets visited), routes (Tisserand network vertex sequences), and date variants (routes that repeat at different dates). These objects can then be summarized, plotted, and compared.

Wander uses the open-source, graph package *graph-tool* for the low-level graph management such as organizing the vertices and edges of the Tisserand network [70]. The grid-like visualizations in this work (such as Figure 3.5) are also created with the assistance of this package. All astrodynamical calculations described in this work are directly included in the **Wander** code base with one notable exception. The Lambert problem is solved with Izzo's algorithm [58] which is imported from the open-source astrodynamics library *pykep* [71].

We now demonstrate the effectiveness of the Tisserand network technique by performing some trajectory searches with **Wander**. We will attempt to find the paths used by Voyager 1, Voyager 2, Galileo, and Cassini. We include here a detailed discussion of the Voyager 2 search. For the other missions we simply provide search inputs and summary results. The detailed discussion of these missions in included in Appendix C.

6.2 Voyager 1 Search

The Voyager 1 mission included flybys of Jupiter and Saturn [72], [73]. The mission provides a fairly simple gravity assist example with known parameters. A complete accounting of actual V_∞ values is typically not available in the public documentation. Therefore, the V_∞ listed in Table 6.3 are approximations derived from Lambert solutions using the positions of the planets in the JPL SPICE ephemerides at the encounter dates. We will not attempt to model any trajectory correction maneuvers that occurred on the actual mission.

6.2.1 Voyager 1 Actual Trajectory

Voyager 1 launched September 5, 1977 on a Titan IIIE-Centaur. The departure ΔV for the Earth to Jupiter leg was provided by the Centaur upper stage and Star 37E solid booster [72]. For comparison with the Tisserand network results, Table 6.3 provides a patched-conic reconstruction of the Voyager 1 encounters based on the published flyby dates.

Table 6.3. Voyager 1 Encounters

Encounter	**Planet**	**V$_\infty$** ($\mathrm{km\,s^{-1}}$)	**Date**
Launch	Earth	10.3	Sep. 5, 1977
1	Jupiter	10.9	Mar. 5, 1979
2	Saturn	15.3	Nov. 12, 1980

6.2.2 Voyager 1 Search Parameters

Table 6.4 summarizes the key inputs used to construct a network for the Voyager 1 problem and search for trajectories. The V_∞ notation 7:1:12 $\mathrm{km\,s^{-1}}$ means the Tisserand network includes V_∞ levels from 7 up to and including 12 $\mathrm{km\,s^{-1}}$ in 1 $\mathrm{km\,s^{-1}}$ increments. The dates in Table 6.4 give boundaries on the alignment dates of the planets (described in Chapter 3). The actual departure or arrival dates may fall outside this window.

Table 6.4. Voyager 1 Search Parameters

Parameter	**Value**
Earth V_∞	7:1:12 $\mathrm{km\,s^{-1}}$
Jupiter V_∞	7:1:12 $\mathrm{km\,s^{-1}}$
Saturn V_∞	9:1:16 $\mathrm{km\,s^{-1}}$
Alignment Start Date	Jan. 1, 1977
Alignment End Date	Dec. 31, 1980
Max Flyby Count	2
Max Repeat Visits	0
Max Time of Flight	4 y
Date Tolerance	10% time of flight

6.2.3 Voyager 1 Results

Table 6.5 summarizes the results of the Voyager 1 search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory. Recall from Chapter 3, a *route* differentiates the particular vertices in the Tisserand network that are passed through along a path (e.g. E10-O, J7-O, S9-O). A *date variant* identifies a duplicate of a route with some difference in the encounter dates. The *Network Routes* and *Date Variants* columns list the number of these occurrences for each path.

The *Launch* V_∞, *Total* ΔV, *Launch Window* and *Arrival Window* columns summarize the extremes of the randomized patched-conic trajectories. The total ΔV is the cumulative propulsive ΔV required to complete gravity-assist turning at any of the encounters after Earth departure. The date columns give the range of departure or arrival dates in month/year format.

Table 6.5. Voyager 1 Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
JS	178	341	4 - 15	0 - 5	06/77 - 12/79	09/79 - 12/83
S	10	40	11 - 15	—	08/76 - 12/79	09/78 - 04/84

Because of the discrete composition of the network and the tolerance in encounter dates, the search results include gaps in time and energy and only represent outlines of closed trajectories. Multiple methods for constructing patched-conic trajectories from these outlines were discussed in Section 3.6. In this case, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution using the procedures outlined in Section 3.6.2. Figure 6.1 shows the patched conic trajectories along the JS path. The launch V_∞ and propulsive ΔV (if any) required for each of the patched-conic trajectories can be computed from the Lambert solutions as described in Chapter 3. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ propulsive ΔV have been excluded from the summary in Table 6.5.

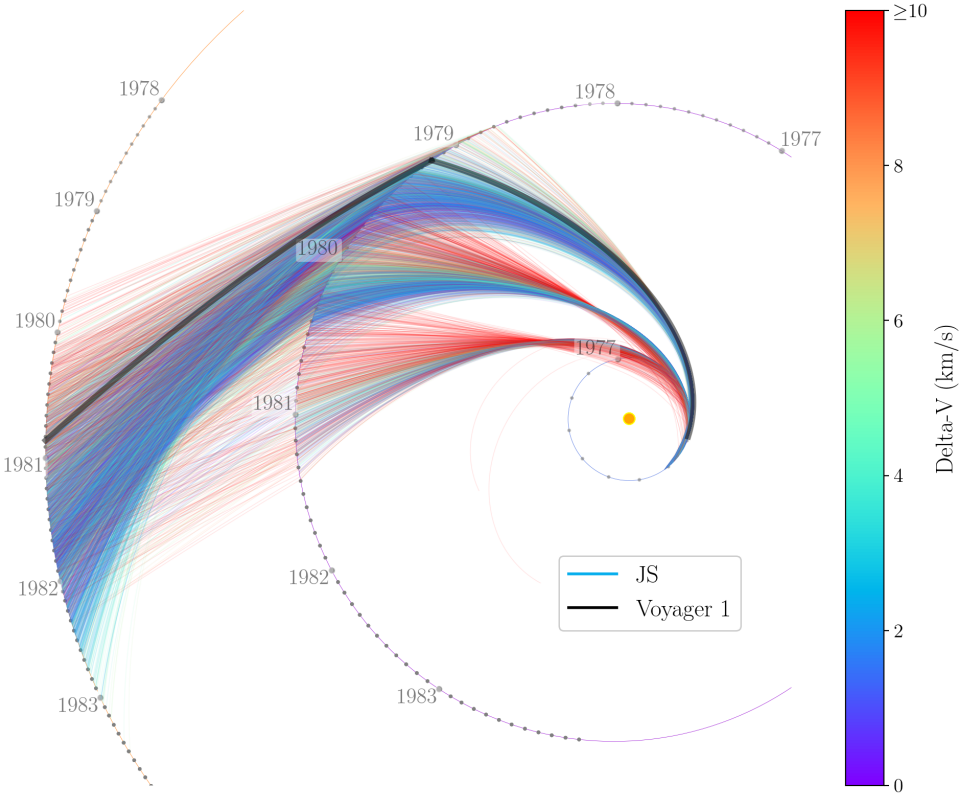


Figure 6.1. The Tisserand network solutions provide the outline for a collection of patched conic trajectories shown here. Only the Earth-Jupiter-Saturn sequences are shown. A patched conic approximation of the true Voyager 1 trajectory is highlighted.

6.3 Voyager 2 Search

The Voyager 2 Grand Tour is the archetypal multi-gravity assist mission. Launched August 20, 1977, the mission provides a good test case for the Tisserand network requiring four linked flybys. We search for a trajectory leaving Earth in late 1977 and arriving at Neptune in 1989. The trajectory should include gravity assists at Jupiter, Saturn, and Uranus. This path is given the name JSUN.

6.3.1 Voyager 2 Actual Trajectory

For reference, we generate a patched-conic version of the Voyager 2 trajectory from the published flyby dates and a SPICE ephemeris. The encounters are tabulated in Table 6.6. This patched-conic reference trajectory will be compared against the search results.

Table 6.6. Voyager 2 Encounters

Encounter	**Planet**	$\mathbf{V}_\infty$ $(\mathrm{km\,s^{-1}})$	**Date**
Launch	Earth	10.2	Aug. 20, 1977
1	Jupiter	7.8	Jul. 9, 1979
2	Saturn	10.7	Aug. 25, 1981
3	Uranus	14.8	Jan. 24, 1986
4	Neptune	16.7	Aug. 25, 1989

6.3.2 Voyager 2 Search Parameters

Table 6.7 summarizes the key inputs used to construct a network for the Voyager 2 search. The dates provided give boundaries on the alignment dates of the planets (described in Chapter 3). The actual departure or arrival dates may fall outside this window.

The V_∞ sequences in Table 6.7 generate the Tisserand graph shown in Figures 6.2 and 6.3. The contours for the outer solar system are compressed in Figure 6.2 so Figure 6.3 focuses on the Tisserand graph in the outer solar system. This difference in scales is an example of the difficulties with manual pathfinding in the Tisserand graph. These graphs are the basis for the Tisserand network in Figure 6.4.

Table 6.7. Voyager 2 Search Parameters

Parameter	**Value**
Earth V_∞	9:1:12 km s^{-1}
Jupiter V_∞	7:1:10 km s^{-1}
Saturn V_∞	8:1:11 km s^{-1}
Uranus V_∞	12:1:15 km s^{-1}
Neptune V_∞	14:1:17 km s^{-1}
Alignment Start Date	Jan. 1, 1972
Alignment End Date	Dec. 31, 1993
Max Flyby Count	6
Max Repeat Visits	0
Max Time of Flight	17 y
Date Tolerance	10% gravity assist body period

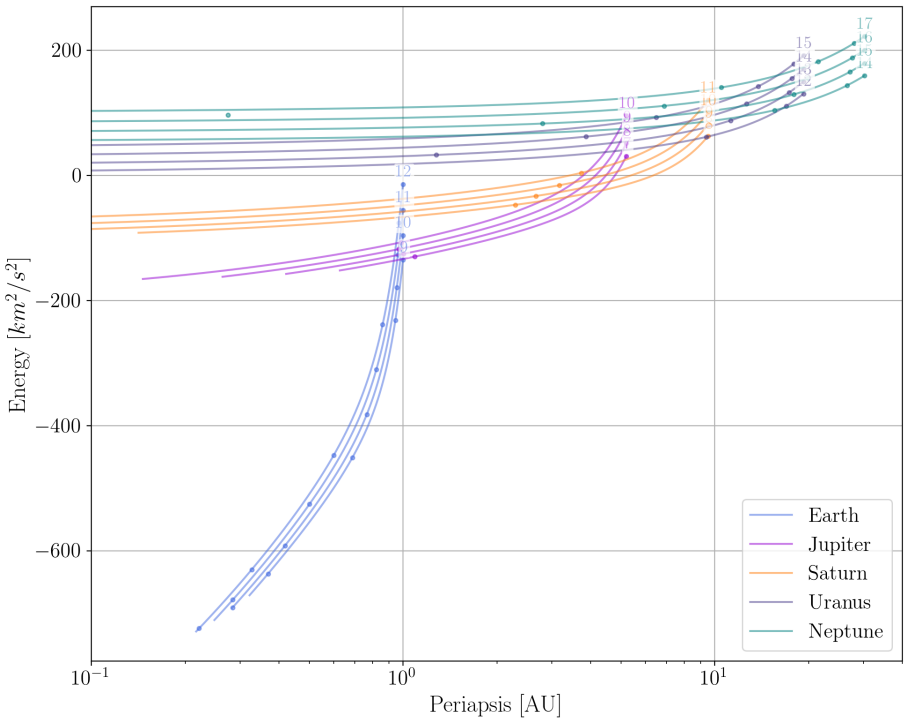


Figure 6.2. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Voyager 2 search.

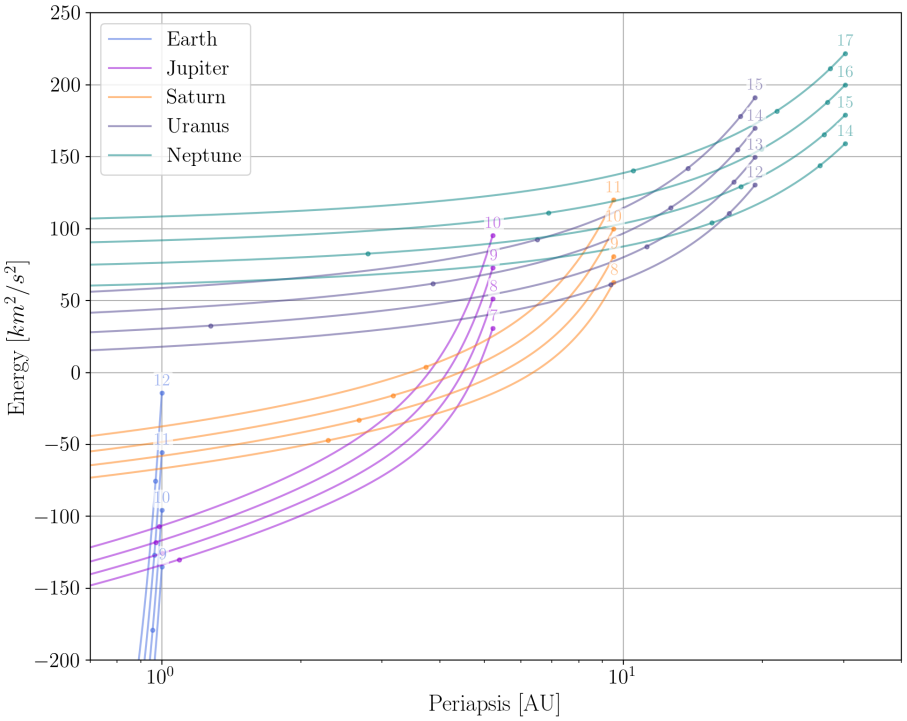


Figure 6.3. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Voyager 2 search. This view focuses on the outer solar system. The Earth contours are visible on the lower left.

6.3.3 Voyager 2 Results

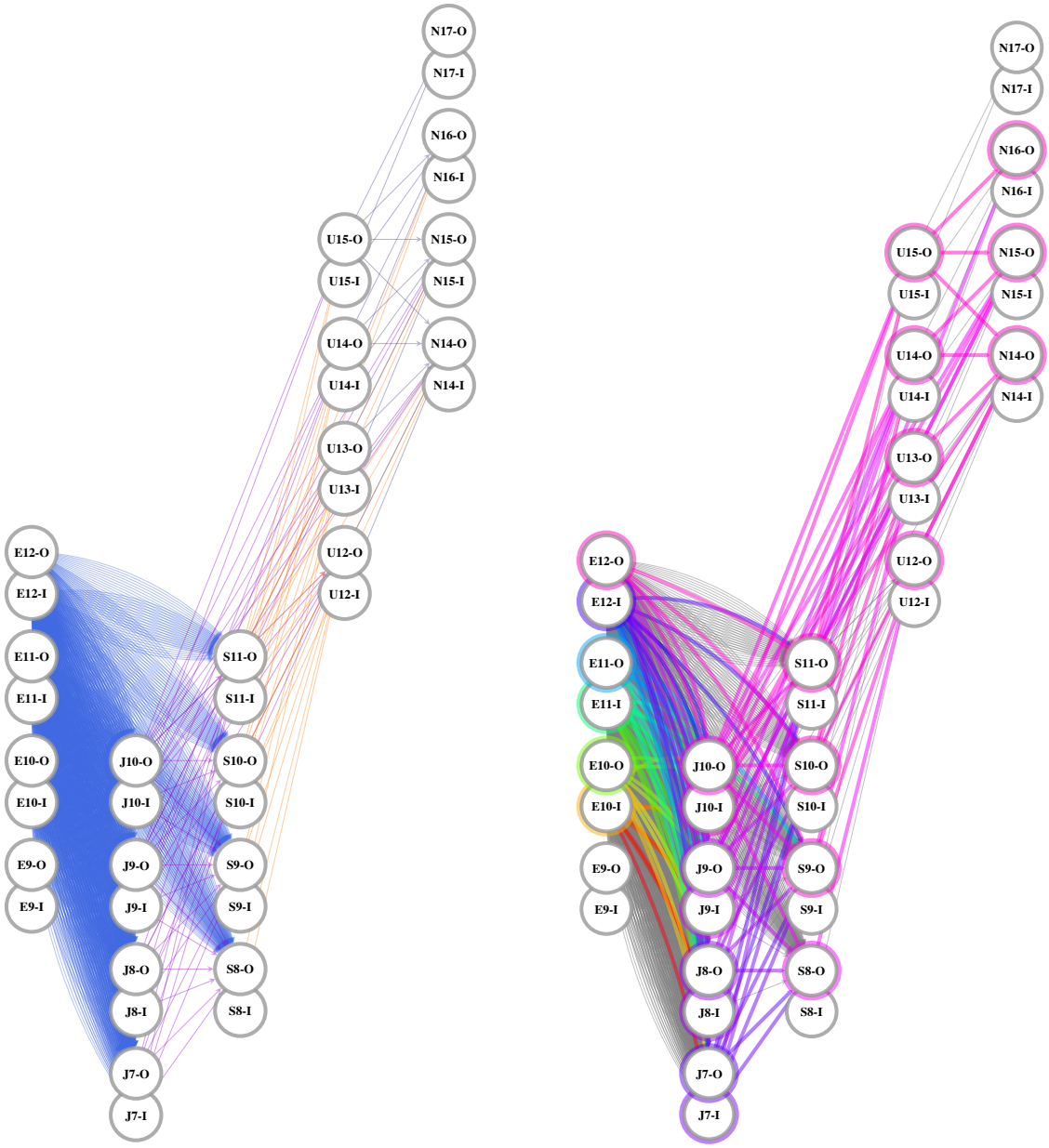
Table 6.8 summarizes the results of the Voyager 2 search. The search discovered paths JSUN, JSN, JUN, JN, SUN, and SN. Within each path are multiple routes. For example, the connected vertices (E10-I, J7-I, S8-O, U12-O, N14-O) and (E12-O, J10-O, S10-O, U14-O, N15-O) are both JSUN paths but they represent different routes through the network vertices. Overall the search found about 500 distinct routes. Including date variations on those routes, the total number of network paths exceeded 1,200.

Table 6.8. Voyager 2 Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞ ($km\,s^{-1}$)**	**Total ΔV ($km\,s^{-1}$)**	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
JSUN	312	690	5 - 15	0 - 5	01/72 - 12/79	09/87 - 07/99
JSN	72	163	6 - 15	0 - 5	02/72 - 12/79	02/90 - 08/93
JUN	96	206	10 - 15	0 - 5	04/72 - 01/81	04/91 - 06/96
JN	24	53	10 - 15	0 - 5	05/72 - 01/81	08/88 - 06/91
SUN	22	120	11 - 15	0 - 5	09/77 - 02/85	11/87 - 07/99
SN	6	34	11 - 15	0 - 5	09/78 - 02/85	06/91 - 03/94

The grid view of the Tisserand network (Figure 6.4a) visualizes the energy connections between flybys of each planet. This view, however, does not clearly show differences in timing between the arrival and departure from any given planet.

Figure 6.5 displays the Tisserand network in a polar view of the two-point orbital arcs for the entire search space. Figure 6.6 shows this same view for the initial Earth-Jupiter-Saturn sequence. The concentric circles outline the orbits of Earth, Jupiter, and Saturn, respectively. Empty markers on these orbits identify a departure from that planet and filled markers identify an arrival. This view shows, more clearly, the timing problem to be solved. However, the orbit view does not reveal information about the V_∞ levels of neighboring encounters. The arrival at Jupiter needs to coincide with the departure for Saturn. The network is filtered to remove arrival/departure time mismatches outside of the desired tolerance. The search algorithm will find arrivals and departures in the filtered network that match in V_∞.



(a) Search network.

(b) Network solutions highlighted.

Figure 6.4. The grid view of the Tisserand network assists in visualizing the energy connections that will be searched to find a possible Voyager 2 path. Multiple lines connecting two vertices indicate more than one date on which the connection exists. This broad visualization is not intended for tracing solutions. For color viewing or inspection of the vertex labels, the reader is referred to the electronic version of this document.

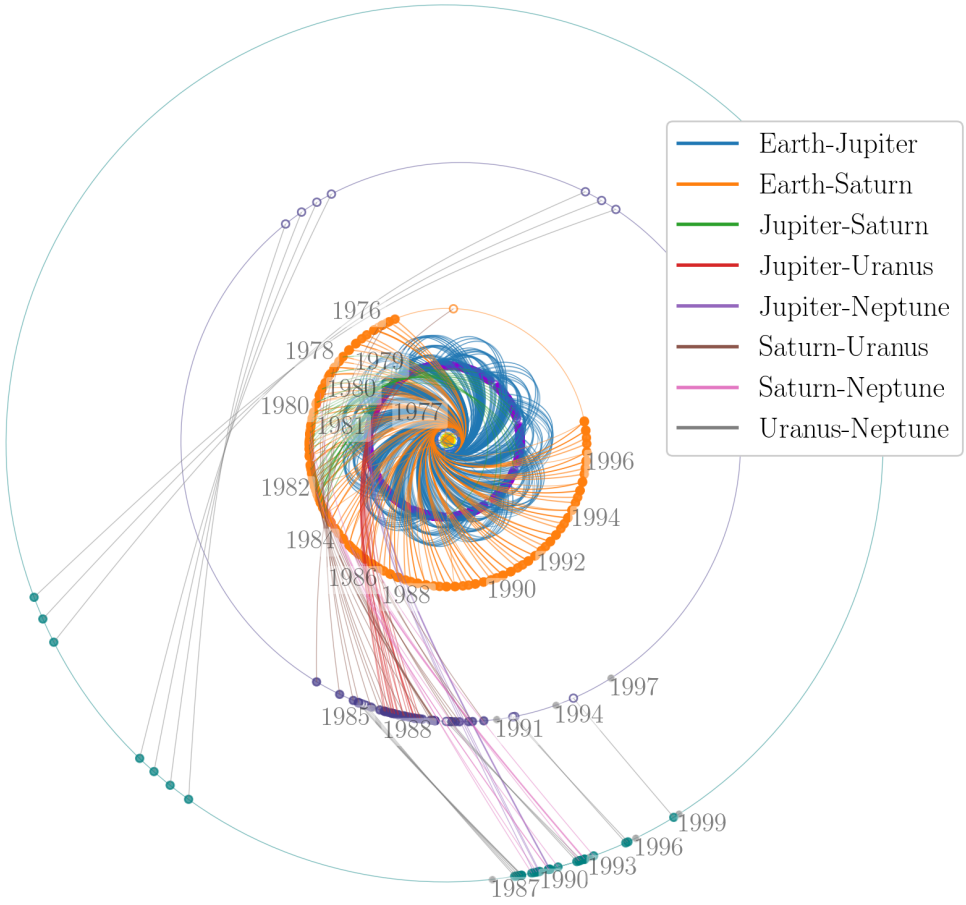


Figure 6.5. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The concentric circles represent the planet orbits with Neptune being outermost. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year.

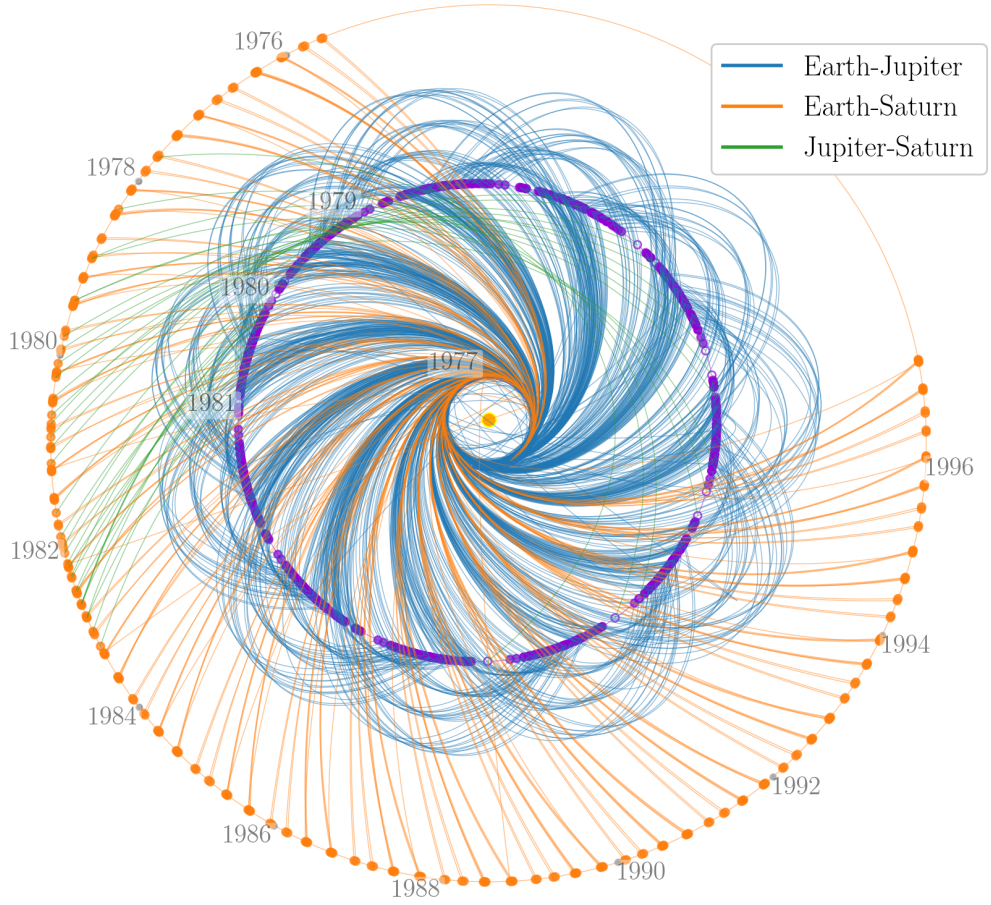


Figure 6.6. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The concentric circles represent the planet orbits with Saturn being outermost. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year.

Figure 6.7 shows a polar view of the Tisserand network search solutions. The network solutions are grouped into similar paths and colored accordingly. Most of the paths result in similar arrival times. The large tolerance in encounter date (10% of the gravity assist body period) accounts for the inclusion of several Uranus-Neptune legs with much later arrivals. The solution routes are also highlighted in the grid view of the Tisserand network in Figure 6.4b.

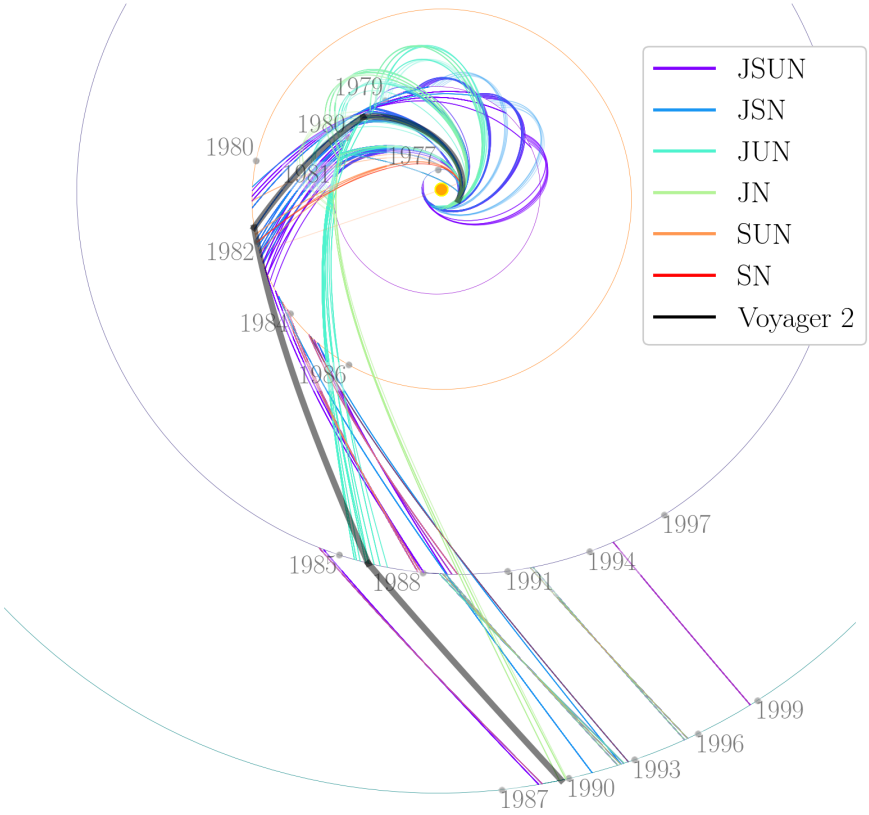


Figure 6.7. The search found six paths (JSUN, JSN, JUN, JN, SUN, and SN) that contain 500 routes. The actual Voyager 2 trajectory is shown in the thick gray line.

The discrete search results include gaps in time and only represent outlines of closed trajectories. Here, we again select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution to close the gaps (see Section 3.6.2). Figure 6.8 shows the resulting patched conic trajectories. The figure shows roughly 3500

patched-conic JSUN trajectories. Because of the random encounter date selection, some trajectories require extremely large and impractical propulsive ΔV in addition to the gravity assist.

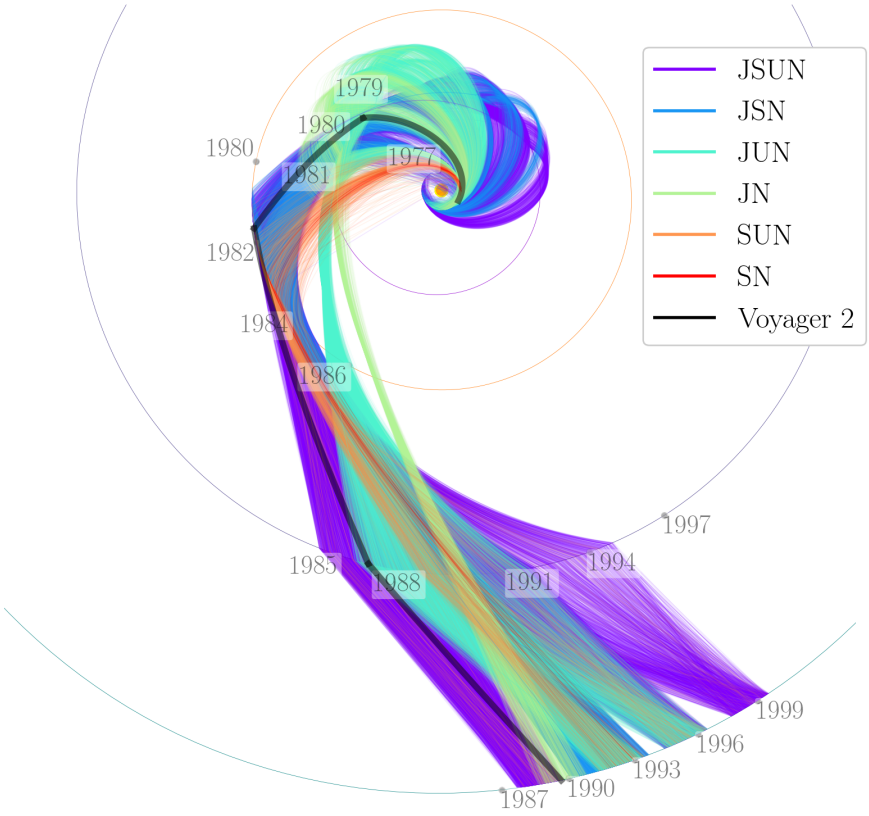


Figure 6.8. The Tisserand network solutions provide the outline for a collection of patched conic trajectories shown here. A patched conic approximation of the true Voyager 2 trajectory is highlighted.

We can evaluate the patched conic trajectories relative to one another by comparing key characteristics. Since we are comparing the results to Voyager 2, which used a JSUN path, we will only consider the JSUN path for further study. Figure 6.9 plots the time of flight against the total mission ΔV for each of the patched conic trajectories generated from the Tisserand network search results. The color of each point corresponds to the launch V_∞. For clarity, trajectories that required more than 15 km s^{-1} launch V_∞ or more than 5 km s^{-1} delta-V were filtered from the solution set. The approximated Voyager 2 trajectory appears as the diamond. In Figure 6.9 attractive trajectories will fall in the lower-left corner and will be colored on the cool end of the spectrum.

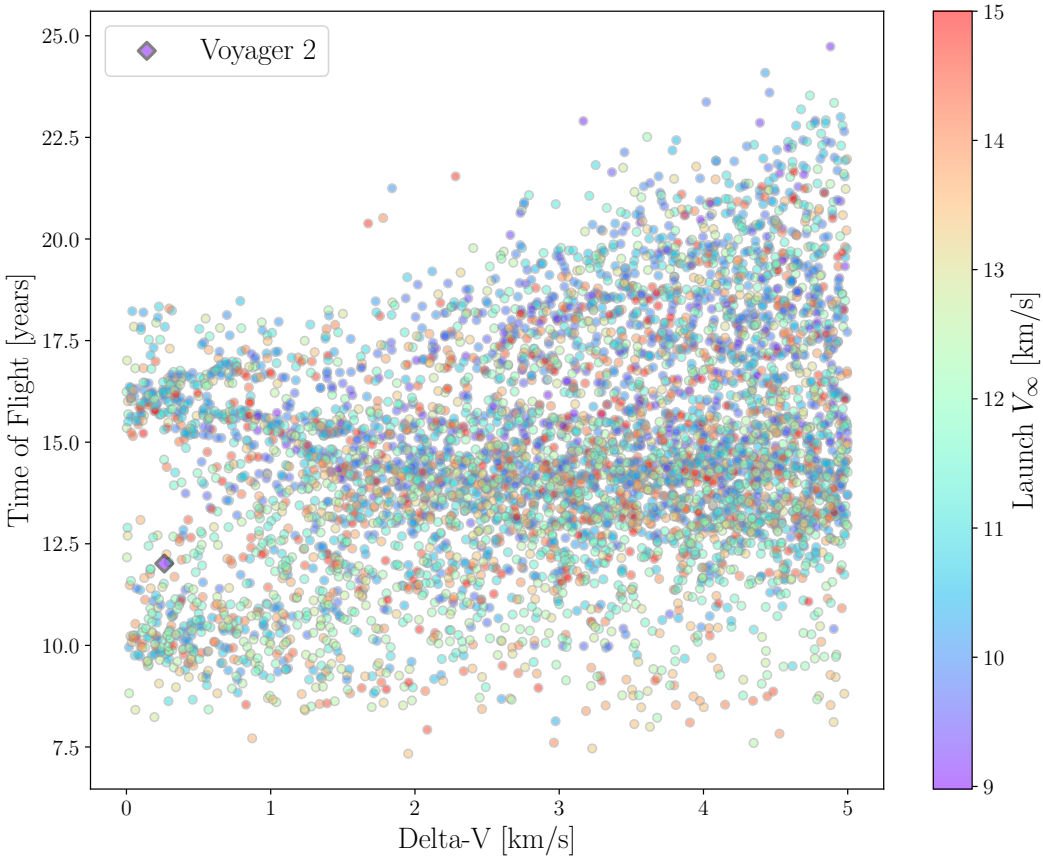


Figure 6.9. The time of flight is plotted against the mission ΔV for each of the patched conic variations on the network solution. The shade of the points corresponds to the total mission launch V_∞. Trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 5 km s^{-1} ΔV are not shown. The actual Voyager 2 trajectory is shown as the diamond.

Figure 6.10 plots the launch V_∞ against the launch date for each trajectory. The color of each point corresponds to the total mission ΔV. The figure shows a series of V-shaped profiles for various launch opportunities. Each opportunity shows a sharp increase in launch V_∞ as the launch date varies from the optimal.

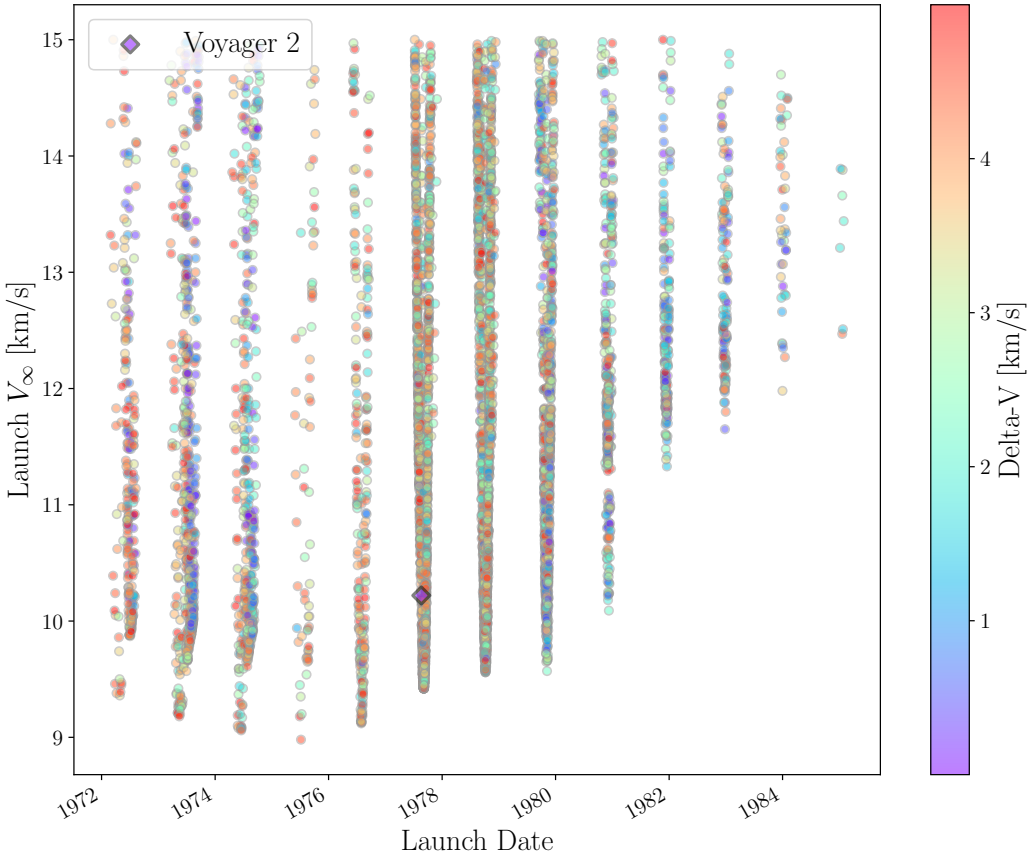


Figure 6.10. The Launch V_∞ is plotted against the Launch Date for each of the patched conic variations on the network solution. The shade of the points corresponds to the total mission ΔV. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown. The actual Voyager 2 trajectory is shown as the diamond.

Figure 6.11 shows the trends in launch V_∞ and ΔV for different arrival dates at Neptune. The figure shows that the group of trajectories that arrive around 1990 provide the earliest arrival and also dominate in both launch V_∞ and ΔV.

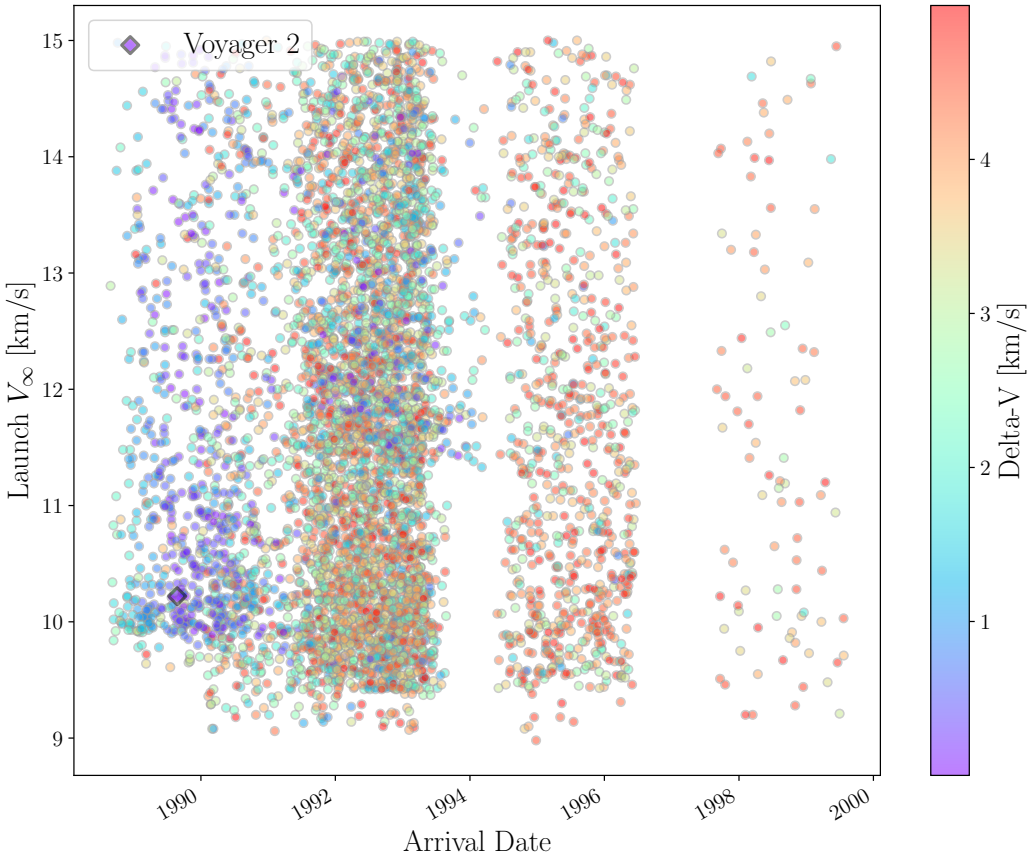


Figure 6.11. The Launch V_∞ is plotted against the Arrival Date for each of the patched conic variations on the network solution. The shade of the points corresponds to the total mission ΔV. Trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 5 km s^{-1} ΔV are not shown. The actual Voyager 2 trajectory is shown as the diamond.

Absent other mission constraints, a trajectory designer would likely choose a trajectory with a low launch V_∞, a low mission ΔV, and a short time of flight. We see that the actual Voyager 2 mission performs well in these criteria relative to the field of possible trajectories.

6.4 Galileo Search

The Galileo mission to Jupiter provides an opportunity to evaluate the Tisserand network in a trajectory search with a resonant transfer. Galileo performed two Earth flybys separated by exactly two years as part of the Venus Earth Earth Gravity Assist (VEEGA) sequence.

The Galileo mission included an atmospheric entry probe and a tour of the Jovian satellites. Here we will examine only the gravity assist journey from Earth to Jupiter and will neglect several trajectory shaping maneuvers.

6.4.1 Galileo Actual Trajectory

Galileo launched aboard Space Shuttle Atlantis on October 18, 1989. The original mission design included a direct trajectory to Jupiter made possible by a Shuttle-Centaur upper stage. The Shuttle-Centaur was a version of the Centaur upper stage designed to be carried to orbit in the Space Shuttle payload bay. In response to the Challenger disaster, the Shuttle-Centaur program was cancelled over safety concerns with the Centaur's liquid-hydrogen fuel. The Galileo mission was redesigned with a less-capable, solid-fueled inertial upper stage. The decrease in launch C_3 capability was the impetus for the VEEGA trajectory [74].

Partial information on the Galileo encounters can be found in D'Amario *et al.* [75] and is tabulated in Table 6.9. The V_∞^{pc} values are based on a patched-conic gravity-assist sequence reconstructed from the published encounter dates. The patched-conic values agree well with the published data.

Table 6.9. Galileo Encounters

Encounter	**Planet**	V_∞ $(\text{km}\,\text{s}^{-1})$	$V_\infty^{\mathbf{pc}}$ $(\text{km}\,\text{s}^{-1})$	**Date**
Launch	Earth	3.1	3.9	Oct. 18, 1989
1	Venus	6.2	6.1	Feb. 10, 1990
2	Earth	8.9	8.8	Dec. 8, 1990
3	Earth	8.9	8.9	Dec. 8, 1992
4	Jupiter	—	5.6	Dec. 7, 1995

6.4.2 Galileo Search Parameters

Table 6.10 summarizes the inputs used to build the Galileo search network. The *Max Repeat Visits* parameter is set to 2 to allow the repeat flybys of Earth in the VEEGA. To make the results more presentable, we add some additional constraints that the Venus encounters must occur in the years 1989 through 1991 and the Earth launch and encounters must occur

in the years 1988 through 1994. If we were looking for a new mission, we might not add these constraints. But for this reconstruction, these constraints will limit the solutions to those more closely resembling the Galileo mission.

Table 6.10. Galileo Search Parameters

Parameter	**Value**
Venus V_∞	4:1:8 $\mathrm{km\,s^{-1}}$
Earth V_∞	3:1:10 $\mathrm{km\,s^{-1}}$
Jupiter V_∞	5:1:8 $\mathrm{km\,s^{-1}}$
Start Date	Jan. 1, 1988
End Date	Dec. 31, 1996
Max Flyby Count	6
Max Repeat Visits	2
Max Time of Flight	9 y
Date Tolerance	5% time of flight
Venus Encounter Window	1989 - 1991
Earth Encounter Window	1988 - 1994

6.4.3 Galileo Results

Table 6.11 summarizes the results of the Galileo search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory.

Table 6.11. Galileo Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
VEEJ	56	121	3 - 15	2 - 10	12/88 - 02/90	02/95 - 06/96
J	16	85	9 - 15	0 - 0	06/88 - 01/94	04/90 - 11/97

As with the other missions, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure 6.12 shows the patched conic trajectories along the VEEJ and J paths. The launch V_∞ and propulsive ΔV (if any) required for each of the patched-conic trajectories can be computed from the Lambert

solutions as described in Chapter 3. Trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 10 km s^{-1} propulsive ΔV have been excluded from the results.

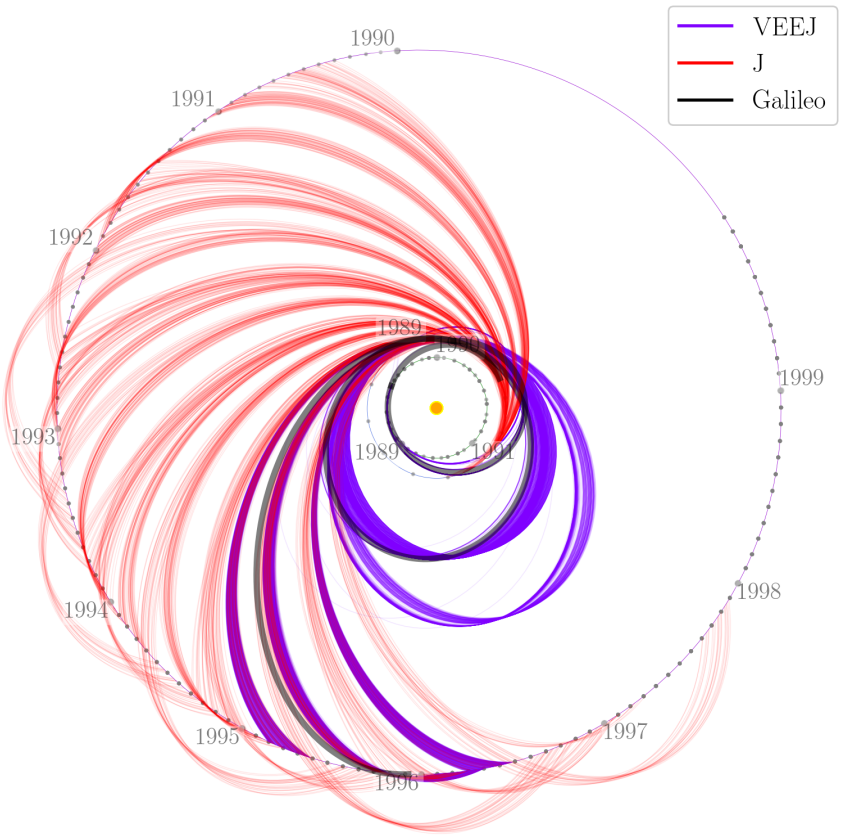


Figure 6.12. The Tisserand network solutions provide the outline for a collection of randomized patched conic trajectories shown here in a polar view of three-dimensional trajectories. The spacecraft trajectories are constructed from Lambert solutions between 3-D ephemerides. A patched conic approximation of the true Galileo trajectory is highlighted.

6.5 Cassini Search

The Cassini mission to Saturn provides a comparison case with a V-Infinity Leveraging Transfer (VILT). We will only attempt to find the trajectory to Saturn and will not recreate the tour of Saturn's satellites.

6.5.1 Cassini Actual Trajectory

Cassini/Huygens was launched on October 15, 1997 on a Titan IVB-Centaur with two solid rocket motor upgrades. The Centaur upper stage provided the ΔV for the departure to Venus [76]. Partial information on the Cassini encounters is available in Goodson *et al.* [65]. Those values are tabulated in Table 6.12. The V_∞^{pc} values are based on a patched conic reconstruction of the gravity assist sequence using the published encounter dates. The patched-conic values agree well with the published data.

Table 6.12. Cassini Encounters

Encounter	**Planet**	V_∞ $(\text{km}\,\text{s}^{-1})$	$V_\infty^{\mathbf{pc}}$ $(\text{km}\,\text{s}^{-1})$	**Date**
Launch	Earth	4.1	4.0	Oct. 15, 1997
1	Venus	6.0	5.9	Apr. 26, 1998
2	Venus	9.4	9.4	Jun. 24, 1999
3	Earth	16.0	16.0	Aug. 18, 1999
4	Jupiter	—	10.6	Dec. 30, 2000
5	Saturn	—	5.3	Jul. 1, 2004

6.5.2 Cassini Search Parameters

Table 6.13 summarizes the inputs used to build the Cassini search network. The network is configured to scan for Venus VILT opportunities starting with a V_∞ of 6 $\text{km}\,\text{s}^{-1}$. The *Max Repeat Visits* parameter is set to 2.

Table 6.13. Cassini Search Parameters

Parameter	Value
Venus V_∞	5, 6, 9, 10, 11 km s^{-1}
Earth V_∞	4.5, 5, 14, 15, 16, 17 km s^{-1}
Jupiter V_∞	9:1:11 km s^{-1}
Saturn V_∞	5:1:7 km s^{-1}
Start Date	Jan. 1, 1997
End Date	Dec. 31, 2010
Max Flyby Count	6
Max Repeat Visits	2
Max Time of Flight	9 y
Date Tolerance	20% time of flight
VILT at Venus	6 km s^{-1}

6.5.3 Cassini Results

Table 6.14 summarizes the results of the Cassini search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory.

Table 6.14. Cassini Search Results Summary

Path	Network Routes	Date Variants	Launch V_∞ (km s^{-1})	Total ΔV (km s^{-1})	Launch Window (mm/yy)	Arrival Window (mm/yy)
VVES	5	27	3 - 12	3 - 10	11/97 - 12/05	11/02 - 10/17
VVEJS	6	6	4 - 11	—	10/97 - 12/97	07/02 - 03/06
JS	42	42	10 - 15	0 - 5	02/98 - 10/00	11/01 - 10/10
VES	46	186	14 - 15	0 - 10	12/96 - 04/06	10/02 - 11/17
VEJS	44	72	14 - 15	0 - 10	12/96 - 05/98	06/02 - 04/06
S	18	234	10 - 15	0 - 0	03/97 - 02/10	08/00 - 04/18

As with the other missions, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure 6.13 shows the patched conic trajectories along the various paths. Patched-conic trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 10 km s^{-1} propulsive ΔV have been excluded from the results.

The Tisserand network was able to find routes similar to the actual VVEJS Cassini path (for example, E5-I, V6-O, V10-I, E16-O, J11-O, S5-O). However, in cases with consecutive flybys of the same planet (such as the Venus VILT in the Cassini design), the ΔV results for the randomized patched-conic trajectories have proven to be sensitive to the actual randomly drawn dates. The particular random draws in the example tabulated here resulted in ΔV above the 10 $\mathrm{km\,s^{-1}}$ limit. In these cases, a more sophisticated method for generating the patched-conic trajectories may be required. This improvement is included in the future work.

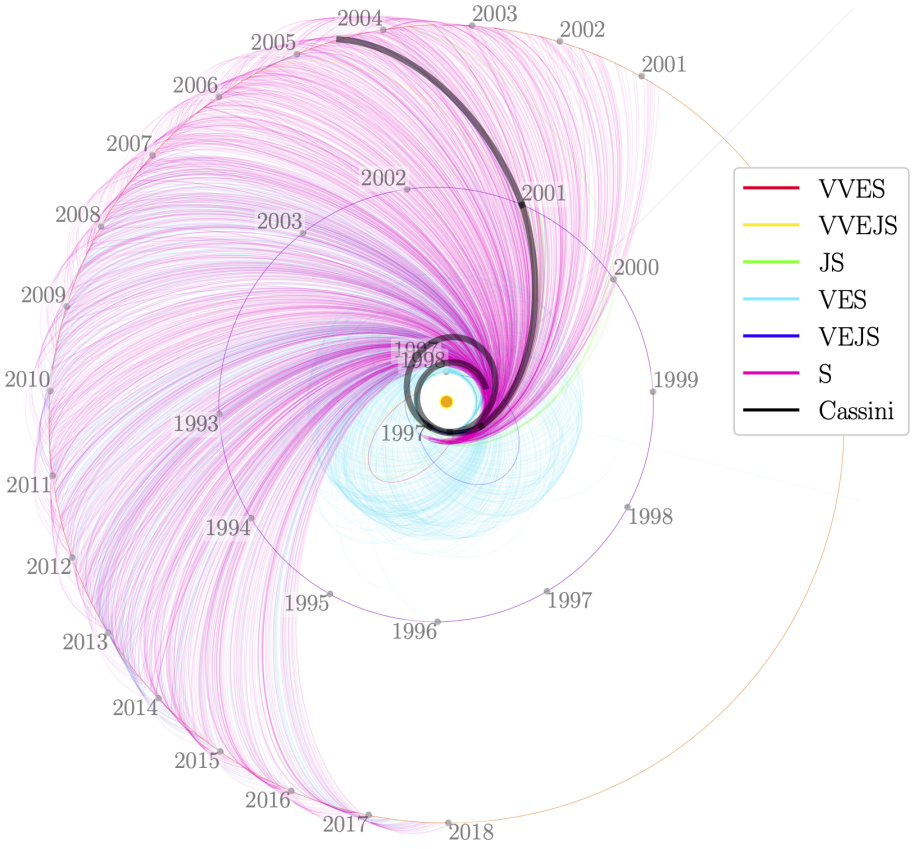


Figure 6.13. The Tisserand network solutions provide the outline for a collection of randomized patched conic trajectories shown here in a polar view of three-dimensional trajectories. The spacecraft trajectories are constructed from Lambert solutions between 3-D ephemerides. A patched conic approximation of the true Cassini trajectory is highlighted.

6.6 Historical Mission Search Summary

The searches in this chapter demonstrated the utility of the Tisserand network in finding paths to the outer solar system. The resonance and VILT models of Chapter 4 were used in the Galileo and Cassini searches, respectively. The networks were provided with the gravity-assist bodies, the time frames to consider, and the desired target planets. The searches produced families of patched-conic trajectories that bracket the actual historical missions.

7. PRELIMINARY TRAJECTORIES TO TRANS-NEPTUNIAN OBJECTS USING THE NETWORK METHOD

As an additional demonstration of the Tisserand-network technique we now attempt a search for trajectories to two dwarf planets: Haumea and Makemake. Both planets are among the largest Trans-Neptunian Objects (TNOs) and are of scientific interest, having only been discovered in the past twenty years.

TNO missions are an active area of research and we will not attempt a detailed evaluation of trajectory options with respect to science objectives and other mission constraints [77]–[80]. Here, we will simply attempt to find some preliminary trajectory candidates using the newly developed method.

To help guide our search, we note the launch energies of the five missions capable of escaping the solar system in Table 7.1 [78].

Table 7.1. High C_3 Missions

Mission	$\mathbf{C_3}$ $(\mathrm{km^2/s^2})$	$\mathbf{V_\infty}$ $(\mathrm{km\,s^{-1}})$
Pioneer 10	95	9.7
Pioneer 11	87	9.3
Voyager 1	105	10.2
Voyager 2	102	10.1
New Horizons	158	12.6

With this history in mind we will include Earth V_∞ contours between 4 $\mathrm{km\,s^{-1}}$ and 14 $\mathrm{km\,s^{-1}}$ ($C_3 = 196$ $\mathrm{km^2/s^2}$). The larger C_3 values are within expected capabilities of the Space Launch System with various upper stages [81].

In the Tisserand network configuration, we make no distinction between a launch or a gravity assist with respect to the Earth departure V_∞. So an Earth V_∞ of 14 $\mathrm{km\,s^{-1}}$ could be a high energy launch or an Earth flyby (for example, part of a VEEGA). Because V_∞ translates directly to the network vertices, in the discussion below we will present results in terms of launch V_∞ rather than the more common C_3.

We limit the search to missions launching by the end of 2040 with a maximum flight time of 25 years. Figure 7.1 shows the positions of the solar system over the period 2020 to

2050. The locations of Neptune and Pluto are not suitable for gravity assists to either of the dwarf planets in our search. To reduce the network edges, we exclude Neptune, Pluto, and Mercury from consideration.

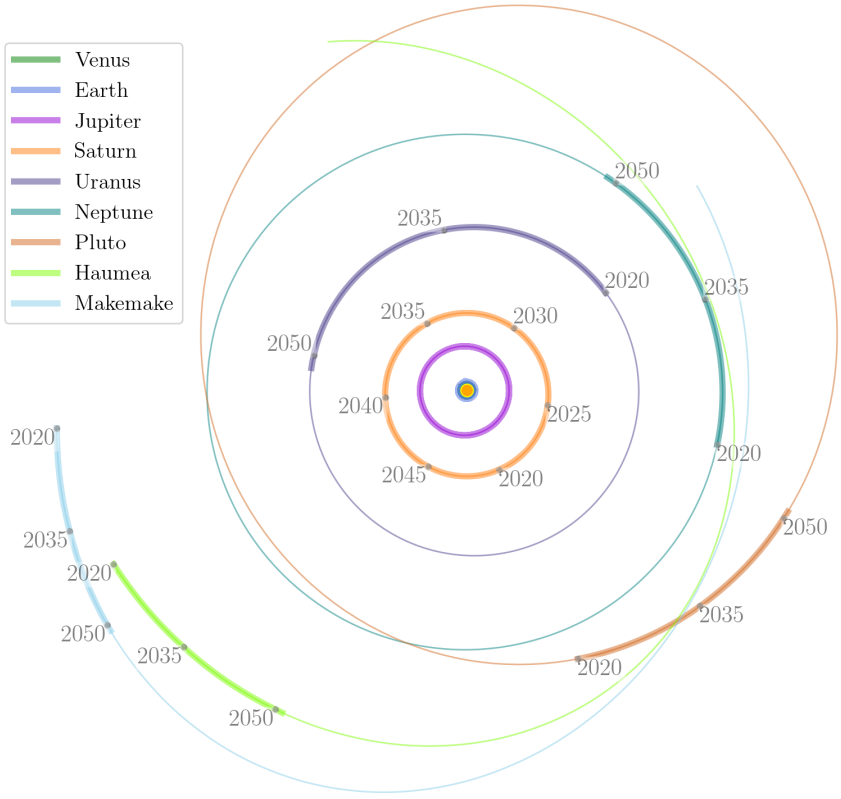


Figure 7.1. The alignment of the solar system from 2020 to 2050. Neptune and Pluto are not well positioned for use in a gravity assist to Haumea or Makemake.

We also note here that the Tisserand graph and Tisserand network assume circular-coplanar orbits. Many TNOs are more inclined relative to the ecliptic plane than the planets. Haumea and Makemake have inclinations of 28 and 29 deg, respectively. The orbits also deviate from the circular assumption. Haumea and Makemake have eccentricities of 0.20 and 0.16, respectively.

As discussed in Chapter 3, patched-conic trajectories are generated from the Tisserand network search results by solving the Lambert problem in a full ephemeris. Therefore, the penultimate gravity assist encounter will include an implicit inclination change.

However, we stress again the preliminary design quality of the assumptions in Section 3.1.3 (patched conics, circular-coplanar orbits, etc.). The intent of these investigations is to generate initial guesses of gravity-assist energies and time frames that are likely to yield results worth examining in higher fidelity tools. These assumptions remain appropriate for this preliminary work. In fact, the approximate arrival epochs resulting from these searches agree with those found in Zangari *et al.* [78] using a purpose-built TNO search tool (also based on patched solutions to Lambert's problem).

7.1 Haumea Search

Haumea is a Kuiper-Belt dwarf planet roughly the size of Pluto. It has a mean orbital radius of approximately 43 AU and a period of 284 years [82]. Haumea is among the fastest rotating known objects in the solar system, completing a rotation once every four hours. The high spin rate gives the planet an ellipsoidal shape. Haumea is orbited by two small satellites and may also have a ring. These characteristics make Haumea an interesting target for exploration despite its great distance [78], [80].

7.1.1 Haumea Search Parameters

Table 7.2 summarizes the key inputs used to construct a network for the Haumea search. The dates in Table 7.2 give boundaries on the alignment dates of the planets (described in Chapter 3). The actual departure or arrival dates may fall outside this window.

7.1.2 Haumea Search Results

The Tisserand network search found five paths to Haumea arriving in the 2030s to 2060s. These paths include 26 different routes. Recall from Chapter 3 that the discrete routes are discontinuous. To evaluate the paths, we generate twenty random but continuous patched-

Table 7.2. Haumea Search Parameters

Parameter	**Value**
Venus V_∞	6:3:14 km s^{-1}
Earth V_∞	4:3:14 km s^{-1}
Mars V_∞	6:3:14 km s^{-1}
Jupiter V_∞	6:2:18 km s^{-1}
Saturn V_∞	8:3:18 km s^{-1}
Uranus V_∞	8:3:18 km s^{-1}
Haumea V_∞	6:3:16 km s^{-1}
Alignment Start Date	Jan. 1, 2025
Alignment End Date	Dec. 31, 2065
Max Flyby Count	8
Venus Max Repeat Visits	2
Earth Max Repeat Visits	2
Other Max Repeat Visits	1
Max Time of Flight	25 y
Date Tolerance	10% time of flight

conic trajectories along each route. Figure 7.2 provides a polar view of these patched-conic trajectories.

Table 7.3 summarizes the results of the Haumea search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory. We use a lower-case h to symbolize the encounter with Haumea. Recall from Chapter 3, a *route* differentiates the particular vertices in the Tisserand network passed through along the path (E13-O, J16-O, h15-O). A *date variant* identifies a duplicate of a route with some difference in the encounter dates. The *Network Routes* and *Date Variants* columns list the number of these occurrences for each path.

The *Launch* V_∞, *Total* ΔV, *Launch Window* and *Arrival Window* columns summarize the extremes of the randomized patched-conic trajectories. The total ΔV is the cumulative propulsive ΔV required to complete gravity-assist turning at any of the encounters after Earth departure. The date columns give the range of departure or arrival dates in month/year format.

The Tisserand network identified two paths that use gravity assists in the inner solar system (VEEJh, VMEEJh). However, both of these paths require excessively high ΔV to

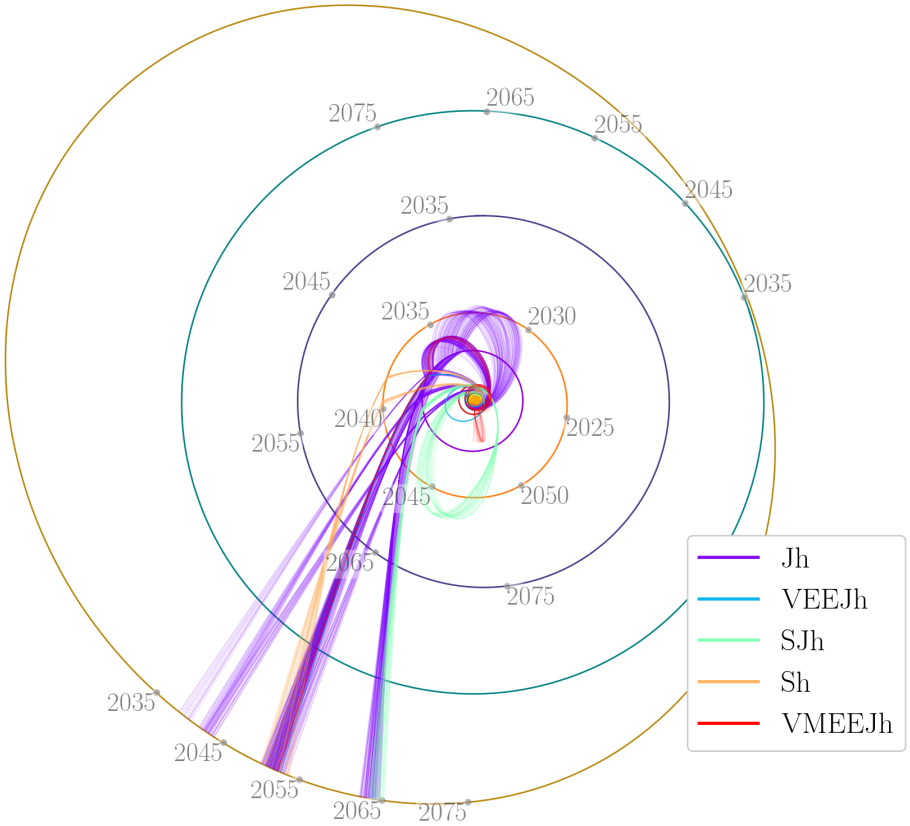


Figure 7.2. The Tisserand network found five paths to Haumea in the search period. These paths include 26 different routes. Twenty patched-conic trajectories were generated for each route. These patched conics are visualized in the figure. The orbits of Jupiter, Saturn, Uranus, Neptune, and Haumea are also displayed.

close the patched-conic trajectories. In this analysis, the only source of mid-mission ΔV is powered flyby ΔV required to complete the turning at an encounter. Stochastic trajectory correction maneuvers are not modeled.

For the remaining discussion, trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV have been filtered out of the results. This removes all of the options using the inner solar system for gravity assist. Figure 7.3 displays Haumea arrival date versus the launch date for the remaining patched-conic trajectories. The simplest path (Jh, Earth-Jupiter-Haumea) provides the most opportunities. The other paths use either Saturn as the only gravity assist or a Jupiter-Saturn gravity assist sequence.

Table 7.3. Haumea Search Results Summary

Path	Network Routes	Date Variants	Launch V_∞ ($km\,s^{-1}$)	Total ΔV ($km\,s^{-1}$)	Launch Window (mm/yy)	Arrival Window (mm/yy)
Jh	18	29	10 - 46	0 - 5	08/25 - 08/44	07/39 - 11/63
VEEJh	1	1	9 - 31	5 - 12+	11/33 - 11/33	09/51 - 09/51
SJh	2	4	11 - 41	1 - 4	11/33 - 01/35	05/64 - 05/64
Sh	4	4	11 - 43	0 - 3	08/34 - 11/36	12/50 - 12/52
VMEEJh	1	1	12 - 19	12+	07/26 - 07/26	08/51 - 08/51

In the absence of specific mission goals, we may assume that trajectory options with low launch energy requirements, low mission ΔV expenditures, and short flight times are most desirable. However, these qualities tend to work against each other; minimizing any two will typically increase the third. Figure 7.4 presents the launch V_∞ versus the ΔV for the Jupiter and Saturn paths. The lowest ΔV paths are almost exclusively Jupiter-Haumea (Jh) paths. However, several Saturn-Haumea paths require a moderate ΔV of 1 to 3 $km\,s^{-1}$.

Figure 7.5 shows the time of flight versus the ΔV for each patched-conic trajectory. The lower left region is again the most desirable. The Jupiter-Haumea path appears in three distinct bands. There are low ΔV options in each band, including the band with lowest time of flight.

Figure 7.6 focuses on the time of flight versus the ΔV like figure 7.5. However, this time we consider only the Jh trajectories and color the points according to the launch V_∞. As we might expect, the trajectories with the shortest time of flight (around 15 years) benefit from higher launch energies. The highest launch energy yet achieved (New Horizons) would fall near the middle of the Launch V_∞ scale in Figure 7.6. This is roughly the launch energy that many of the fastest Haumea trajectories require.

The Voyager missions would fall on the lower end of the Launch V_∞ spectrum in Figure 7.6. There are many trajectories in the middle band (around 18 to 20 year flight times) that require 10 to 11 $km\,s^{-1}$ launch V_∞. The upper band includes some lower launch V_∞ and lower ΔV requirements, but the flight time is longer (22 to 25 years).

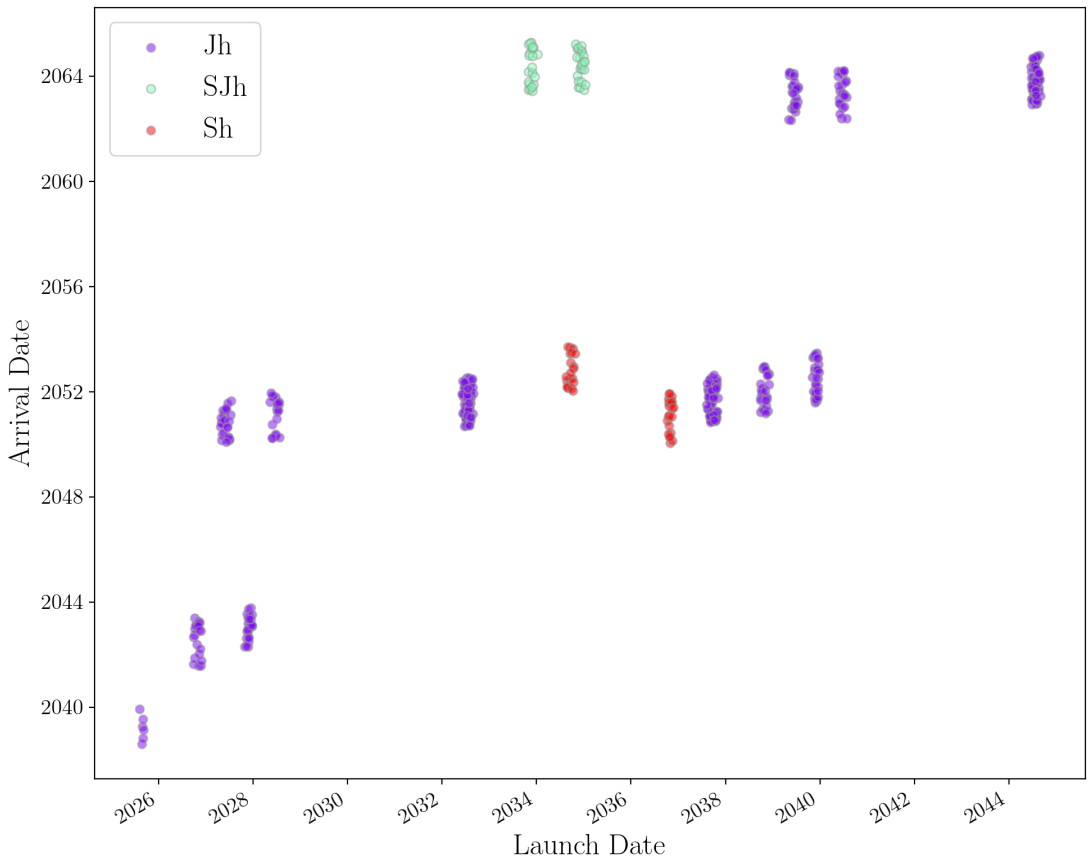


Figure 7.3. The arrival date is plotted against the launch date for each of the patched conic variations on the network solution. Trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 5 km s^{-1} ΔV are not shown.

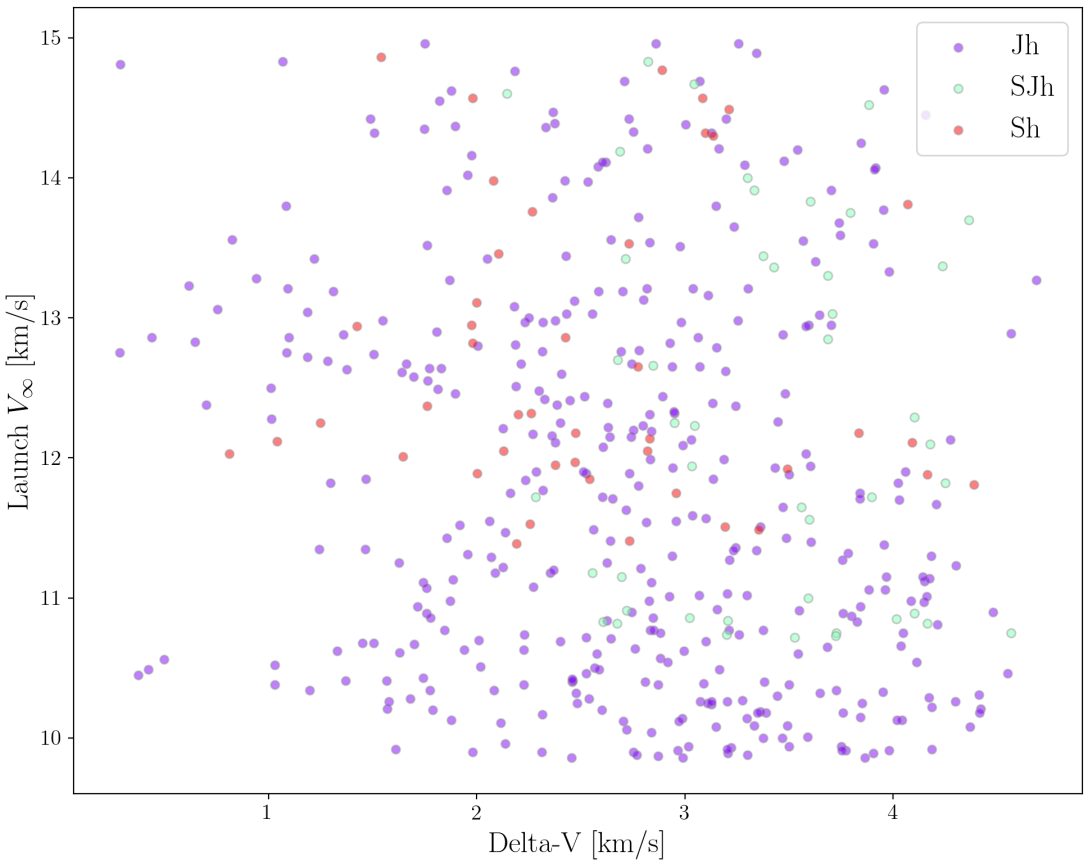


Figure 7.4. The launch V_∞ is plotted against the cumulative ΔV for each of the patched conic variations on the network solution. The Jupiter-Haumea path requires the lowest launch cost and lowest mission ΔV. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

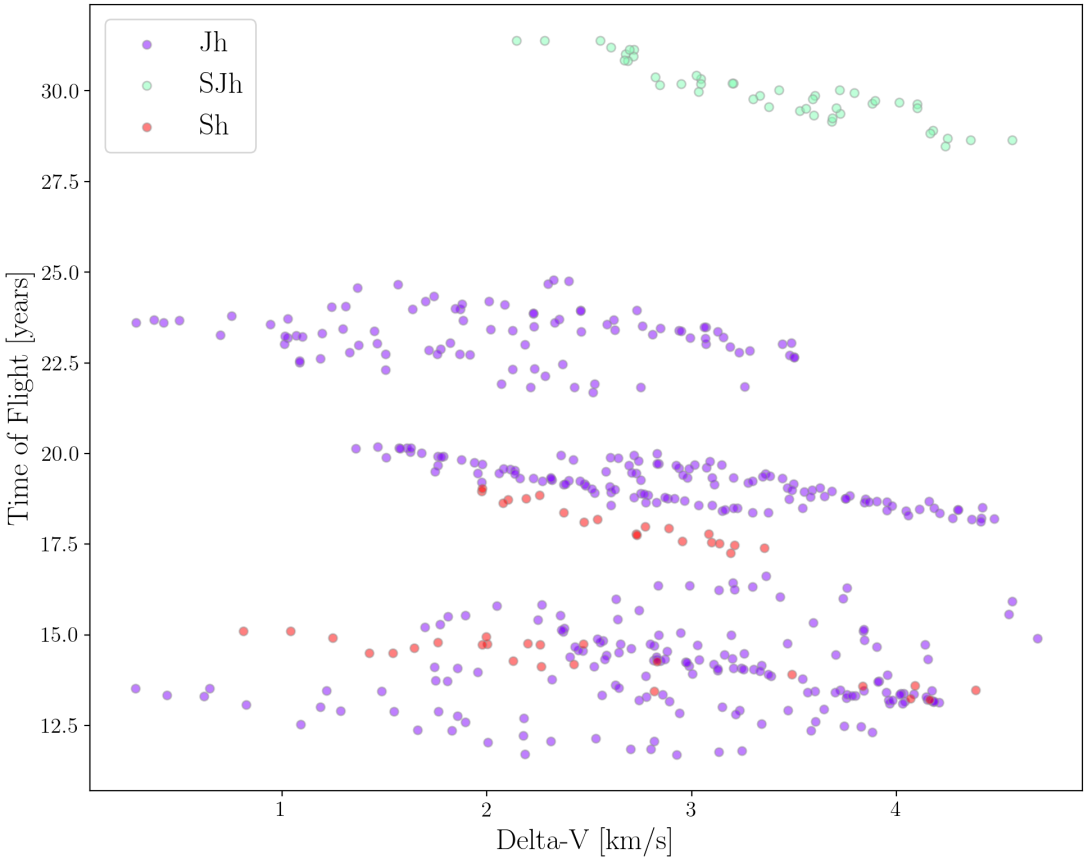


Figure 7.5. The time of flight is plotted against the cumulative ΔV for each of the patched conic variations on the network solution. Jupiter-Haumea trajectories are available with low ΔV. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

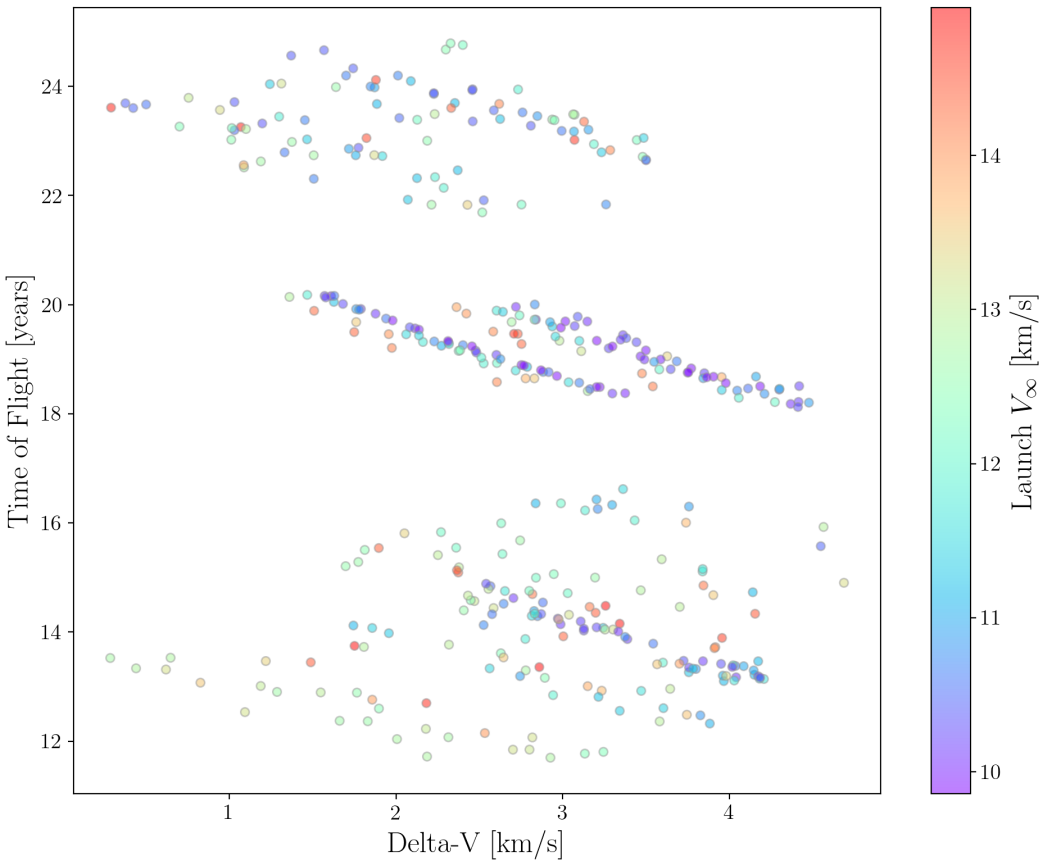


Figure 7.6. The time of flight is plotted against the cumulative ΔV for the Jupiter-Haumea path only. The points are colored according to launch V_∞. The fastest and least ΔV costly trajectories in the lower left require large launch energy. Trajectories requiring more than 15 km s^{-1} launch V_∞ or more than 5 km s^{-1} ΔV are not shown.

7.2 Makemake Search

Makemake is a classical Kuiper Belt Object (KBO). The planet is slightly smaller than Pluto and was first observed in 2005. With a semi-major axis of approximately 45 AU, Makemake has an orbital period of 305 years [82]. The Hubble Space Telescope observed a likely satellite, nicknamed MK2, in 2016.

7.2.1 Makemake Search Parameters

Table 7.4 summarizes the key inputs used to construct a network for the Makemake search.

Table 7.4. Makemake Search Parameters

Parameter	**Value**
Venus V_∞	6:3:14 km s^{-1}
Earth V_∞	4:3:14 km s^{-1}
Mars V_∞	6:3:14 km s^{-1}
Jupiter V_∞	6:2:18 km s^{-1}
Saturn V_∞	8:3:18 km s^{-1}
Uranus V_∞	8:3:18 km s^{-1}
Makemake V_∞	6:3:16 km s^{-1}
Alignment Start Date	Jan. 1, 2025
Alignment End Date	Dec. 31, 2065
Max Flyby Count	8
Venus Max Repeat Visits	2
Earth Max Repeat Visits	2
Other Max Repeat Visits	1
Max Time of Flight	25 y
Date Tolerance	10% time of flight

The dates provided give boundaries on the alignment dates of the planets (described in Chapter 3). The actual departure or arrival dates may fall outside this window.

7.2.2 Makemake Search Results

The Tisserand network search found nine paths to Makemake arriving in the 2040s to 2070s. These paths include 41 different routes. Recall from Chapter 3 that the discrete routes

are discontinuous. To evaluate the paths, we generate 20 random but continuous patched-conic trajectories along each route. Figure 7.7 provides a polar view of these patched-conic trajectories.

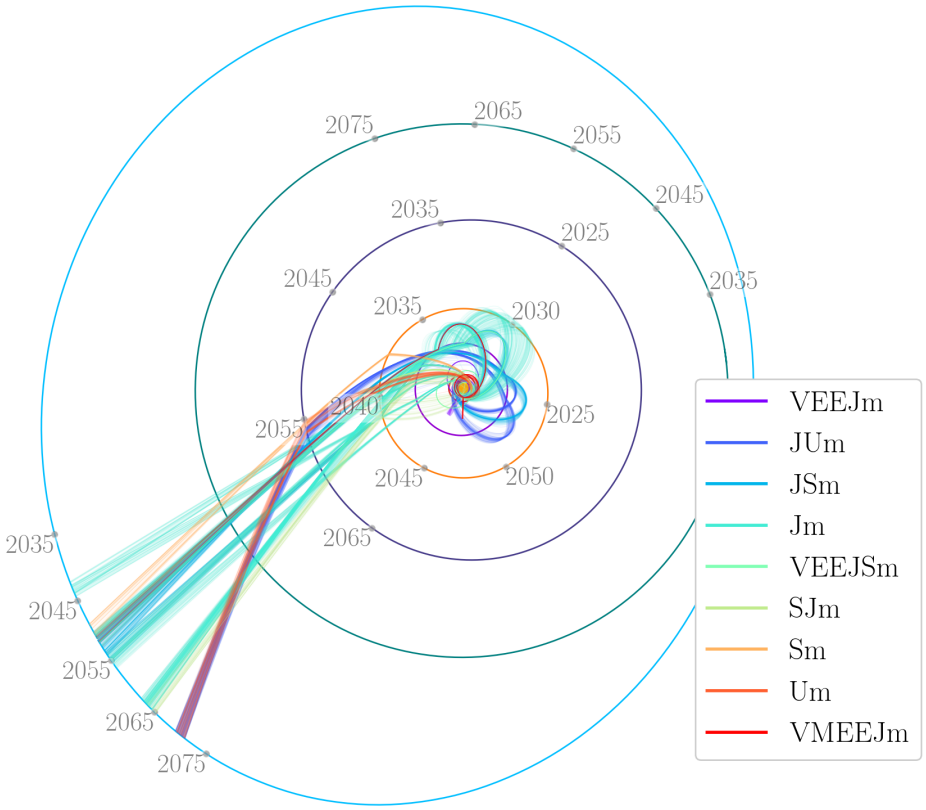


Figure 7.7. The search found nine paths to Makemake in the search period. These paths include 41 different routes. Twenty patched-conic trajectories were generated for each route. These patched conics are visualized in the figure. The orbits of Jupiter, Saturn, Uranus, Neptune, and Makemake are also displayed.

Table 7.5 summarizes the results of the Makemake search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory. We use a lower-case m to symbolize the encounter with Makemake (reserving M for Mars). Recall from Chapter 3, a *route* differentiates the particular vertices in the Tisserand network passed through along the path (E13-O, J18-O, m12-O). A *date variant* identifies a duplicate of a route with some difference in the encounter dates. The *Network Routes* and *Date Variants* columns list the

number of these occurrences for each path. The *Launch* V_∞, *Total* ΔV, *Launch Window* and *Arrival Window* columns summarize the extremes of the randomized patched-conic trajectories.

Table 7.5. Makemake Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
VEEJm	2	2	4 - 23	4 - 12+	07/27 - 01/28	07/51 - 07/51
JUm	8	8	9 - 44	2 - 9	12/38 - 04/42	02/70 - 02/70
JSm	12	12	10 - 35	3 - 7	01/28 - 10/37	01/51 - 06/54
Jm	10	25	10 - 27	0 - 12	03/26 - 06/44	12/42 - 09/63
VEEJSm	2	2	9 - 30	8 - 12+	06/27 - 11/33	01/51 - 06/54
SJm	2	2	11 - 47	1 - 4	09/35 - 11/35	05/64 - 05/64
Sm	2	2	12 - 28	2 - 4	09/34 - 10/34	09/49 - 09/49
Um	2	4	12 - 32	1 - 3	09/46 - 11/47	02/70 - 02/70
VMEEJm	1	1	12 - 16	5 - 12+	07/26 - 07/26	07/51 - 07/51

The search identified some paths that use gravity assists in the inner solar system (VEEJm, VEEJSm, VMEEJm). However, all of these paths require excessively high ΔV to close the patched-conic trajectories. In this analysis, the only source of mid-mission ΔV is powered flyby ΔV required to complete the turning at an encounter. Trajectory correction maneuvers are not modeled.

For the remaining discussion, trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV have been filtered out of the results. This removes all of the options using the inner solar system for gravity assist. Figure 7.8 displays Makemake arrival date versus the launch date for the remaining patched-conic trajectories. The simplest path (Jm, Earth-Jupiter-Makemake) provides the most opportunities.

Lacking specific mission goals, we may assume that trajectory options with low launch energy requirements, low mission ΔV expenditures, and short flight times are most desirable. Figure 7.9 presents the launch V_∞ versus the ΔV for the various paths discovered by the search. The lower left region of low launch V_∞ and low mission ΔV appears to be dominated by the Jupiter-Makemake path (Jm).

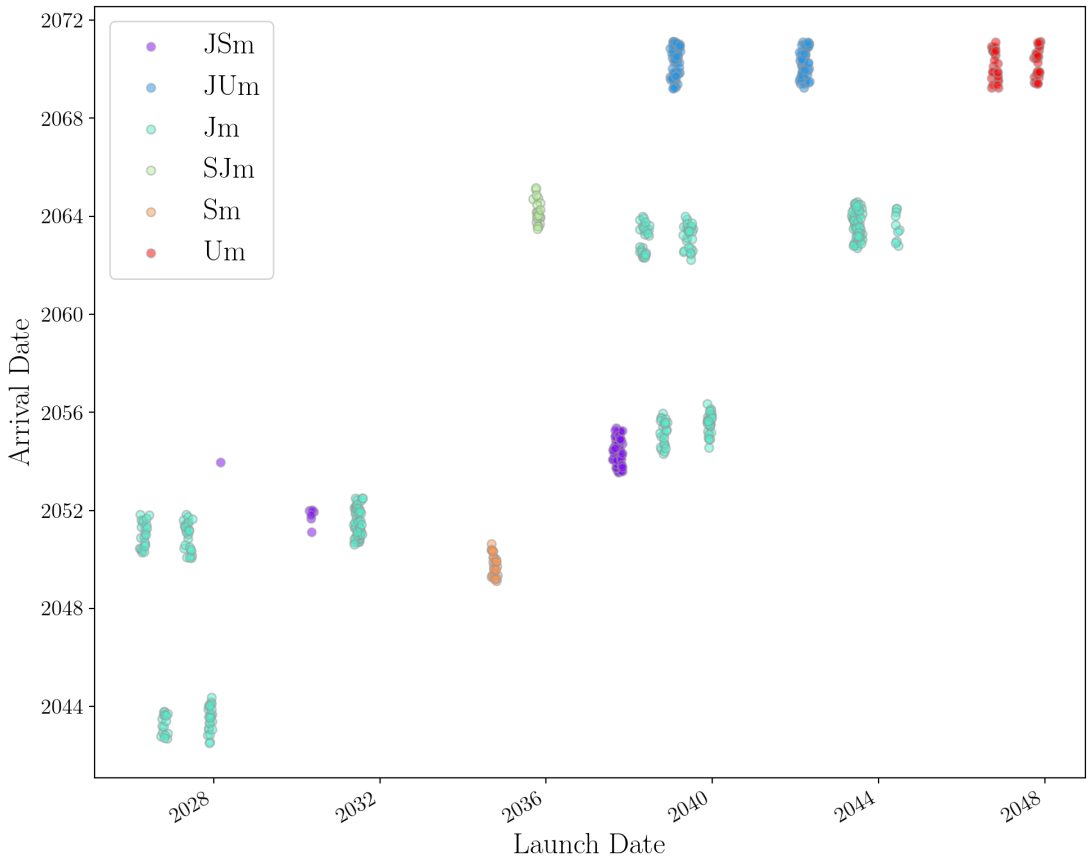


Figure 7.8. The arrival date is plotted against the launch date for each of the patched conic variations on the network solution. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

Figure 7.10 shows the time of flight versus the ΔV for each patched-conic trajectory. The lower left region is again the most desirable. There appear to be many low ΔV options following the Jupiter-Makemake path, including some with low time of flight. However, we cannot assume that the patched-conic Jm trajectories in the lower left of Figure 7.10 are the same trajectories in the lower left of 7.9.

Figure 7.11 focuses on the time of flight versus the ΔV considering only the Jm trajectories. The points are colored according to the launch V_∞. As we might expect, the trajectories with the shortest time of flight (around 16 years) have launch energies on the upper end of the spectrum. We note that the highest launch energy yet achieved (New Horizons) would fall near the middle of the Launch V_∞ scale in Figure 7.11.

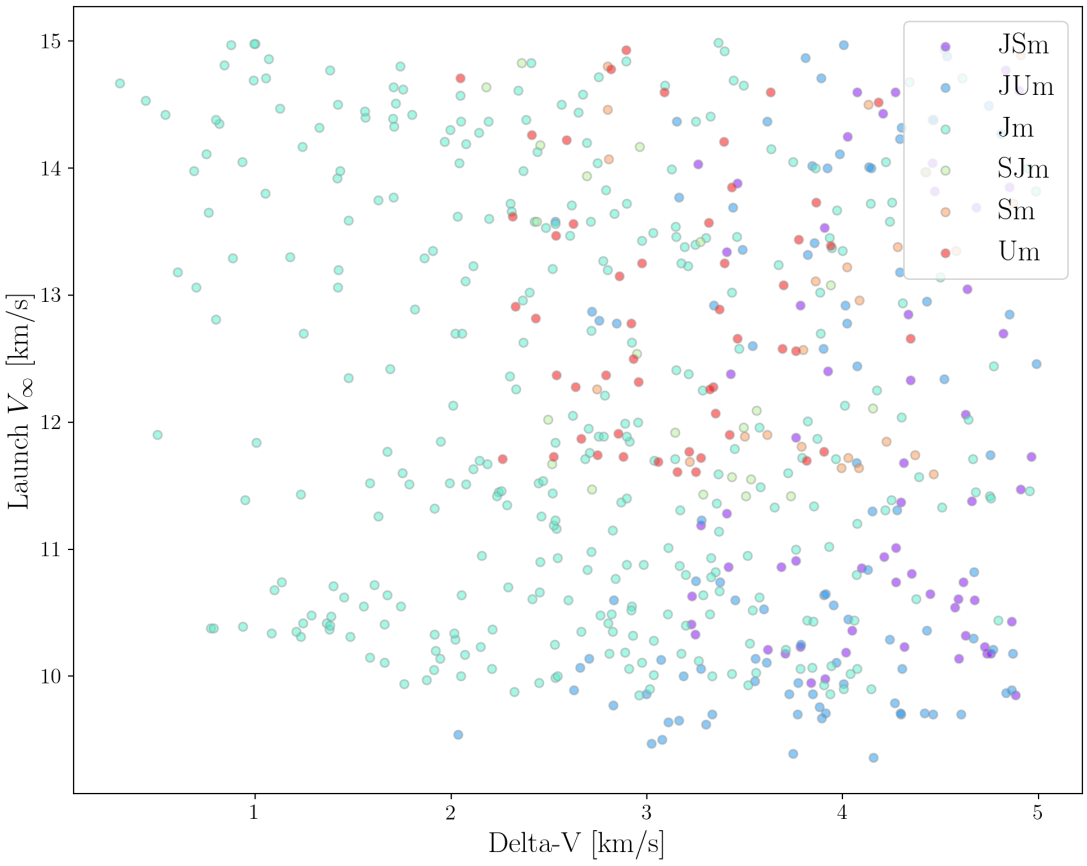


Figure 7.9. The launch V_∞ is plotted against the cumulative ΔV for each of the patched conic variations on the network solution. The Jupiter-Makemake path dominates the lower left region which represents lower launch cost and lower mission ΔV. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

The lower end of the Launch V_∞ spectrum in Figure 7.11 represents Voyager class launch energy. There are many trajectories in the middle band (around 20 year flight time) that are within mission experience. The upper band includes lower launch V_∞ and lower ΔV requirements, but the flight time is extended to about 25 years.

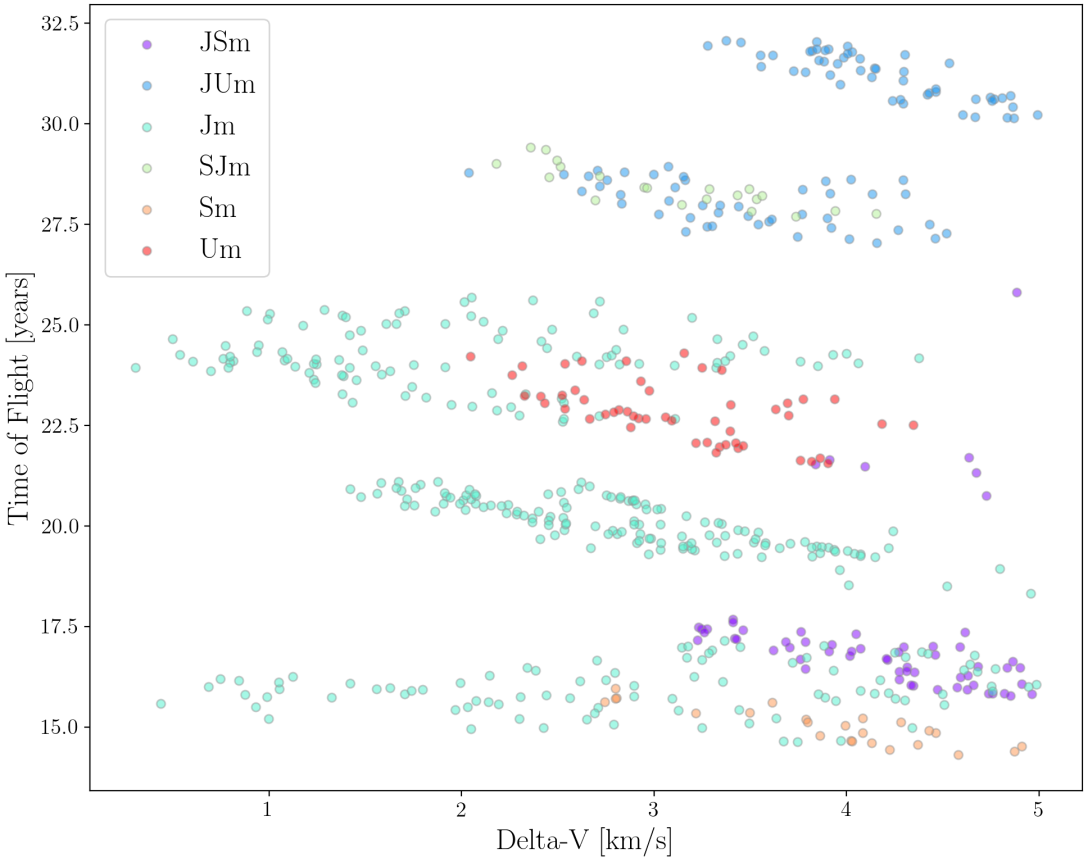


Figure 7.10. The time of flight is plotted against the cumulative ΔV for each of the patched conic variations on the network solution. Jupiter-Makemake trajectories are available with low ΔV. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

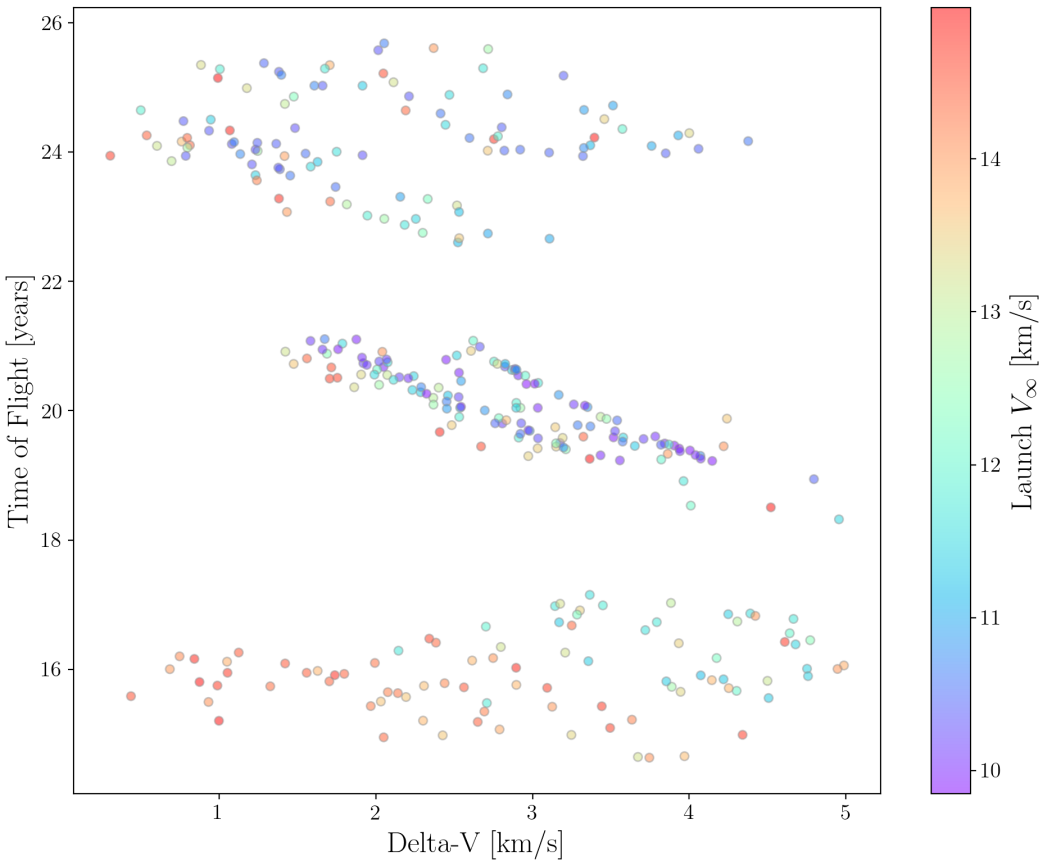


Figure 7.11. The time of flight is plotted against the cumulative ΔV for the Jupiter-Makemake path only. The points are colored according to launch V_∞. The fastest and least ΔV costly trajectories in the lower left require large launch energy. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown.

7.3 Trans-Neptunian Object Search Summary

The Tisserand network can be extended to include any solar system object with an available ephemeris. The example searches in this chapter identified preliminary paths to Haumea and Makemake using the same techniques applied to the planets in the previous chapters. The searches found multiple opportunities for missions in the near future with moderate ΔV requirements and launch energy requirements within flight experience and nascent launch vehicle capabilities.

8. CONCLUSION

This dissertation has presented a new method for performing gravity-assist trajectory searches. The Tisserand network captures the necessary astrodynamical information within its vertices and edges. Fundamental graph traversal techniques were tailored into search algorithms for the gravity-assist trajectory problem. The multiple gravity assist trajectory problem has a tendency to grow in size rapidly as more options are considered. The efficiency of the search algorithms was discussed and methods to manage problem growth and search time were presented. These algorithms were then demonstrated on networks designed for specific missions (both new and old).

8.1 Future Work

The present research has generated some related opportunities for exploration. Unfortunately, time does not permit a detailed examination of every interesting side topic. A few open areas of study are introduced below.

One of the original motivations for this research was the problem of finding gravity assist trajectories for tours of the giant planet satellites. Such tours require dozens of moon flybys to achieve the desired trajectory turning. Manual inspection of a Tisserand graph is impractical for more than a few gravity assists. This problem calls for an automated means of identifying energy-feasible sequences.

Therefore, the search for satellite tours with the Tisserand network method may be a fruitful area of research. To extend the basic method described in this work to a planetary satellite system we need the gravitational parameters of the satellites, the mean radii of each satellite orbit, and an appropriate limit on flyby radius for each satellite. The system central body must be changed from the Sun to the planet of interest. This is effectively only a change in gravitational parameter. Future researchers should, of course, evaluate the assumptions and limitations in Chapter 3 in light of the system of interest and analysis goals.

The Tisserand network provides a graph theory framework for gravity assist design problems. The present work has been focused on search techniques within this framework. However, there may be additional information to be gained about the nature of the design space

from analysis of the network itself. For example, studies of the connectivity and clustering of the network may yield new insights about the importance of certain bodies or synodic periods for the trajectory availability.

Additional analysis and experimentation with the broken conics resulting from the Tisserand network search might show that these results can provide a reasonable estimate of the flight times for the associated patched-conic trajectories. For example, the sum of the flight times from each leg of the Tisserand network solution might predict (or bound) the flight time of the optimized patched-conic trajectory using the procedure outlined in Equation 3.33. If such a relationship could be established, then we might be justified in performing network searches with Dijkstra's algorithm to find the fastest path to a planet. A shortest-path search is much more computationally efficient than the all-paths search examined in this work. A k-shortest paths search might also provide a group of k network paths which is likely to contain the fastest patched-conic trajectory.

8.1.1 Feature Enhancements

The resonant model discussed in Chapter 4 only includes $2n\pi$ transfers. This model requires that sequential encounters of the same planet occur at the same inertial location after the completion of a whole number of orbits of both the planet and the spacecraft. Adding a non-resonant ($n\pi$) repeat flyby model would increase the pathfinding potential of the Tisserand network [34], [35].

The VILT model in Chapter 4 assumes that the leveraging maneuver is tangent to the spacecraft orbit and occurs at apoapsis or periapsis of the heliocentric orbit. The model also assumes that any leveraging maneuver occurs between successive gravity assists from a single planet. Removing these assumptions could provide additional search capability. For example, the techniques used to scan for VILT opportunities could be expanded for the case where the arrival body is different than the departure body.

It may also be possible to expand the existing VILT model to apply to general Deep Space Maneuvers (DSMs). A DSM is a maneuver that occurs during the heliocentric orbit (not in the vicinity of the planet). Coupled with an extension to different-body transfers, a more

general DSM model could loosen constraints on the maneuver location. The key output of any such model is the time of flight between the two flybys for weighting the network edges. Some simplifying assumptions will be required to limit the possible scenarios.

The Monte Carlo method for generating patched-conic trajectories is fast and provides a reasonable characterization of each network path. The optimization technique can refine the random patched-conic trajectories into a smaller set of more interesting trajectories. However, the optimization procedure becomes slower as more gravity assist encounters are added. Optimizing many randomized patched-conic trajectories with several gravity assists can become computationally expensive. Moreover, preliminary experimentation with this method has shown that many of the randomized trajectories will optimize to the same result. Reducing the number of Monte Carlo cases per network path would reduce execution time but might miss important features of some paths. A study to find the best number of Monte Carlo cases could further refine the final results to highlight nearly-ballistic trajectories without an undue increase in execution time.

As an alternative to the Monte Carlo patched-conic trajectory generation, a future improvement might be to integrate a more structured grid search for gravity assists that do not require significant propulsive ΔV. The grid search can be localized to a small search space based on the network search results.

The Tisserand network could be extended to include transfers with inclination changes. This update might be desirable to provide better preliminary trajectories for satellite tours, missions to Trans-Neptunian Objects, or other missions intended to deviate significantly from the ecliptic plane.

The Tisserand graph was developed from the basic relationships between the gravity-assist pump angle and the heliocentric orbit in Section 2.1. Strange [34] includes a similar discussion of the relationships between the gravity-assist pump and crank angles and the resulting spacecraft orbits. These relationships might be employed to add an inclination component to the Tisserand network approach. Campagnola and Russell [67] also provide some discussion of a three dimensional Tisserand-Poincaré graph that includes inclination.

8.1.2 Performance Improvements

The bounded all paths and trace searches presented in Chapter 5 have so far only been implemented in Python. Considerable speed improvements can be expected by translating these algorithms to a compiled language.

The transitive closure modification to the bounded search suggested in Chapter 5 would improve search speed regardless of the language implementation. This improvement would use the transitive closure to verify the viability of a prospective branch before proceeding.

Future researchers might explore areas where search speed can be improved through the use of parallel computing. Algorithm 1 has been parallelized, but in many basic searches the multiprocessing overhead outweighs the performance benefit. For dense networks, the weighting procedure can become costly. This process repeats a common task on each edge in the network and would be a good target for multiprocessing.

8.2 Summary of the Contribution

The Tisserand network developed in this research provides a new approach to gravity-assist trajectory design that solves the pathfinding and pathsolving problems simultaneously. The method is configurable to allow trajectory searches for any solar system destination with minimal inputs.

New network-oriented models for resonance and V-infinity leveraging transfers expand the types of trajectories that can be found with the Tisserand network analysis. The models also provide opportunities for further expansion of the network technique.

The new method automatically performs the task of finding energy-feasible gravity-assist paths in a fraction of the time required by manual Tisserand graph analysis. The phasing information included in the network leads to search results that are feasible with respect to both energy and scheduling constraints. The search results are suitable for preliminary comparisons and can focus higher fidelity analyses on promising candidates. The Tisserand network enables a mission designer to find more preliminary trajectories of better quality and in less time than was previously possible.

REFERENCES

[1] G. Flandro, "From Instrumented Comets to Grand Tours - on the History of Gravity Assist," in *39th Aerospace Sciences Meeting and Exhibit*, Reno, NV: American Institute of Aeronautics and Astronautics, Jan. 2001.

[2] R. B. Negri, "A Historical Review of the Theory of Gravity-Assists in the Pre-Spaceflight Era," *Journal of the Brazilian Society of Mechanical Sciences and Engineering*, vol. 42, no. 406, pp. 1–10, 2020.

[3] K. A. Ehricke, "Instrumented Comets—Astronautics of Solar and Planetary Probes," in *VIIIth International Astronautical Congress Barcelona 1957*, Berlin: Springer, 1958, pp. 74–126.

[4] M. A. Minovitch, "The Invention That Opened the Solar System to Exploration," *Planetary and Space Science*, vol. 58, no. 6, pp. 885–892, May 2010.

[5] G. Flandro, "Fast Reconnaissance Missions to the Outer Solar System Utilizing Energy Derived from the Gravitational Field of Jupiter," *Acta Astronautica*, vol. 12, no. 4, pp. 329–337, Aug. 1966.

[6] G. Flandro, "Discovery of the Grand Tour Voyager Mission Profile," in *Planets Beyond: Discovering the Outer Solar System*, New York: John Wiley & Sons, Inc., 1988.

[7] D. W. Swift, *Voyager Tales: Personal Views of the Grand Tour*. Reston, Va: American Institute of Aeronautics and Astronautics, 1997.

[8] S. N. Williams, "Automated Design of Multiple Encounter Gravity-Assist Trajectories," M.S. thesis, Purdue University, West Lafayette, IN, 1990.

[9] N. J. Strange and J. M. Longuski, "Graphical Method for Gravity-Assist Trajectory Design," *Journal of Spacecraft and Rockets*, vol. 39, no. 1, pp. 9–16, Jan. 2002.

[10] D. de la Torre Sangrà, E. Fantino, R. Flores, O. Calvente Lozano, and C. García Estelrich, "An Automatic Tree Search Algorithm for the Tisserand Graph," *Alexandria Engineering Journal*, (in press), 2020.

[11] K. M. Hughes, J. W. Moore, and J. M. Longuski, "Preliminary Analysis of Ballistic Trajectories to Neptune via Gravity Assists from Venus, Earth, Mars, Jupiter, Saturn, and Uranus," in *AAS/AIAA Astrodynamics Specialist Conference*, Hilton Head Island, SC, Aug. 2013.

[12] A. Bellome, J.-P. S. Cuartielles, L. Felicetti, and S. Kemble, "Modified Tisserand Map Exploration for Preliminary Multiple Gravity Assist Trajectory Design," in *Proceedings of the 71st International Astronautical Congress*, Oct. 2020.

[13] D. V. Lantukh, "Preliminary Design of Spacecraft Trajectories for Missions to Outer Planets and Small Bodies," PhD Thesis, University of Texas, Austin, TX, Aug. 2015.

[14] K. M. Hughes, "Gravity-Assist Trajectories to Venus, Mars, and the Ice Giants: Mission Design with Human and Robotic Applications," PhD Thesis, Purdue University, West Lafayette, IN, 2016.

[15] A. J. Mudek, J. W. Moore, K. M. Hughes, S. J. Saikia, and J. M. Longuski, "Ballistic and High-Thrust Trajectory Options to Uranus Considering 50 Years of Launch Dates," in *AAS/AIAA Astrodynamics Specialist Conference*, Stevenson, WA, Aug. 2017.

[16] D. Landau, S. Campagnola, and E. Pellegrini, "Star Searches for Patched-Conic Trajectories," *The Journal of the Astronautical Sciences*, vol. 69, no. 6, pp. 1613–1648, Nov. 2022.

[17] A. J. Mudek, "50-Year Catalogs of Uranus Trajectory Options with a New Python-Based Rapid Design Tool," PhD Thesis, Purdue University, West Lafayette, IN, 2022.

[18] I. M. Ross and C. N. D'Souza, "Hybrid Optimal Control Framework for Mission Planning," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 4, pp. 686–697, Jul. 2005.

[19] M. Vasile, J. M. R. Martin, L. Masi, *et al.*, "Incremental Planning of Multi-Gravity Assist Trajectories," *Acta Astronautica*, vol. 115, pp. 407–421, Oct. 2015.

[20] J. Englander, "Automated Trajectory Planning for Multiple-Flyby Interplanetary Missions," PhD Thesis, University of Illinois, Urbana-Champaign, 2013.

[21] D. H. Ellison, "Robust Preliminary Design for Multiple Gravity Assist Spacecraft Trajectories," PhD Thesis, University of Illinois, Urbana-Champaign, 2018.

[22] T. Crain, R. H. Bishop, W. Fowler, and K. Rock, "Interplanetary Flyby Mission Optimization Using a Hybrid Global-Local Search Method," *Journal of Spacecraft and Rockets*, vol. 37, no. 4, pp. 468–474, Jul. 2000.

[23] M. Vavrina, "A Hybrid Genetic Algorithm Approach to Global Low-Thrust Trajectory Optimization," M.S. thesis, Purdue University, West Lafayette, IN, 2008.

[24] 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, pp. 2096–2107, Nov. 2015.

[25] M. Ceriotti and M. Vasile, "MGA trajectory planning with an ACO-inspired algorithm," *Acta Astronautica*, vol. 67, no. 9-10, pp. 1202–1217, Nov. 2010.

[26] D. Izzo, "Global Optimization and Space Pruning for Spacecraft Trajectory Design," in *Spacecraft Trajectory Optimization*, ser. Cambridge Aerospace Series 29, B. A. Conway, Ed., Cambridge: Cambridge University Press, 2010, pp. 178–201.

[27] G. A. Tsirogiannis, "A Graph Based Methodology for Mission Design," *Celestial Mechanics and Dynamical Astronomy*, vol. 114, no. 4, pp. 353–363, Dec. 2012.

[28] E. Trumbauer and B. Villac, "Heuristic Search-Based Framework for Onboard Trajectory Redesign," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 1, pp. 164–175, Jan. 2014.

[29] A. Das-Stuart, K. Howell, and D. Folta, "Rapid Trajectory Design in Complex Environments Enabled by Reinforcement Learning and Graph Search Strategies," *Acta Astronautica*, vol. 171, pp. 172–195, Jun. 2020.

[30] A. Das, "Artificial Intelligence Aided Rapid Trajectory Design in Complex Dynamical Environments," PhD Thesis, Purdue University, West Lafayette, IN, 2019.

[31] J. R. Stuart, "A Hybrid Systems Strategy for Automated Spacecraft Tour Design and Optimization," PhD Thesis, Purdue University, West Lafayette, IN, 2014.

[32] J. R. Stuart, K. C. Howell, and R. S. Wilson, "Design of End-to-End Trojan Asteroid Rendezvous Tours Incorporating Scientific Value," *Journal of Spacecraft and Rockets*, vol. 53, no. 2, pp. 278–288, Mar. 2016.

[33] J. A. Sims, J. M. Longuski, and A. J. Staugler, "V-infinity Leveraging for Interplanetary Missions: Multiple-Revolution Orbit Techniques," *Journal of Guidance, Control, and Dynamics*, vol. 20, no. 3, pp. 409–415, May 1997.

[34] N. J. Strange, "Analytical Methods for Gravity-Assist Tour Design," PhD Thesis, Purdue University, West Lafayette, IN, 2016.

[35] C.-Y. Wu and R. P. Russell, "Reachable Set of Low-Delta-v Trajectories Following a Gravity-Assist Flyby," *Journal of Spacecraft and Rockets*, (in press), pp. 1–18, Jan. 2023.

[36] R. R. Bate, D. D. Mueller, and J. E. White, *Fundamentals of Astrodynamics*. New York: Dover Publications, 1971.

[37] H. D. Curtis, *Orbital Mechanics for Engineering Students* (Elsevier Aerospace engineering series), 1. ed., reprinted. Amsterdam: Elsevier/Butterworth-Heinemann, 2008.

[38] G. R. Hintz, *Orbital Mechanics and Astrodynamics*. Cham: Springer International Publishing, 2015.

[39] D. Vallado, *Fundamentals of Astrodynamics and Applications* (Space Technology Library). El Segundo, Dordrecht: Microcosm Press, Kluwer Academic Publishers, 2001.

[40] R. H. Battin, *An Introduction to the Mathematics and Methods of Astrodynamics* (AIAA education series), Rev. ed. Reston, VA: American Institute of Aeronautics and Astronautics, 1999.

[41] A. Prado, "A Comparison of the Patched-Conics Approach and the Restricted Problem for Swing-Bys," *Advances in Space Research*, vol. 40, no. 1, pp. 113–117, 2007.

[42] R. B. Negri, A. F. B. d. A. Prado, and A. Sukhanov, "Studying the Errors in the Estimation of the Variation of Energy by the Patched-Conics Model in the Three-Dimensional Swing-By," *Celestial Mechanics and Dynamical Astronomy*, vol. 129, no. 3, pp. 269–284, Nov. 2017.

[43] A. E. Roy, *Orbital Motion*, 3rd ed. Bristol, England ; Philadelphia: A. Hilger, 1988.

[44] N. Bradley and R. P. Russell, "A Continuation Method for Converting Trajectories from Patched Conics to Full Gravity Models," *The Journal of the Astronautical Sciences*, vol. 61, no. 3, pp. 227–254, Sep. 2014.

[45] J. Miller and C. Weeks, "Application of Tisserand's Criterion to the Design of Gravity Assist Trajectories," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Monterey, California: American Institute of Aeronautics and Astronautics, Aug. 2002.

[46] F. Tisserand, *Traité De Mécanique Céleste*. Gauthier-Villars et fils, 1896, vol. 4.

[47] J. Danby, *Fundamentals of Celestial Mechanics*. New York: Macmillan, 1962.

[48] V. Szebehely, *Theory of Orbits: The Restricted Problem of Three Bodies*. 1967.

[49] A. V. Labunsky, O. V. Papkov, and K. G. Sukhanov, *Multiple Gravity Assist Interplanetary Trajectories*. CRC Press, 1998.

[50] J. M. Longuski and S. N. Williams, "Automated Design of Gravity-Assist Trajectories to Mars and the Outer Planets," *Celestial Mechanics and Dynamical Astronomy*, vol. 52, no. 3, pp. 207–220, 1991.

[51] G. Chartrand, *Introductory Graph Theory*. New York: Dover Publications, 1985.

[52] R. J. Trudeau, *Introduction to Graph Theory*. New York: Dover Publications, 1993.

[53] A. Gibbons, *Algorithmic Graph Theory*. Cambridge University Press, 1985.

[54] R. Sedgewick, *Algorithms* (Addison-Wesley series in computer science). Addison-Wesley, 1988.

[55] J. W. Moore and J. M. Longuski, "Network Analysis of Tisserand Graphs for Automated Gravity-Assist Pathfinding," in *AAS/AIAA Astrodynamics Specialist Conference*, Lake Tahoe, CA, Aug. 2020.

[56] J. L. Horsewood and J. D. Vickery, "Mission Window Definition for Jupiter Swingbys to the Outer Planets.," *Journal of Spacecraft and Rockets*, vol. 6, no. 5, pp. 525–531, May 1969.

[57] C. Acton, N. Bachman, B. Semenov, and E. Wright, "A Look Towards the Future in the Handling of Space Science Mission Geometry," *Planetary and Space Science*, vol. 150, pp. 9–12, Jan. 2018.

[58] D. Izzo, "Revisiting Lamberts Problem," *Celestial Mechanics and Dynamical Astronomy*, vol. 121, no. 1, pp. 1–15, Jan. 2015.

[59] K. Qadir, "Multi-Gravity Assist Design Tool for Interplanetary Trajectory Optimization," M.S. thesis, Lulea University, Sweden, 2009.

[60] G. Hollenbeck, "New Options for Outer Planet Exploration," in *Conference on the Exploration of the Outer Planets*, St. Louis, MO: American Institute of Aeronautics and Astronautics, Sep. 1975.

[61] J. A. Sims, "Delta-V Gravity-Assist Trajectory Design: Theory and Practice," PhD Thesis, Purdue University, West Lafayette, IN, 1998.

[62] S. Campagnola, N. J. Strange, and R. P. Russell, "A Fast Tour Design Method Using Non-Tangent V-Infinity Leveraging Transfer," *Celestial Mechanics and Dynamical Astronomy*, vol. 108, no. 2, pp. 165–186, Oct. 2010.

[63] F. Peralta and S. Flanagan, "Cassini Interplanetary Trajectory Design," *Control Engineering Practice*, vol. 3, no. 11, pp. 1603–1610, Nov. 1995.

[64] S. Flanagan and F. Peralta, "Cassini 1997 VVEJGA Trajectory Launch/Arrival Space Analysis," in *Proceedings of the AAS/AIAA Astrodynamics Conference*, Victoria, Canada, Aug. 1993, pp. 1587–1607.

[65] T. Goodson, D. Gray, Y. Hahn, and F. Peralta, "Cassini Maneuver Experience - Launch and Early Cruise," in *Guidance, Navigation, and Control Conference and Exhibit*, Boston, MA: American Institute of Aeronautics and Astronautics, Aug. 1998.

[66] D. Dunham, J. McAdams, and R. Farquhar, "NEAR Mission Design," *Johns Hopkins APL Technical Digest (Applied Physics Laboratory)*, vol. 23, no. 1, pp. 18–33, Jan. 2002.

[67] S. Campagnola and R. P. Russell, "Endgame Problem Part 2: Multibody Technique and the Tisserand-Poincaré Graph," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 476–486, Mar. 2010.

[68] D. Kraft, "A Software Package for Sequential Quadratic Programming," Deutsche Forschungs- und Versuchsanstalt fur Luft- und Raumfahrt, Oberpfaffenhofen, Tech. Rep. DFVLR-FB 88-28, Jul. 1988.

[69] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.

[70] T. P. Peixoto, "The Graph-Tool Python Library," *figshare*, 2014.

[71] D. Izzo, "Pygmo and Pykep: Open Source Tools for Massively Parallel Optimization in Astrodynamics (the Case of Interplanetary Trajectory Optimization)," in *Proceedings of the 5th International Conference on Astrodynamics Tools and Techniques*, Noordwijk, Netherlands, Jan. 2012.

[72] B. Evans, *NASA's Voyager Missions: Exploring the Outer Solar System and Beyond*. Cham: Springer International Publishing, 2022.

[73] R. Beebe, *Jupiter: The Giant Planet* (Smithsonian Library of the Solar System), 2nd ed. Washington: Smithsonian Institution Press, 1997.

[74] M. Meltzer, "Mission to Jupiter: A History of the Galileo Project," NASA History Division, Washington, DC, Tech. Rep. SP-2007-4231, 2007.

[75] L. A. D'Amario, L. E. Bright, and A. A. Wolf, "Galileo Trajectory Design," *Space Science Reviews*, vol. 60, no. 1, pp. 23–78, May 1992.

[76] C. T. Russell, Ed., *The Cassini-Huygens Mission: Overview, Objectives and Huygens Instrumentarium Volume 1*. Dordrecht: Springer Netherlands, 2003.

[77] R. Mcgranaghan, B. Sagan, G. Dove, A. Tullos, J. Lyne, and J. Emery, "A Survey of Mission Opportunities to Trans-Neptunian Objects," *Journal of the British Interplanetary Society*, vol. 64, pp. 296–303, Sep. 2011.

[78] A. M. Zangari, T. J. Finley, S. Alan Stern, and M. B. Tapley, "Return to the Kuiper Belt: Launch Opportunities from 2025 to 2040," *Journal of Spacecraft and Rockets*, vol. 56, no. 3, pp. 919–930, May 2019.

[79] A. Gleaves, R. Allen, A. Tupis, *et al.*, "A Survey of Mission Opportunities to Trans-Neptunian Objects - Part II, Orbital Capture," in *AIAA/AAS Astrodynamics Specialist Conference*, Minneapolis, MN: American Institute of Aeronautics and Astronautics, Aug. 2012.

[80] J. Poncy, J. Fontdecaba Baig, F. Feresin, and V. Martinot, "A Preliminary Assessment of an Orbiter in the Haumean System: How Quickly Can a Planetary Orbiter Reach Such a Distant Target?" *Acta Astronautica*, vol. 68, no. 5-6, pp. 622–628, Mar. 2011.

[81] D. A. Smith, "Space Launch System (SLS) Mission Planner's Guide," NASA, Tech. Rep. ESD30000, 2018.

[82] M. E. Brown, "The Largest Kuiper Belt Objects," in *The Solar System Beyond Neptune*, M. A. Barucci, H. Boehnhardt, D. P. Cruikshank, and A. Morbidelli, Eds., Tuscon, AZ: University of Arizona Press, 2008, pp. 335–344.

[83] M. Okutsu and J. M. Longuski, "Mars Free Returns via Gravity Assist from Venus," *Journal of Spacecraft and Rockets*, vol. 39, no. 1, pp. 31–36, Jan. 2002.

[84] A. F. Heaton, N. J. Strange, J. M. Longuski, and E. P. Bonfiglio, "Automated Design of the Europa Orbiter Tour," *Journal of Spacecraft and Rockets*, vol. 39, no. 1, pp. 17–22, Jan. 2002.

[85] A. F. Heaton and J. M. Longuski, "Feasibility of a Galileo-Style Tour of the Uranian Satellites," *Journal of Spacecraft and Rockets*, vol. 40, no. 4, pp. 591–596, Jul. 2003.

[86] S. Campagnola and Y. Kawakatsu, "Jupiter Magnetospheric Orbiter: Trajectory Design in the Jovian system," *Journal of Spacecraft and Rockets*, vol. 49, no. 2, pp. 318–324, Mar. 2012.

[87] W. R. Johnson, "Analysis and Design of Aeroassisted Interplanetary Missions.," PhD Thesis, Purdue University, West Lafayette, IN, 2003.

[88] W. R. Johnson and J. M. Longuski, "Design of Aerogravity-Assist Trajectories," *Journal of Spacecraft and Rockets*, vol. 39, no. 1, pp. 23–30, Jan. 2002.

[89] K.-H. Chen, K. Kloster, and J. Longuski, "A Graphical Method for Preliminary Design of Low-Thrust Gravity-Assist Trajectories," in *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, Honolulu, HI: American Institute of Aeronautics and Astronautics, Aug. 2008.

[90] V. Maiwald, "Applicability of Tisserand Criterion for Optimization of Gravity-Assist Sequences for Low-Thrust Missions," in *66th International Astronautical Congress*, Jerusalem, Israel, 2015.

[91] V. Maiwald, "A New Method for Optimization of Low-Thrust Gravity-Assist Sequences," *CEAS Space Journal*, vol. 9, no. 3, pp. 243–256, Sep. 2017.

[92] S. Campagnola and R. P. Russell, "Endgame Problem Part 1: V-Infinity-Leveraging Technique and the Leveraging Graph," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 2, pp. 463–475, Mar. 2010.

[93] S. Campagnola, A. Boutonnet, J. Schoenmaekers, D. J. Grebow, A. E. Petropoulos, and R. P. Russell, "Tisserand-Leveraging Transfers," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 4, pp. 1202–1210, Jul. 2014.

[94] D. G. Yárnoz, C. H. Yam, S. Campagnola, and Y. Kawakatsu, "Extended Tisserand-Poincaré Graph and Multiple Lunar Swingby Design with Sun Perturbation," in *Proceedings of the 6th International Conference on Astrodynamics Tools and Techniques*, Darmstadt, Germany, 2016.

[95] M. Pugliatti, "The Extended Tisserand-Poincaré Graph for Multi-Body Trajectory Design," M.S. thesis, Delft University of Technology, Delft, Netherlands, 2018.

[96] D. R. Jones, S. Hernandez, and M. Jesick, "Low Excess Speed Triple Cyclers of Venus, Earth, and Mars," in *AAS/AIAA Astrodynamics Specialist Conference*, Stevenson, WA, 2017.

[97] S. Hernandez, D. R. Jones, and M. Jesick, "One Class of Io-Europa-Ganymede Triple Cyclers," in *AAS/AIAA Astrodynamics Specialist Conference*, Stevenson, WA, 2017.

[98] Y. Chen, H. Baoyin, and J. Li, "Accessibility of Main-Belt Asteroids via Gravity Assists," *Journal of Guidance, Control, and Dynamics*, vol. 37, no. 2, pp. 623–632, Mar. 2014.

A. THE TISSERAND GRAPH

This appendix includes some background material on the use and application of the Tisserand graph. Figure A.1 is a sample Tisserand graph including the planets Venus, Earth, Mars, and Jupiter.

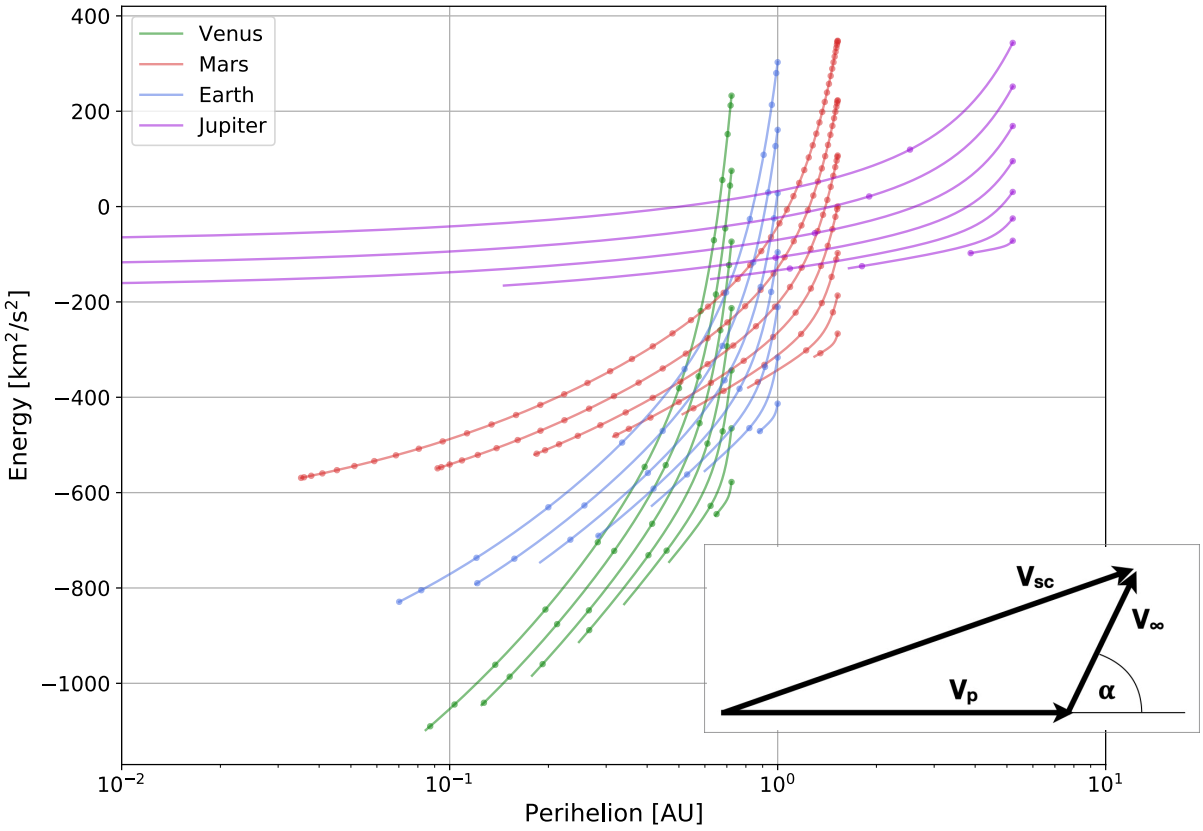


Figure A.1. A sample Tisserand graph including Venus, Earth, Mars, and Jupiter. The contours for each planet represent the locus of heliocentric orbit parameters for flybys at a given V_∞. Intersections of the contours identify pairs of flybys for two planets connected by the same heliocentric orbit. The inset shows the relationship between the V_∞ vector and the planet velocity, spacecraft heliocentric velocity, and pump angle (V_p, V_sc, and α, respectively).

A.1 Demonstration

Here we review how to graphically construct a flyby sequence in the Tisserand graph. Figure A.2 is a close-up view of Figure A.1 with a sample path highlighted. In both figures,

the lower-right contour in each color represents a flyby of the given planet at a V_∞ of 1 km/s. The contours increase up and to the left at 3 km/s intervals. The *pump* angle, α, is the angle between the flyby V_∞ vector and the planet's heliocentric velocity vector. A single contour traces out the heliocentric orbit parameters that result from a flyby at the given V_∞ as the pump angle varies from 0 to π. We will use the nomenclature "Earth-4" or "E4" to refer to the Earth contour where V_∞ is 4 km/s. We will refer to intersections by their component contours. So "V7E4" is the intersection of the Venus 7 km/s contour and the Earth 4 km/s contour.

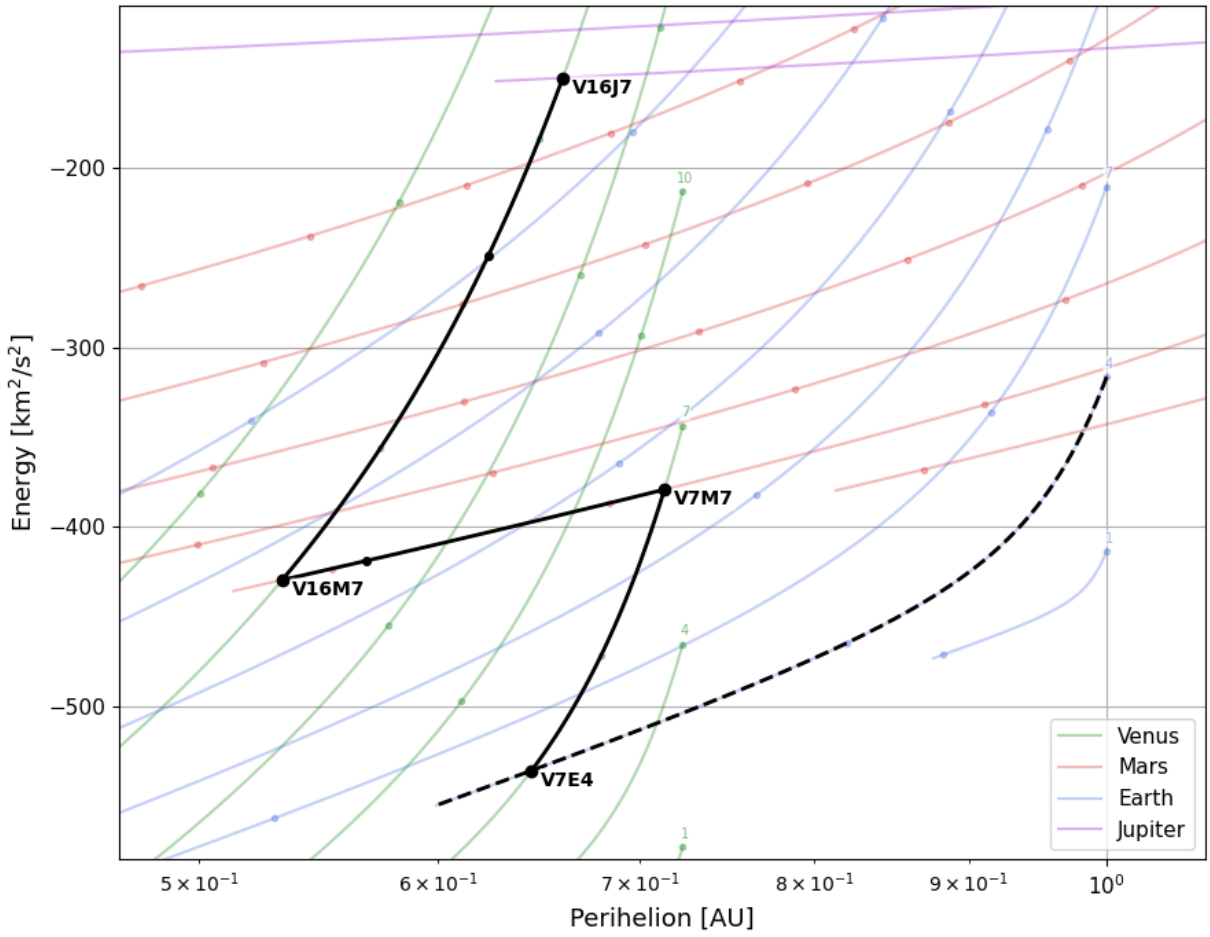


Figure A.2. A small section of a Tisserand graph highlights a potential path from Earth to Jupiter. Starting at the V7E4 node and proceeding along the V_∞ contours through V7M7, V16M7, and V16J7 indicates that a path VMMVVJ exists (from an energy perspective).

To start tracing out the sample trajectory, we assume an Earth departure V_∞ of 4 km/s (represented by the dashed line in Figure A.2). We can visually scan the dashed line for an intersection with another contour. In this case, the V7E4 intersection is highlighted. This intersection represents a heliocentric orbit that leaves Earth at 4 km/s and encounters Venus at 7 km/s.

By following the Venus-7 contour upward we consider the family of Venus flybys with a V_∞ of 7 km/s but with decreasing pump angles. Eventually, we find an intersection (V7M7) with the Mars-7 contour. Here the pump angle at Venus is sufficient to intersect the Mars orbit. In particular, this intersection represents a heliocentric orbit that departs Venus at 7 km/s and encounters Mars at 7 km/s.

Tracing the Mars-7 contour to the left we eventually find the intersection with the Venus-16 contour. However, before reaching that intersection we notice the black dot on the Mars-7 contour. This dot represents the maximum turning angle (or distance along the contour) that can be achieved with a single flyby of Mars (due to a practical limit on the periapsis of the flyby hyperbola). In other words, the transfer from Mars-7 to Venus-16 requires two Mars flybys.

From the V16M7 intersection, we trace the Venus-16 contour upward until we encounter the V16J7 intersection. Here we note again that two Venus flybys are required to boost the heliocentric energy sufficiently high enough to reach the Jupiter-7 contour. This final leg indicates that a series of two flybys of Venus at 16 km/s will result in a heliocentric orbit that reaches Jupiter with a Jupiter-centric V_∞ of 7 km/s. To summarize, we have identified a "path" from Earth to Jupiter that we label VMMVVJ. By convention, the initial E for the Earth departure is implied. We know the planet and the planet-centric V_∞ of each encounter on the path.

A.2 History, Applications, and Advancements

The following is a partial list of authors who have employed and extended the Tisserand graph for mission planning.

A.2.1 Early Development

An early study by Strange and Longuski develops the Tisserand graph for ballistic trajectories from fundamental gravity-assist principles [9]. The authors also present a search algorithm focused on finding minimum time-of-flight paths from the possible paths in the Tisserand graph. Miller and Weeks explore the connection between the Tisserand criterion and the Jacobi integral and describe the use of the Tisserand criterion to match gravity-assist trajectories [45]. Okutsu and Longuski discovered a Mars free-return abort trajectory that meets mission constraints by adding a flyby of Venus [83].

A.2.2 Application of the Tisserand Graph to Satellite Tours

Heaton et al. design a tour for the proposed Europa Orbiter using the Tisserand graph [84]. Heaton and Longuski investigate the feasibility of tours of the Uranian satellites with the help of the Tisserand graph [85]. Campagnola and Kawakatsu use the Tisserand graph to design a tour of the Jovian system [86].

A.2.3 Extension of Tisserand Analysis Methods for Aero-Gravity-Assist

Johnson and Longuski apply the Tisserand graph to aero-gravity-assist trajectories [87], [88]. In this application, the V_∞ at approach and departure are no longer identical. However, the aero-gravity-assist extends the amount of trajectory turning that can be achieved from a single flyby.

A.2.4 Extension of Tisserand Analysis Methods for Low-Thrust

Chen *et al.* [89] extend the Tisserand analysis technique to low-thrust trajectories. In this analysis, the constant V_∞ contours are replaced by numerically integrated trajectories assuming representative low-thrust control laws. The addition of thrust violates a basic assumption of the Tisserand criterion derivation. Maiwald develops a correction to the Tisserand criterion for low-thrust applications and an optimization method for combined low-thrust gravity-assist missions [90], [91].

A.2.5 Extension to Other Dynamical Models

The Tisserand graph was developed for circular, coplanar orbits about the central body—in other words, for analytical solutions in the patched two-body problem. Campagnola and Russell extend the Tisserand graph by adding numerical solutions to V_∞-leveraging maneuvers and create the Tisserand-leveraging graph [67]. Campagnola and Russell also extend the method to allow patched trajectories in the Circular Restricted 3-body Problem [92], [93]. The adapted tool is called the Tisserand-Poincaré graph. Yárnoz *et al.* [94] and Pugliatti [95] make a further extension by including the perturbing effect of a second primary body to form the Extended Tisserand-Poincaré graph.

A.2.6 Pathfinding for Gas Giant Catalogs

Hughes *et al.* [11], Mudek *et al.* [15], and Mudek [17] assemble catalogs of gravity-assist trajectories to Uranus and Neptune over the next several decades. These studies use the Tisserand graph as a preliminary step to identify candidate flyby paths. The candidate paths are then evaluated for feasibility by solving each path for specific launch dates and launch V_∞.

A.2.7 Cycler and Asteroid Applications

Jones *et al.* [96] use the Tisserand graph to construct triple cyclers of Venus, Earth, and Mars. Hernandez *et al.* [97] employ the Tisserand graph to identify potential triple cyclers in the Jovian moons. Chen *et al.* [98] use a Tisserand graph to study the accessibility of main belt asteroids using gravity assists from Earth and Mars.

B. DERIVATIONS

This appendix includes some mathematical derivations that are important for implementing the Tisserand network.

B.1 Time of Flight for Transfer Arcs

This section derives the time equations given in Strange and Longuski [9] and reproduced in Table 3.2 and below for convenience. Consider the geometry in Figure B.1

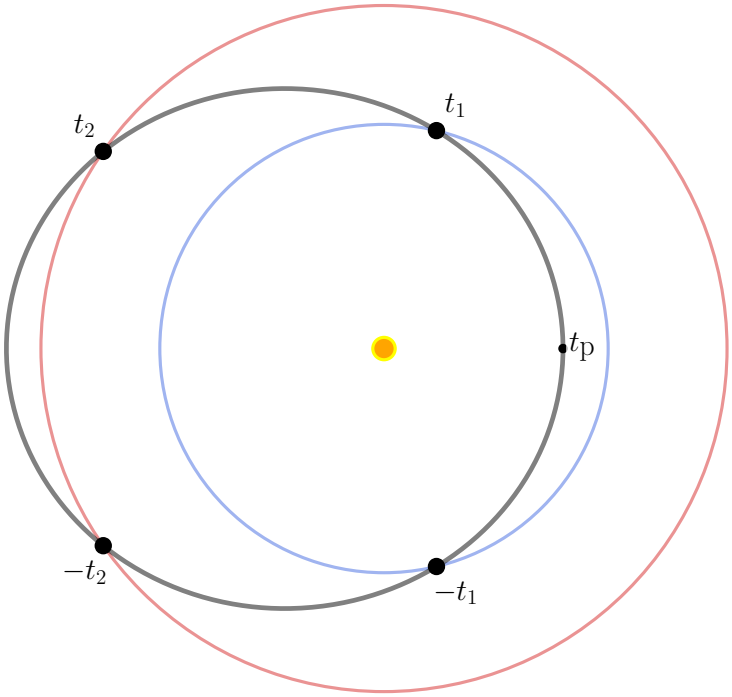


Figure B.1. An example of the four intersection points

Let us measure the times of the encounters relative to the time of periapsis passage on the transfer orbit, $t_p = 0$. Therefore, t_1 is the time required to reach the outbound crossing of the inner planet orbit and t_2 is the time required to reach the outbound crossing of the outer planet orbit. Similarly, $-t_2$ and $-t_1$ are the times required to reach periapsis from

the inbound crossings of the outer and inner planet orbits, respectively. There are eight combinations of up/down and inbound/outbound encounters. The eight different transfer arcs are visualized in Figures B.2 and B.3.

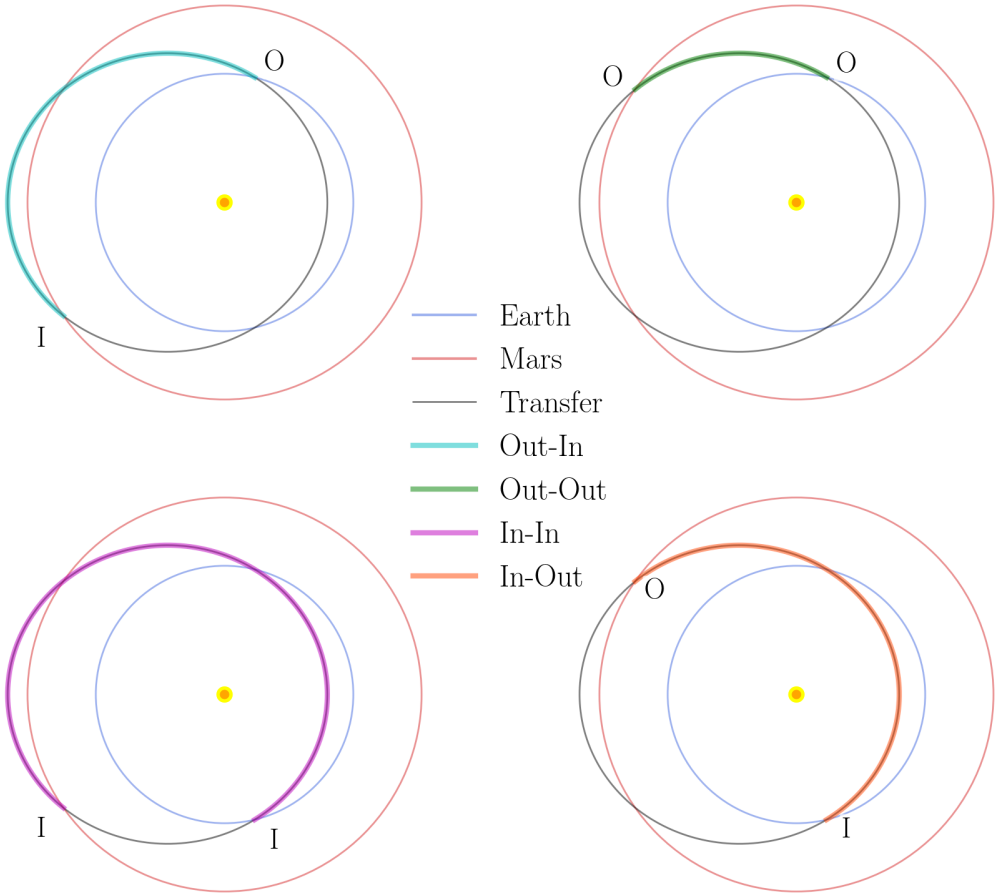


Figure B.2. An example of the four possible upward transfer arcs between planets in circular, coplanar orbits. This transfer orbit is defined by the intersection of the Earth-10 and Mars-7 contours and crosses each planetary orbit at two points (denoted I for inbound and O for outbound). The two points on one orbit connect to the two points on the other orbit to create four unique arcs in the transfer orbit. The transfer time depends on the arc and the direction of travel.

The time of flight for each arc is calculated in the following sections. Table B.1 summarizes the results.

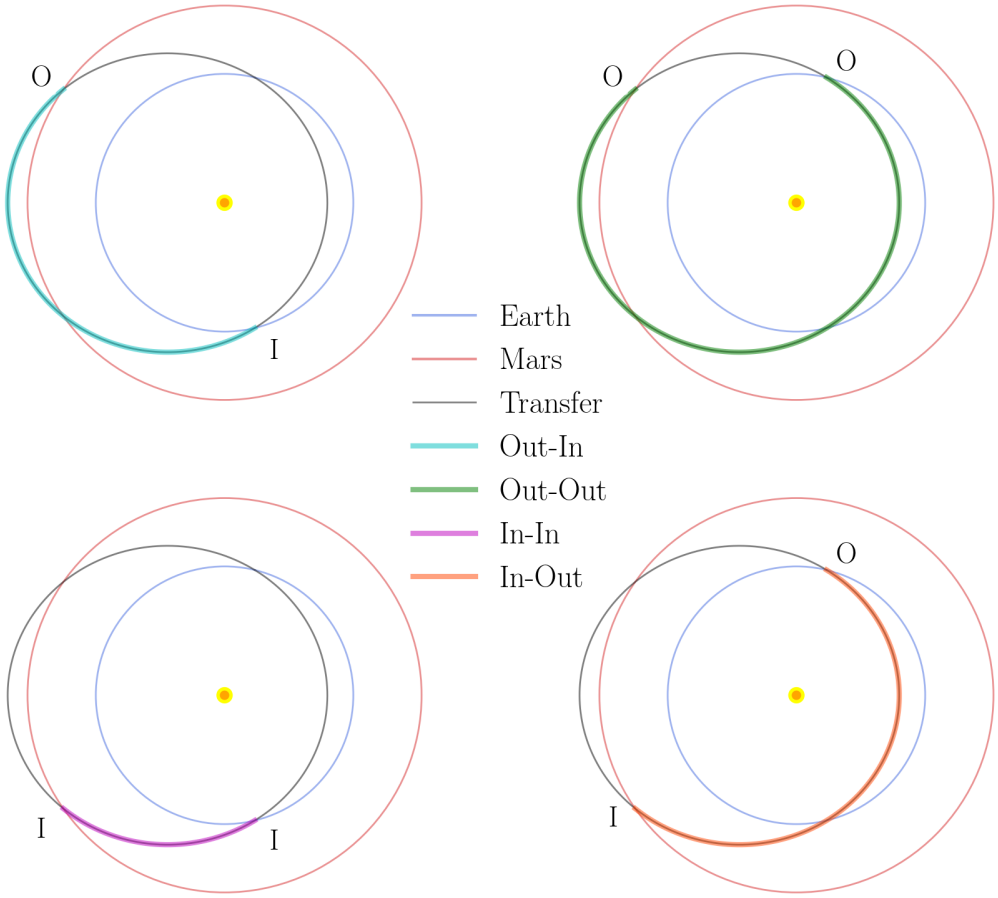


Figure B.3. An example of the four possible downward transfer arcs between planets in circular, coplanar orbits. This transfer orbit is defined by the intersection of the Earth-10 and Mars-7 contours and crosses each planetary orbit at two points (denoted I for inbound and O for outbound). The two points on one orbit connect to the two points on the other orbit to create four unique arcs in the transfer orbit. The transfer time depends on the arc and the direction of travel.

B.1.1 Upward, Outbound to Outbound

The transfer from t_1 to t_2 is simply

$$t_{\text{flight}} = t_2 - t_1 \,. \tag{B.1}$$

Table B.1. Time of Flight for Possible Arcs

Upward	Downward	Time of Flight
I-I	O-O	$P + t_1 - t_2$
O-I	O-I	$P - t_1 - t_2$
I-O	I-O	$t_1 + t_2$
O-O	I-I	$t_2 - t_1$

B.1.2 Upward, Inbound to Outbound

The transfer from $-t_1$ through periapsis to t_2 is

$$\begin{aligned} t_{\text{flight}} &= |-t_1| + t_2 \\ t_{\text{flight}} &= t_1 + t_2 \,. \end{aligned} \tag{B.2}$$

B.1.3 Upward, Outbound to Inbound

To find the time from t_1 to $-t_2$ we start with the full orbital period and subtract the time from periapsis to planet 1, t_1, and the time from planet 2 to periapsis, $|-t_2| = t_2$:

$$\begin{aligned} t_{\text{flight}} &= P - t_1 - |-t_2| \\ t_{\text{flight}} &= P - t_1 - t_2 \,. \end{aligned} \tag{B.3}$$

B.1.4 Upward, Inbound to Inbound

To find the time from $-t_1$ through periapsis to $-t_2$ we subtract the (downward) travel time from $-t_2$ to $-t_1$ from the full orbital period:

$$\begin{aligned} t_{\text{flight}} &= P - (-t_1 - (-t_2)) \\ t_{\text{flight}} &= P + t_1 - t_2 \,. \end{aligned} \tag{B.4}$$

B.1.5 Downward, Inbound to Inbound

The transfer from $-t_2$ to $-t_1$ is simply

$$\begin{aligned} t_{\text{flight}} &= -t_1 - (-t_2) \\ t_{\text{flight}} &= t_2 - t_1 \,. \end{aligned} \tag{B.5}$$

B.1.6 Downward, Inbound to Outbound

The transfer from $-t_2$ through periapsis to t_1 is

$$\begin{aligned} t_{\text{flight}} &= |-t_2| + t_1 \\ t_{\text{flight}} &= t_1 + t_2 \,. \end{aligned} \tag{B.6}$$

B.1.7 Downward, Outbound to Inbound

To find the time from t_2 to $-t_1$ we start with the full orbital period and subtract the time from periapsis to planet 2, t_2, and the time from planet 1 to periapsis, $|-t_1| = t_1$:

$$\begin{aligned} t_{\text{flight}} &= P - |-t_1| - t_2 \\ t_{\text{flight}} &= P - t_1 - t_2 \,. \end{aligned} \tag{B.7}$$

B.1.8 Downward, Outbound to Outbound

To find the time from t_2 through periapsis to t_1 we subtract the (upward) travel time from t_1 to t_2 from the full orbital period:

$$\begin{aligned} t_{\text{flight}} &= P - (t_2 - t_1) \\ t_{\text{flight}} &= P + t_1 - t_2 \,. \end{aligned} \tag{B.8}$$

B.2 Delta-V Estimate for Linked Lambert Problem Solutions

When creating patched-conic trajectories from a set of linked interplanetary Lambert problem arcs, the magnitude of the inbound and outbound $\mathbf{V}_\infty$ at the intermediate gravity-assist bodies are, in general, not equal. A propulsive ΔV will be required make up any change in V_∞ magnitude and any change in direction in excess of the "free" rotation provided by gravity assist.

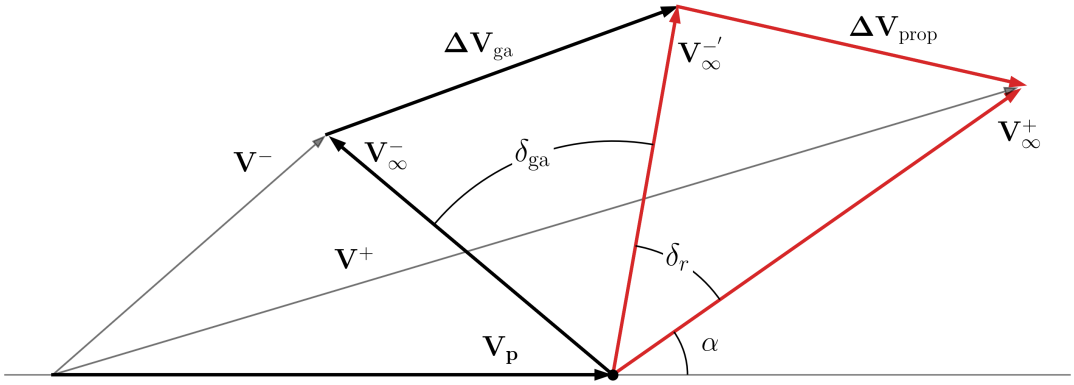


Figure B.4. A schematic of a flyby including a propulsive change in V_∞. The maximum bending angle provided through gravity assist is less than the desired bending. $\mathbf{V}_\infty^-$ may be rotated as far as $\mathbf{V}_\infty^{-\prime}$. To complete the bending and any magnitude change a propulsive ΔV is required. The magnitude of the ΔV is found from the red triangle.

The angle between the $\mathbf{V}_\infty$ vectors is found from their dot product:

$$\cos\delta = \frac{\mathbf{V}_\infty^- \cdot \mathbf{V}_\infty^+}{|\mathbf{V}_\infty^-||\mathbf{V}_\infty^+|}\,. \tag{B.9}$$

The maximum bending angle achievable through gravity assist alone is found from Equation 2.18 to be

$$\delta_{\mathrm{ga}} = 2\arcsin\frac{\mu}{\mu + r_{\min}\left(V_\infty^-\right)^2}\,, \tag{B.10}$$

where $r_{\min}$ is the minimum allowable flyby radius for the planet. If $\delta_{ga} \geq \delta$ then the trajectory bending can be accomplished via gravity assist and only the difference in magnitude needs to be made up propulsively, so

$$\Delta V_{\text{prop}} = \left| V_{\infty}^{+} - V_{\infty}^{-} \right| . \tag{B.11}$$

If $\delta_{\text{ga}} < \delta$ then there is some residual angle, δ_r that must be made up propulsively:

$$\delta_r = \delta - \delta_{\text{ga}} . \tag{B.12}$$

The situation is shown schematically in Figure B.4. The gravity assist is capable of rotating $\mathbf{V}_{\infty}^{-}$ to $\mathbf{V}_{\infty}^{-'}$. The magnitude of the required propulsive ΔV can be found from the Law of Cosines on the red triangle in Figure B.4:

$$\Delta V_{\text{prop}} = \sqrt{\left(V_{\infty}^{-}\right)^2 + \left(V_{\infty}^{+}\right)^2 - 2 V_{\infty}^{-} V_{\infty}^{+} \cos\left(\delta_r\right)} . \tag{B.13}$$

C. DETAILED HISTORICAL MISSION SEARCHES

This appendix includes a more detailed look at some of the historical mission searches summarized in Chapter 6. The Voyager 1, Galileo, and Cassini missions are discussed in more detail. The full analysis of Voyager 2 is included in Chapter 6.

C.1 Voyager 1 Search

The Voyager 1 mission included flybys of Jupiter and Saturn [72], [73]. The mission provides a fairly simple gravity assist example with known parameters. We will not attempt to model any trajectory correction maneuvers that occurred on the actual mission.

C.1.1 Voyager 1 Search Parameters

Table C.1 summarizes the key inputs used to construct a network for the Voyager 1 problem and search for trajectories. The V_∞ notation 7:1:12 $\mathrm{km\,s^{-1}}$ means the Tisserand network includes V_∞ levels from 7 up to and including 12 $\mathrm{km\,s^{-1}}$ in 1 $\mathrm{km\,s^{-1}}$ increments. The

Table C.1. Voyager 1 Search Parameters

Parameter	**Value**
Earth V_∞	7:1:12 $\mathrm{km\,s^{-1}}$
Jupiter V_∞	7:1:12 $\mathrm{km\,s^{-1}}$
Saturn V_∞	9:1:16 $\mathrm{km\,s^{-1}}$
Alignment Start Date	Jan. 1, 1977
Alignment End Date	Dec. 31, 1980
Max Flyby Count	2
Max Repeat Visits	0
Max Time of Flight	4 y
Date Tolerance	10% time of flight

dates provided give boundaries on the alignment dates of the planets (described in Chapter 3). The actual departure or arrival dates may fall outside this window.

C.1.2 Voyager 1 Actual Trajectory

Voyager 1 launched September 5, 1977 on a Titan IIIE-Centaur. The departure ΔV for the Earth to Jupiter leg was provided by the Centaur upper stage and Star 37E solid booster [72]. For comparison with the Tisserand network results, Table C.2 provides a patched-conic reconstruction of the Voyager 1 encounters based on the published flyby dates.

Table C.2. Voyager 1 Encounters

Encounter	**Planet**	$\mathbf{V_\infty}$ $(\mathrm{km\,s^{-1}})$	**Date**
Launch	Earth	10.3	Sep. 5, 1977
1	Jupiter	10.9	Mar. 5, 1979
2	Saturn	15.3	Nov. 12, 1980

C.1.3 Voyager 1 Results

Figure C.1 shows the Tisserand graph created for the Voyager 1 problem. This Tisserand graph is the basis for the Tisserand network in Figure C.2a. The Earth departures at V_∞ of 7, 8, and 9 $(\mathrm{km\,s^{-1}})$ do not intersect the contours of the other planets in the Tisserand graph. This fact is mirrored by the isolated E7, E8 and E9 vertices in the Tisserand network. The grid view of the Tisserand network makes explicit the energy connections between flybys of each planet. This view, however, does not clearly show differences in timing between the arrival and departure from any given planet.

Figure C.3 displays the Tisserand network in a polar view of the two-point orbital arcs. The concentric circles outline the orbits of Earth, Jupiter, and Saturn, respectively. Empty markers on these orbits identify a departure from that planet and filled markers identify an arrival. This view shows, more clearly, the timing problem to be solved. The arrival at Jupiter needs to coincide with the departure for Saturn. The network is filtered to remove arrival/departure time mismatches outside of the desired tolerance. The search algorithm will find arrivals and departures in the filtered network that match in V_∞.

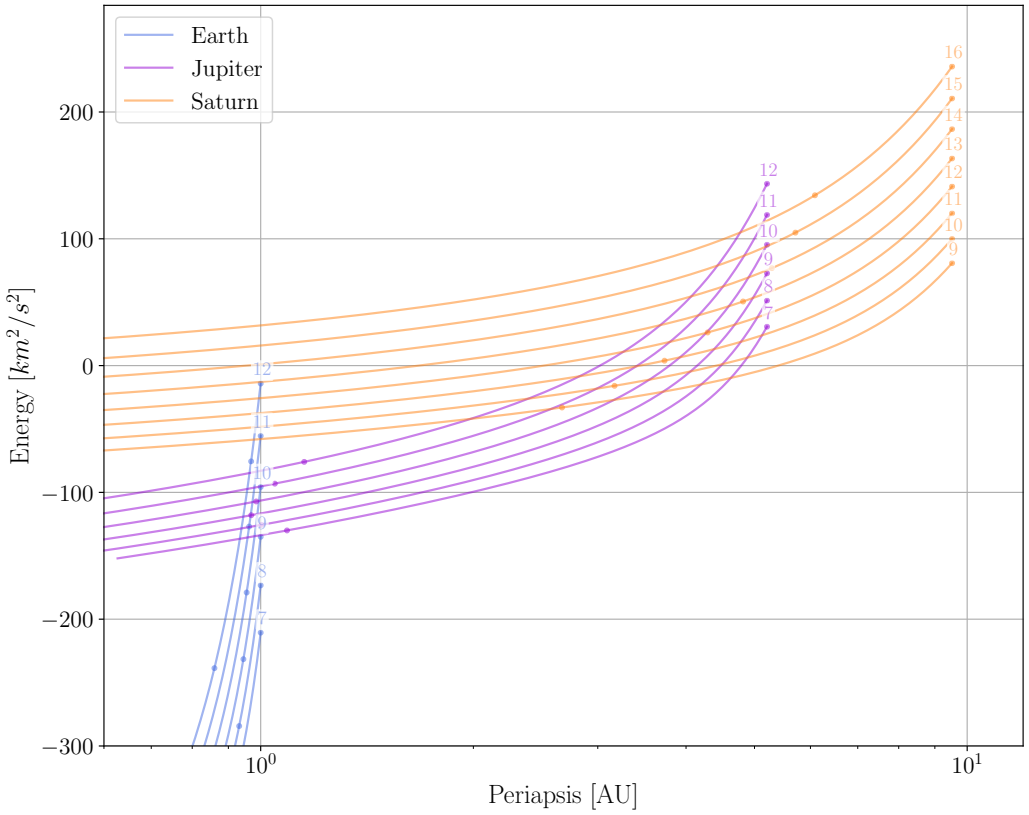
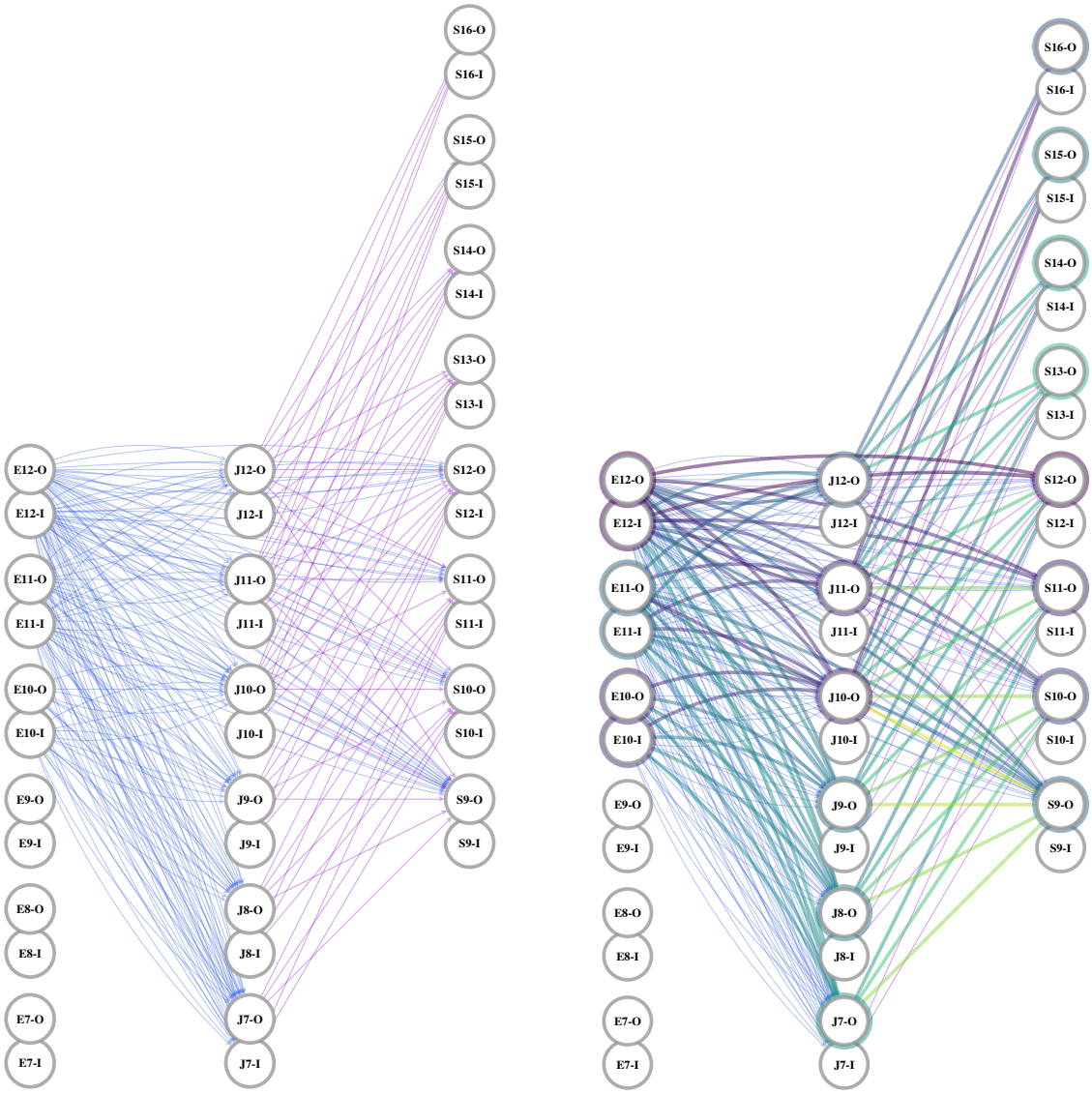


Figure C.1. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Voyager 1 search.



(a) Search Network. (b) Network solutions highlighted.

Figure C.2. The grid view of the Tisserand network assists in visualizing the energy connections that will be searched to find a possible Voyager 1 path. Multiple lines connecting two vertices indicate more than one date on which the connection exists.

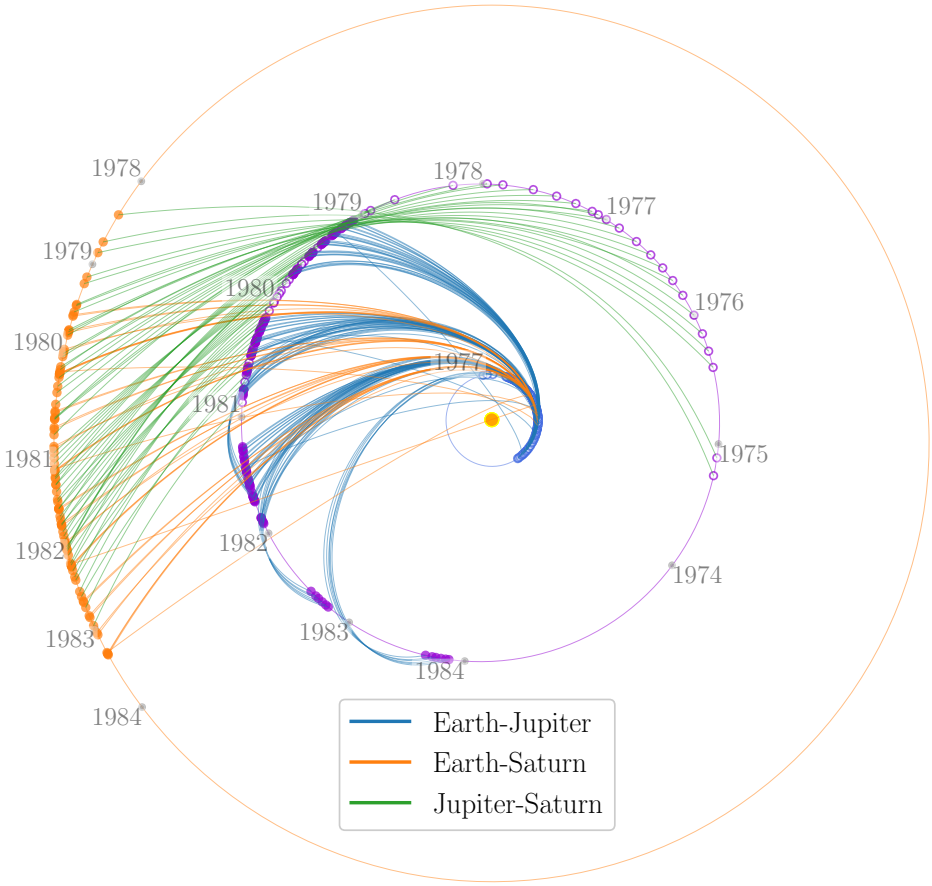


Figure C.3. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The circular orbits represent the orbits of Earth, Jupiter, and Saturn, respectively. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year.

Table C.3 summarizes the results of the Voyager 1 search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory. Recall from Chapter 3, a *route* differentiates the particular vertices in the Tisserand network that are passed through along a path (e.g. E10-O, J7-O, S9-O). A *date variant* identifies a duplicate of a route with some difference in the encounter dates. The *Network Routes* and *Date Variants* columns list the number of these occurrences for each path.

The *Launch* V_∞, *Total* ΔV, *Launch Window* and *Arrival Window* columns summarize the extremes of the randomized patched-conic trajectories. The total ΔV is the cumulative propulsive ΔV required to complete gravity-assist turning at any of the encounters after Earth departure. The date columns give the range of departure or arrival dates in month/year format.

Table C.3. Voyager 1 Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
JS	178	341	4 - 15	0 - 5	06/77 - 12/79	09/79 - 12/83
S	10	40	11 - 15	—	08/76 - 12/79	09/78 - 04/84

Figure C.4 shows the Tisserand network search solutions. This includes Earth-Jupiter-Saturn (JS) paths and direct Earth-Saturn (S) paths. Within each path are multiple routes. For example, the connected vertices E10-I,J7-O,S9-O and E10-O,J7-O,S10-O are both JS paths but they represent different routes through the network vertices. The various routes are visible in Figure C.2b which shows the search results in the Tisserand network grid view.

Because of the discrete composition of the network and the tolerance in encounter dates, the search results include gaps in time and only represent outlines of closed trajectories. Section 3.6 discusses several methods of filling in these outlines. Here, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure C.5 shows the resulting patched conic trajectories.

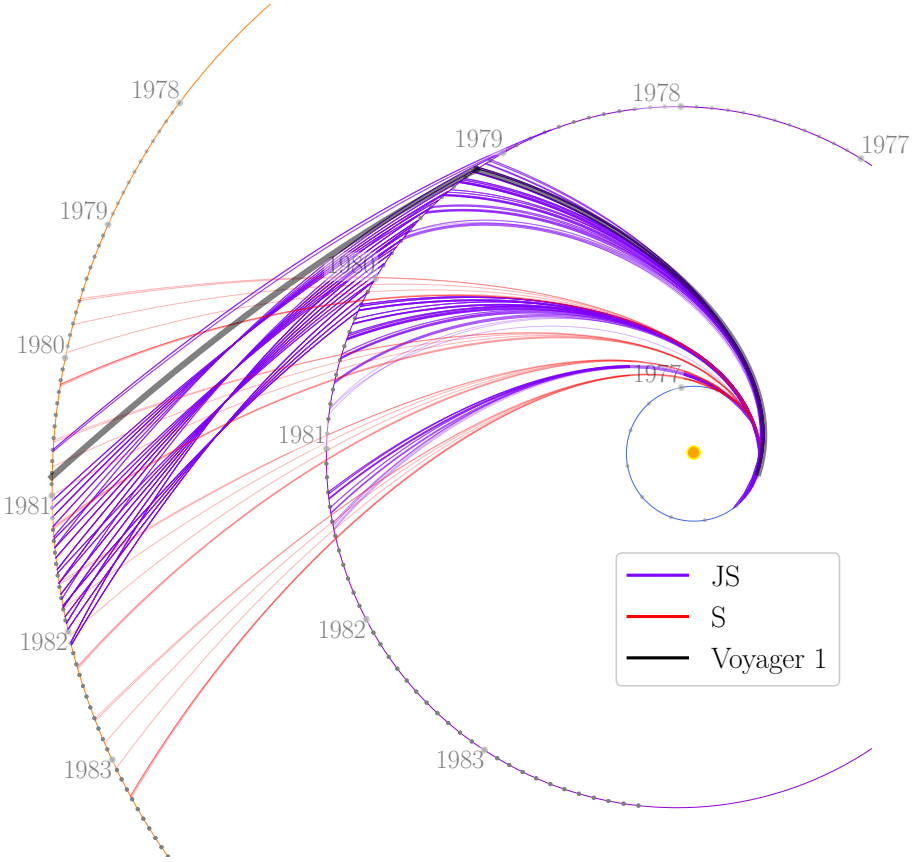


Figure C.4. The search found two paths (JS and S) that contain 161 routes. The actual Voyager 1 trajectory is shown in the thick gray line.

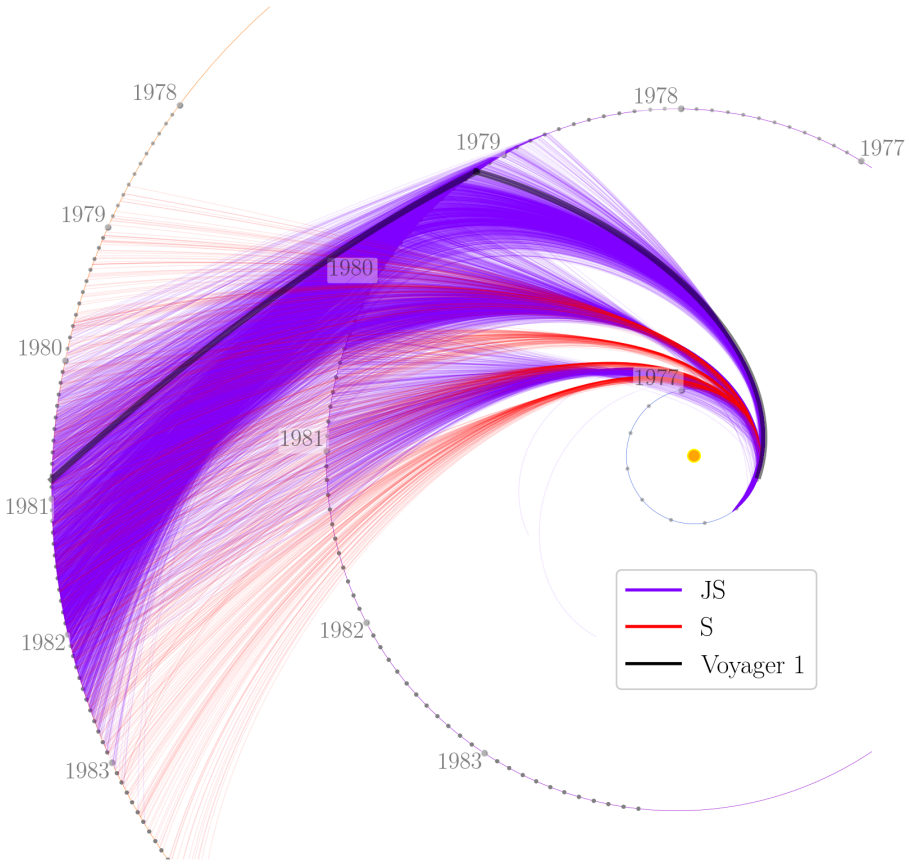


Figure C.5. The Tisserand network solutions provide the outline for a collection of patched conic trajectories shown here. A patched conic approximation of the true Voyager 1 trajectory is highlighted.

We can evaluate the patched conic trajectories relative to one another by comparing key characteristics. Figure C.6 plots the time of flight against the total mission ΔV for each of the patched conic trajectories generated from the Tisserand network search results. The color of each point corresponds to the launch V_∞. For clarity, trajectories that required more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV were filtered from the solution set. The approximated Voyager 1 trajectory appears as the diamond. In Figure C.6 attractive trajectories will fall in the lower-left corner and will be on the cool end of the spectrum.

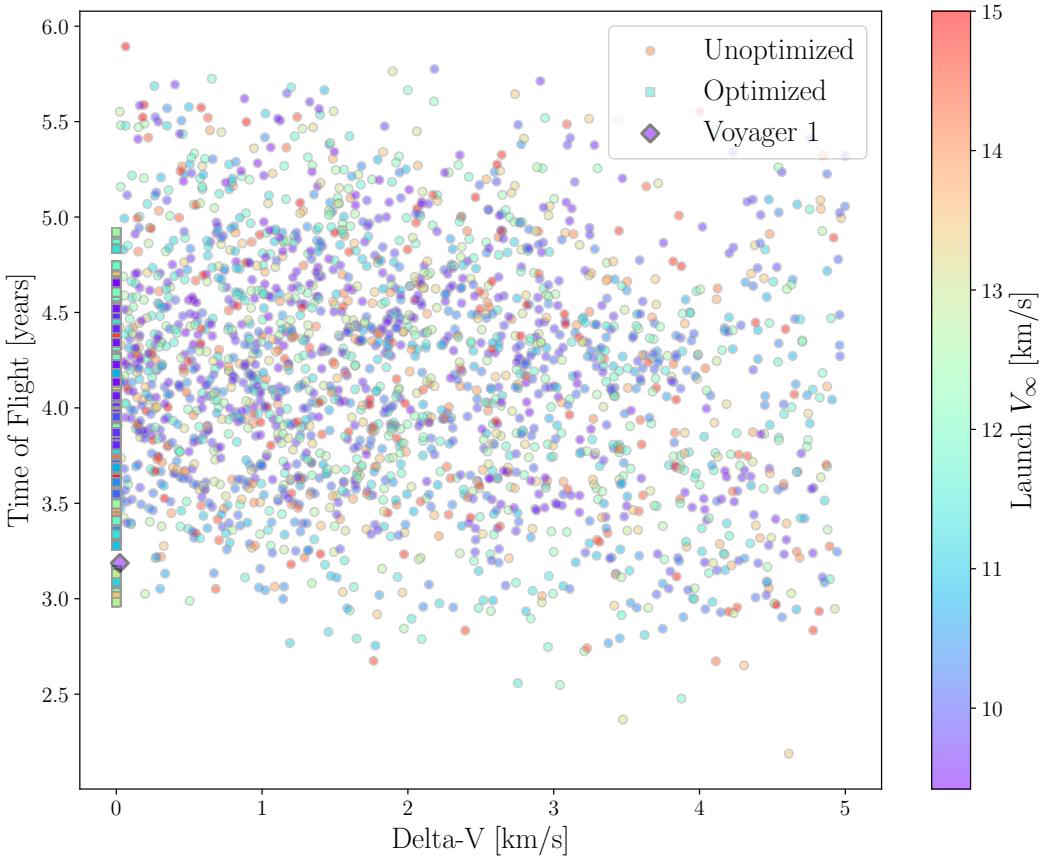


Figure C.6. The time of flight is plotted against the mission ΔV for each of the patched conic variations on the network solution. The shade of the points corresponds to the total mission launch V_∞. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown. The actual Voyager 1 trajectory is shown as the diamond.

Another practical parameter to consider is the launch date. Figure C.7 plots the launch V_∞ against the launch date for each trajectory. The color of each point corresponds to the

total mission ΔV. The data show that selecting the right launch date has a pronounced effect on the required launch V_∞.

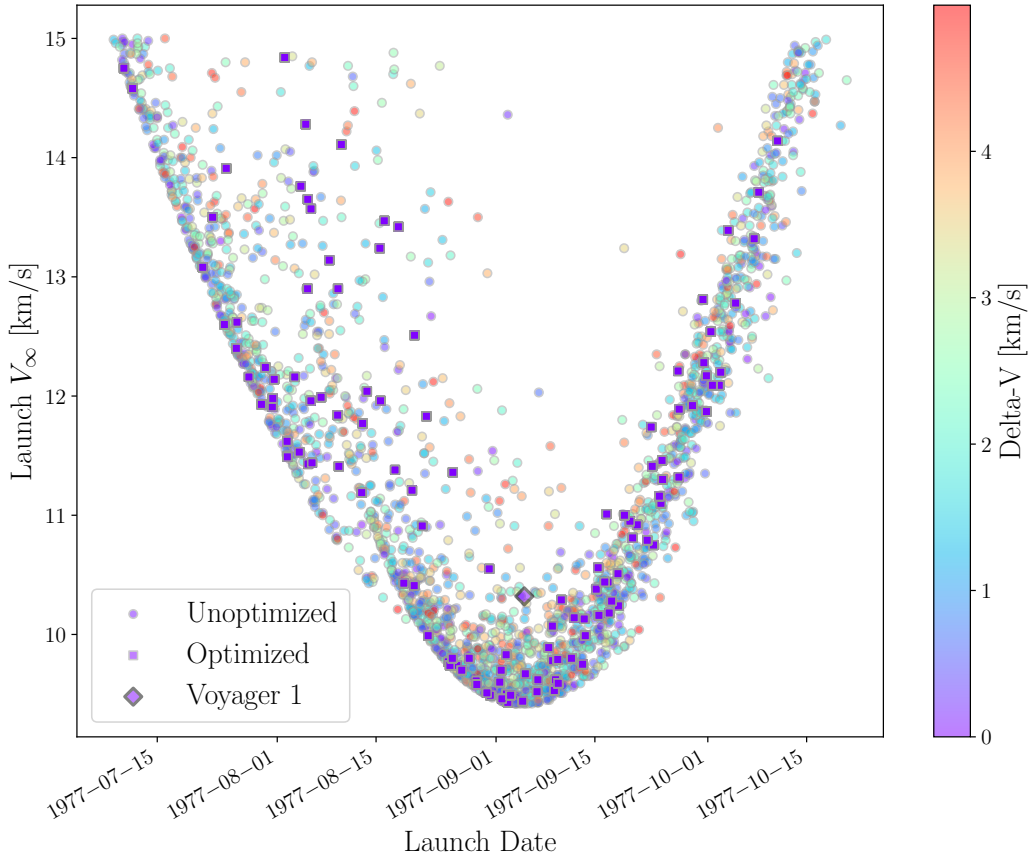


Figure C.7. The Launch V_∞ is plotted against the Launch Date for each of the patched conic variations on the network solution. The shade of the points corresponds to the total mission delta-V. Trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 5 $\mathrm{km\,s^{-1}}$ ΔV are not shown. The actual Voyager 1 trajectory is shown as the diamond.

Absent other mission constraints, a trajectory designer would likely choose a trajectory with a low launch V_∞, a low mission ΔV, and a short time of flight. We see that the actual Voyager 1 mission performs well in these criteria relative to the field of possible trajectories.

C.2 Galileo Search

The Galileo mission to Jupiter provides an opportunity to evaluate the Tisserand network in a trajectory search with a resonant transfer. Galileo performed two Earth flybys separated by exactly two years as part of the Venus Earth Earth Gravity Assist (VEEGA) sequence. The Galileo mission included an atmospheric entry probe and a tour of the Jovian satellites. Here we will examine only the gravity assist journey from Earth to Jupiter and will neglect several trajectory shaping maneuvers.

C.2.1 Galileo Search Parameters

Table C.4 summarizes the inputs used to build the Galileo search network. The *Max Repeat Visits* parameter is set to 2 to allow the repeat flybys of Earth in the VEEGA. To make the results more presentable, we add some additional constraints that the Venus encounters must occur in the years 1989 through 1991 and the Earth launch and encounters must occur in the years 1988 through 1994. If we were looking for a new mission, we might not add these constraints. But for this reconstruction, these constraints will limit the solutions to those more closely resembling the Galileo mission.

Table C.4. Galileo Search Parameters

Parameter	**Value**
Venus V_∞	4:1:8 $\mathrm{km\,s^{-1}}$
Earth V_∞	3:1:10 $\mathrm{km\,s^{-1}}$
Jupiter V_∞	5:1:8 $\mathrm{km\,s^{-1}}$
Start Date	Jan. 1, 1988
End Date	Dec. 31, 1996
Max Flyby Count	6
Max Repeat Visits	2
Max Time of Flight	9 y
Date Tolerance	5% time of flight
Venus Encounter Window	1989 - 1991
Earth Encounter Window	1988 - 1994

C.2.2 Galileo Actual Trajectory

Galileo launched aboard Space Shuttle Atlantis on October 18, 1989. The original mission design included a direct trajectory to Jupiter made possible by a Shuttle-Centaur upper stage. The Shuttle-Centaur was a version of the Centaur upper stage designed to be carried to orbit in the Space Shuttle payload bay. In response to the Challenger disaster, the Shuttle-Centaur program was cancelled over safety concerns with the Centaur's liquid-hydrogen fuel. The Galileo mission was redesigned with a less-capable, solid-fueled inertial upper stage. The decrease in launch C_3 capability was the impetus for the VEEGA trajectory [74].

Partial information on the Galileo encounters can be found in D'Amario *et al.* [75] and is tabulated in Table C.5. The V_∞^{pc} values are based on a patched-conic gravity-assist sequence reconstructed from the published encounter dates. The patched-conic values agree well with the published data.

Table C.5. Galileo Encounters

Encounter	**Planet**	V_∞ $(\text{km}\,\text{s}^{-1})$	$V_\infty^{\mathbf{pc}}$ $(\text{km}\,\text{s}^{-1})$	**Date**
Launch	Earth	3.1	3.9	Oct. 18, 1989
1	Venus	6.2	6.1	Feb. 10, 1990
2	Earth	8.9	8.8	Dec. 8, 1990
3	Earth	8.9	8.9	Dec. 8, 1992
4	Jupiter	—	5.6	Dec. 7, 1995

C.2.3 Galileo Results

Figure C.8 shows the Tisserand graph created for the Galileo problem. The contours for the outer solar system are typically compressed and we can see this effect for Jupiter in Figure C.8. This difference in scales is an example of the difficulties with manual pathfinding in the Tisserand graph. We can make an immediate observation that an Earth V_∞ of roughly 9 $\text{km}\,\text{s}^{-1}$ will be required to reach Jupiter (see inset). The contours in Figure C.8 are the basis for the Tisserand network in Figure C.9.

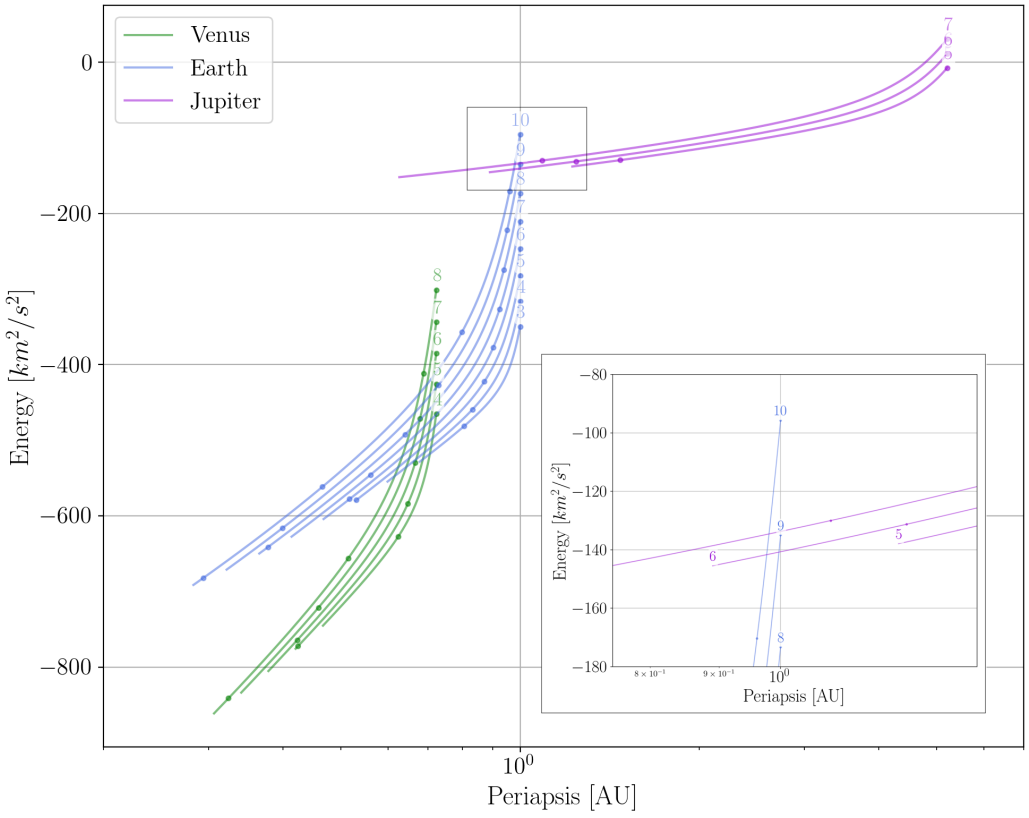
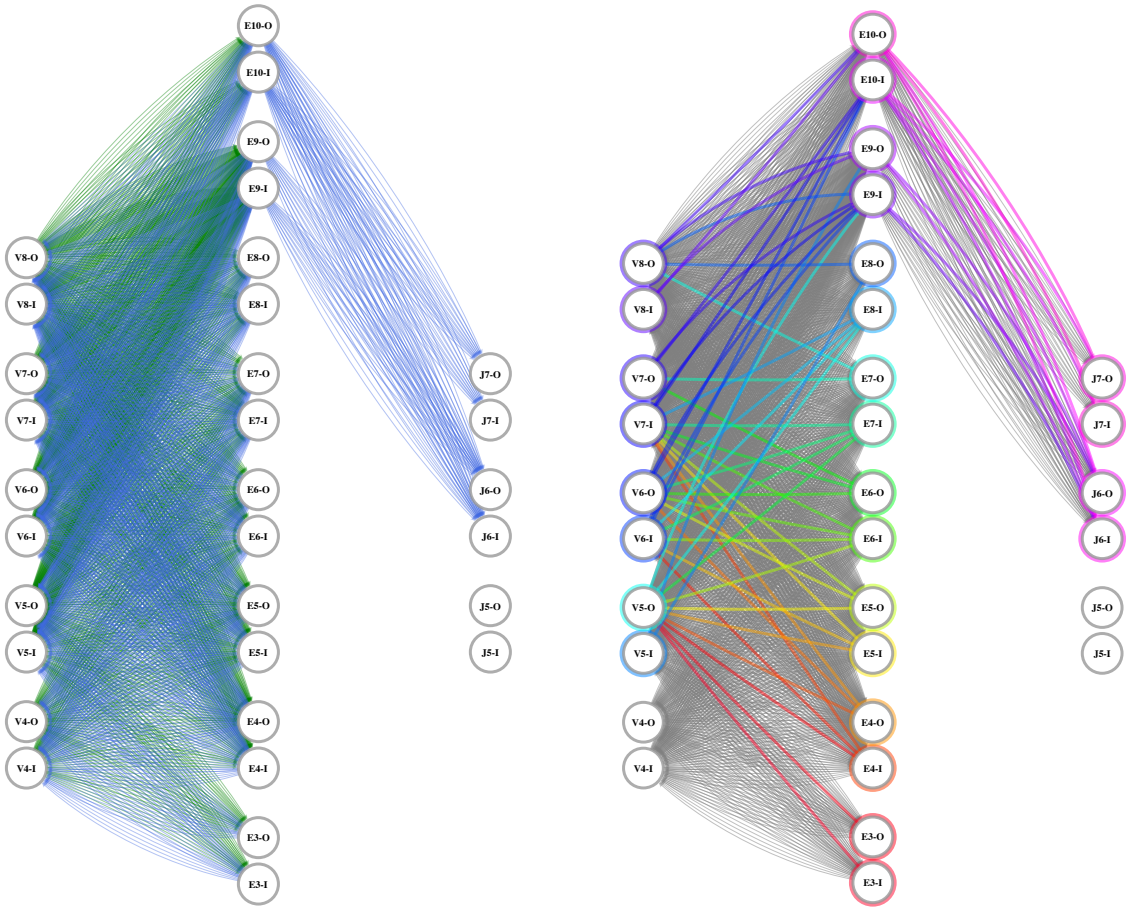


Figure C.8. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Galileo search.

The grid view of the Tisserand network (Figure C.9) visualizes the energy connections between flybys of each planet. This view, however, does not clearly show differences in timing between the arrival and departure from any given planet.

Figure C.10 displays the Tisserand network in a polar view of the two-point orbital arcs. The concentric circles outline the orbits of Earth and Jupiter. Figure C.11 shows a similar view focusing on Earth and Venus. Empty markers on these orbits identify a departure from that planet and filled markers identify an arrival. This view shows, more clearly, the timing problem to be solved. The network is filtered to remove arrival/departure time mismatches outside of the desired tolerance. The search algorithm will find arrivals and departures in the filtered network that match in V_∞.



(a) Search Network.

(b) Network solutions highlighted.

Figure C.9. The grid view of the Tisserand network assists in visualizing the energy connections that will be searched to find a possible Galileo path. Multiple lines connecting two vertices indicate more than one date on which the connection exists.

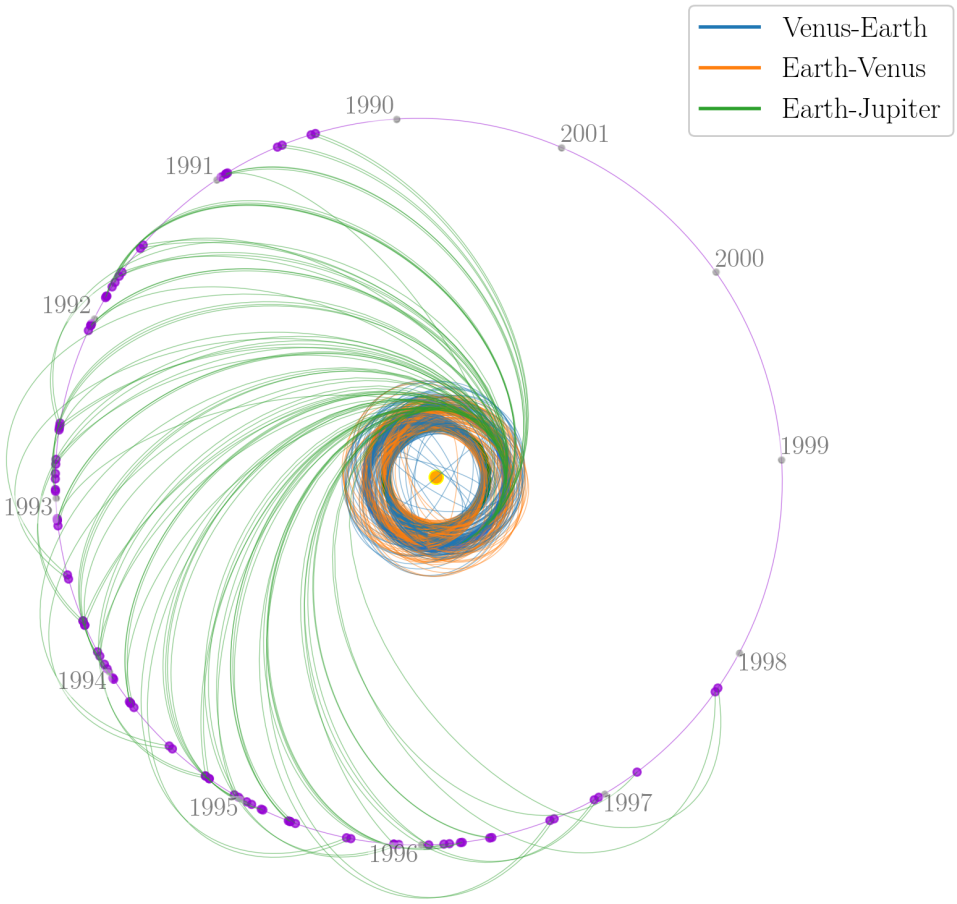


Figure C.10. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible Galileo path. The outer circle represents Jupiter's orbit. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year.

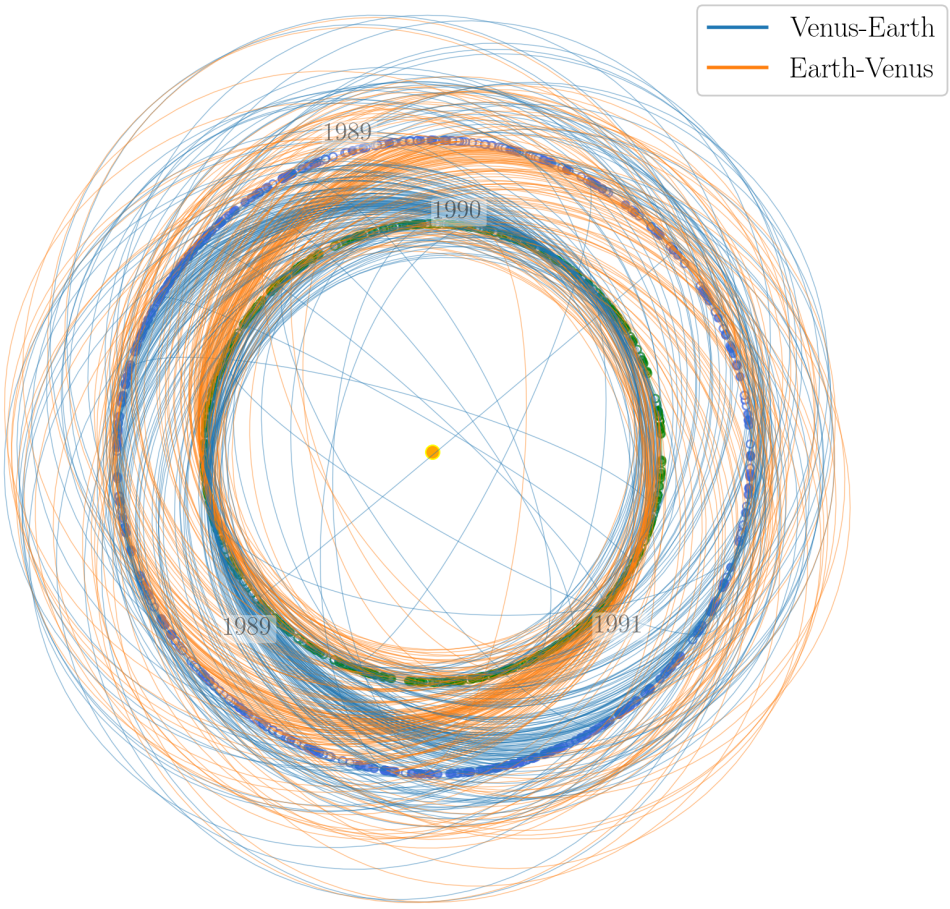


Figure C.11. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The concentric circles represent the orbits of Venus and Earth. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year. In this search, there are many possible transfers between Earth and Venus over the course of a few years.

Table C.6 summarizes the results of the Galileo search. The *Path* column lists the sequence of planets encountered on the gravity-assist trajectory.

Table C.6. Galileo Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
VEEJ	56	121	3 - 15	2 - 10	12/88 - 02/90	02/95 - 06/96
J	16	85	9 - 15	0 - 0	06/88 - 01/94	04/90 - 11/97

Figure C.12 shows the Tisserand network search solutions. The search discovered two paths: VEEJ and J. Within each path are multiple routes. Overall the search found about 70 distinct routes. Including date variations on those routes, the total number of network paths was 206. The various routes are visible in Figure C.9b which shows the search results in the Tisserand network grid view.

As with the other missions, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure C.13 shows the patched conic trajectories along the VEEJ and J paths. The launch V_∞ and propulsive ΔV (if any) required for each of the patched-conic can be computed from the Lambert solutions as described in Chapter 3. Patched-conic trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 10 $\mathrm{km\,s^{-1}}$ propulsive ΔV have been excluded from the summary in Table C.6.

The Tisserand network was successful at finding Galileo-like paths (VEEJ) including some patched-conic trajectories with similar launch and arrival dates as seen in Figure C.13.

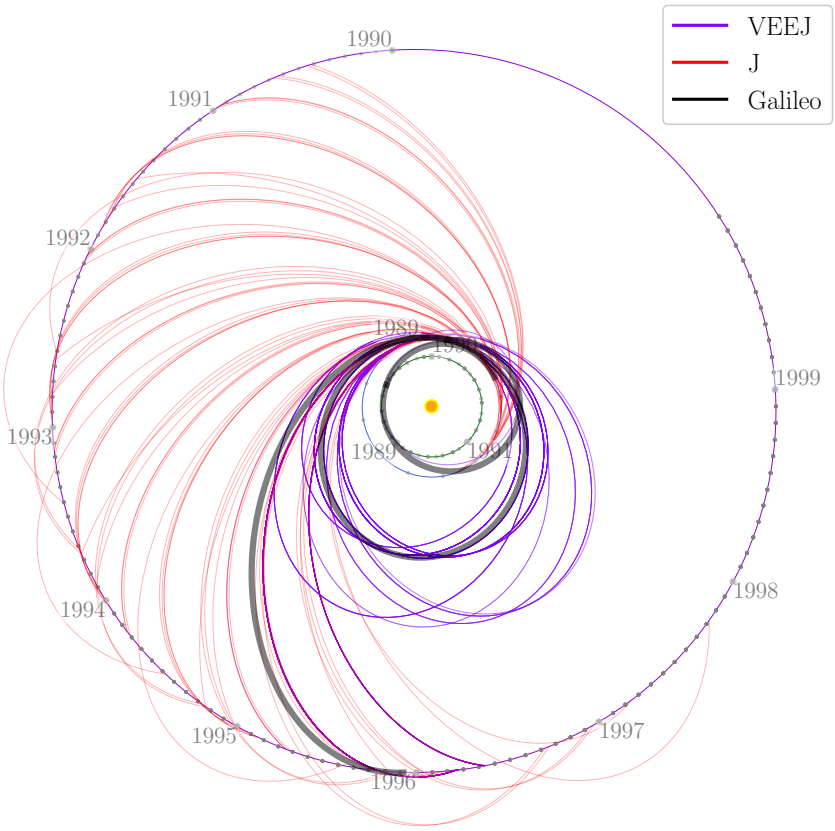


Figure C.12. The search found two paths (VEEJ and J) that contain 72 routes with approximately 200 date variants. The discontinuous network routes are plotted for the VEEJ and J paths. The actual Galileo trajectory is shown in the thick gray line.

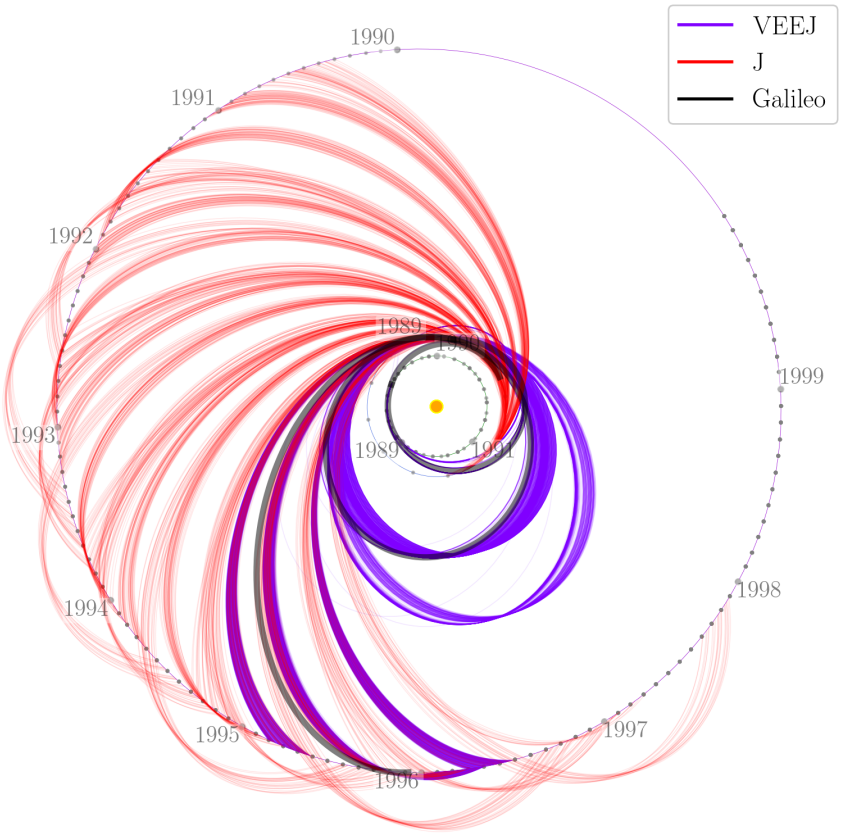


Figure C.13. The Tisserand network solutions provide the outline for a collection of randomized patched conic trajectories shown here in a polar view of three-dimensional trajectories. The spacecraft trajectories are constructed from Lambert solutions on a 3-D ephemeris. A patched conic approximation of the true Galileo trajectory is highlighted.

C.3 Cassini Search

The Cassini mission to Saturn provides a comparison case with a V-Infinity Leveraging Transfer (VILT). We will only attempt to find the trajectory to Saturn and will not recreate the tour of Saturn's satellites.

C.3.1 Cassini Search Parameters

Table C.7 summarizes the inputs used to build the Cassini search network. The network is configured to scan for Venus VILT opportunities starting with an V_∞ of 6 $\mathrm{km\,s^{-1}}$.

Table C.7. Cassini Search Parameters

Parameter	**Value**
Venus V_∞	5, 6, 9, 10, 11 $\mathrm{km\,s^{-1}}$
Earth V_∞	4.5, 5, 14, 15, 16, 17 $\mathrm{km\,s^{-1}}$
Jupiter V_∞	9:1:11 $\mathrm{km\,s^{-1}}$
Saturn V_∞	5:1:7 $\mathrm{km\,s^{-1}}$
Start Date	Jan. 1, 1997
End Date	Dec. 31, 2010
Max Flyby Count	6
Max Repeat Visits	2
Max Time of Flight	9 y
Date Tolerance	20% time of flight
VILT at Venus	6 $\mathrm{km\,s^{-1}}$

C.3.2 Cassini Actual Trajectory

Cassini/Huygens was launched on October 15, 1997 on a Titan IVB-Centaur with two solid rocket motor upgrades. The Centaur upper stage provided the ΔV for the departure to Venus [76]. Partial information on the Cassini encounters is available in Goodson *et al.* [65]. Those values are tabulated in Table C.8. The V_∞^{pc} values are based on a patched conic reconstruction of the gravity assist sequence using the published encounter dates. The patched-conic values agree well with the published data.

Table C.8. Cassini Encounters

Encounter	**Planet**	V_∞ (km s^{-1})	$V_\infty^{\mathbf{pc}}$ (km s^{-1})	**Date**
Launch	Earth	4.1	4.0	Oct. 15, 1997
1	Venus	6.0	5.9	Apr. 26, 1998
2	Venus	9.4	9.4	Jun. 24, 1999
3	Earth	16.0	16.0	Aug. 18, 1999
4	Jupiter	—	10.6	Dec. 30, 2000
5	Saturn	—	5.3	Jul. 1, 2004

C.3.3 Cassini Results

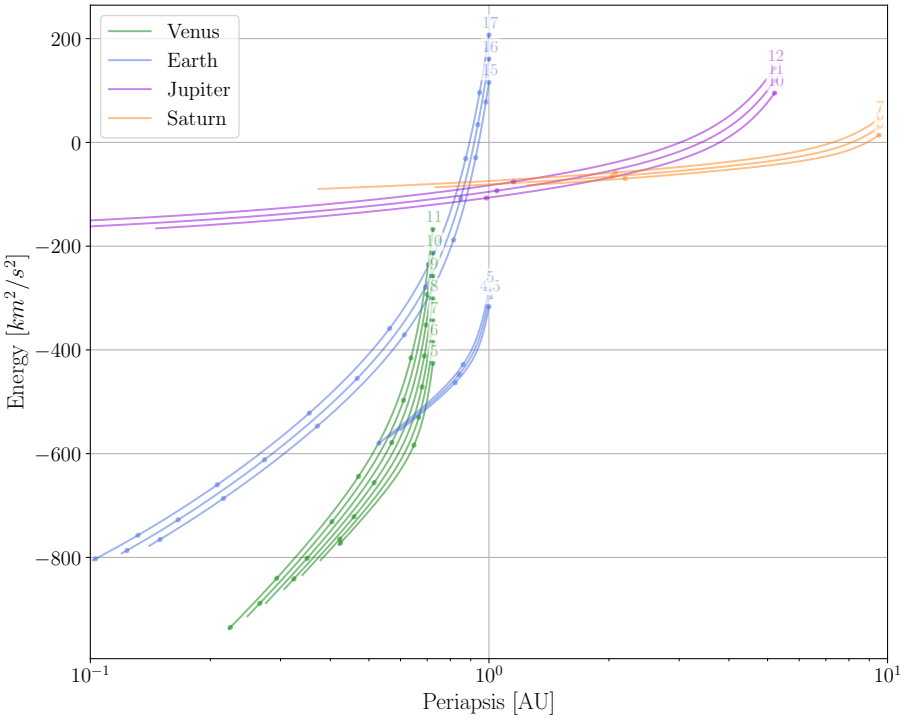


Figure C.14. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Cassini search.

Figures C.14 and C.15 show the Tisserand graph created for the Cassini problem. The contours for the outer solar system are compressed in Figure C.14 so Figure C.15 focuses on the Tisserand graph in the outer solar system. This difference in scales is an example of the

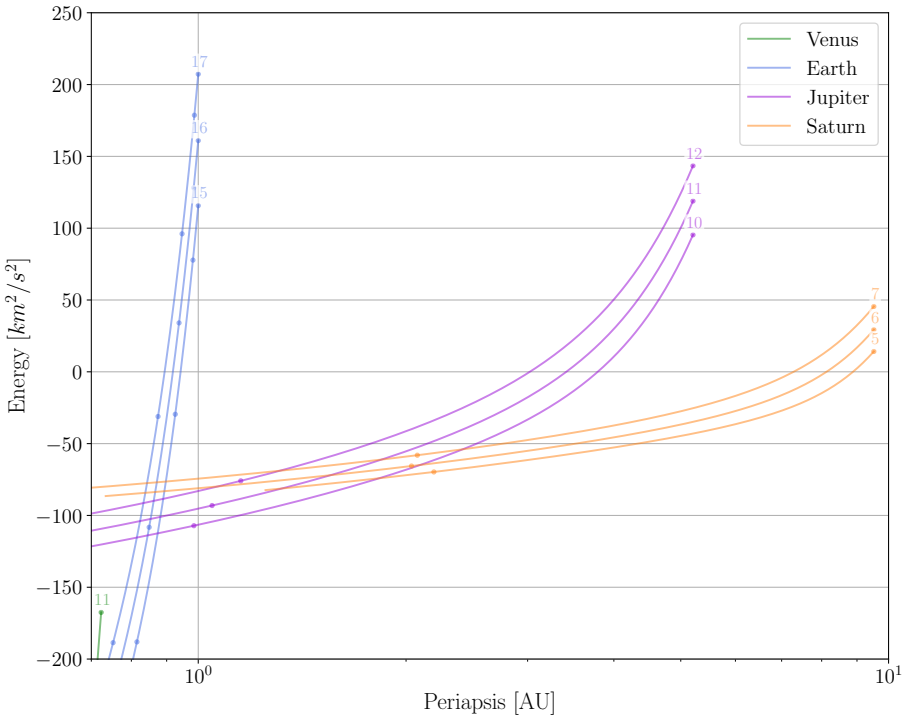
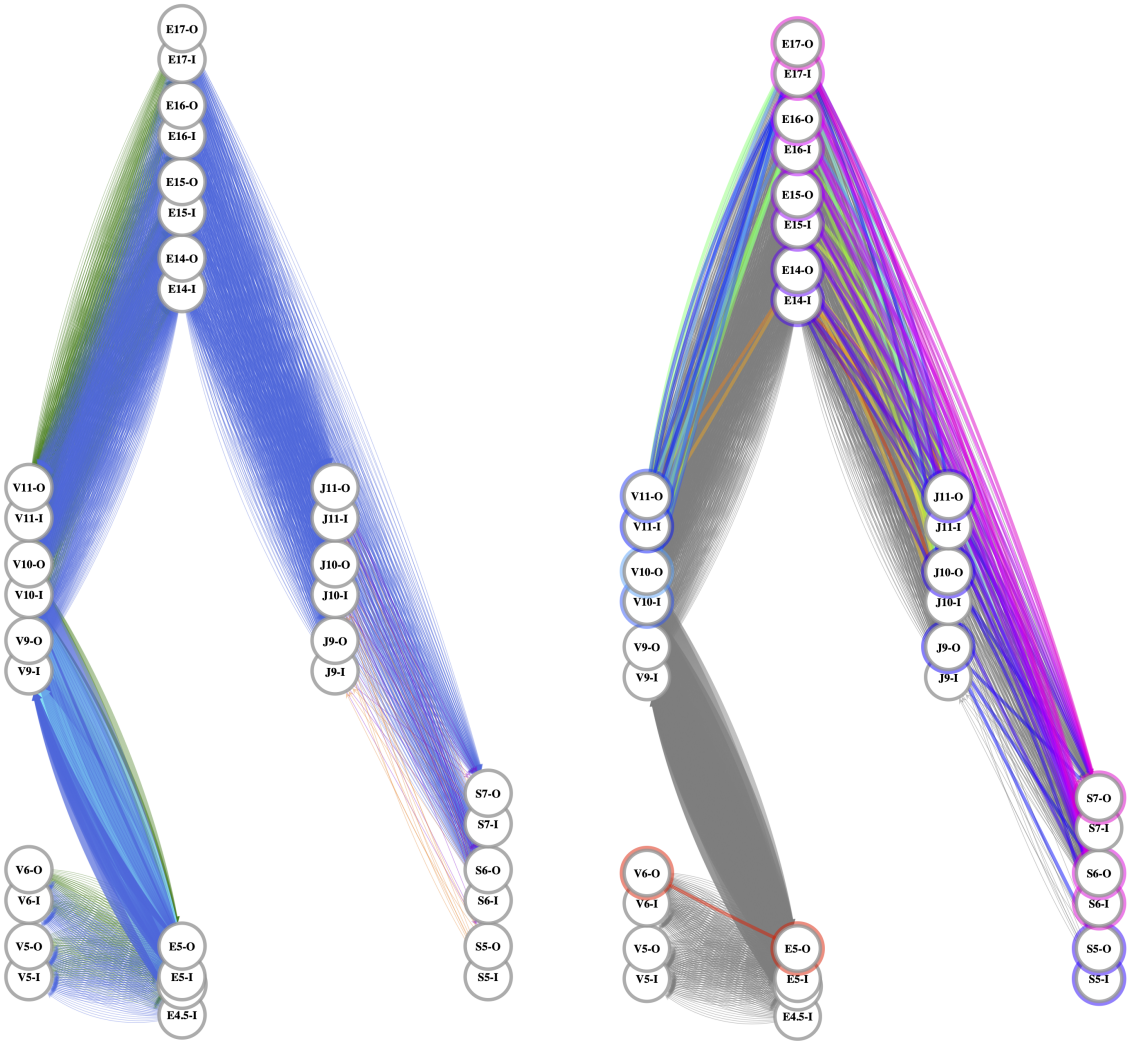


Figure C.15. The Tisserand graph displays the discrete V_∞ levels that will be considered in the Cassini search. This view focuses on the outer solar system.

difficulties with manual pathfinding in the Tisserand graph. These graphs are the basis for the Tisserand network in Figure C.16.

The grid view of the Tisserand network (Figure C.16a) visualizes the energy connections between flybys of each planet. This view, however, does not clearly show differences in timing between the arrival and departure from any given planet.

Figure C.17 displays the Tisserand network in a polar view of the two-point orbital arcs. The concentric circles outline the orbits of Jupiter, and Saturn, respectively. Figure C.18 provides a similar view of the inner solar system. Empty markers on these orbits identify a departure from that planet and filled markers identify an arrival. The network is filtered to remove arrival/departure time mismatches outside of the desired tolerance. The search algorithm will find arrivals and departures in the filtered network that match in V_∞.



(a) Search Network. (b) Network solutions highlighted.

Figure C.16. The grid view of the Tisserand network assists in visualizing the energy connections that will be searched to find a possible Cassini path. Multiple lines connecting two vertices indicate more than one date on which the connection exists.

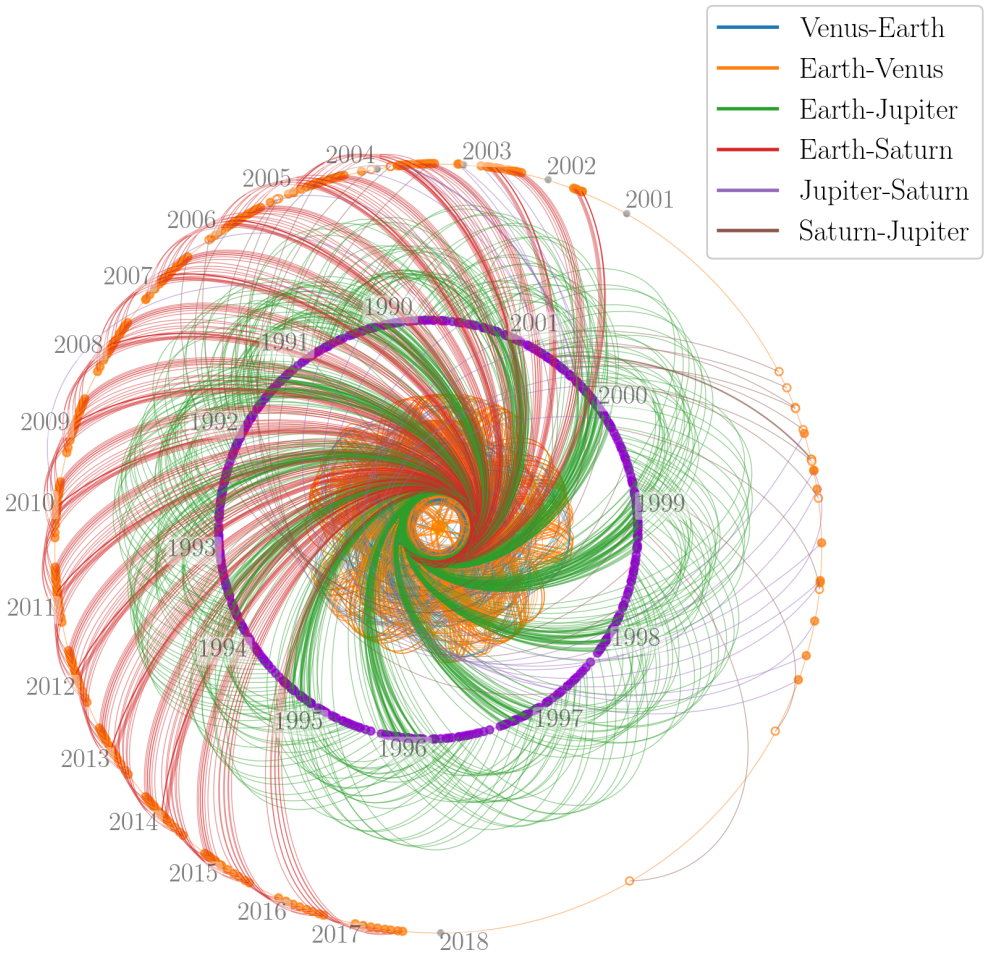


Figure C.17. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The outer circle represents Saturn's orbit. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year.

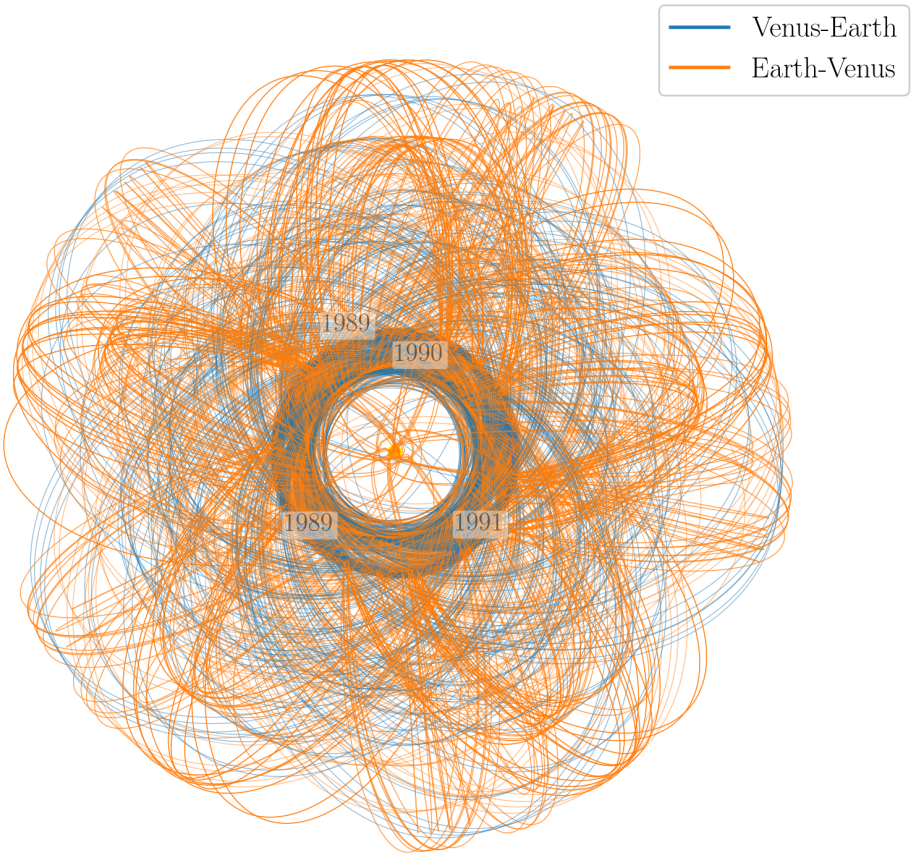


Figure C.18. The polar view of the Tisserand network assists in visualizing the time connections that will be searched to find a possible path. The concentric circles represent the orbits of Venus and Earth. Empty markers on the orbits represent a departure and filled markers represent an arrival. Date ticks represent January 1st of each year. In this search, there are many possible transfers between Earth and Venus over the course of a few years.

Table C.9 summarizes the results of the Cassini search. The *Path* column lists the sequence of planets encountered on the gravity assist trajectory.

Table C.9. Cassini Search Results Summary

Path	**Network Routes**	**Date Variants**	**Launch V_∞** ($\mathrm{km\,s^{-1}}$)	**Total ΔV** ($\mathrm{km\,s^{-1}}$)	**Launch Window (mm/yy)**	**Arrival Window (mm/yy)**
VVES	5	27	3 - 12	3 - 10	11/97 - 12/05	11/02 - 10/17
VVEJS	6	6	4 - 11	—	10/97 - 12/97	07/02 - 03/06
JS	42	42	10 - 15	0 - 5	02/98 - 10/00	11/01 - 10/10
VES	46	186	14 - 15	0 - 10	12/96 - 04/06	10/02 - 11/17
VEJS	44	72	14 - 15	0 - 10	12/96 - 05/98	06/02 - 04/06
S	18	234	10 - 15	0 - 0	03/97 - 02/10	08/00 - 04/18

As with the other missions, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure C.20 shows the patched conic trajectories along the various paths. The launch V_∞ and propulsive ΔV (if any) required for each of the patched-conic can be computed from the Lambert solutions as described in Chapter 3. Patched-conic trajectories requiring more than 15 $\mathrm{km\,s^{-1}}$ launch V_∞ or more than 10 $\mathrm{km\,s^{-1}}$ propulsive ΔV have been excluded from the summary in Table C.9.

Figure C.19 shows the Tisserand network search solutions. The search discovered six paths. These include the VVEJS path used by Cassini, in addition to VVES, JS, VES, VEJS, and S. Within each path are multiple routes. Overall the search found about 160 distinct routes. Including date variations on those routes, the total number of network paths was 567. The various routes are visible in Figure C.16b, which shows the search results in the Tisserand network grid view. The V6 to V10 VILT is not shown in this view.

The Tisserand network was able to find routes similar to the actual VVEJS Cassini path (for example, E5-I, V6-O, V10-I, E16-O, J11-O, S5-O). However, in cases with consecutive flybys of the same planet (such as the Venus VILT in the Cassini design), the ΔV results for the randomized patched-conic trajectories have proven to be sensitive to the actual randomly drawn dates. The particular random draws in the example tabulated here resulted in ΔV

above the 10 $\mathrm{km\,s^{-1}}$ limit. In these cases, a more sophisticated method for generating the patched-conic trajectories may be required. This improvement is included in the future work.

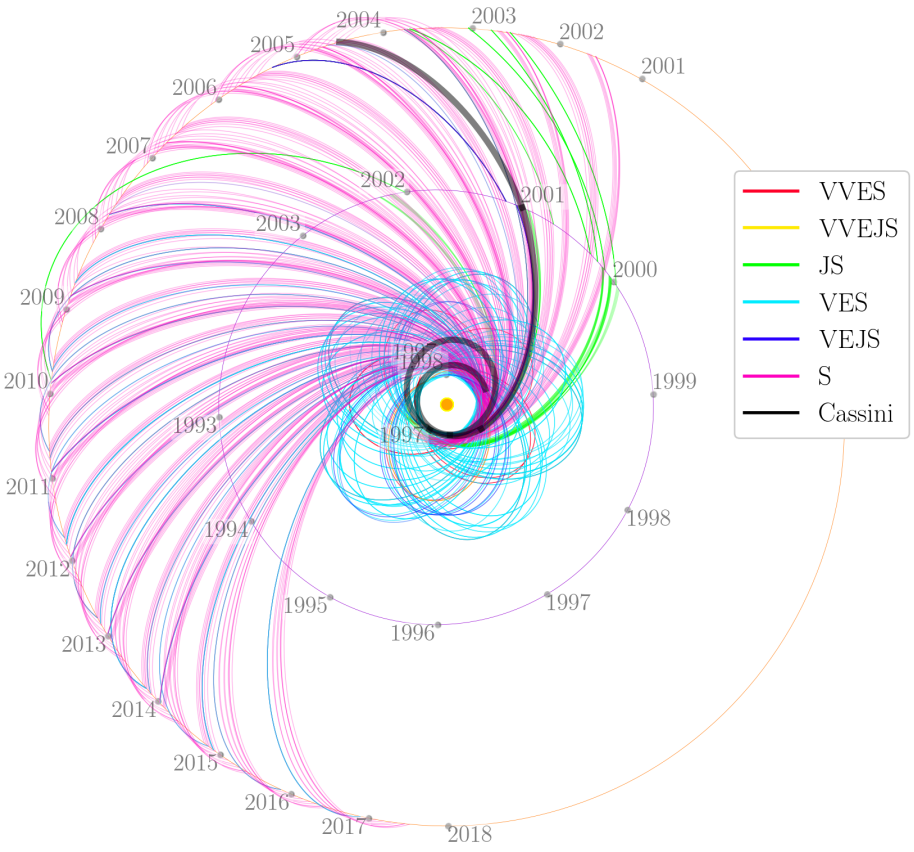


Figure C.19. The search found six paths (VVES, VVEJS, JS, VES, VEJS, and S) that contain 160 routes with approximately 570 date variants. The actual Cassini trajectory is shown in the thick gray line.

Because of the discrete composition of the network and the tolerance in encounter dates, the search results include gaps in time and only represent outlines of closed trajectories. These outlines can be filled in by several means as discussed in Section 3.6. Here, we select 20 random encounter date sequences within the date ranges identified by each Tisserand network solution. Figure C.20 shows the resulting patched conic trajectories.

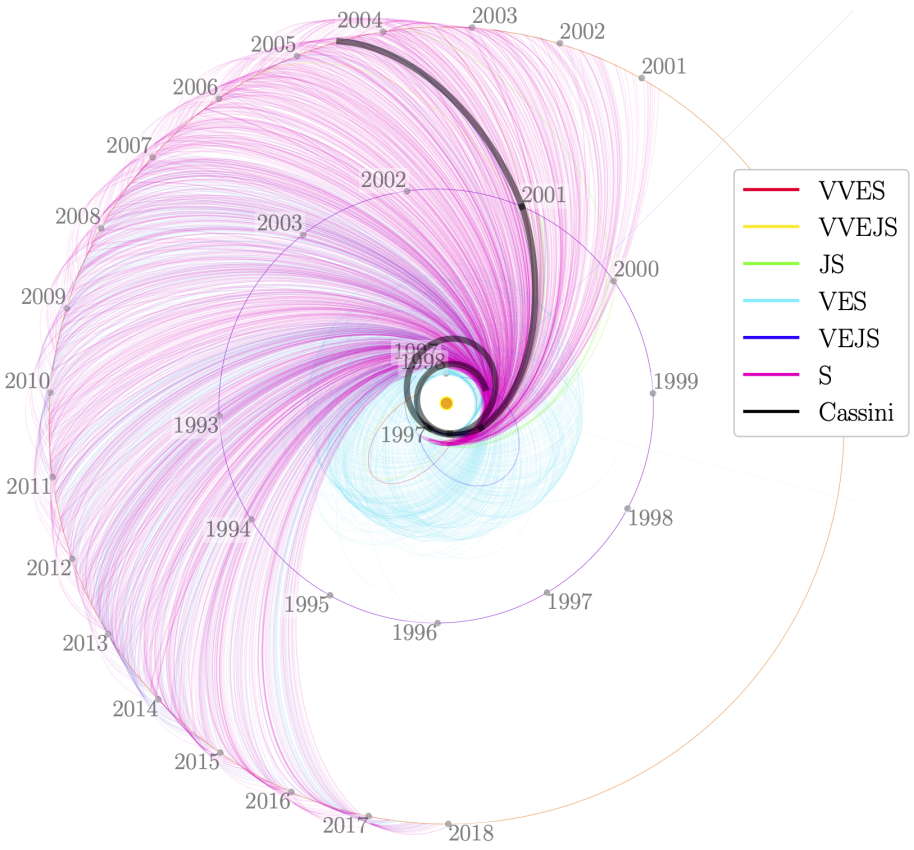


Figure C.20. The Tisserand network solutions provide the outline for a collection of randomized patched conic trajectories shown here in a polar view of three-dimensional trajectories. The spacecraft trajectories are constructed from Lambert solutions on a 3-D ephemeris. Some of the random encounter dates result in out-of-plane trajectories. A patched conic approximation of the true Cassini trajectory is highlighted.

VITA

James Moore earned his Bachelor of Science Mechanical Engineering from the University of Houston in 1997. After graduating, Jim began his career at the NASA Johnson Space Center working on the entry, descent, and landing phase of the Space Shuttle. In this position, Jim supported launch and landing operations in the Mission Control Center for multiple Space Shuttle missions. Near the end of the Space Shuttle program Jim joined the team designing and testing the parachute system for the Orion spacecraft.

During this time, Jim earned a Masters in Physics from the University of Houston Clear Lake in 2007 and a Masters in Aerospace Engineering from Purdue University in 2008. Well into his professional career, Jim began pursuing a Ph.D. in Aeronautics and Astronautics from Purdue with Prof. Jim Longuski as his advisor. While working toward his Ph.D., Jim has continued to work full time on projects including the Dream Chaser lifting body spacecraft, the OSIRIS-REx asteroid sample return mission, and the Nova-C lunar lander.

Outside of his work and studies, Jim enjoys spending time with his wife and four children, skiing and hiking near their home in Colorado.

INDEX