UNIVERSITÄT BREMEN

MASTER THESIS

ANALYSIS ABOUT CONSTRAINTS FOR LOW-THRUST GRAVITY-ASSIST SEQUENCING

Autor:
Hauke HANSEN

Erstprüfer:
Prof. Dr. Claus BRAXMAIER

Betreuer:
Volker MAIWALD

Angefertigt zum Erwerb des akademischen Grades

Master of Science

Bearbeitet in der Abteilung

Systemanalyse Raumsegment
Deutsches Zentrum für Luft- und Raumfahrt Bremen

23.05.2017

Universität Bremen

DLR

Eidesstattliche Erklärung

Ich, Hauke HANSEN, versichere hiermit wahrheitsgemäß, dass ich die vorliegende Arbeit, „ANALYSIS ABOUT CONSTRAINTS FOR LOW-THRUST GRAVITY-ASSIST SEQUENCING“, bis auf die offizielle Betreuung, ohne fremde Hilfe angefertigt habe. Die benutzte Literatur ist vollständig angegeben.

Unterschrift:

_____

Datum:

_____

UNIVERSITÄT BREMEN

Abstract

Fachbereich 4: Systems Engineering
Deutsches Zentrum für Luft- und Raumfahrt Bremen

Master of Science

ANALYSIS ABOUT CONSTRAINTS FOR LOW-THRUST GRAVITY-ASSIST SEQUENCING

by Hauke HANSEN

Matrikelnummer: 2469978
Email: *haukehansen89@t-online.de*

Exploring the solar system by sending spacecraft to different bodies and planets has significant scientific value. Two different means of enabling demanding missions to targets throughout the solar system are the application of gravity-assist manoeuvres and the usage of electrical low-thrust propulsion systems. Determining optimal trajectories for these kind of missions is a challenging task.

A new method of determining low-thrust gravity-assist sequences, while incorporating the gravity-assist partner into the optimisation process, was proposed by Maiwald [1]. The developed search method is simple to use and shows promising results. The quality of the solution can be improved however. To accomplish this, two constraints are proposed, which are designed to reduce the size of the search space.

The first constraint is meant to maximise the energy which is transferred to the spacecraft by conducting a gravity assist. The second constraint limits the possible next gravity-assist partner in an user specified range.

Within the present thesis, the implementation and evaluation of these constraints is realised. At first, the constraints are deployed into the method and a series of simulations, using an example mission from Earth to Jupiter, is conducted. The results are evaluated and discussed afterwards.

The evaluation of the tests revealed that the constraints have beneficial influences on the convergence properties of the solutions. The quality of the resulting low-thrust gravity-assist trajectories could be improved by utilisation of the implemented constraints.

Contents

List of Figures

List of Tables

List of Abbreviations and Symbols

Abbreviations

NASA	National Aeronautics and Space Administration
JPL	Jet Propulsion Laboratory
ESA	European Space Agency
JAXA	Japan Aerospace Exploration Agency
DLR	Deutsches Zentrum für Luft- und Raumfahrt
GOLT	Gravity-assist Optimisation for Low-thrust Trajectories
PPT	Pulsed Plasma Thruster
HET	Hall-Effect Thruster
SMART-1	Small Missions for Advanced Research in Technology
NSTAR	NASA Solar Technology Application Readiness
STOUR	Satellite Tour Design Program
EA	Evolutionary Algorithm
DE	Differential Evolution
PP	Partner Pool (Constraint)
AU	Astronomical Unit
MJD	Modified Julian Date
S/C	Spacecraft

Symbols

Δv	m/s	Velocity change
σ		Standard deviation
v_e	m/s	Exhaust velocity
m_f	kg	Combustion cut-off mass
m_0	kg	Initial mass
m_p	kg	Propellant mass
$r(\Theta)$	km	Radial position of spacecraft in respect to Θ
Θ	$^\circ$	Angular position of spacecraft
r_{per}	km	Pericenter distance
r_{SOI}	km	Radius of the sphere of influence
r_{min}	km	Minimum possible distance for gravity assist
F	N	Thrust force
α	$^\circ$	Angle between hyperbolic excess velocity at departure and velocity of planet
v^h_{pl}	m/s	Heliocentric velocity of planet
v^h_{in}	m/s	Heliocentric encounter velocity of incoming S/C
v^h_{out}	m/s	Heliocentric encounter velocity of departing S/C
v_∞	m/s	Hyperbolic excess velocity
$v^{pl}_{\infty,in}$	m/s	Hyperbolic excess velocity at arrival

$v_{\infty,out}^{pl}$	m/s	Hyperbolic excess velocity at departure
δ	$^\circ$	Planet-centric turning angle
I_{sp}	s	Specific impulse
g_0	m/s^2	Standard gravity acceleration at Earth ($9.81\,m/s^2$)
a	km	Semi-major axis
e		Eccentricity
i	$^\circ$	Inclination
μ	m^3/s^2	Standard gravitational parameter
T		Tisserand's Parameter
P	s	Orbital period
$\overrightarrow{a}_t$	m/s^2	Thrust acceleration of the S/C

Used in DE:

N_p	Population size
D	Number of parameters of every population member
$\overrightarrow{x}_i^g$	Population member i in generation g
$\overrightarrow{x}_{r0}$	Random population member 0
$\overrightarrow{x}_{r1}$	Random population member 1
$\overrightarrow{x}_{r2}$	Random population member 2
$\overrightarrow{v}_i^g$	Mutation vector for member i in generation g
W	Differential weight (also denoted as F in literature)
$\overrightarrow{u}_i^g$	Trial vector for member i in generation g
Cr	Crossover constant

Used in GOLT:

d_{peri}		First parameter of the Δv-gain constraint
d_{velo}		Second parameter of the Δv-gain constraint
$v_{\infty,start}$	m/s	Iinitial hyperbolic excess velocity of S/C
$v_{\infty,end}$	m/s	Hyperbolic excess velocity of S/C at end of mission
$LD_{mission}$	MJD	Launch date of the mission
$ToF_{mission}$	days	Overall mission flight time
$N_{rev,mission}$		Number of revolutions in the mission
GA_{ID}		Array with a sequence of gravity-assist partners
LD_{leg}	MJD	Launch date of the trajectory leg
ToF_{leg}	days	Flight time of the trajectory leg
$N_{rev,leg}$		Number of revolutions in the leg

1 Introduction

Exploring the solar system by sending spacecraft to different objects, like planets, moons or asteroids, is an important part of the basic research about the formation and the characteristics of the solar system and by that, the world we are living in. Ambitious and demanding missions to distant objects like *Cassini-Huygens*, *Voyager 1 & 2*, *New Horizons*, *Hayabusa 1 & 2* and *Rosetta* are rare, due to the tremendous amount of labour and thus money necessary, but significant in their scientific value. [2]

Planing and executing these type of missions takes long periods of time and hence requires the stability and experience of publicly founded governmental space programs (e.g. NASA, ESA, JAXA). The ESA mission *Rosetta*, for example, reached its target, the comet 67P/Churyumov-Gerasimenko, in 2014 after a flight time of around 10 years. Developing the mission took even longer – the first ideas to conduct a *Rosetta*-like mission date back to the mid 1980s. This sums up to a total timespan of around 30 years from the beginning of planing to the end of service in 2016 when the probe landed on the comet. [3]

Besides the development and assembly of the actual spacecraft and the operational costs, one major matter of expense is the cost of launching the payload into orbit [4]. These high costs are one of the factors which steady the high entry barrier for launching new scientific spacecraft. With the emergence of privately funded space programmes in recent years (the so called *NewSpace* industry, e.g. SpaceX, BlueOrigin), the goal of this industry to decrease the cost of space flight and the resulting competition imposed on the launch provider market, this high entry barrier will most likely get smaller over the next years [5]. This might be an important factor in increasing the amount of proposed exploration missions to other bodies in the solar system. If less budget needs to be allocated for launching the spacecraft, it might be possible that more research facilities and companies will develop and conduct a greater amount of missions.

However, the accessibility of space flight is only one part of the costly and labour intensive process of designing and flying spacecraft. Another part of the design process, which is especially important for exploration missions to other places in solar system and which is within the scope of this thesis, is the procedure of calculating and optimising the trajectory to reach the desired body. The mission analysis requires a great amount of expertise and experience of the researchers to enable the calculation of the most efficient trajectories [1]. In accordance to the possible simplification of the access to space, it is desirable to develop methods which simplify the process of designing demanding flight trajectories.

An important variable in the design process of flight trajectories, especially those to other solar system bodies, is the energy which is required to reach the target. This energy has to

be provided by the spacecraft itself, which is normally placed into an orbit around Earth, or an escape trajectory leading away from it, by the launch vehicle, i.e. the rocket. The energy necessary to alter the orbit to the desired trajectory is depicted in form of the necessary velocity change (Δv). Taking the ideal rocket equation, Eq. (1.1), into account – in which Δv is the maximum change in velocity, v_e the effective exhaust velocity of the engine, m_0 the initial mass of the spacecraft including propellant mass and m_f the dry mass of the spacecraft – it can be seen that the possible change in velocity is highly dependent on the amount of fuel stored on the spacecraft ($m_0 - m_f$) and the engine's exhaust velocity. [6]

$$\Delta v = v_e \cdot ln\left(\frac{m_0}{m_f}\right) \quad [\mathrm{m/s}] \tag{1.1}$$

The propellant mass is strictly limited by the total mass budget of the spacecraft. It is not possible to store an arbitrary amount of fuel on the probe to enable the execution of the required orbital manoeuvres, because of the interdependencies between the different subsystems of the craft. Cutting back on e.g. the usable science payload mass in order to increase the propellant mass is not preferable in most cases. Consequently, another means of enabling high-energy missions is to use engines, which are as efficient as possible. [7]

In general, two different flight proven propulsion technologies exist – high-thrust, low-efficiency chemical propulsion and low-thrust, high-efficiency electrical propulsion [8]. Both systems have specific advantages and disadvantages, which affect the respective main area of application. Chemical propulsion systems are primarily used as main drive in most satellites and interplanetary spacecraft, whereas electrical propulsion systems are mainly used in the attitude control system and as station keeping devices [9].

But electric propulsion systems are getting more important in recent years and missions like *Dawn* and *Hayabusa 1 & 2* utilised electric thrusters as the main drive to reach their interplanetary targets [2]. Due to the high efficiency of the electric engines the propellant mass could be drastically decreased, offering an advantage over conventional chemical drive systems [10].

The calculation of low-thrust trajectories, however, is a complex process because of the manner these engines are applied. In consequence to the small thrust output, electrical thruster are turned on for months or even years straight, altering the orbit of the spacecraft in a slow and steady way. In contrast to the brief intervals in which high-thrust engines are turned on to perform orbital manoeuvres, low-thrust engines are typically thrusting for a significant fraction of the whole mission duration. This constant change in velocity entails that the resulting trajectories normally have helical-like shapes. The task of determining the thrust magnitude and direction for every point in time in order to obtain an optimal trajectory, in terms of flight duration and fuel consumption, can become very difficult. The subject of low-thrust engines and the calculation of the resulting trajectories will be further discussed in Chapter 2.1. [11, 10]

Another very important means of improving mission performance, and by that facilitating many high-energy missions in the first place, is the utilisation of gravity-assist manoeuvres.

The basic concept is to perform a close fly-by at a planet (or moon) which shifts the direction of the spacecraft's velocity vector in the heliocentric reference frame. Throughout this shift in direction, orbital energy of the planet is transferred to the spacecraft, which provides additional energy for the alteration of the flight trajectory in the intended way. [12]

Gravity assists are commonly used in many high-energy missions like the already mentioned *Rosetta* mission, which performed four fly-bys (three at Earth and one at Mars) to reach the highly elliptical orbit of the target comet [3]. The application of gravity-assist manoeuvres enables a lot of different missions which would not be possible otherwise, mainly due to the limitations of currently available launcher technologies and the resulting strict boundaries of the payload mass, i.e. the mass of the spacecraft. More details on gravity-assist manoeuvres are provided in Chapter 2.2.

Because of the positive effects gravity-assist manoeuvres and low-thrust propulsion technologies have on the overall mission performance, it is worthwhile to investigate and develop methods which are capable of calculating optimal low-thrust trajectories while incorporating gravity assists [1].

1.1 A New Method for Low-Thrust Gravity-Assist Optimisation

Currently available methods which calculate low-thrust gravity-assist trajectories have one flaw in common: the gravity-assist partners are not included as an optimisation variable. This means that the trajectories between the gravity-assist partners are optimised, but the sequence of partner planets has to be provided to the optimiser. The knowledge about beneficial gravity-assist sequences for the respective mission profile has to be available a priori. This restricts the search for feasible solutions to sequences which are already known or were investigated beforehand. Furthermore, these methods rely heavily on the expertise and experience of the respective mission analyst. [1]

Low-thrust gravity-assist optimisation methods like the ones proposed by Crain, Bishop and Fowler [13], Zhao, Shang, Cui and Huang [14] and McConaghy, Debban, Petropoulos and Longuski [15] are based on this concept. To overcome this issue, the *System Analysis Space Segment* department of the *German Aerospace Center* (DLR) in Bremen is conducting research on a method which includes the gravity-assist partners into the optimisation procedure. This new method was first published in a 2017 paper by Maiwald [1].

In this paper, it is explained how gravity-assist sequences are mapped out for conventional impulsive missions by utilisation of Tisserand's Criterion and how this method could be transferred to low-thrust missions. The implications and restrictions which arise by attempting an application on low-thrust trajectories will be further discussed in Chapter 2.2.2. The basic idea of the proposed method is to force the optimiser to evaluate trajectories with specific gravity-assists partners and to benchmark the resulting trajectories to a no-gravity-assist trajectory. Which planet will serve as gravity-assist partner at which position in the

sequence is open to optimisation. This enables the optimiser to specifically use the gravity-assist partner as a control variable.

The trajectories between the gravity-assist manoeuvres are modelled by utilisation of a shape-based approach where the shape of the trajectory is described as an analytical function, Eq. (1.2) presented by Wall and Conway [16]. [1]

$$r(\Theta) = \frac{1}{a + b\Theta + c\Theta^2 + d\Theta^3 + e\Theta^4 + f\Theta^5 + g\Theta^6} \tag{1.2}$$

The variable r defines the radial and Θ the angular position of the spacecraft in a heliocentric reference frame. The coefficients $a - g$ are provided by the optimiser and describe the shape of the trajectory. Using a shape-based trajectory model, at the expense of accuracy, the calculation time can be reduced in comparison to trajectory propagation methods [17].

Currently, as the basic evaluation of this method is the first goal, a 2-dimensional model is used to reduce the complexity and calculation efforts. For missions including bodies in the solar system which posses relatively small inclinations, this simplification is acceptable. For the actual optimisation of the trajectories (i.e. calculation of the coefficients $a - g$) a heuristic approach utilising an evolutionary algorithm is used. An in-depth analysis of the whole optimisation method is provided in Chapter 2.4. [1]

The method has been coded in C++ to be further analysed and to enable an evaluation of the performance. The final program is called G*ravity-assist* O*ptimisation for* L*ow-thrust* T*rajectories* (GOLT) and the first realised tests show promising results. Using a mission from Earth to Jupiter as an example, the calculated trajectories result in a reduction of the required Δv of up to 22 % for a 2-leg trajectory, which incorporates one gravity-assist manoeuvre, in respect to the 1-leg trajectory, which functions as the benchmark.

However, the search method can be further improved in regard of the resulting trajectories' quality, which is randomly fluctuating, and the reproducibility of results. This is why two constraints are proposed which are designed to decrease the size of the search space. The implementation and evaluation of these constraints is the foundation of this thesis. [1]

1.2 Work Objectives and Outline

The primary work objective is to implement the constraints (specified in Chapters 1.2.1 and 1.2.2). The second objective is to conduct a series of simulations to evaluate whether the constraints entail any positive effects on the resulting trajectories. These simulations will use a similar mission profile as used by Maiwald, to enable a comparability of the results. Finally, it will be analysed and discussed if the search space could be decreased sufficiently to enable a better convergence of the results to the (global) optimum.

Within Chapter 2 the theoretical background this thesis is based on is provided. Beginning with current low-thrust engine technologies and the calculation of low-thrust trajectories (Chapter 2.1). Afterwards, the functional principle of gravity-assist manoeuvres and

the application of Tisserand's Criterion on low-thrust missions are explained (Chapter 2.2). Evolutionary algorithms are examined within Chapter 2.3, subsequently followed by the examination of the search method (GOLT) within Chapter 2.4.

Within Chapter 3 the implementation of the constraints is demonstrated, followed by an explanation about the method of data acquisition, i.e. the design approach of the simulation series which are conducted to evaluate the usefulness of the constraints.

Within Chapter 4 the results of the testing phases are presented and discussed within Chapter 5.

Finally, the implementation and application of the constraints is summarised within Chapter 6. The remaining open issues are outlined, an outlook to possible further investigations is provided and the work done within this thesis is concluded.

1.2.1 First Constraint: Δv Gain

The idea of the first constraint is to stick to gravity-assist manoeuvres of higher quality, i.e. manoeuvres which yield the most energy. Because it is assumed, that by forcing the optimiser to utilise manoeuvres which gain a maximum amount of Δv, the Δv requirements of the overall mission could be decreased. Maximising the outcome of one gravity assist might, for instance, enable a reduction of the total number of necessary manoeuvres to reach the target. Furthermore, it might be possible that more cost effective flight paths are produced in terms of Δv and flight time.

On the other hand, limiting the search space to high-energy yielding gravity assists could complicate the determination of useful subsequent manoeuvres. Conducting one good gravity assist might reduce the chance to find a beneficial next one. The spacecraft's arrival at the target planet at high velocities might also implicate the necessity of large deceleration manoeuvres at the end of the mission. The implications of using the Δv-gain constraint thus need to be carefully analysed. [1]

The currently implemented method for determining the parameters of the gravity-assist manoeuvre (i.e. the hyperbolic excess velocity v_∞, the planet-centric turning angle δ and the pericenter distance r_{per}) is to randomly choose them. They are interlinked with the trajectory leading to a gravity assist and can be arbitrarily altered by the optimiser. Eq. (1.3) depicts how the Δv gained by a gravity-assist manoeuvre is calculated. v_∞ represents the spacecraft's hyperbolic excess velocity as it reaches the gravitational influence of the body, r_{per} the pericenter distance of the fly-by hyperbola and μ_{pl} the gravitational constant of the respective planet [18].

$$\Delta v = \frac{2v_\infty}{1 + v_\infty^2 \cdot \frac{r_{per}}{\mu_{pl}}} \tag{1.3}$$

The three variables (δ, v_∞ and r_{per}) are not independent, as two of them are sufficient to completely describe a gravity-assist manoeuvre. E.g. by knowing the values of v_∞ and r_{per} the turning angle can be calculated (see Chapter 3.2.1 for more information). Because of this interdependence the turning angle is neglected during the remarks within this chapter.

The pericenter distance is constrained by the minimum possible distance (limited by the radius of the body, the atmosphere or other barriers like the radiation belts of Jupiter) and the sphere of gravitational influence of the respective body (r_{SOI}, radius of the sphere of influence). The hyperbolic excess velocity is defined to lie between 50 m/s and 10,000 m/s, as this range of velocities is believed to be sufficient for most gravity-assist manoeuvres.

Figure 1.1 depicts the possible Δv gain for a fly-by at Earth in respect to the pericenter distance and the hyperbolic excess velocity. The graph is a graphical representation of Eq. (1.3) and shows which combinations of r_p and v_∞ provide which Δv gain. The pericenter distance in this graph is specified between 6,500 km and 200,000 km. The minimum distance of 6,500 km represents the rounded value of Earth's radius (6,370 km). The sphere of influence, on the other hand, is reaching up to 925,000 km [19], which is not displayed to its full extend because the graph would become illegible.

The first observation is that the possible Δv gain is increasing when the pericenter distance is decreased. The closer the spacecraft is flying to the center of gravitational pull, the greater the gravitational attraction gets. This leads to a higher deflection of the velocity vector in the heliocentric reference frame and thus to a greater amount of gained Δv (see Chapter 2.2).

Furthermore, it can be observed that a hyperbolic excess velocity of 0 m/s entails that Δv is

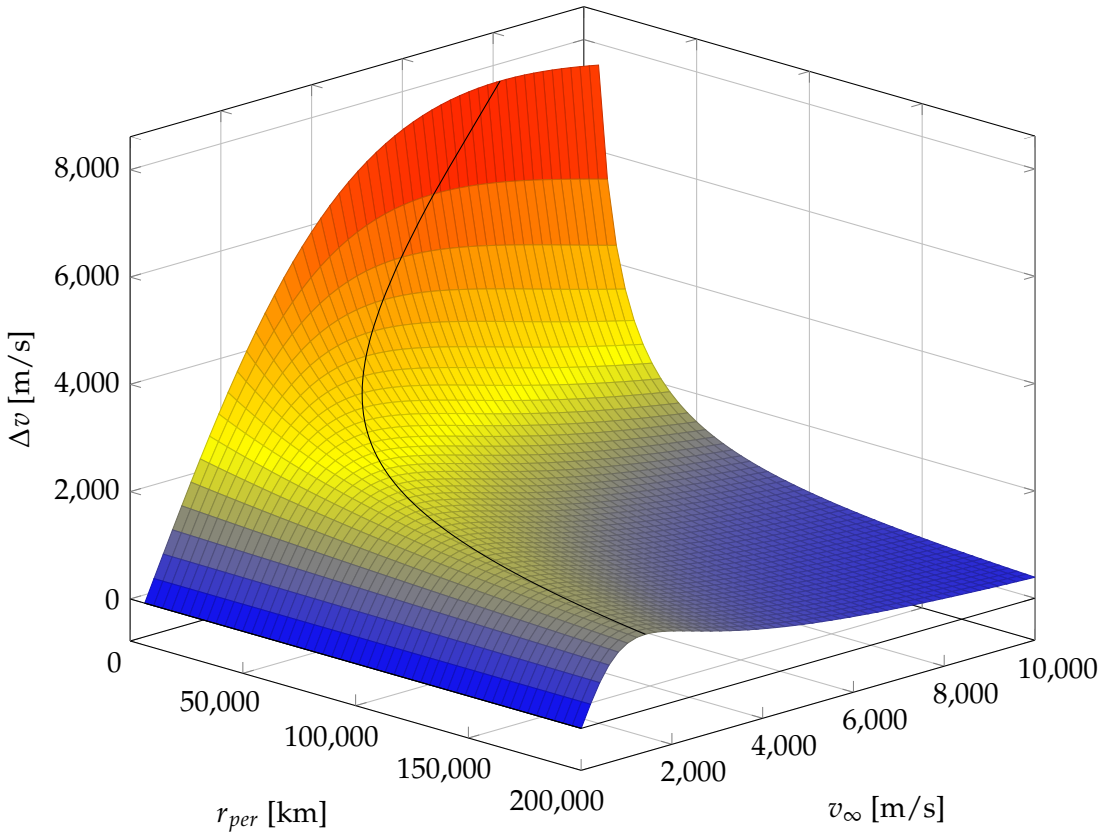

FIGURE 1.1: *Δv gain by performing a gravity assist at Earth, in respect to the fly-by distance r_{per} and the hyperbolic excess velocity v_∞. The black line represents the maximum possible Δv gain for every pericenter distance r_{per}.*

also 0 m/s. If the spacecraft has no relative velocity in respect to the planet at a distance just outside the sphere of influence, it would never reach the planet and no gravity assist would be performed. Hyperbolic excess velocities which are on the other end of the scale (reaching 10,000 m/s) also have a disadvantageous influence on the Δv gain of the manoeuvre. If the spacecraft is passing the planet at a very high velocity, the time frame in which it is under the gravitational influence gets shorter, thus reducing the effect of the gravity assist.

Consequently, there is a specific v_∞ for every r_{per} which yields the maximum amount of Δv. To define this extreme value, the derivative of Eq. (1.3) with respect to v_∞ has to be calculated, see Eq. (1.4). [20]

$$\frac{\delta \Delta v}{\delta v_\infty} = \frac{2}{1 + v_\infty^2 \cdot \frac{r_{per}}{\mu_{pl}}} - \frac{4 \cdot v_\infty^2 \cdot \frac{r_{per}}{\mu_{pl}}}{\left(1 + v_\infty^2 \cdot \frac{r_{per}}{\mu_{pl}}\right)^2} \tag{1.4}$$

The first derivative has to be equal to 0 to find the position of the extreme value, Eq. (1.5). [20]

$$0 = \frac{2}{1 + v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}} - \frac{4 \cdot v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}}{\left(1 + v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}\right)^2} \qquad \Big| \cdot \left(1 + v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}\right) \tag{1.5}$$

$$\Leftrightarrow \quad 2 = \frac{4 \cdot v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}}{1 + v_{\infty,ex}^2 \cdot \frac{r_{per}}{\mu_{pl}}}$$

$$\Leftrightarrow \quad 2 + 2 \cdot v_{\infty,ex}^2 \frac{r_{per}}{\mu_{pl}} = 4 \cdot v_{\infty,ex}^2 \frac{r_{per}}{\mu_{pl}}$$

$$\Leftrightarrow \quad 1 = v_{\infty,ex}^2 \frac{r_{per}}{\mu_{pl}}$$

$$\Leftrightarrow \quad v_{\infty,ex} = \sqrt{\frac{\mu_{pl}}{r_{per}}} \tag{1.6}$$

Due to the fact that v_∞, μ_{pl} and r_{per} cannot have negative values, the extreme value described by Eq. (1.6) is the positive maximum depicted by the black line in Figure 1.1. By inserting Eq. (1.6) into Eq. (1.3) the maximum possible Δv gain is calculated. Interestingly, this value is equal to the respective hyperbolic excess velocity ($v_{\infty,ex}$) and also the circular orbital velocity for the respective pericenter distance Eq. (1.7). This entails that by setting a specific pericenter distance, the maximum amount of Δv which could be gained by the gravity assist can be easily calculated. [20]

$$\Delta v_{max} = v_{\infty,ex} = v_{circular} = \sqrt{\frac{\mu_{pl}}{r_{per}}} \tag{1.7}$$

The search method in its current form does not take these considerations into account. The present proceeding is to randomly determine the values of r_p and v_∞ in the specified ranges, and by that randomly setting the resulting Δv gain.

As can be seen in Figure 1.1 there are large areas in which the gravity assist does not have a significant effect. For example, a randomly chosen fly-by at high velocity and a high pericenter distance influences the trajectory just in a very slight way. The first proposed constraint is designed to disregard these areas and to force the optimiser to utilise gravity-assist manoeuvres which gain more Δv and thus have greater value for the overall mission, by possibly decreasing e.g. the flight time, the number of gravity assists and the required fuel mass. [1]

1.2.2 Second Constraint: Partner Pool

The second proposed constraint is based on the assumption that for certain mission profiles only certain gravity-assist partners are advantageous. A trajectory from Earth to Mercury, for example, will most likely not benefit from a gravity assist at Saturn or Neptune. Or a trajectory leading to the outer rim of the solar system does not benefit from propagating back to gravity assists at the inner planets. The idea is to implement a method which lets the user specify in which range (in the direction and against the direction of flight) gravity-assist partners can be chosen by the optimiser. By doing this a lot of useless gravity-assist manoeuvres are neglected and it is assumed that by reducing the search space in this manner, the solutions will converge more quickly to the optimum. [1]

2 State of the Art

The following chapters are intended to provide the necessary background information this thesis is based on.

2.1 Low Thrust

Within this chapter the basic principle of low-thrust engines will be explained and which types are used for the propulsion of interplanetary spacecraft. Afterwards, low-thrust trajectories are illustrated and a quick method of trajectory approximation is presented.

2.1.1 Electrical Propulsion Systems

Electrical powered spacecraft propulsion systems generally work by using electrical energy to expel an (ionised) propellant, the reaction mass, at very high velocities to exert a force on the spacecraft [9]. Electrical thrusters are able to generate exhaust velocities, and with that specific impulses, which are multiple times greater than the ones produced by chemical propulsion systems. The advantages of these systems thus lie in the fuel efficiency. [8]

In general, three different types of electrical propulsion systems are differentiated, distinguished by the force which propels the reaction mass.

- **Electrothermal** systems use electromagnetic fields to heat up a propellant, which is then expanded through a nozzle to generate thrust. Arcjets (heating the propellent by generation of an electric discharge, i.e. an electrical arc) and resistojets (heating the propellant through an electrical resistor) are two commonly used types of these engines. Hydrazine is commonly used as reaction mass. With specific impulses around 500 s to 1,000 s and thrust levels which are smaller than 1 N, these engines are generally used as station keeping devices and attitude control systems [21].
- **Electromagnetic** systems are utilising the Lorentz force to accelerate ions inside a electromagnetic field. A simple and often used form of electromagnetic thrusters are the pulsed plasma thruster (PPT), which discharge an electrical current through a solid propellant to ablate and sublimate parts of the propellant. The resulting gas is turned into a plasma by an electrical arc and is expelled by application of an electromagnetic field, which exerts a strong Lorentz force on the plasma. PPT work in short pulses and generate specific impulses around 2,000 s to 3,000 s. The generated thrust is in the low mN and µN range, which is why these engines are mainly used for attitude control and the propulsion of very small spacecraft. [8, 21]

- **Electrostatic** systems ionise a gas (most of the time xenon) and accelerate it by application of an electric field. These engines generate thrust levels in the range of 25 mN to 600 mN with specific impulses ranging from 1,500 s to 5,000 s. Besides attitude control and station keeping are electrostatic ion engines also used as main propulsion systems for (interplanetary) spacecraft. Two different thruster types deployed for this task are the Hall-effect-thruster (HET), used on the ESA mission *SMART-1* and the electrostatic ion thruster, e.g. *NSTAR* used on the NASA mission *Dawn*. [8]

2.1.2 Low-Thrust Trajectories

To investigate what differentiates low-thrust trajectories from conventional Hohmann transfer orbits and which difficulties arise in computing these type of trajectories, at first the differences in the accumulation of Δv will be further explained. To do this, a hypothetical spacecraft is introduced with a total mass m_0 of 1,000 kg, a propellant mass m_p of 500 kg and a dry mass m_f of also 500 kg. Two different types of engines are utilised. A conventional chemical vacuum engine with an specific impulse I_{sp} of 320 s which produces 400 N of thrust and a electrical thruster with a specific impulse of 2,500 s which generates 100 mN of thrust.

To enable a statement about how much Δv can be generated by these thrusters, the possible duration of thrusting has to be calculated first. As a simplification it is assumed, that the thrust level and the specific impulse are constant during the whole operation of the engine.

To derive the propellant mass used per second, the thrust force F has to be divided by the exhaust velocity of the engine (the exhaust velocity is derived by multiplying the specific impulse with the standard gravity acceleration on Earth, $g_0 \approx 9.81\,\mathrm{m/s^2} \rightarrow v_e = I_{sp} \cdot g_0$) [8]. The total amount of propellant m_p then has to be divided by the propellant mass used per second to deviate the total burn duration t_b. Eq. (2.1) shows how the thrusting duration t_b can be described.

$$t_b = m_p \cdot \left(\frac{F}{v_e}\right)^{-1} \tag{2.1}$$

Reorganise this equation to single out the propellant mass m_p on one side, Eq. (2.2), and insert it into the basic rocket equation, (Eq. (1.1) in Chapter 1), to obtain a formula which calculates the generated Δv for the whole thrusting time t_b, Eq. (2.3).

$$m_p = \frac{t_b \cdot F}{v_e} \tag{2.2}$$

$$\Delta v = v_e \cdot ln\left(\frac{m_p + m_f}{m_f}\right)$$

$$\Rightarrow \Delta v = v_e \cdot ln\left(\frac{\frac{t_b \cdot F}{v_e} + m_f}{m_f}\right) \tag{2.3}$$

Figure 2.1 shows the graphical representation of this formula for the respective engine types. On the left side is the chemical thruster which can thrust for around 4,000 s ($\approx$ 66 min) and on the right side is the electrical thruster which operates for around 1,400 days ($\approx$ 3.8 years).

The vast differences in the resulting Δv are very obvious in this representation. The chemical engine is able to generate around 2,000 m/s in roughly an hour, which allows to treat orbital manoeuvres as impulsive changes in velocity and thus simplifies the calculation of trajectories. The electrical thruster operates for nearly 4 years and the spacecraft changes its velocity by 17,000 m/s in this time.

This constant change in velocity, and by that the constant alteration of the orbital parameters, complicates the computation of optimal trajectories for low-thrust missions. Furthermore, during the complete time of operation the engine can be throttled up or down and the direction of thrust can be arbitrarily changed. This opens an infinite search space where at each point in time the thruster can be operational in every direction and with every possible thrust level.

An example for a low-thrust trajectory is shown in Figure 2.2. This picture, published by the Jet Propulsion Laboratory (JPL), depicts the trajectory the *Dawn* spacecraft took on its journey to Vesta and Ceres [22]. The continuous thrust exerted by the propulsion system generates a helical-shaped orbit. Mission design of low-thrust trajectories therefore requires a quick method of approximation, as there is no analytical solution for a problem of this complexity. There are a very large number of different solutions to accomplish the mission, with a large amount of different launch dates, flight times and thrusting phases. [16]

Different approaches are described for the computation of low-thrust trajectories. For instance the impulsive Δv model proposed by Sims and Flanagan [11], where a low-thrust trajectory is modelled as a sequence of impulsive manoeuvres. This basic trajectory model

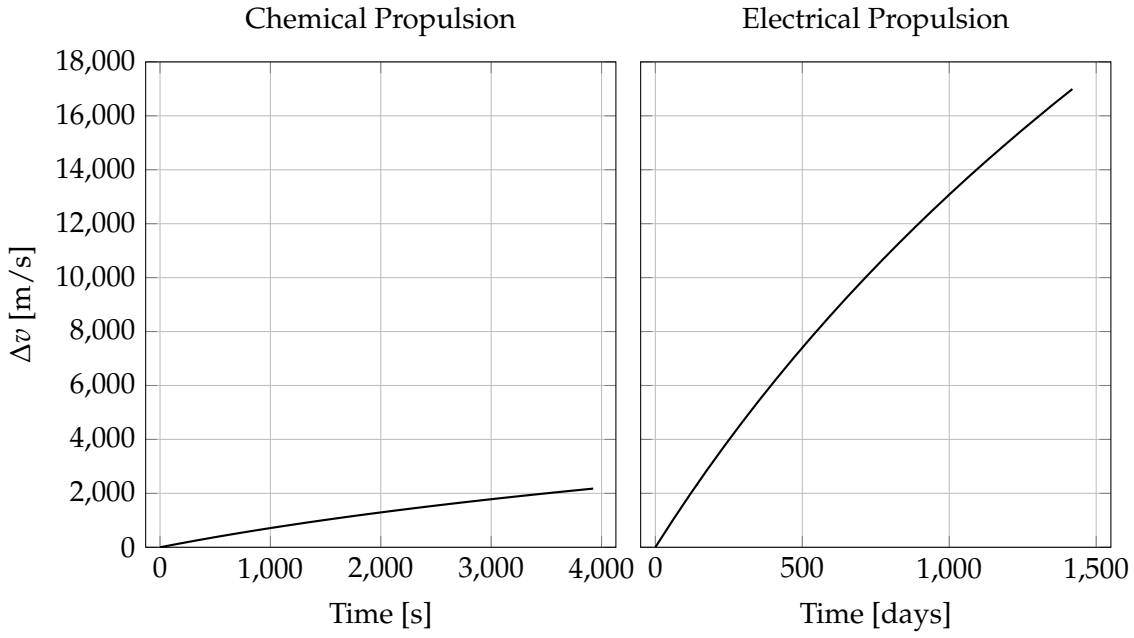

FIGURE 2.1: *Difference in accumulated Δv for a 1,000 kg spacecraft with 500 kg propellant. Using a chemical thruster (400 N, I_{sp}: 320 s) on the left side and an electrical thruster (100 mN, I_{sp}: 2,500 s) on the right side.*

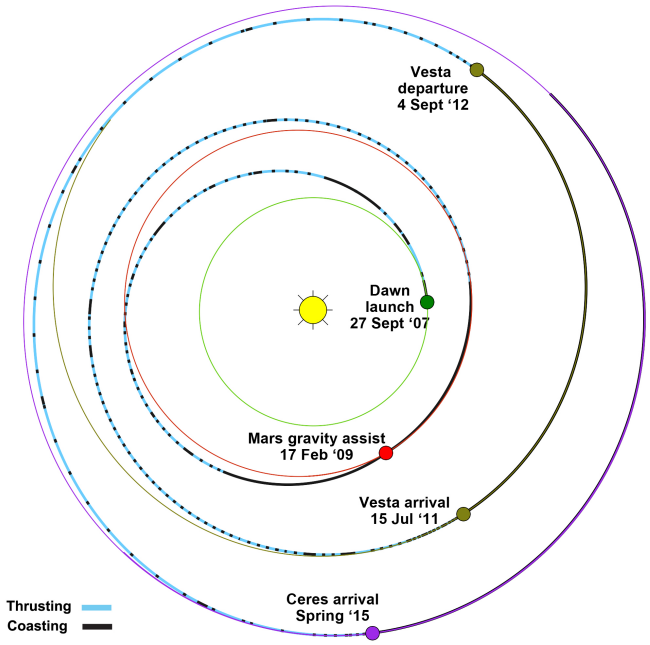

FIGURE 2.2: *Trajectory of the Dawn spacecraft launched in 2007. The thrusting phases are depicted in blue color and the coasting phases in black. Source: NASA - JPL URL: http://dawn.jpl.nasa.gov/mission/timeline_trajectory.html [22]*

was for example used by Yam, Biscani and Izzo [23] to develop a global optimisation algorithm for low-thrust missions.

Shape-Based Approach

The trajectory model used in GOLT is the *shape-based approach* described by Wall and Conway [16]. The proposed method is based on approximation of the trajectory by an inverse polynomial, Eq. (2.4), with seven free parameters ($a - g$) which incorporate the boundary conditions, like launch date, flight time, number of revolutions around the solar system, and describe the flight path of the spacecraft. As mentioned in Chapter 1, the radial position of the spacecraft is described by r and the angular position by Θ. [16]

$$r(\Theta) = \frac{1}{a + b\Theta + c\Theta^2 + d\Theta^3 + e\Theta^4 + f\Theta^5 + g\Theta^6} \tag{2.4}$$

Within the search method GOLT, the complete trajectory is divided into legs, where each leg describes the path from one gravity-assist body to the next one. Accordingly, for n gravity-assist manoeuvres the trajectory consists of $n + 1$ legs. Every leg is represented by one particular polynomial described by the parameters $a - g$. This method enables a fast evaluation of a vast amount of different trajectories, as the optimiser can quickly link the boundary conditions (e.g. date of departure at body 1, date of arrival at body 2, flight time between

the bodies, arrival and departure velocities, etc.) and calculate a transfer trajectory from one body to the other. The resulting trajectories, however, are not perfectly accurate, which is why this approach is used as a quick method for pre-evaluating different solutions, that serve afterwards as an initial guess in a more precise optimisation process. This final optimisation is not within the scope of this thesis and will thereby not be further discussed. [16]

To provide an example how these shape-based trajectories may look, a 1-leg trajectory, that was calculated during the later presented testing phases (see Chapter 4), is shown in Figure 2.3. The trajectory represents a direct insertion of the spacecraft into the orbit of Jupiter, starting at Earth, with one revolution around the center of the solar system. The solution of the inverse polynomial, the required Δv, the launch date (LD displayed as Modified Julian Date, MJD) and the flight time (ToF) are depicted to the right side of the graph.

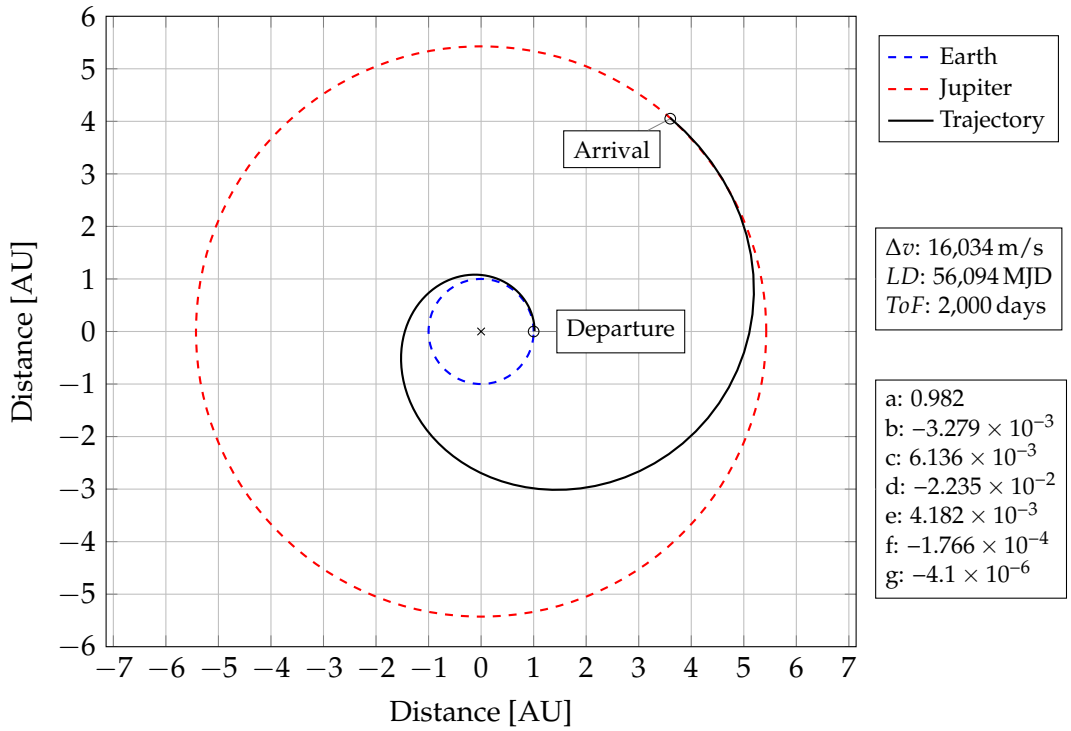

FIGURE 2.3: *Example for a shape-based, 1-leg trajectory from Earth to Jupiter. The coefficients which describe the inverse polynomial, Eq. (2.4), are depicted on the right side of the graph. The axes are scaled in astronomical units (AU). The slightly elliptical orbits of Jupiter and Earth are approximated as circles in this graph.*

2.2 Gravity Assist

Missions leading to other bodies in the solar system can be expensive in terms of flight time and Δv requirements. A key component in reducing theses costs is the application of gravity assists. These manoeuvres provide a way to harness orbital energy of a planet as a means

to alter the trajectory of the spacecraft. The practical application of gravity assists enabled many highly demanding exploration mission to deep space bodies. [24, 25]

The functional principle of gravity assists is best explained by considering the implied velocity vectors of the spacecraft and the planet, see Figure 2.4. In a planet-centric reference frame an incoming spacecraft will travel on a hyperbolic path around the planet. The gravitational attraction of the planet curves the trajectory of the spacecraft. Provided that the velocity of the planet v_{pl} remains constant throughout the encounter, the magnitudes of the incoming and the outgoing velocities remain constant during this deflection [26]. However, assuming a constant velocity of the planet is not entirely correct. As the planets in the solar system are located on slightly eccentric orbits, their respective orbital velocities change over time. But for the short amount of time the gravity-assist encounter takes place, the planet's velocity change is negligible. [12]

How this curvature in the trajectory enables the transfer of energy to the spacecraft's orbit

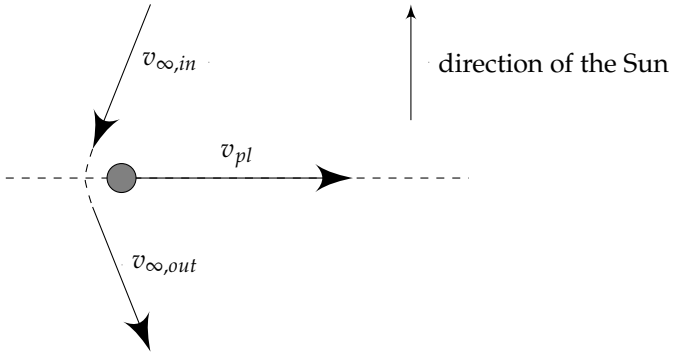

FIGURE 2.4: *Planet-centric view of a gravity assist. The incoming spacecraft ($v_{\infty,in}$) travels on a hyperbola around the planet (velocity depicted by v_{pl}) and obtains a deflected trajectory at departure ($v_{\infty,out}$). The magnitudes of the incoming and outgoing velocities remain constant. [24]*

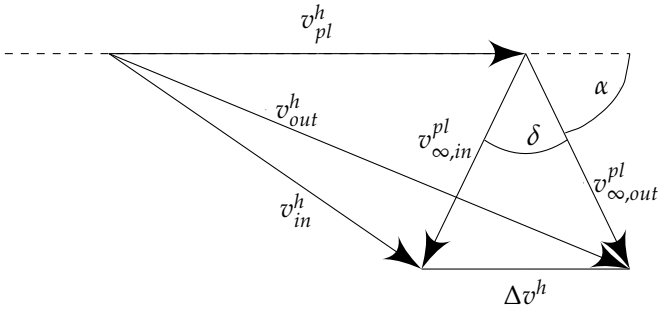

FIGURE 2.5: *Velocity vector diagram of a gravity assist. Heliocentric values marked with ^{h} and planet centric with ^{pl}. v^h_{pl}: velocity vector of planet - v^h_{in}: incoming - v^h_{out}: outgoing velocity of the spacecraft - Δv^h: Δv gain - $v^{pl}_{\infty,in}$: hyperbolic excess velocity at arrival - $v^{pl}_{\infty,out}$: departure - α: angle between hyperbolic excess velocity at departure and velocity of planet - δ: turning angle.*

can be seen when the heliocentric and the planet-centric velocity vectors are regarded together. Figure 2.5 shows a combined view of the heliocentric vectors (marked with ^{h}) and the planet centric (marked with ^{pl}). Adding the velocity vector of the planet v_{pl}^{h} with the incoming, respectively the outgoing, velocity vector of the spacecraft, derives the heliocentric velocity vectors of the spacecraft before (v_{in}^{h}) and after (v_{out}^{h}) the gravity assist. It can be seen that the outgoing velocity's vector v_{out}^{h} is larger than the incoming one. The spacecraft thus gained energy, i.e. velocity, by conducting the gravity assist. [24]

The magnitude of the difference between v_{in}^{h} and v_{out}^{h} depicts the velocity change induced by the gravity assist (Δv^{h}). The turning angle δ describes the angle between the planet-centric velocities of the spacecraft. α describes the angle between the hyperbolic excess velocity at departure and the heliocentric velocity of the planet. This angle determines whether the spacecraft gains or looses energy in the heliocentric reference frame. An angle of 0° provides the most energy, as the departing velocity vector of the spacecraft is aligned with the velocity of the planet – the gained Δv is at its maximum. An angle of 180° on the other hand reduces the speed of the spacecraft by the greatest amount, resulting in a heliocentric orbit of lower energy. The angle can be arbitrarily altered, by changing the direction of the incoming spacecraft in respect to the planet. If the spacecraft is still very far away from the planet, it is possible by application of small corrective manoeuvres to modify the trajectory in a way that every desired α can be generated. [24]

2.2.1 Gravity-Assist Sequencing Using Tisserand's Criterion

A technique to plan out gravity-assist sequences, from an energetic point of view, is the application of Tisserand's Criterion. This criterion was described by French astronomer Félix Tisserand in 1889 as an invariant quantity in the orbits of comets, that got perturbed by the gravitational influence of Jupiter [27]. The specific function of the orbital parameters a (semi-major axis), e (eccentricity) and i (inclination) remained constant before and after the encounter with the planet. By using this function, see Eq. (2.5), observed comets could be identified, although their orbital period and perihelion changed throughout the encounter with Jupiter. The Tisserand parameter T stays approximately constant for these different observations: [24, 28]

$$T = \frac{r_{pl}}{a} + 2\sqrt{\frac{a(1-e^2)}{r_{pl}}} \cdot cos(i) \approx const \tag{2.5}$$

r_{pl} denotes the heliocentric radius of the planet in this equation, i.e. the distance to the Sun. Tisserand's Criterion and its graphical representation that was developed for the *Europa Orbiter Tour* [29], are nowadays commonly used to determine possible sequences which can be achieved by using gravity-assist manoeuvres. An example for these so called Tisserand graphs (in this case a $P - r_p$-plot) is depicted in Figure 2.6. In this illustration, encounters with Venus, Earth and Mars are displayed. The X-axis shows the heliocentric pericenter and the Y-axis the orbital period. Every point on the graph thus refers to one specific heliocentric

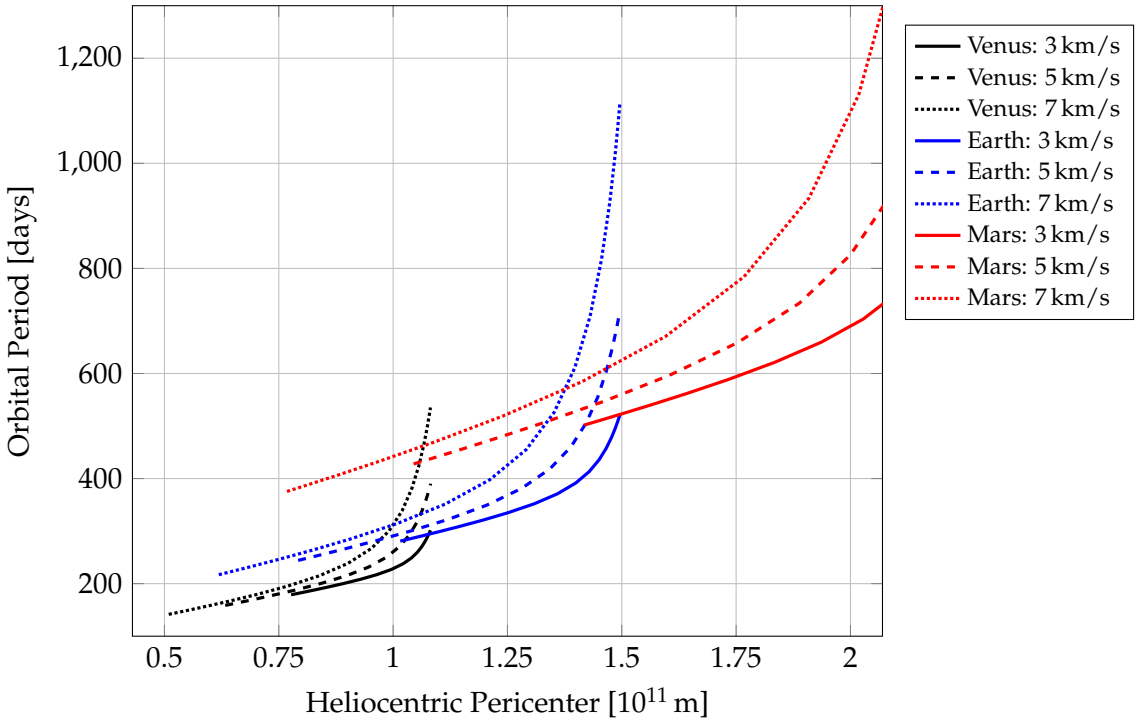

FIGURE 2.6: *Tisserand graph ($P - r_p$-plot) of gravity assists at Venus, Earth and Mars. The orbital period is plotted over the heliocentric pericenter and graphs for hyperbolic excess velocities of 3 km/s, 5 km/s and 7 km/s are depicted.*

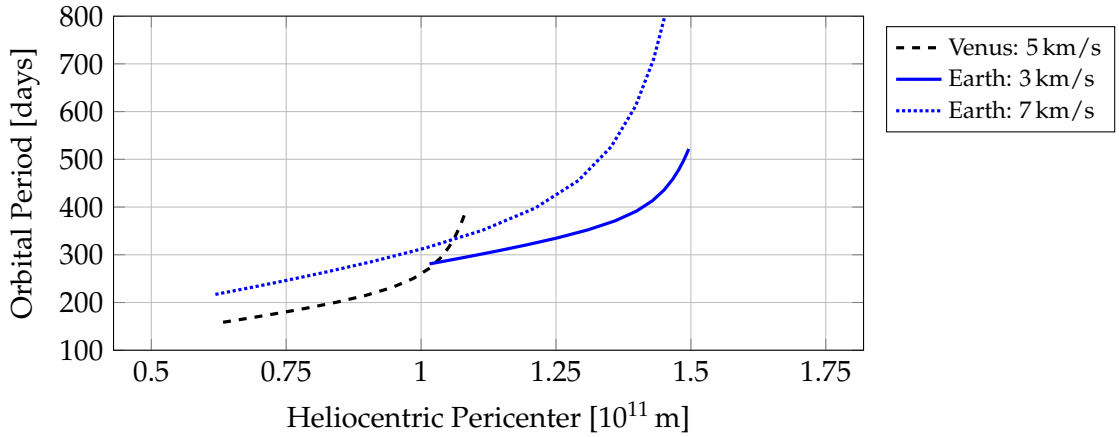

FIGURE 2.7: *Example for a VEEGA sequence: The spacecraft leaves the Earth at 3 km/s and performs a fly-by at Venus to enable a return to Earth at a higher hyperbolic excess velocity of 7 km/s. Multiple consecutive gravity assists at Earth enable a heliocentric orbit of high energy orbit.*

orbit described by pericenter distance and orbital period. The different lines show which orbits can be obtained by conducting a gravity assist at the respective planet with a specific hyperbolic excess velocity. The highest point of the line relates to the angle α being 0° and the lowest point to being 180°. [24]

By following the paths through the intersection points, possible gravity-assist sequences can

be determined. A classical path is the VEEGA sequence (Venus, Earth, Earth, see Figure 2.7) which was used by the *Galileo* spacecraft [24]. The sequence starts with a trajectory leading away from Earth with a velocity of 3 km/s. Then a gravity assist at Venus is conducted with a hyperbolic excess velocity of 5 km/s which enables a return to Earth with a higher velocity, e.g. 7 km/s in this example. By conducting two consecutive gravity assists at Earth, the heliocentric orbit of the spacecraft can be significantly raised to reach planets like Mars, Jupiter or beyond. Following this sequence requires only minimal propulsion by the spacecraft for corrective manoeuvres in deep space to achieve the desired fly-by parameters. [24]

It shall be noted, that the possible flight paths derived by this method are just necessary conditions and not sufficient, as the phasing of the planets is not considered. Tisserand graphs only show the possible trajectories from an energetic point of view – if a desired flight path is possible in the first place. The generated sequences get fed into an optimisation algorithm which includes the phasing of the planets and calculates optimal trajectories for the respective sequences. An example of these optimisers is the *satellite tour design program* (STOUR) developed by JPL for the *Galileo* mission [30]. [24]

2.2.2 Application of Gravity-Assist Sequencing on Low-Thrust Trajectories

Tisserand's Criterion is derived under the premise of the *restricted, circular three body system*. This means that the masses of the involved bodies (planet, sun and spacecraft) are of different orders of magnitude and constant, the orbits are circular, the only forces involved are of gravitational origin and the energy relation depicted by Eq. (2.5) is true for large distances between the planet and the spacecraft.

This premise is violated in several aspects by attempting to apply the criterion on low-thrust trajectories. The applicability on low-thrust missions and the necessary changes to the calculations are explained in three associated papers by Maiwald from 2015 [28], 2016 [20] and 2017 [1]. Using Tisserand's Criterion to map out possible gravity-assist paths through the solar system always includes violations of the restricted, circular three body system to some extend, as there are more than three bodies involved and the respective orbits are usually eccentric. To account for the application of continuous thrust, a modified Tisserand's Criterion was derived by Maiwald, which introduces an additional term into the Eq. (2.6). [28, 20, 1]

$$\frac{1}{2a_2} + \sqrt{a_2(1-e_2^2)} \cdot cos(i_2) = \frac{1}{2a_1} + \sqrt{a_1(1-e_1^2)} \cdot cos(i_1) + 2 \int_{t_1}^{t_2} \overrightarrow{v} \cdot \overrightarrow{a}_t dt \tag{2.6}$$

The value of Tisserand's Criterion after the encounter (state 2) is equal to the value of Tisserand's Criterion before the encounter (state 1) extended by the energy of the thrusting between these two instances, i.e. the integral part. The semi-major axes are scaled with the solar distance of the planet, so r_{pl} does not appear in this equation. [28, 20, 1]

The thrust energy part of this equation is specific to the respective trajectory, which is why the application of the modified Tisserand's Criterion requires knowledge about the exact flight path of the spacecraft. Accordingly, the modified Criterion cannot be used in the same way as for impulsive missions – to assess beneficial gravity-assist sequences in the first place – because the resulting trajectory has to be known to calculate Eq. (2.6). This means that Tisserand graphs cannot be used for an a priori mapping of low-thrust missions, because the introduction of thrust enables a linkage between the respective Tisserand graph lines in a non-specified way. A possible link (without any calculated values) which could be obtained by application of thrust is depicted in Figure 2.8. [28, 20, 1]

Theoretically, the modified Tisserand's Criterion can be used for planning low-thrust missions, but practically the absence of a priori information about the actual trajectory reduces the applicability, as non-specific interlinkages between the different orbital states are conceivable.

2.3 Evolutionary Algorithms

The method of low-thrust gravity-assist optimisation investigated in this thesis utilises an evolutionary algorithm, specifically differential evolution, as the optimisation method. This chapter provides information about evolutionary algorithms in general and exemplifies the characteristics of differential evolution in Chapter 2.3.1.

Evolutionary computation denotes a broad range of different probabilistic global optimisation procedures that utilise models of genetics and natural selection as major elements in

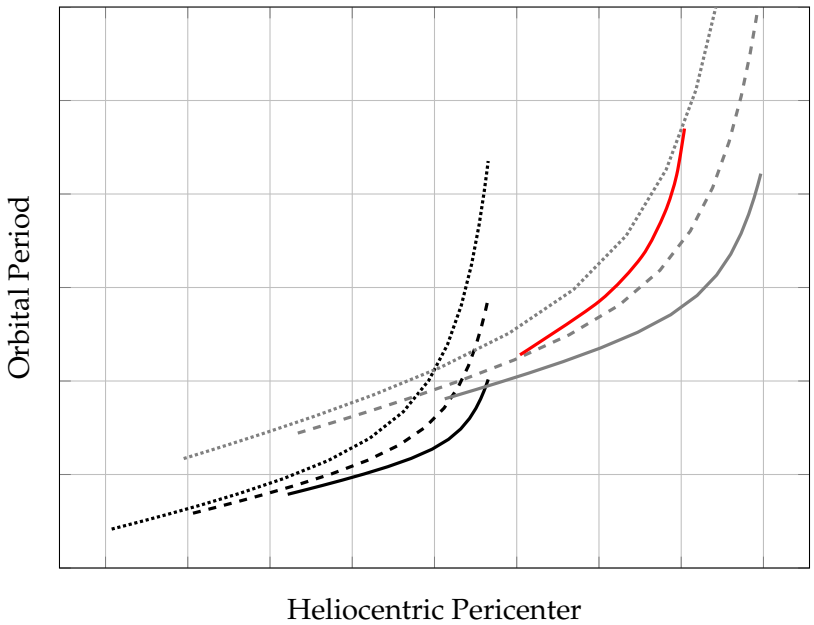

FIGURE 2.8: *Possible change of the orbital parameters of a spacecraft which utilises low-thrust propulsion (red line). No actual values were calculated, as this graph is meant to illustrate the idea.*

their design and implementation [31]. A great variety of different approaches were developed like genetic algorithms, evolution strategies, evolutionary programming and genetic programming [32]. The conceptual basis of these different algorithms is mostly the same and will be explained in this chapter.

Evolutionary algorithms (EA) use a vocabulary derived from biology, especially the evolution of species. The key part of an EA is a *population* of elements which represent the potential solutions of the given optimisation problem [32]. This population of solutions is optimised in an iterative process (over the course of *generations*) which rewards better solutions and sanctions worse. The idea is to integrate an evolutionary pressure to strive for the best possible solution. The algorithm has to incorporate three core principles of the Darwinian evolution in order for a natural selection of the best solution to occur. [33, 32, 31, 34]

- **Heredity** – a process must be in place which allows the properties of individuals (*parents*) to be passed on to their successors (*children*).
- **Variation** – a variety of the traits has to be present in the population and a mechanism to introduce new (random) variations.
- **Selection** – there must be a mechanism which selects the superior solutions which are closer to the targeted optimal solution, i.e. that are more *fit* than the other solutions. [33]

To enable the concept of *selection* and to assess which solutions are better than the others, a fitness function needs to be designed, which allocates a certain numerical fitness value to every population member [33]. Superior solutions get a higher fitness value than inferior solutions. How this fitness function is implemented is greatly dependent on the actual optimisation problem. In the case of low-thrust trajectory optimisation a solution which requires less Δv and/or less flight time could for instance indicate a solution of higher fitness. [33, 31]

The derivation of new population members in a new generation is done by selecting two or more solutions as *parents*. The traits of the parents are crossed over to form a new set of properties which will be passed on the the *child* solution. The implementation of this *crossover* mechanism is also dependent on the optimisation problem and the data structure of the elements. A possible crossover technique might be to take half of the data of one parent solution and combine this with half of the data of the other solution. The selection process of new parent solutions is normally combined with the calculated fitness value. This means that solutions with a higher fitness are more likely to serve as parents. This method enables the *heredity* of good solutions to a new generation and the combination of good solutions to maybe even better ones. [33, 31]

A mechanism to introduce *variation* into the solutions is normally incorporated in two different ways. By randomly generating the initial population, a certain initial variety is present in the first place. The bigger the number of population members is, the greater this initial variation of the solutions is. In highly complex, multi dimensional search spaces, this initial

variety is most like not sufficient to find the optimal solution. Therefore a second variation mechanism has to be introduced, the *mutation*. The mutation mechanism is interconnected with the heredity of properties from parents to children. When properties are passed on from one generation to the next, new random values are introduced into the traits of the children at a certain probability. A mutation rate of 1 % for example means that every inherited property (an integer value for instance) has a chance of 1 % to be overwritten by a new and random value. The random mutation of properties prevents (or should prevent) the optimiser from getting stuck at certain parts of the search space without any means of randomly generating new ways to find the global optimum. [33, 31]

A simple EA is working in the following way: [31]

Create an initial population

Repeat

Calculate fitness of every population member

Select parents from the population, with a fitness bias

Generate children by crossing over the parents

Mutate the children based on mutation rate

Place the children into the population

Until best solution is found or the results do not get better over multiple generations

EA are used in a great variety of different optimisation problems due to their robustness and their ability to find (near) global optima in many different high-dimensional multi-modal search spaces [32].

2.3.1 Differential Evolution

The specific evolutionary optimisation algorithm used in GOLT is called *Differential Evolution* (DE). DE was developed by Storn and Price [35] to minimise objective functions of non-differentiable, non-linear, multi-modal, noisy problems with continuous or discrete parameters. The optimiser works with a population of N_p vectors, where each vector has D parameters or dimensions. $\overrightarrow{x}_i^g$ describes the population member i in generation g, with $\{0 \leq i \leq N_p - 1\}$ and $\{0 \leq g \leq g_{max}\}$. DE generates new population members as perturbations of existing vectors. [36]

The mechanism is, that for every population member of the current generation (the target vector $\overrightarrow{x}_i^g$) , three random members of the population are selected $(\overrightarrow{x}_{r0}, \overrightarrow{x}_{r1}, \overrightarrow{x}_{r2})$ and perturbed to the mutation vector ($\overrightarrow{v}_i^g$), utilising the formula depicted in Eq. (2.7). W denotes the *differential weight* and is a real valued number, chosen by the user, between 0 and 2. [36]

$$\overrightarrow{v}_i^g = \overrightarrow{x}_{r0} + W \cdot (\overrightarrow{x}_{r1} - \overrightarrow{x}_{r2}) \tag{2.7}$$

Each vector of the current population is then recombined with the respective mutant vector to produce a trial vector $\overrightarrow{u}_i^g$. This *crossover* step requires a user specified crossover constant

$Cr \in [0, 1]$. The recombination works by iterating through every parameter D of the target vector and always producing a random number between 0 and 1. For every parameter, this random number is compared to Cr. If the random number is less or equal than the crossover constant, the parameter of the new trial vector is set to be the respective parameter of the mutant vector. If the random number is bigger than the crossover constant, the parameter of the current population vector is copied to the trial vector. The crossover constant thus denotes a probability by which the mutated values are transferred to the trial vector. [36]

The last step is to compare the fitness function of the trial vector $\overrightarrow{u}_i^g$ to the original target vector $\overrightarrow{x}_i^g$. If the fitness is higher, the trial vector replaces the target vector in the population of the next generation $g + 1$. These three steps, *Mutation*, *Crossover* and *Selection/Evaluation*, are performed for every population member of the current generation to generate the population of the next generation. This optimisation is proceeded until a stopping criterion is met. This criterion might be a maximum number of generation or a number of subsequent generations where no trial vectors of higher fitness were found. [36]

2.4 GOLT

This chapter is dedicated to provide insights into the underlying mechanics of the search method GOLT (G*ravity-assist* O*ptimisation for* L*ow-thrust* T*rajectories*). At first the basic concept of optimisation is described, followed by a specification of the local and global control variables and the concept of incorporating the selection of gravity-assist partners into the optimisation. Afterwards, preliminary simulation results are presented.

2.4.1 Concept of the Optimisation Method

As elaborated in Chapter 2.2.2, Tisserand's Criterion cannot be used for an a priori mapping of feasible low-thrust gravity-assist trajectories, as the resulting trajectory has to be known in order to calculate the modified Tisserand's Criterion. Because of this, as mentioned in Chapter 1.1, the idea of the search method is to force the optimiser to investigate gravity-assist trajectories with given fly-by partners, chosen randomly from a pool of possible partners. The mission sequence is thereby represented as a list of encounter partners, starting with the launch body and ending at the target planet. The rest of the sequence, beyond these two fixed bodies, is open for optimisation.

The trajectories in between the gravity-assist bodies are defined by velocity and position vectors and are part of the optimisation process. The complete mission is separated into segments, or legs, between the gravity-assist bodies, which each are defined by the respective departure and arrival body. The gravity-assist manoeuvres function as handover points, were the arrival body, encounter date and the departure velocity vector are passed on as starting constraints for the subsequent segment. As a quick approximation method, the shape-based approach is used (see Chapter 2.1.2). The resulting polynomial, described by

the coefficients $a - g$, is released as a set of 100 data points, each containing a value for $r(\Theta)$, Θ and the thrust acceleration a_t. [1]

The overall mission is described by four control variables – the mission launch date $LD_{mission}$, the flight time $ToF_{mission}$, the number of revolutions around the barycentre $N_{rev,mission}$ and the sequence of gravity assists GA_{ID}. The range in which these variables can be altered by the optimiser is provided by the user. A launch window gets specified, a maximum and minimum flight time, a maximum number of revolutions and the amount of gravity-assist manoeuvres. The number of revolutions around the solar system is needed in the shape-based trajectory model, as the final angle Θ_{final} is described as the angle Θ plus the complete revolutions multiplied with 2π ($\Theta_{final} = \Theta + N_{rev} \cdot 2\pi$). Derived from these mission variables, every trajectory segment is also described by a respective set of variables (LD_{leg}, ToF_{leg} and $N_{rev,leg}$). The variables describing the trajectory segments are constrained by the overall mission, as the particular launch dates, flight times and numbers of revolutions cannot exceed the limitations set by the mission variables. [1]

A 1-leg trajectory is thereby defined by three control variables that just obtain integer values, as the launch date and time of flight are described as whole days, without consideration of smaller time frames. Without incorporating gravity-assist manoeuvres, the search space is thus finite, which means that the optimisation method should be able to consistently find the same optimal trajectory.

However, this only the case as long as no hyperbolic excess velocities at start and end of the mission are defined. The user is able to specify these values to simulate, for instance, an initial trajectory leading away from Earth, a spacecraft is brought to by the launch vehicle. The magnitude of these velocities is defined by the user, the direction on the other hand is randomly defined by the optimiser. These random initial and finial velocities open an infinite search space, preventing the calculation of the exact same 1-leg trajectory.

For every gravity-assist manoeuvre, however, two additional real-valued variables describing the manoeuvre are introduced. As mentioned in the Chapters 1.2.1 and 3.2.1 two of the three variables (δ, v_∞ and r_{per}) are sufficient to specify a gravity assist. As the currently implemented (and later changed, see Chapter 3.2.1) method of determining these variables relies on choosing δ and v_∞, and calculating r_{per} based on the chosen values, it can be stated that for every gravity-assist manoeuvre, the variables δ and v_∞ are added to the system. As these variables are real valued an infinite amount of combinations is feasible.

The variable GA_{ID}, which contains the sequence of gravity-assist bodies, strongly influences the other variables describing the gravity assist, due to its discreteness. Values for variables like δ, v_∞ or the segment flight time for one particular gravity assist (e.g. at Earth), are most likely useless if applied to a gravity assist at a different planet (e.g. at Mars), as soon as the GA_{ID} variable gets changed by the optimiser.

For that reason, there has to be a distinction between variables which can be globally optimised and variables which are just locally viable for one gravity-assist manoeuvre. The local variables get overwritten in every iteration of the optimisation, because they highly depend

TABLE 2.1: *Global and local control variables used for the optimisation.*

Variable	Description
global variables	
$LD_{mission}$	Launch date of the overall mission and the first leg of the trajectory
$ToF_{mission}$	Overall flight time of the mission
$N_{rev,mission}$	Number of revolutions around the barycentre of the whole mission
GA_{ID}	Sequence of gravity-assist partners
local variables	
δ	Turning angle around which the planet-centric hyperbolic excess velocity is turned due to the gravitational influence of the planet
v_∞	Hyperbolic excess velocity of the spacecraft as it enters the sphere of influence of the planet (consists of X- and Y-component)
LD_{leg}	Launch date of the particular trajectory leg
ToF_{leg}	Flight time of the particular trajectory leg
$N_{rev,leg}$	Number of revolutions performed in the particular trajectory leg

on the global variables and especially the respective gravity-assist partner. Table 2.1 lists the control variables of the optimisation process. [1]

Theoretically, a search approach is possible, in which every variable gets varied in a structured manner to completely explore the search space. Practically, this is not viable, as for every additional gravity-assist manoeuvre a new, infinite search space gets introduced, making a structured search infeasible. Therefore a heuristic search approach, based on an evolutionary algorithm, is chosen. The first iteration of the method, which is investigated in this thesis, utilises Differential Evolution (DE) as the designated optimisation algorithm. As the search space is unknown and cannot be described by an analytical and differentiable function, using DE is a viable method [37]. However, the capability of the evolutionary algorithm gets limited by using local variables, which cannot be optimised over the course of generations. [1]

Furthermore, the number of control variables depends on the number gravity-assist manoeuvres, but the optimisation algorithm requires a constant number of control variables, as it works by recombining the current population members to produce new solutions. This is why, for every number of gravity assists (from 0 to the maximum amount specified by the user), the optimiser initialises a completely new population of solution candidates.

As the values of the control variables which provide good results for a trajectory with a certain number of gravity assists, are most likely not beneficial for a different amount of gravity-assist manoeuvres, this approach is feasible. A flight time and launch date which, for instance, enable a good 1-leg trajectory, are not necessarily as profitable for a 2-leg trajectory, as the phasing of the planet has to be considered and it might not be possible for the spacecraft to reach the fly-by planet in that certain time frame. Thus, for every different number of gravity assists a new population is created and an optimised trajectory is produced and released. [1]

Initially, DE was designed to minimise a certain cost- or error-function [36]. In this case,

however, the maximisation of a fitness-function is implemented, that is calculated based on the the inverse of the Δv requirement of the mission. A lower Δv requirement results in a higher fitness value and vice versa. [1]

Modelling the gravity-assist manoeuvre is done by application of Tisserand's Criterion. As explained above, the criterion cannot be used for an a priori investigation of the feasible gravity-assist partners, but the actual manoeuvre, i.e. the perturbation of the spacecraft's trajectory, can be approximated. For the relatively short amount of time the spacecraft is in close vicinity to the planet, the velocity change induced by the thruster is negligible. Thereby the integral part of the modified Tisserand's Criterion, Eq. (2.6), is small and the gravity assist can be modelled by shifting the heliocentric orbit of the spacecraft along the respective Tisserand graph line. The position and velocity of the planets are based on the real ephemerides, located in text files available to the program. [1]

The settings specified by the user are provided to the search method via a text file. The complete contents of this file are listed in Appendix A.

2.4.2 Preliminary Simulation Results

Preliminary test runs conducted by Maiwald [1] show promising results concerning the usefulness of the search method. The following settings were specified for these trial runs:

- Trajectory from Earth to Jupiter
- $LD_{mission}$: 56,000 MJD
- Launch window: 360 days
- ToF: between 2,000 days and 3,000 days
- $N_{rev,max}$: 4
- Maximum number of gravity assists: 2 (i.e. 1- to 3-leg missions)
- Population size: either 50 or 200
- Maximum number of generations: 1,000
- Crossover Constant Cr: 0.75 and Differential Weight W: 1

In total, 30 calculation have been executed to get an overview about the to be expected quality of the resulting trajectories with different populations sizes and numbers of gravity-assist manoeuvres. The 2-leg trajectories generated by using 50 population members were better than the 1-leg trajectory benchmark in 60 % of the cases. The trajectories produced by 200 population members, on the other hand, exceeded the benchmark in all cases. The Δv requirement of the mission could be decreased by performing one gravity-assist manoeuvre. The 1-leg trajectory required 16,256 m/s in all cases and the average Δv requirement of the 2-leg trajectories were 16,033 m/s (50 population members) and 14,083 m/s (200 population members). Most of the time the gravity-assist partner was the Earth. Mars was the designated partner of just two trajectories and was only found in the runs with 50 population members. The 3-leg trajectories however did not enable better results than the 1-leg

benchmark. The averagely required Δv of these trajectories was between 41,363 m/s (50 population members) and 32,151 m/s (200 population members). [1]

These tests revealed that the method is basically working and can produce good results for trajectories which incorporate one gravity-assist manoeuvre. It could be observed that utilising more population members has a very beneficial influence on the solutions. An initial population which ranges over a greater random area of the search space enables the calculation of notably better trajectories. However, by increasing the number of gravity assists to more than one, the computation of trajectories that perform better than the 1-leg benchmark gets considerable harder. None of the 3-leg trajectories could exceed the 1-leg benchmark. Furthermore, the quality of the solutions is randomly changing over successive calculations and it was not possible to reproduce a good solution in a later calculation. Whether the proposed constraints enable an overall improvement of the trajectory quality and variation, will be further evaluated within the next chapters. [1]

3 Methodology

Within this chapter the working steps required to successfully implement and test the constraints are described. In the first section, the proceedings to accomplish this task are illustrated. Within the second part, the considerations which went into the implementation of the constraints are explained. Afterwards, the conducted series of simulations are elucidated. The test set-up and the insights which are hoped to be obtained by the different testing phases are outlined.

3.1 Proceedings

The first step in implementing the constraints is to gain in-depth knowledge about the inner workings of the search method. To determine where the constraints will be incorporated into the actual source code, it is crucial to fully understand the whole search process. Furthermore, a literature review is conducted to assess the state of the art of the research fields relevant to this thesis – low-thrust propulsion and trajectory design / gravity-assist manoeuvres and gravity-assist sequencing / evolutionary algorithms.

A new input parameter, called *debug_mode*, got implemented in the course of this process, which reduces the amount of messages logged via the output console during the execution of the program. This reduced amount of output messages increases the performance of the search method by 45 %, resulting in a significantly smaller calculation time.

After finishing the thorough examination of the search method and locating the positions where the constraints can be inserted, the mathematical expressions modelling the desired search method behaviour are developed. The final functionality and the implementation process of the constraints is described in the following Chapter 3.2.

The evaluation of the usefulness of the implemented constraints is done by conducting a series of simulations. These tests are meant to systematically explore the implications of applying the constraints. The decisions and considerations which went into the design of these simulations are described within Chapter 3.3.

Then, after completely carrying out the simulations, the gathered data is processed into graphs and tables which enable the generation of quantifiable statements about the search method's behaviour while applying the constraints. The results comprised of the edited data are illustrated within Chapter 4.

Based on the data, it is investigated and discussed whether the constraints work in the intended way. The implications, benefits and characteristics which arise by application of the constraints are presented in Chapter 5.

3.2 Implementation of the Constraints

The Chapters 3.2.1 and 3.2.2 are providing information about the implementation process of the respective constraints. The mathematical considerations required for the incorporation are explained and the currently used parts of the search method are compared to the new, modified version which includes the constraints.

3.2.1 Implementation: Δv Gain

The idea of the first constraint is to neglect gravity assists which do not transfer much energy to the orbit of the spacecraft. As illustrated in Chapter 1.2.1 the optimiser will randomly choose the defining variables of a gravity-assist manoeuvre (the pericenter distance - r_{per}, the turning angle δ and the hyperbolic excess velocity at arrival - v_∞) without considering how much energy could be gained.

To acknowledge how the currently implemented method works, some additional mathematical expressions have to be regarded. At first a formula is needed which calculates the turning angle based on the values of r_{per} and v_∞. This can be derived by taking the vector diagram shown in Chapter 2.2 into account, see Figure 3.1 [1]. The values which are marked with ^{h} are, as before, valid inside the heliocentric reference frame – the velocity of the planet (v^h_{pl}), the incoming (v^h_{in}) and outgoing (v^h_{out}) velocity of the spacecraft and the heliocentric Δv gain (Δv^h). The planet-centric values (marked with ^{pl}) are the respective hyperbolic excess velocities at arrival and departure ($v^{pl}_{\infty,in}$ and $v^{pl}_{\infty,out}$). The turning angle δ is not specific to one of the reference frames.

As the magnitudes of $v^{pl}_{\infty,in}$ and $v^{pl}_{\infty,out}$ do not change during the gravity assist, it is not important to distinct between the two in the following considerations ($|v^{pl}_{\infty,in}| = |v^{pl}_{\infty,out}| = |v_\infty|$).

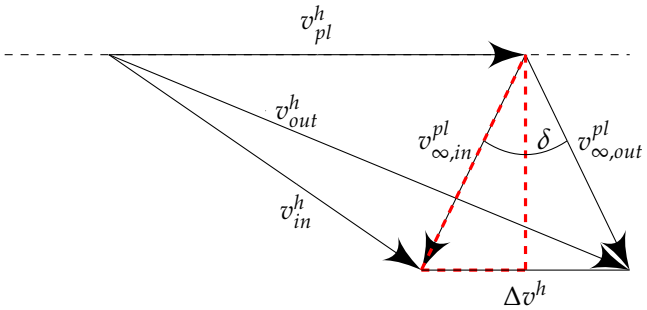

FIGURE 3.1: *Velocity vector diagram of a gravity assist. Heliocentric values marked with ^{h} and planet centric with ^{pl}. v^h_{pl}: velocity vector of planet - v^h_{in}: incoming - v^h_{out}: outgoing velocity of the spacecraft - Δv^h: Δv gain - $v^{pl}_{\infty,in}$: hyperbolic excess velocity at arrival - $v^{pl}_{\infty,out}$: departure - δ: turning angle.*

The relation depicted in Eq. (3.1) can be derived based on the red triangle in the diagram.

$$sin\left(\frac{\delta}{2}\right) = \frac{\Delta v}{2 \cdot v_\infty} \tag{3.1}$$

Reorganising this equation to place the turning angle on one side and applying the formula for the Δv gain (shown in Chapter 1.2.1, Eq. (1.3)), the relation described in Eq. (3.2) is resulting. This equation links the planet-centric variables and is currently used in the search method.

$$\delta = 2 \cdot sin^{-1}\left(\frac{1}{1 + v_\infty \cdot \frac{r_{per}}{\mu_{pl}}}\right) \tag{3.2}$$

If this equation is reorganised again, to calculate r_{per} based on the other two variables, the equation shown in Eq. (3.3) is resulting. This equation is also presently used.

$$r_{per} = \left(\frac{1}{sin\left(\frac{\delta}{2}\right)} - 1\right) \cdot \frac{\mu_{pl}}{v_\infty^2} \tag{3.3}$$

In a simplified way, the currently implemented method defines the gravity-assist variables in the following manner:

- Choose a new hyperbolic excess velocity at arrival in the sphere of influence of the planet $v_{\infty,in}$. This is a random value between 50 m/s and 10,000 m/s.
- After setting the magnitude of the arrival velocity, the direction is set by randomly defining the components of the vector in X- and Y-direction in a way that the magnitude of the velocity remains constant. $v_\infty = \sqrt{v_{\infty,x}^2 + v_{\infty,y}^2}$. The vector gets basically turned in a random direction which relates to a random determination of the angle α.
- Calculate the minimum and the maximum turning angle δ which are possible by arriving at this velocity. This is done by utilising Eq. (3.2) and setting the pericenter distance to r_{min} for δ_{min} and r_{SOI} for δ_{max}.
- The resulting values for δ_{min} and δ_{max} are the boundaries in which a random turning angle is chosen. Furthermore, this value is randomly set to be either positive or negative, to randomise whether the spacecraft gains or looses energy in the heliocentric reference frame through the gravity assist.
- The calculated turning angle δ is afterwards applied into Eq. (3.3) to calculate the respective pericenter distance.
- Now the gravity-assist manoeuvre is defined in a random way and the program can proceed to further calculations.

To implement the proposed constraint this sequence is altered in a way which enables the utilisation of high-energy yielding gravity assists. As mentioned, two of the three variables (r_{per}, δ and v_∞) are sufficient to describe the whole manoeuvre. For that reason the constrained search method just uses the pericenter distance and the hyperbolic excess velocity,

as these two values are more intuitively usable than the turning angle. Furthermore, the sequence of the calculations is adjusted and simplified.

At first the pericenter distance is randomly chosen, followed by the definition of the hyperbolic excess velocity. Both values are located within a specific range, which can be varied by the user, see below. After defining these two variables, the gravity-assist manoeuvre is completely described and the remaining parameters (i.e. the turning angle, the Δv gain etc.) can be calculated based on these two values and the program can proceed.

To enable the user to specify in what extent the constraint shall be able to change the behaviour of the old search method, two input values are defined. The first one is called d_{peri} and describes in the range of 0 % to 100 % where the pericenter of the fly-by could be set. 0 % means in this context that the pericenter is always at the minimal possible distance r_{min}, limited by body surface, atmosphere or other barriers. 100 % means, that the pericenter is randomly placed between r_{min} and the sphere of influence r_{SOI}, thus matching the currently implemented method.

The second value is called d_{velo} and describes the range around the extreme value of the hyperbolic excess velocity ($v_{\infty,ex}$, derived in Chapter 1.2.1) in which the actual velocity can be randomly chosen. d_{velo} can also obtain values between 0 % and 100 %. 0 % means in this case that the velocity is always set to the extreme value given in Eq. (1.6). 100 % on the other hand widens the range to $v_{\infty,ex} \pm v_{\infty,ex}$. Accordingly, the velocity will be randomly set to be between 50 m/s and $2 \cdot v_{\infty,ex}$.

50 m/s acts as the lower border because smaller velocities, and most of all 0 m/s, do not make very much sense, as the possible gained energy gets negligible at this point (see Chapter 1.2.1). The mathematical expressions which describe the calculation of the values for r_{per} and v_∞ are provided in the Eq. (3.4) and (3.5).

$$r_{min} \leq \quad r_{per} \quad \leq r_{min} + d_{peri} \cdot (r_{SOI} - r_{min}) \qquad d_{peri} \in \{0, 1\} \tag{3.4}$$

$$v_{\infty,ex} \cdot (1 - d_{velo}) \leq \quad v_\infty \quad \leq v_{\infty,ex} \cdot (1 + d_{velo}) \qquad d_{velo} \in \{0, 1\} \tag{3.5}$$

The following steps are executed in the modified sequence, in place of the instructions listed above, also see Figure 3.2:

- Use the parameter d_{peri} to choose a new pericenter distance r_{per} based on Eq. (3.4).
- Calculate the extreme value of the hyperbolic excess velocity $v_{\infty,ex}$, Eq. (1.6), to gain knowledge about the maximum amount of possible Δv gain.
- Utilise Eq. (3.5) to choose the actual hyperbolic excess velocity which deviates from the extreme value in the way specified by the parameter d_{velo}. If the velocity is below 50 m/s: set it to 50 m/s.
- Turn the incoming velocity vector in a random direction, similar to the step in the unmodified method. Through this step, the angle α is randomly determined, thus specifying whether energy will be gained or lost in the heliocentric reference frame.

- The gravity assist is now completely described and the resulting Δv gain and turning angle δ can be calculated before the program proceeds.

As an example how the search space for gravity assists can be limited by this constraint, Figure 3.2 shows the same graph as in Chapter 1.2.1 with the addition of the restrictions on the variables r_{per} and v_∞. In this example the values for d_{peri} and d_{velo} are set to be 10 % and 20 %. This means that the pericenter distance can be set between 6,500 km and 91,850 km ($6{,}500\,\text{km} + 0.1 \cdot (925{,}000\,\text{km} - 6{,}500\,\text{km}) = 91{,}850\,\text{km}$, Eq. (3.4)). The two lines (derived from Eq. (3.5)) depicting the upper and lower bound of the velocity restriction are also plotted and marked as $v_{\infty,max}$ and $v_{\infty,min}$. Thus, the green color depicts the area in which the parameters are allowed to be set by the optimiser.

It can be seen that by using these parameter settings (10 % for d_{peri} and 20 % for d_{velo}) the area, in which the variables of the gravity assist are possibly located, gets significantly smaller. If this reduction has any positive effects on the resulting trajectories will be tested and analysed in the following chapters.

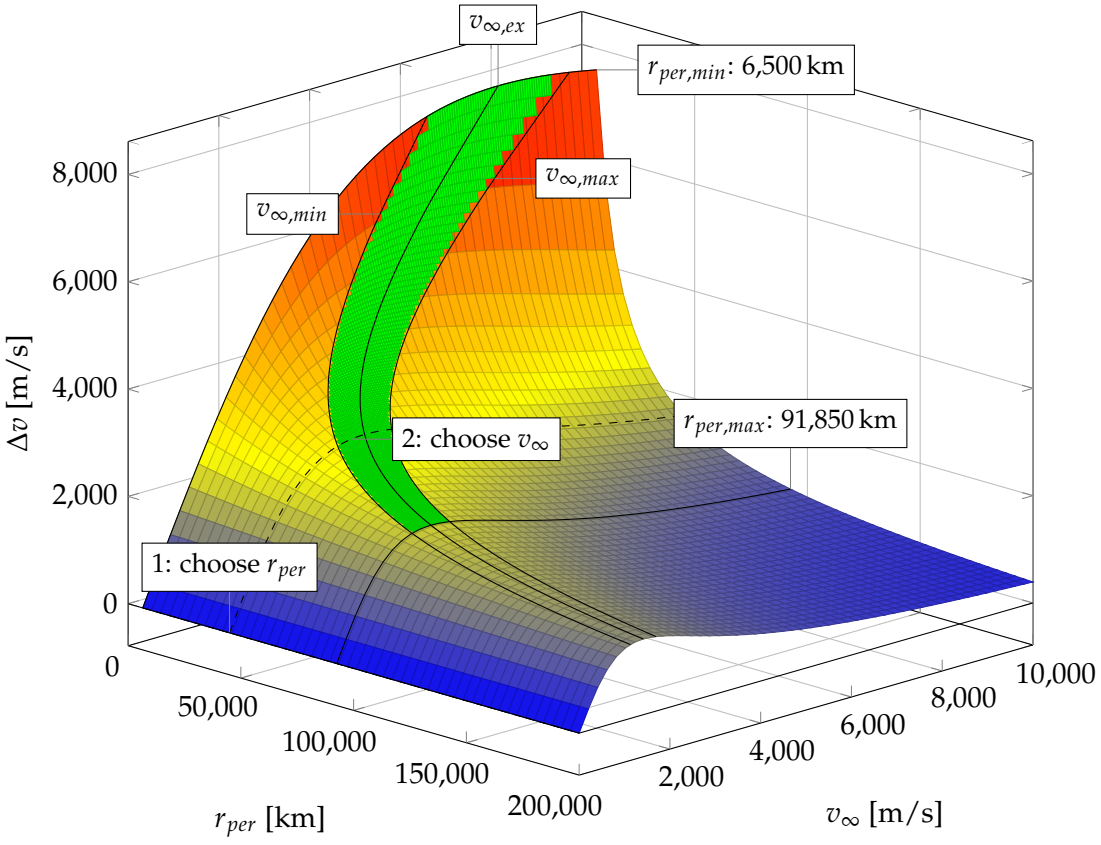

FIGURE 3.2: *Possible Δv gain of a gravity assist at the Earth by using the constraint (derived from Figure 1.1). The restricted ranges of r_{per} (r_{min} to r_{max}) and v_∞ ($v_{\infty,min}$ to $v_{\infty,max}$) are listed in the graph and marked as black lines. The green color described the area in which the optimiser is allowed to set the gravity-assist variables. In this case the parameter d_{peri} is set to 10 % and d_{velo} to 20 %. The search method first chooses r_{per} (marked as black dashed line) and sets v_∞ afterwards.*

3.2.2 Implementation: Partner Pool

The idea of the Partner Pool (*PP*) constraint is to restrict the possible gravity-assist partners, to prevent the investigation of sequences with are most likely disadvantageous, like Earth – Mercury – Neptune – Venus – Saturn, to name an arbitrarily selected extreme example.

The optimiser chooses a new gravity-assist partner in accordance to Eq. (3.6) in its unmodified, current state. The eight planets are numbered from 1 to 8 – 1 being Mercury and 8 Neptune. The variables $GA_{Partner,new}$ and $GA_{Partner,current}$ are consequently integer values between 1 and 8. In the calculation, the ID of the current planet is subtracted by 1 and an random value (X) between 0 and 3 is added.

$$GA_{Partner,new} = (GA_{Partner,current} - 1) + X \qquad X \in \{0, 1, 2, 3\} \tag{3.6}$$

The resulting behaviour is that the optimiser can choose gravity-assist partners in a fixed range around the current planet. The range can be characterised as follows: starting from the current gravity-assist partner (or the body at which the trajectory origins), the new gravity-assist partner can be either one step away towards the center of the solar system or two steps away in an outwards direction. For example, Eq. (3.7), if the current gravity-assist partner is the Earth (ID 3), the new partner ID can be set to 2, 3, 4 or 5. Respectively, Venus, Earth, Mars or Jupiter.

$$\begin{aligned} & GA_{Partner,new} = (3 - 1) + X \qquad X \in \{0, 1, 2, 3\} \\ \Rightarrow\ & GA_{Partner,new} \in \{2, 3, 4, 5\} \end{aligned} \tag{3.7}$$

This approach is obviously very restrictive and may be sufficient for trajectories which do not involve planets which are wide apart. But it is important, that the user is able to specify in which range around the current planet the next gravity-assist partner can be selected. To accomplish this, two new input values are defined, *forwards* and *backwards*. These two variables define how many "jumps" are allowed in which flight direction of the mission. A jump is in this context defined as the selection of a gravity-assist partner planet which is in direct neighbourhood to the current planet (like Earth – Mars or Saturn – Jupiter).

The parameter *forwards* defines the number of jumps in alignment with the mission direction and *backwards* the number against the direction of flight. Thus, if the overall mission trajectory is outbound in the solar system (like a mission from Earth to Saturn), *forwards* limits the number of jumps in the outbound direction, whereas *backwards* restricts the number of jumps in the inbound direction. For a flight direction towards the center of the solar system this is the other way around. The values can differ between 0 and 7, as the eight main planets of the solar system are supported currently. The mission flight direction does not need to be specifically provided by the user, as it is calculated based on the trajectory's start and end body.

The modified calculations of the new partner IDs are provided in Eq. (3.8) and 3.9.

$$direction = outbound:$$
$$GA_{Partner,new} = (GA_{Partner,current} - backwards) + X \qquad X \in \{0, \ldots, forwards + backwards\} \tag{3.8}$$

$$direction = inbound:$$
$$GA_{Partner,new} = (GA_{Partner,current} - forwards) + X \qquad X \in \{0, \ldots, forwards + backwards\} \tag{3.9}$$

The calculations are slightly different for the respective mission directions. For the case of an outbound mission directory, the current planet ID is subtracted by the *backwards* parameter and a random number between 0 and $forwards + backwards$ is added. The term in brackets $(GA_{Partner,current} - backwards)$ acts as the lower limit of the possible gravity-assist partners, because by adding the random number X, the new ID can only get bigger or stay the same. The random number lies in an interval between 0 and $forwards + backwards$, as the addition of the two parameters provides the whole range of allowed possible next gravity-assist partners.

E.g. for a mission trajectory from Earth to Saturn (outbound direction), with the parameters *forwards* and *backwards* set to 3 and 2, the modified method calculates the gravity-assist partner IDs shown in Eq. (3.10) as possible candidates for the first manoeuvre.
(current partner ID: Earth = 3, $forwards + backwards = 5$)

$$\begin{aligned} & GA_{Partner,new} = (3-2) + X \qquad X \in \{0,1,2,3,4,5\} \\ \Rightarrow\ & GA_{Partner,new} \in \{1,2,3,4,5,6\} \end{aligned} \tag{3.10}$$

It can be seen that by using relatively high numbers for the *forwards* and *backwards* parameters, the first gravity-assist manoeuvre is allowed to happen at nearly every planet in the solar system (Mercury to Saturn).

If the mission direction is towards the center of the solar system, the lower limit of possible next partners is determined by subtracting the *forwards* parameter from $GA_{Partner,current}$, Eq. (3.9). As an example a mission from Earth to Mercury is regarded and the parameters *forwards* and *backwards* are set to 1 and 0. By doing this, a backwards propagation is prevented. The calculation of the first possible gravity-assist partner is shown in Eq. (3.11). In this case at least one gravity assist at Venus has to be conducted before the spacecraft is able to reach Mercury.
(current partner ID: Earth = 3, $forwards + backwards = 1$)

$$\begin{aligned} & GA_{Partner,new} = (3-1) + X \qquad X \in \{0,1\} \\ \Rightarrow\ & GA_{Partner,new} \in \{2,3\} \end{aligned} \tag{3.11}$$

These calculations have to be incorporated into two different locations inside the program.

The first location is where a new gravity-assist partner is selected, after execution of the previous fly-by, or at the beginning of the mission. The other location, where this constraint needs to be implemented, is within the differential evolution algorithm. During the mutation step, the gravity-assist partner of the new mutation vector is assembled by application of the mutation equation (see Chapter 2.3.1, Eq. (2.7)). To implement the constraint, the normal behaviour of the differential evolution has to be changed to prevent using gravity-assist partners which are outside of the scope defined by the user. Thus, the calculations shown in Eq. (3.8) and (3.9) have to be incorporated into the mutation mechanism. This leads to a differing behaviour of the mutation in contrast to the original algorithm. As this deviance is just in respect to the selection of a new gravity-assist partner, the rest of the DE algorithm is not affected by this change.

By introduction of the two parameters, the possible gravity-assist sequences can be adjusted by the user according to the respective mission profile. If this approach has any positive effects on the resulting trajectories will be tested and analysed, in combination with the first constraint, within the next chapters.

3.2.3 Remarks About the Source Code

The implemented, modified source code, which incorporates the constraints into the search method, is not provided within this thesis for several reasons. The whole program is composed of more than 6,000 lines of code and most of the changes performed in this context were of administrative nature. For example, implementing the new input values and additional calculations (like the flight direction of the mission), which are executed in several different locations within the source code. The new procedures that calculate the constrained parameters of the gravity-assist manoeuvres are thoroughly explained in Chapter 3.2. These explanations are sufficient to conceive the modifications integrated into the search method. Additionally supplying the source code would not be practical without providing the context of the remaining program. This would involve an extensive illustration of the details which would exceed the confines of this thesis.

3.3 Data Acquisition

In order to determine the influence of the proposed and implemented constraints on the resulting trajectories, a series of tests is carried out. The goals of these tests are, on one hand, to identify the magnitude of influence exerted by the usage of the constraints, and on the other hand, to define which set of parameters enable the best results. The most important value that represents the quality of the result is the required Δv for a calculated trajectory. Further results, which are investigated, are for example the encounter date and the partner planets of the gravity-assist manoeuvres.

For a better comparability of the results with the work done by Maiwald [1], the following, similar search method settings are chosen. The trajectory begins at Earth and ends at Jupiter and the launch date is set to be 56,000 MJD, which is equal to the 14th of March in 2012. Furthermore, the flight time is defined to not exceed 3,000 days and the launch window is either 100 days for Phase 1 or 360 days for Phases 2 and 3. The settings for Phase 2 and 3 are exactly the same as used by Maiwald, so these results can be directly compared. The

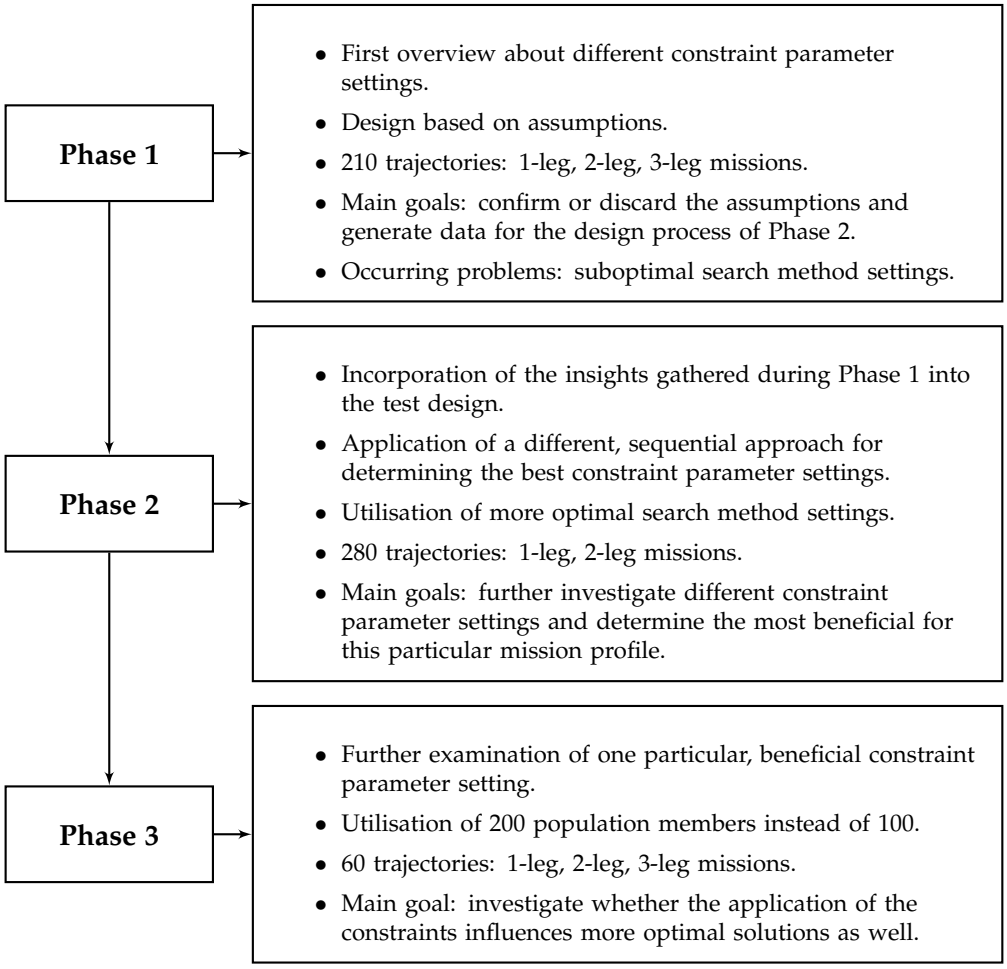

FIGURE 3.3: *Overview about the three testing phases with the respective main characteristics and goals.*

TABLE 3.1: *Constraint parameters which are altered during the test runs.*

Parameter	**Abbreviation**	**Values**
Application of Δv-gain constraint	Δv *gain*	Yes / No
Pericenter distance in the range between r_{min} and r_{SOI}, in %	d_{peri}	0 to 100
Range around the extreme value $v_{\infty,ex}$ in which the actual hyperbolic excess velocity v_∞ lies, in %	d_{velo}	0 to 100
Application of the *PP* constraint	*PP*	Yes / No
Maximum jumps in the direction of flight	*forwards*	0 to 7
Maximum jumps against the direction of flight	*backwards*	0 to 7

differences and the aim of Phase 1 are discussed further in Chapter 3.3.1.

To account for the random variations in the quality of the results, and to collect statistically significant data, each set of parameters is calculated 10 (Phase 1 and 2) or 30 (Phase 3) times. Furthermore, each testing phase consists of a control group in which no constraints are applied. This control group is used as a benchmark to provide comparable results for the influence, the application of the constraints entails. The parameters, which characterise the behaviour of the constraints, and which are altered during the different test runs, are listed in Table 3.1. Figure 3.3 contains a graphical overview about the three testing phases' goals and properties.

3.3.1 Phase 1

The aim of Phase 1 is to gain an overview concerning the expectable benefits the application of the constraints implicates. The starting conditions of this phase are a little bit different from Phases 2 and 3 which is why the resulting Δv values cannot be directly compared. However, a direct comparison between these phases is not intended. Instead, the target is to gain knowledge about the most beneficial parameter sets, and to investigate whether these benefits are applicable for 2-leg and 3-leg missions alike. It is examined if parameter sets which produce good results for 2-leg missions are equally advantageous for 3-leg missions. The main settings of Phase 1 are listed in Table 3.2. The complete settings file is specified in Appendix A.

In Phase 1, other than Phases 2 and 3, there are values set for $v_{\infty,start}$ and $v_{\infty,end}$. This means that, at the beginning of the mission, the spacecraft is on a random flight path leading away from Earth with a hyperbolic excess velocity v_∞ of 1,000 m/s. In the same way, the encounter at Jupiter is considered complete, as soon as a trajectory is found which ends with a v_∞ of 50 m/s. The maximum number of gravity-assist manoeuvres is set to 2. The search method will thus produce trajectories with one, two and three legs, respectively zero, one and two gravity assists. To limit the time the computation takes to finish, the evolutionary optimisation algorithm is set to a population size of 100 and a maximum number of 200

TABLE 3.2: *Starting conditions of Phase 1.*

Parameter	**Value**
Start Body	Earth
End Body	Jupiter
Launch Date	56,000 MJD
Launch Window	100 days
Maximum Flight Time	3,000 days
Minimum Flight Time	1,000 days
Maximum Revolutions	3
$v_{\infty,start}$	1,000 m/s
$v_{\infty,end}$	50 m/s
Maximum Number of Gravity Assists	2
Population Size	100
Maximum Number of Generations	200

generations. With these settings one trajectory computation takes around four hours on the testing computer specified in Appendix B.

Assumptions

As no information is available, how the search method responds to varying constraint parameter settings, the design process of the first phase has to be based on two assumptions. Each assumption concerns the basic implications of applying one of the constraints. See the following list and Figure 3.4:

- It is assumed that changes in the values of d_{peri} and d_{velo} between 0 % and 10 % have a great impact on the quality of the result. The parameters are set 1 %, 2 %, 5 % and 10 % to examine this assumption and evaluate whether this influence is positive or negative. Consequently, it is assumed that values greater than 10 % are leading to similar results like the unchanged version, thus eliminating the purpose of the Δv-gain constraint. Both parameters are set to 50 % to investigate this argumentation.
- The *PP* constraint is assumed to have an advantageous impact on the quality of the results, only if the allowed jumps in both directions of flight are limited to values smaller and equal to 2. The constraint is meant to decrease the size of the search space, thus allowing gravity assists to happen at planets far away from the target Jupiter, is assumed to be a hindrance.

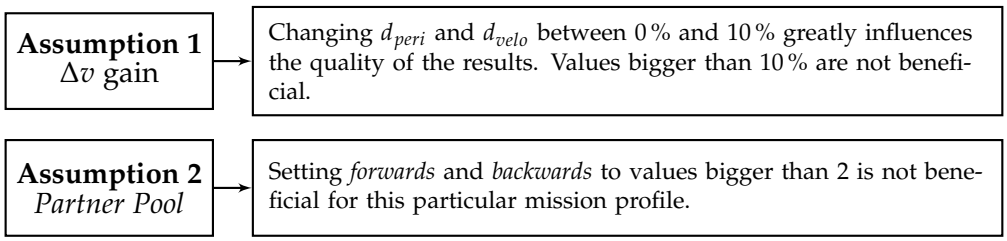

FIGURE 3.4: *Overview about the two assumptions made prior to Phase 1.*

TABLE 3.3: *Trial plan Phase 1.*

Number	Δv ***gain***	d_{peri}	d_{velo}	***PP***	***forwards***	***backwards***
1	no	-	-	no	-	-
2	yes	0	0	no	-	-
3	yes	1	0	no	-	-
4	yes	2	0	no	-	-
5	yes	5	0	no	-	-
6	yes	10	0	no	-	-
7	yes	50	0	no	-	-
8	yes	0	1	no	-	-
9	yes	0	2	no	-	-
10	yes	0	5	no	-	-
11	yes	0	10	no	-	-
12	yes	0	50	no	-	-
13	yes	1	1	no	-	-
14	yes	2	2	no	-	-
15	yes	5	5	no	-	-
16	yes	10	10	no	-	-
17	yes	50	1	no	-	-
18	no	-	-	yes	1	1
19	no	-	-	yes	2	1
20	no	-	-	yes	2	2
21	yes	50	1	yes	1	1

Trial Plan

Phase 1 consists of 21 different parameter sets, whereby every set is computed 10 times. This leads to a total amount of 210 computations with each one producing a 1-leg, a 2-leg and a 3-leg trajectory. Table 3.3 shows the complete trial plan for this phase.

The first run, without any constraints in use, acts as a benchmark to measure the impact the different constraint parameter sets have. The runs 2 to 17 examine the different settings of the Δv-gain constraint. The first of these runs investigates what happens when the optimiser is forced to maximise the Δv which can be gained through a gravity-assist manoeuvre. To achieve this, the parameters d_{peri} and d_{velo} are both set to 0 %. Accordingly, the pericenter distance is always set to be the lowest possible distance r_{min} and v_∞ is set to be the respective extreme value specified by Eq. (1.6).

The next five runs are examining different settings for the pericenter distance part of the constraint. In these runs the pericenter distance is altered between 1 % and 50 %, whereas d_{velo} is always set to 0 %. This means that for a randomly chosen pericenter distance the hyperbolic excess velocity is always set to the extreme value. The next five runs are intended to investigate the impact of changing d_{velo}. Therefore the pericenter distance is always set to the minimum possible value (0 %) and d_{velo} is altered between 1 % and 50 %.

The next five runs combine these two parts of the Δv-gain constraint. The parameters are consecutively increased from 1 % to 10 % and the trial number 17 utilises the parameters of

the best results from the previous runs, which are the runs with the numbers 7 and 8 (see Chapter 4).

The runs 18 to 20 are investigating three different settings concerning the *PP* constraint. Assuming, it is not beneficial for an Earth - Jupiter mission to allow the execution of gravity-assist manoeuvres at planets which are far away from these two bodies (like Neptune for example), the possible gravity-assist partners are limited to either *forwards:1* and *backwards:1* (18), *forwards:2* and *backwards:1* (19), or *forwards:2* and *backwards:2* (20). During these three trial runs the Δv-gain constraint is not activated.

The last trial run deploys the best parameter settings of the previous runs. It shall be investigated whether the application of the most beneficial parameters for the respective constraints leads to even better results. Why these settings where chosen can be seen in the presentation of the results in Chapter 4.

3.3.2 Phase 2

TABLE 3.4: *Starting conditions of Phase 2.*

Parameter	**Value**
Start Body	Earth
End Body	Jupiter
Launch Date	56,000 MJD
Launch Window	360 days
Maximum Flight Time	3,000 days
Minimum Flight Time	2,000 days
Maximum Revolutions	4
$v_{\infty,start}$	0 m/s
$v_{\infty,end}$	0 m/s
Maximum Number of Gravity Assists	1
Population Size	100
Maximum Number of Generations	1,000

The second phase of testing is, like Phase 1, designed to investigate a large range of different constraint parameter sets, to determine which are most beneficial. As stated in Chapter 3.3 the search method settings are in line with the settings used by Maiwald [1] to enable comparability. The greatest change in the settings, opposed to Phase 1, is that the values of $v_{\infty,start}$ and $v_{\infty,end}$ are set to 0 m/s. This means that there is no initial hyperbolic excess velocity defined at the beginning of the mission, and the encounter with Jupiter is finished as soon as a trajectory is found that leads to a v_∞ of 0 m/s. Furthermore, the maximum number of generations is increased to 1,000, because the 200 generations used in the computation of Phase 1 were found to not be sufficient to always find the best solution (see Chapter 5.1.2). Table 3.4 shows the starting parameters of Phase 2. The complete settings file is specified in the Appendix A.

TABLE 3.5: *Trial plan Phase 2.*

Number	Δv ***gain***	d_{peri}	d_{velo}	***PP***	***forwards***	***backwards***
1	no	-	-	no	-	-
2	yes	0	0	no	-	-
3	yes	10	0	no	-	-
4	yes	20	0	no	-	-
5	yes	30	0	no	-	-
6	yes	40	0	no	-	-
7	yes	50	0	no	-	-
8	yes	60	0	no	-	-
9	yes	70	0	no	-	-
10	yes	80	0	no	-	-
11	yes	90	0	no	-	-
12	yes	100	0	no	-	-
13	yes	20	10	no	-	-
14	yes	20	20	no	-	-
15	yes	20	30	no	-	-
16	yes	20	40	no	-	-
17	yes	20	50	no	-	-
18	yes	20	60	no	-	-
19	yes	20	70	no	-	-
20	yes	20	80	no	-	-
21	yes	20	90	no	-	-
22	yes	20	100	no	-	-
23	yes	20	40	yes	1	1
24	yes	20	40	yes	2	1
25	yes	20	40	yes	2	2
26	no	-	-	yes	1	1
27	no	-	-	yes	2	1
28	no	-	-	yes	2	2

Trial Plan

Phase 2 consists of 28 different parameter sets, each calculated 10 times, leading to a total amount of 280 calculations. To enable a timely completion of this testing phase, and because the evaluation of Phase 1 (see Chapter 4 and 5) revealed that the quality of the results for 2-leg and 3-leg mission behave in a similar manner for a given set of constraint parameters, only 1-leg and 2-leg missions are calculated during this phase. The maximum number of gravity assists is thus set to 1, resulting in a calculation time of about two hours per solution.

One assumption made prior to the execution of Phase 1 is, that small changes (in the range from 1 % and 10 %) of the Δv-gain parameters entail great changes in the results. As later discussed within Chapter 5.1.1, this assumption is not necessarily correct.

To account for this, Phase 2 comprises a sequential and thorough investigation of the possible parameter sets for the Δv-gain constraint. Starting with setting d_{velo} to 0 % and increasing d_{peri} from 0 % to 100 % within the runs number 2 to 12. A quick pre-evaluation revealed that

the best value for d_{peri} is around 20 %. Therefore, d_{peri} is set to be 20 % for the runs 13 to 22, where d_{velo} is increased from 10 % to 100 %.

Analogous to this, the value for d_{velo} is set to be 40 % for the subsequent runs 23 to 25. These three runs evaluate the impact of the *PP* constraint, using the same values as before in Phase 1. The last three runs are using the same *PP* constraint parameters while the Δv-gain constraint is deactivated.

3.3.3 Phase 3

TABLE 3.6: *Starting conditions of Phase 3.*

Parameter	**Value**
Start Body	Earth
End Body	Jupiter
Launch Date	56,000 MJD
Launch Window	360 days
Maximum Flight Time	3,000 days
Minimum Flight Time	2,000 days
Maximum Revolutions	4
$v_{\infty,start}$	0 m/s
$v_{\infty,end}$	0 m/s
Maximum Number of Gravity Assists	2
Population Size	200
Maximum Number of Generations	1,000

In Phase 3 the same settings as in Phase 2 are utilised. The only differences are, on one hand, that the maximum amount of gravity-assist manoeuvres is set to 2, and, on the other hand, that 200 population members are used for the evolutionary optimization algorithm. Table 3.6 shows the starting parameters of Phase 3 (see Appendix A for the complete settings file).

Phase 3 is designed to further investigate one specific parameter set which was found to be advantageous during the evaluation of Phase 2. Because of this Phase 3 comprises only two different settings, one without application of the constraints (acting as a benchmark), and one utilising the constraint parameters, shown in Table 3.7. These parameters were chosen because they were found the beneficial for this particular mission profile. These findings will be later presented and discussed.

As opposed to the other two testing phases, during Phase 3 each setting is computed 30 times to further reduce the impact of statistical anomalies. This means that 60 computations are done altogether. As shown by Maiwald [1] (see Chapter 2.4.2), the number of population members directly influences the quality of the computed trajectories. Accordingly, it is assumed that calculations with 200 population members should produce better results than the calculations of Phase 2.

Furthermore, the number of computed gravity-assist manoeuvres is raised to two, to confirm the observation made in Phase 1, that 2-leg and 3-leg missions behave in similar ways

TABLE 3.7: *Trial plan Phase 3.*

Number	Δv ***gain***	d_{peri}	d_{velo}	***PP***	***forwards***	***backwards***
1	no	-	-	no	-	-
2	yes	20	20	yes	1	1

to the change of the parameter settings. The chosen settings increased the computation time to around 12 hours to 15 hours.

4 Results

The results of the three testing phases are successively presented within the following chapter to provide an objective overview about the influences, different constraint parameter settings entail.

4.1 Phase 1

TABLE 4.1: *Resulting average Δv requirements and standard deviation σ of the 2-leg and 3-leg trajectories of Phase 1. The average Δv requirement for all of the 1-leg trajectories is around 16,040 m/s.*

No.	Δv ***gain***	***PP***	**2-leg:** $\overline{\Delta v}$	m/s σ	**3-leg:** $\overline{\Delta v}$	m/s σ
1	-	-	17,779	1,506	42,451	17,722
2	0 / 0	-	18,621	1,339	51,705	13,459
3	1 / 0	-	17,604	1,590	46,616	11,723
4	2 / 0	-	17,583	2,606	46,087	10,597
5	5 / 0	-	17,168	556	39,976	14,217
6	10 / 0	-	17,097	808	40,355	10,221
7	50 / 0	-	16,627	709	35,986	12,405
8	0 / 1	-	18,597	1,658	44,874	13,288
9	0 / 2	-	19,374	1,638	59,693	20,786
10	0 / 5	-	18,618	1,842	49,249	8,124
11	0 / 10	-	19,315	2,070	55,120	26,246
12	0 / 50	-	19,174	1,350	48,369	15,724
13	1 / 1	-	20,127	2,627	49,063	14,819
14	2 / 2	-	18,125	881	46,105	6,937
15	5 / 5	-	17,626	886	36,842	8,443
16	10 / 10	-	17,877	1,602	41,210	13,055
17	50 / 1	-	16,694	585	35,529	12,092
18	-	1 / 1	17,581	575	39,386	10,520
19	-	2 / 1	18,278	2,063	41,799	22,477
20	-	2 / 2	18,502	1,922	38,847	9,718
21	50 / 1	1 / 1	16,386	406	30,209	7,578

Phase 1 consists of 21 different parameter settings which have been calculated 10 times each. To assess the performance of every particular run the average value of the 10 Δv-requirements is calculated. To supplement these average values the respective standard deviations are also determined. The resulting values are shown in Table 4.1. To provide an

overview about the utilised parameter settings per run, the first two columns represent an abridged version of Table 3.3, where the settings for the Δv gain and the *PP* constraint are depicted. Figure 4.1 (2-leg trajectories) and Figure 4.2 (3-leg trajectories) depict the graphical representation of the average values and the respective standard deviations. The red dotted line in Figure 4.1 additionally displays the required Δv of the computed 1-leg missions. The search method always found 1-leg trajectories which required approximately 16,040 m/s with minimal deviations.

The computation of the 3-leg trajectories revealed the problem that during some calculations no sufficient optimum was found. This lead to Δv requirements which were way too high to be reasonable (up to 1.8×10^6 m/s). To avoid corrupting the data, every trajectory which requires a lot more than 100,000 m/s is ignored. This means that overall 13 3-leg trajectories are not included in the evaluation of the results. The runs number 11 and 12 are the worst in this respect with two and three non-converging trajectories.

Observations

- Nearly all 2-leg trajectories and all of the 3-leg trajectories have greater Δv requirements than the trajectories without any gravity-assist manoeuvres. Altogether, just 15 (7.1 %) 2-leg trajectories were found which require less than 16,040 m/s Δv. None of these trajectories were computed during the benchmark run number 1. Instead, the runs number 3, 4, 7 and 21 each produced at least two trajectories which exceeded the 1-leg trajectory.

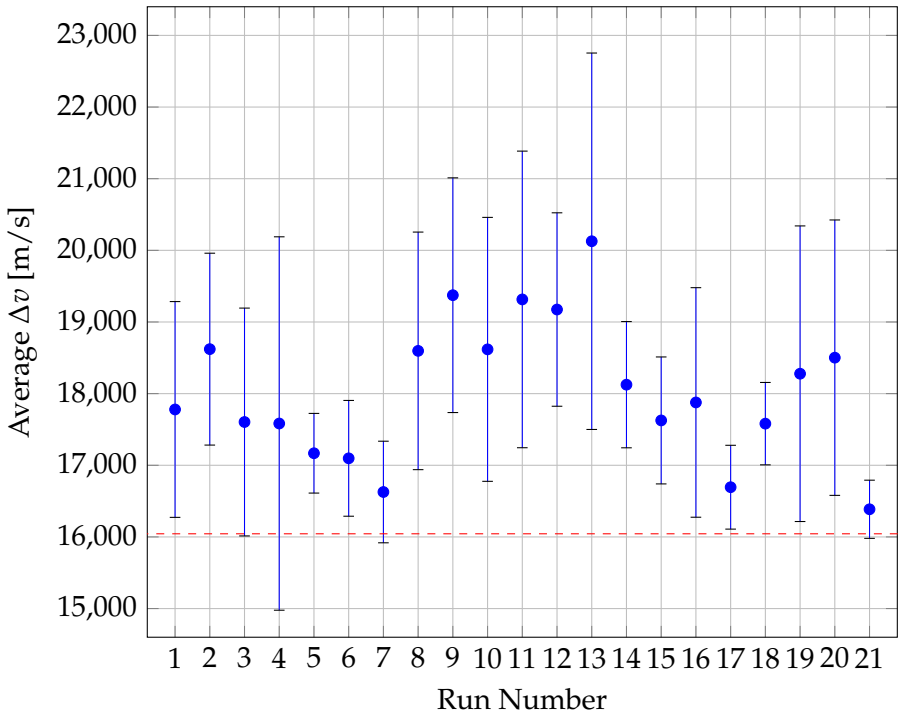

FIGURE 4.1: *Average Δv requirements for the 2-leg trajectories of the 21 runs of Phase 1. The red dotted line represents the produced 1-leg trajectories.*

- The usage of the constraints has a notably impact on the quality of the results. This influence can be positive or negative based on which constraint parameters have been chosen. As for the 2-leg trajectories, the average Δv requirement of 17,779 m/s (without constraints) could be improved up to 7.8 % (in run 21) whereby the standard deviation improved by 73 %. On the other hand, different parameter settings (e.g. run 13) worsen the average Δv by 13.2 % and the standard deviation by 74.4 %. The observation can also be made for the 3-leg trajectories. The best result (run 21) improved the average Δv requirement by 28.8 % and the standard deviation by 57.2 %. One of the worst results (run 9) produced trajectories which require 40.6 % more Δv in average, with a 17.3 % higher standard deviation.
- As for the Δv-gain constraint, increasing the first parameter (d_{peri}), which controls the possible distance of the pericenter of a gravity assist, has a beneficial influence on the Δv requirement. Run number 7 for example, which uses a value of 50 % for d_{peri}, leads to a reduction of the average Δv and the standard deviation of 6.5 % and 53 % for the 2-leg trajectory. The respective 3-leg mission shows the same behaviour with a 15.2 % and 30 % reduction of the average Δv and the standard deviation.
- The second parameter of the Δv-gain constraint (d_{velo}), which controls the hyperbolic excess velocity to maximise the gained Δv by a gravity assist, has an opposing influence on the results. The best results are achieved, when this value is 0 %. Especially the 2-leg missions show a increasing worsening of the resulting Δv requirements during

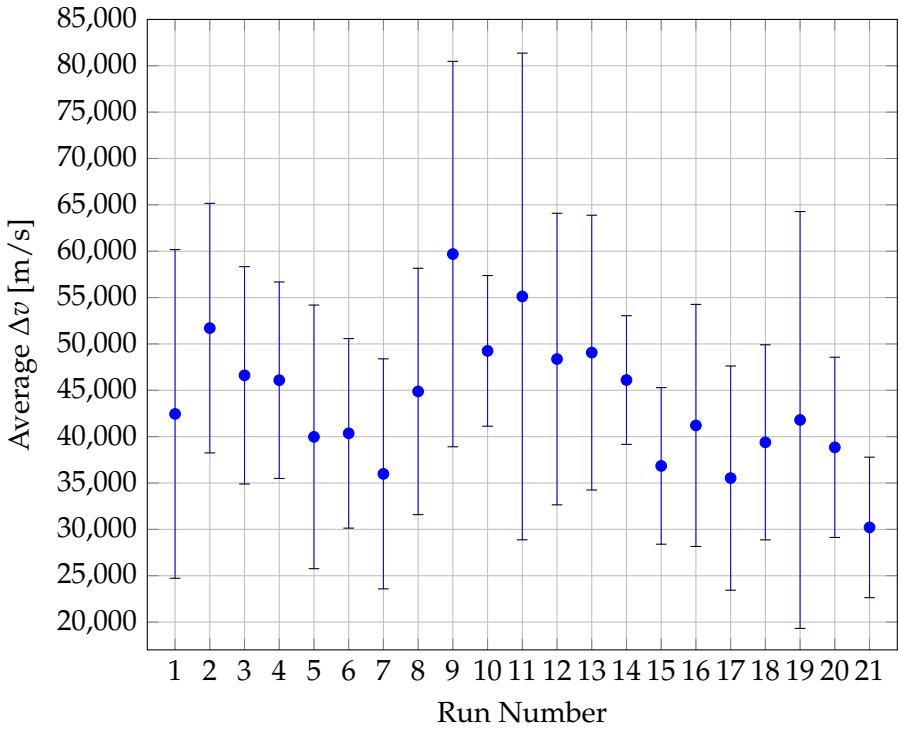

FIGURE 4.2: *Average Δv requirement for the 3-leg missions of the 21 runs of Phase 1.*

the runs 8 through 12.

- Combining the incrementation of the two parameters in the runs 13 through 15 from 1 % to 5 % improves the required Δv and the respective standard deviation. This improvement is beginning to stop at the run number 16 in which both parameters are set to 10 %. The Δv requirement and the respective standard deviation are getting higher than in the previous run for 2-leg and 3-leg trajectories alike.
- By comparing the related runs, in which the two parameters are either set to 0 % or the same value (for 1 % runs number 3, 8 and 13; for 2 % runs number 4, 9 and 14; for 5 % runs number 5, 10 and 15; for 10 % runs number 6, 11 and 16), it can be seen that the best results origin by increasing d_{peri} (runs number 3, 4, 5 and 6) and setting d_{velo} to 0 %. The worst results are in the group of the runs where d_{peri} is set to 0 % and d_{velo} is changed (8, 9, 10 and 11). The four runs where both parameters obtain values different from 0 % produce mediocre results which lie between the two others. These statements are mostly true for the 2-leg missions. The biggest exception being the 2-leg trajectories of run 13. The 3-leg missions do show a similar, but not exactly the same behaviour (e.g. runs 3, 8 and 13, where run 8 produced the best results).
- Concerning the influence of the *PP* constraint, the data shows that the setting of run 18 (*forwards:1* and *backwards:1*) yields results which are better than the benchmark (run 1). The runs number 19 and 20 perform worst than run 18. Except for the 3-leg trajectories produced during run 20, which are similar to the 3-leg trajectories of run 18.
- The combination of the best parameter sets of the previous runs produces the overall best results for 2-leg and 3-leg missions alike in run 21.

Gravity Assist Partners and Encounter Dates

To further investigate the properties of the 2-leg missions, Figure 4.3 depicts the gravity-assist partner, the encounter date and the resulting Δv requirement of all 210 calculated trajectories. The graph shows that only 33 (15.7 %) gravity-assist manoeuvres are performed at the Earth, whereas the majority uses Mars as gravity-assist partner. Further, each planet seems to have two main groups of encounter dates. As for Earth these two groups are located around 56,125 MJD and around 56,470 MJD. These two groups of encounters are roughly one year apart from another.

The only obvious exception to this is a gravity assist happening at 56,671 MJD. Also, there are four gravity assists happening at around 56,250 MJD which might indicate a third group of encounter dates at Earth. The second group of gravity assists at the Earth (at 56,470 MJD) produces the best results of the whole testing phase, but very few trajectories were found which utilise this particular fly-by opportunity. The seven of these trajectories which require less than 16,040 m/s were found during the runs 3, 4, 10 and 19.

The two groups of encounter dates with Mars as gravity-assist partner are more widespread than the gravity assists at Earth. The first and most commonly used group is between 56,600 MJD and 56,800 MJD and the second group is around 56,950 MJD. Better results are

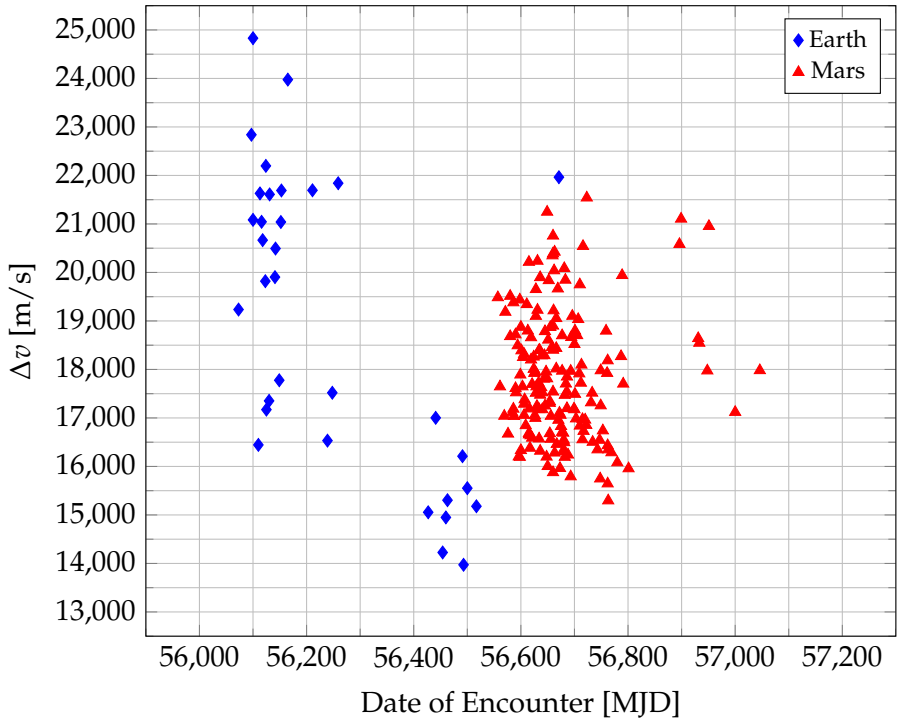

FIGURE 4.3: *Date of encounter and gravity-assist partner, with the respective Δv requirement, for the 2-leg missions of Phase 1.*

produced by the first group of the Mars encounters. But none of these trajectories can compete with the best results from second group of Earth gravity assists and only a few exceed the 1-leg trajectory benchmark of 16,040 m/s.

Trajectory Examples

The Figures 4.4, 4.5, 4.6 and 4.7 depict four trajectory examples which were calculated during Phase 1. Figure 4.4 illustrates a 1-leg trajectory. It can be seen how the spacecraft travels on a helical shape from Earth to Jupiter, while completing two revolutions around the barycentre. In general, all of the 1-leg solutions are very similar to the displayed, only varying in a small range due to the hyperbolic excess velocities at start and finish.

Figure 4.5 shows the best 2-leg trajectory of Phase 1 with a Δv requirement of 13,971 m/s. The spacecraft follows the Earth on its orbital path around the Sun for more than a year (421 days), before conducting a gravity assist at Earth and proceeding towards Jupiter. However, most 2-leg trajectories of Phase 1 use Mars as gravity-assist partner. Figure 4.6 thus depicts the best trajectory found which utilises a fly-by at Mars. This trajectory requires 15,295 m/s Δv and it can be seen how the spacecraft travels for two years away from Earth's orbital path towards the encounter with Mars on a spiral-shaped trajectory.

The best 3-leg solution of Phase 1 is displayed in Figure 4.7. This particular trajectory requires just 18,757 m/s Δv and with that only around 1,000 m/s more than the overall best 3-leg solution found during Phase 3 (see 4.3). The first gravity assist is conducted at Earth 144 days after departure and the second one at Mars over 600 days later.

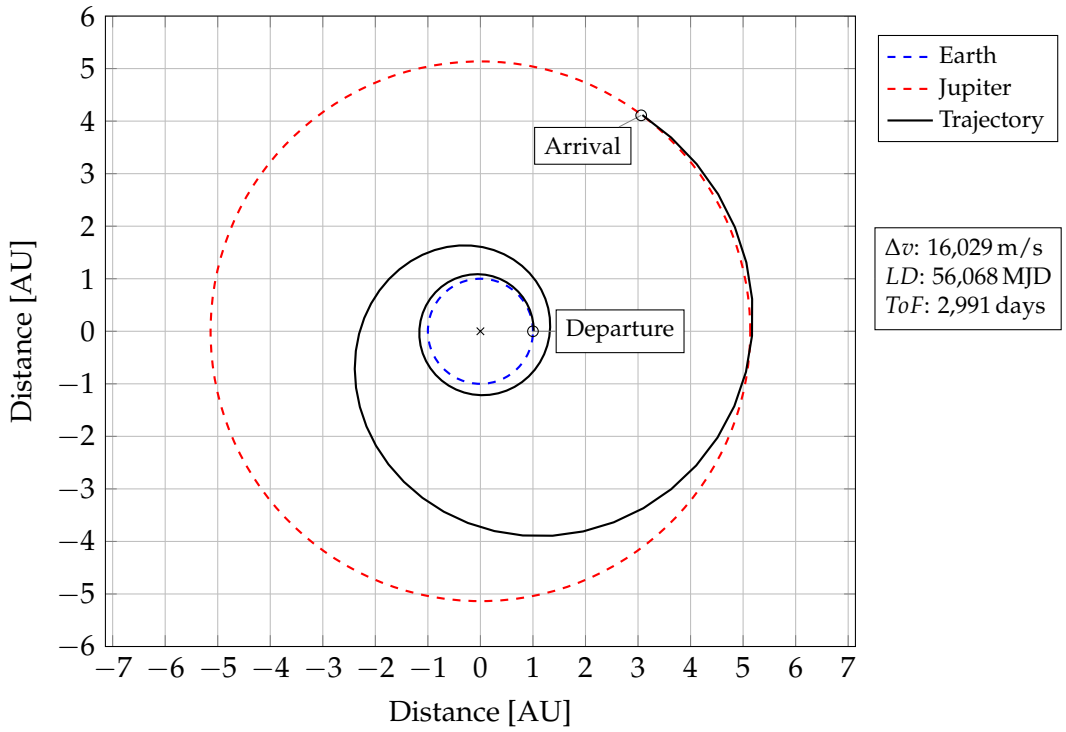

FIGURE 4.4: *1-leg trajectory calculated during Phase 1. The Δv requirement, the launch date (LD) and the flight time (ToF) are listed on the right side.*

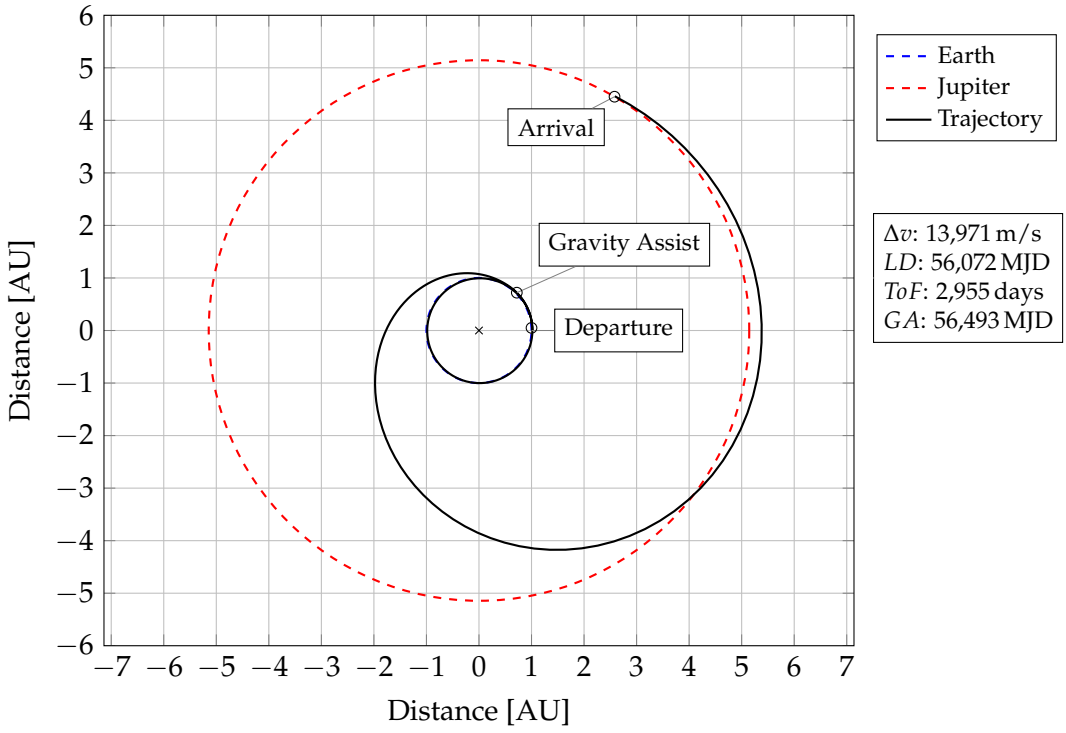

FIGURE 4.5: *Best 2-leg trajectory produced during Phase 1. The date of the encounter at Earth (GA) is additionally listed.*

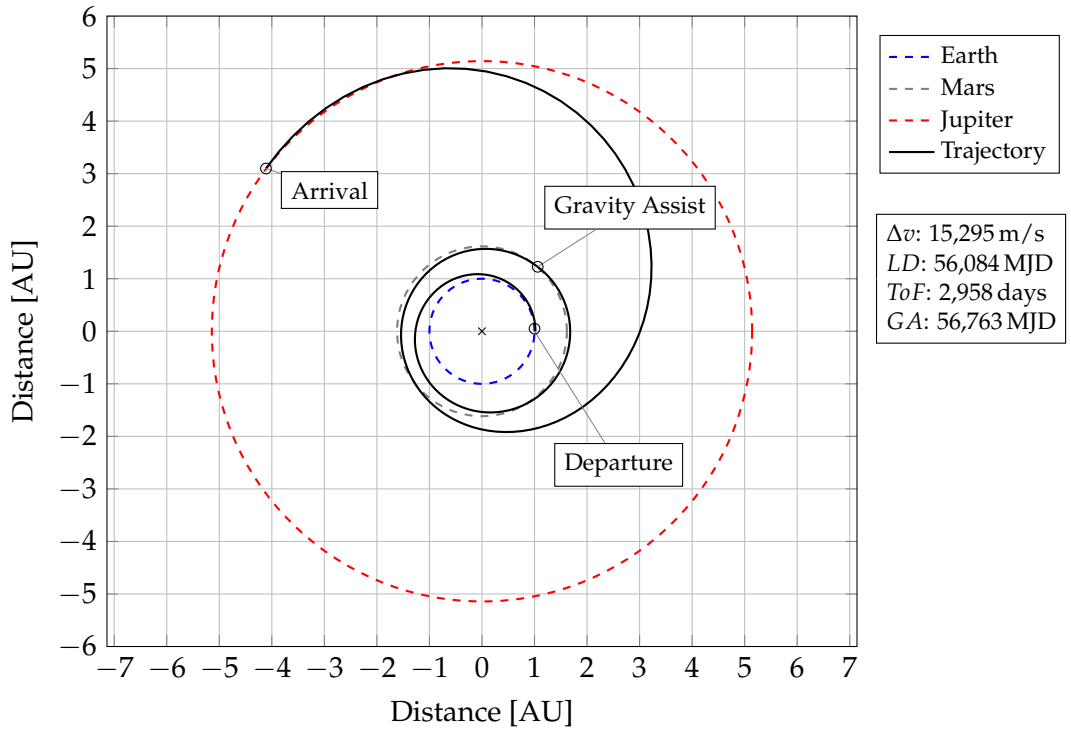

FIGURE 4.6: *Best 2-leg trajectory calculated during Phase 1 which incorporates a gravity assist at Mars.*

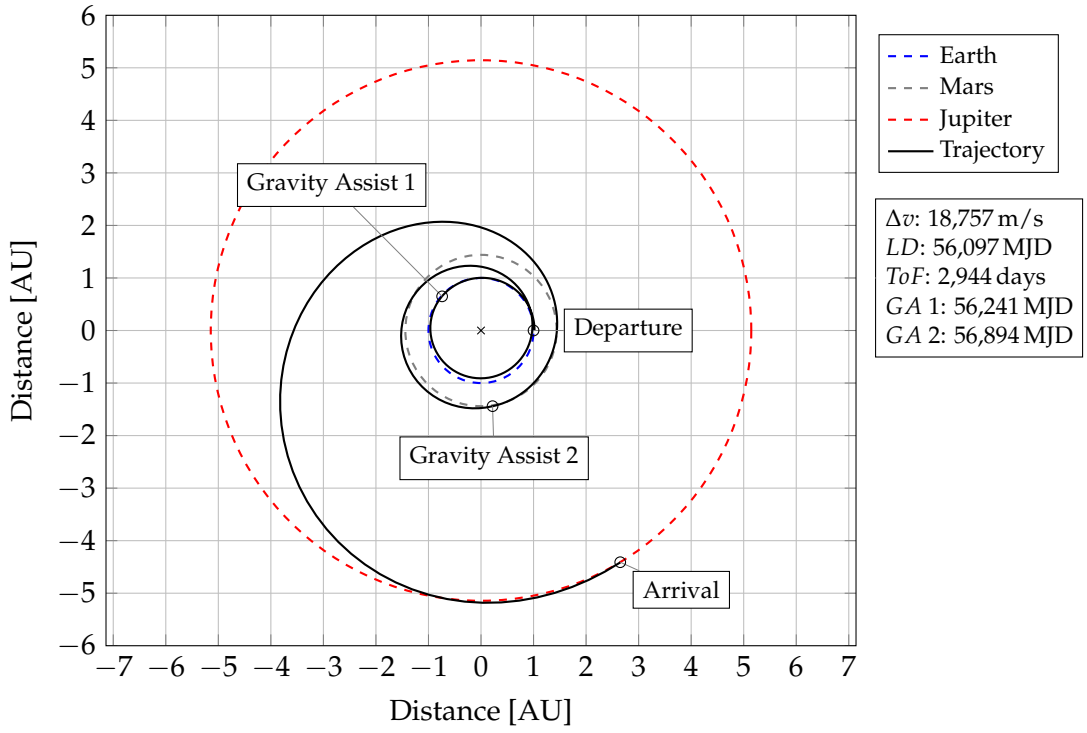

FIGURE 4.7: *Best 3-leg trajectory produced during Phase 1. The dates of the first (GA 1) and second (GA 2) gravity assist are listed as well.*

4.2 Phase 2

TABLE 4.2: *Resulting average Δv requirements and standard deviation σ of the 2-leg trajectories of Phase 2. The Δv requirement for all of the 1-leg trajectories is exactly 16,034 m/s.*

No.	Δv ***gain***	***PP***	**2-leg:** $\overline{\Delta v}$	m/s σ	**No.**	Δv ***gain***	***PP***	**2-leg:** $\overline{\Delta v}$	m/s σ
1	-	-	15,125	1,000	15	20 / 30	-	14,852	681
2	0 / 0	-	15,870	727	16	20 / 40	-	14,787	448
3	10 / 0	-	14,720	1,217	17	20 / 50	-	14,813	807
4	20 / 0	-	14,935	348	18	20 / 60	-	15,086	618
5	30 / 0	-	15,072	773	19	20 / 70	-	15,169	549
6	40 / 0	-	15,277	754	20	20 / 80	-	14,546	718
7	50 / 0	-	15,032	579	21	20 / 90	-	14,865	614
8	60 / 0	-	15,092	683	22	20 / 100	-	14,963	778
9	70 / 0	-	15,007	446	23	20 / 40	1 / 1	14,217	838
10	80 / 0	-	15,284	585	24	20 / 40	2 / 1	14,920	719
11	90 / 0	-	15,303	521	25	20 / 40	2 / 2	14,786	908
12	100 / 0	-	15,632	485	26	-	1 / 1	14,612	1,087
13	20 / 10	-	15,145	849	27	-	2 / 1	15,558	957
14	20 / 20	-	14,796	879	28	-	2 / 2	14,198	1,281

Phase 2 of testing consists of 28 different parameter sets which were each calculated 10 times. Opposed to Phases 1 and 3, only 2-leg trajectories are calculated to decrease the overall computing time of the 280 single calculations of this phase. Similar to Phase 1, Table 4.2 shows the average Δv requirements plus the respective standard deviation of all runs. The graphical representation is given in Figure 4.8. The red dotted line in Figure 4.8 again shows the Δv requirement of the calculated 1-leg trajectories. In this testing phase the search method always found the exact same 1-leg trajectory which needs 16,034 m/s of Δv.

Observations

- Unlike in Phase 1, most of the 2-leg trajectories require less Δv than the 1-leg trajectory. The single best trajectory, which was found during run number 28, requires just 12,635 m/s Δv which is equal to an improvement of 21.2 %.
- The influence of the changing constraint parameters is similar, but not equal, to the observed behaviour during Phase 1. As for the Δv-gain constraint, changing the d_{peri} parameter from 0 % to 100 % (runs number 2 to 12) is revealing a curve similar to a bathtub, whereby the worst results are produced at the left and right side of the curve. Setting d_{peri} to 0 % or 100 % increases the averagely needed Δv by 4.9 % or 3.6 %. Good results with the smallest deviation are produced by setting d_{peri} to 20 %. This decreases the required Δv by 1.3 % and the standard deviation is reduced by 65.2 %. Increasing the parameter further leads to a successively increasing Δv requirement.

- Changing the second parameter of the Δv-gain constraint (d_{velo}) shows, that in this case, the parameter does not have a very great influence on the resulting trajectories. The average Δv values in the runs 13 to 22 lay within a range of around 360 m/s. Except for run 20 which requires a little bit less Δv in average. This low spread complicates the decision which parameter setting is the most beneficial. Nevertheless, for the following runs 23, 24 and 25 it was chosen to set d_{velo} to 40 %, because of the low standard deviation of this particular run.
- Utilising the *PP* constraint in the runs 23 to 25 reveals that, similar to Phase 1, the best results are produced by setting the constraint parameter to *forwards:1* and *backwards:1*. In the runs 24 and 25 trajectories were calculated which require, in average, considerably more Δv.
- The last three runs use solely the *PP* constraint. Run number 26 produced results which require 2.8 % more Δv with a 29.7 % higher standard deviation in comparison to run number 23, that uses the same *PP* constraint parameters, but in addition to using the Δv-gain constraint. Run number 27 produces even worst results, compared directly to run number 24 (4.3 % and 33.1 % enlargement of the average Δv and the standard deviation) . However, this pattern does not continue through run number 28, where some really good and some really bad solutions were calculated. Resulting in a very high standard deviation and a noticeable lower average Δv requirement (compared to run 25, the required Δv drops by 4 % and the standard deviation increases by 41.7 %).

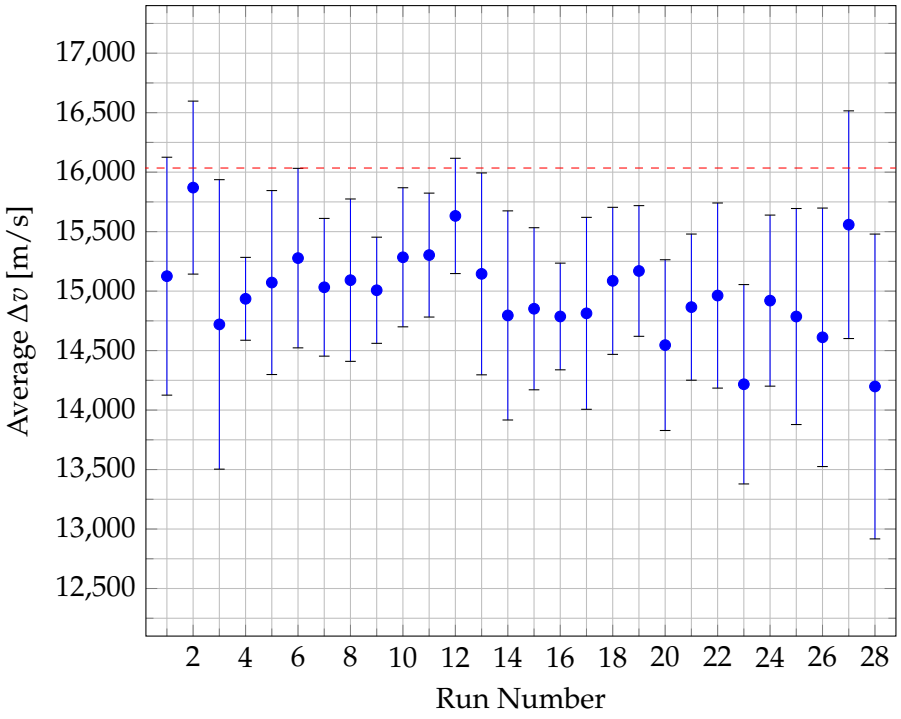

FIGURE 4.8: *Average Δv requirement for the 2-leg missions of the 28 runs of Phase 2. The red dotted line represents the produced 1-leg trajectory.*

Gravity Assist Partners and Encounter Dates

Analogous to Figure 4.3, which showed the 2-leg trajectory gravity-assist partners and encounter dates of Phase 1, depicts Figure 4.9 the same graph for Phase 2. It can be seen that only 31 (11.1 %) of the gravity assists were performed at Mars. Most of these encounters at Mars happen around 56,950 MJD and some around 56,700 MJD. These two groups of encounter dates match the two groups found during Phase 1. The difference is that the most commonly used Mars encounter dates from Phase 1 are barely used during Phase 2 and vice versa. A similar behaviour can be observed at the Earth encounters. The group of encounters which is most commonly used during Phase 2 (around 56,470 MJD), was also found during Phase 1, but only used a few times. However analogous to Phase 1, this particular gravity-assist manoeuvre enables very good results. The best of these trajectories was found during run 28.

A second large group of encounter dates at Earth is located roughly one year later around 56,850 MJD. This group of encounter dates also enables the calculation of good trajectories, but was not found during Phase 1. The best trajectory of this encounter date group, which happens at 56,880 MJD, was calculated during run 23. But similar to Phase 1, there are two groups of encounters which are close together, one around 56,125 MJD and the other around 56,250 MJD. As in Phase 1, these encounters are not enabling particularly good trajectories. Further, there are some encounters which lie between one of these main groups, but none of these gravity-assist encounters enable any good results.

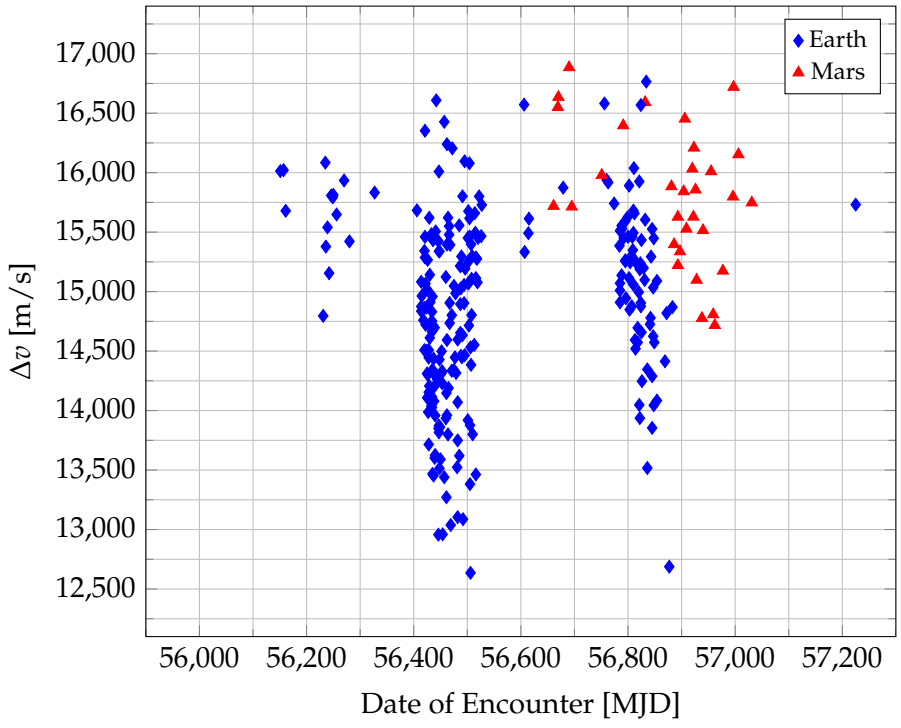

FIGURE 4.9: *Date of encounter and gravity-assist partner, with the respective Δv requirement, for the 2-leg missions of Phase 2.*

Trajectory Examples

During Phases 2 and 3 the search method produced always the same 1-leg trajectory, due to the finite search space while utilising no gravity assist. As elaborated in Chapter 2.4 and opposed to Phase 1, the search space for 1-leg trajectories is finite as long as no hyperbolic excess velocities at start and end ($v_{\infty,start}$, $v_{\infty,end}$) are defined. The calculated 1-leg trajectory is shown in Figure 2.3 in Chapter 2.1.2.

Two examples of the trajectories produced during Phase 2 are provided. The first, depicted in Figure 4.10, displays the overall best solution with a Δv requirement of 12,635 m/s. The solution incorporates a gravity assist at a similar encounter date as the best trajectory of Phase 1 (around 56,500 MJD) but a later launch date, which is why the spacecraft follows the Earth's orbital path for just 171 days before conducting the fly-by and proceeding to Jupiter. The best solution of Phase 1 entailed that the spacecraft followed the Earth for more than a year. Furthermore, it can be seen that the spacecraft's path exceeds Jupiter's orbit by around 0.5 AU before arriving at the target. This indicates that the trajectory might be further improvable as this wide arc above Jupiter's orbit should be expensive in terms of flight time and Δv.

Figure 4.11 depicts the worst 2-leg trajectory found during Phase 2. This solution incorporates a gravity assist at Mars and the flight time is 208 days shorter than for the best trajectory. The Δv requirement, on the other hand, is significantly higher at 16,884 m/s.

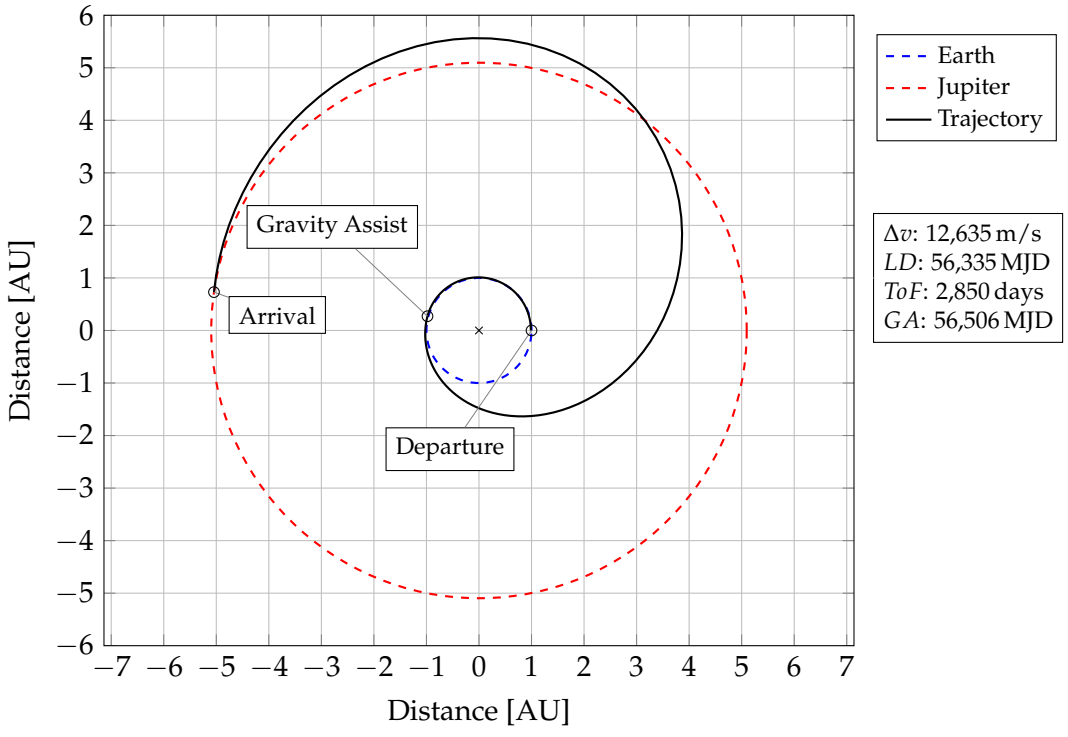

FIGURE 4.10: *Best 2-leg trajectory found during Phase 2.*

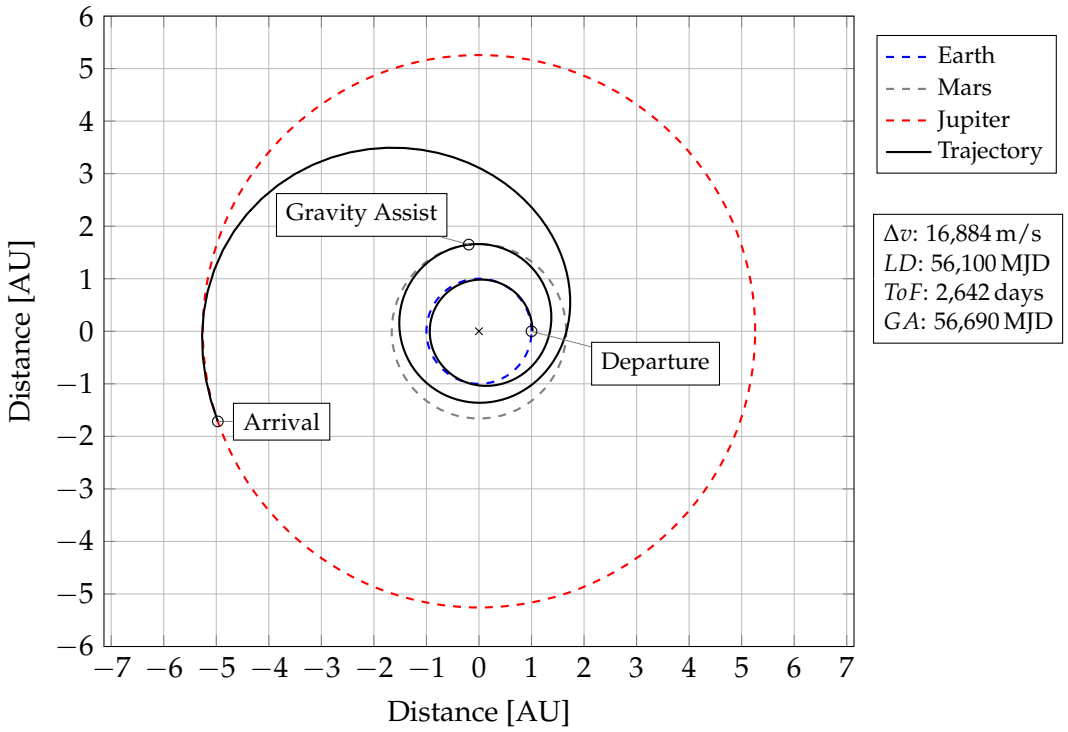

FIGURE 4.11: *Worst 2-leg trajectory solution of Phase 2.*

4.3 Phase 3

Phase 3 consists of only two different runs. One acting as a benchmark where no constraints are used and the other one using the parameters shown in Table 4.3. Furthermore, the table shows the average Δv requirements and the respective standard deviations of the calculated 2-leg and 3-leg missions.

In Phase 3, 30 calculations were conducted per parameter setting, which means that overall 60 different 2-leg and 3-leg trajectories were computed. Figure 4.12 and Figure 4.13 show, on the left hand side, the Δv requirements of all 60 single trajectories and, on the right hand side, the respective average Δv and standard deviation. The left graph is separated at run 30. The runs number 1 to 30 use the regular search method, while during the runs number 31 to 60 the constraints are applied. The red dotted line again shows the computed 1-leg trajectory which requires a Δv of 16,034 m/s.

TABLE 4.3: *Resulting average Δv requirements and standard deviation σ of the 2-leg and 3-leg trajectories of Phase 3. The average Δv requirement for all of the 1-leg trajectories is exactly 16,034 m/s.*

No.	Δv ***gain***	***PP***	**2-leg:** $\overline{\Delta v}$	m/s σ	**3-leg:** $\overline{\Delta v}$	m/s σ
1	-	-	14,100	931	34,323	8,168
2	20 / 20	1 / 1	14,071	563	26,164	5,301

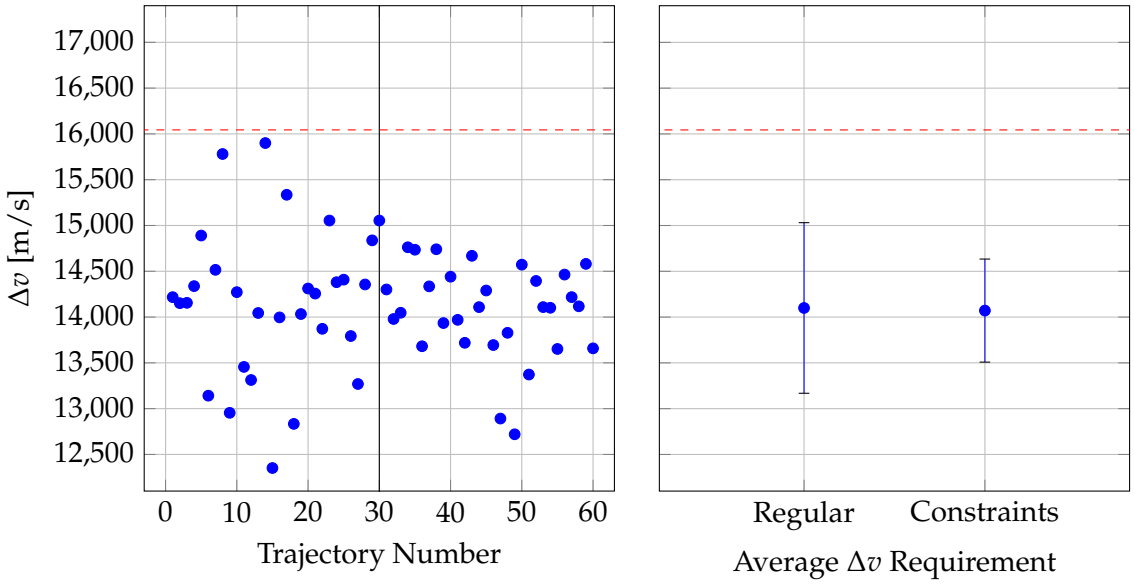

FIGURE 4.12: *Δv requirement for the 2-leg missions of Phase 3. Every single trajectory is listed on the left side and the average values with the respective standard deviation on the right side.*

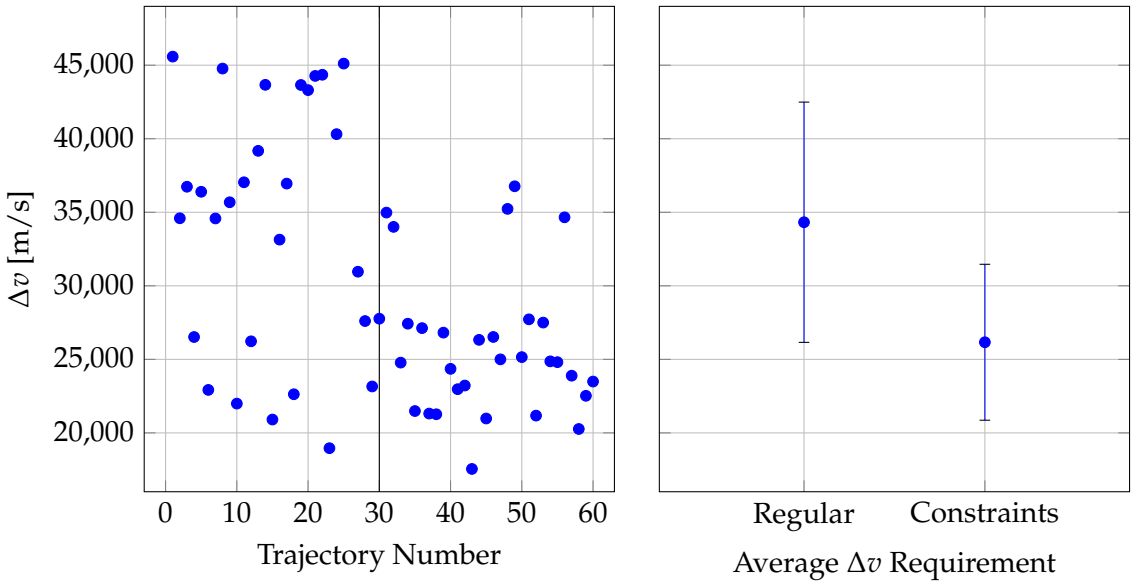

FIGURE 4.13: *Δv requirement for the 3-leg missions of Phase 3. Every single trajectory is listed on the left side and the average values with the respective standard deviation on the right side.*

Observations

- Using 200 population members for the DE algorithm reduces the overall Δv requirements of all trajectories. Other than in Phase 2, none of the 2-leg trajectories required more Δv than the 1-leg trajectory.
- As for the 2-leg trajectories, using the constraints does not reduce the averagely needed

Δv by much, but the standard deviation is reduced by 39.5 %. This can be seen very clearly in the graphs of Figure 4.12. The unconstrained search method produces results which have a relatively wide spread, ranging from around 12,000 m/s to 16,000 m/s. As the constraints get activiated, this range is reduced to values between 12,500 m/s and 15,000 m/s.

- The benefits of using constraints are even more notable for the 3-leg missions (Figure 4.13). The application of the constraints entails a 23.8 % reduction of the average Δv and a 35.1 % reduction of the respective standard deviation. One of the calculated trajectories even gets in close proximity to the 1-leg trajectory benchmark, by requiring only 17,550 m/s Δv.

Gravity Assist Partners and Encounter Dates

To further examine the gravity-assist encounters of the 2-leg trajectories, Figure 4.14 shows, analogous to Phases 1 and 2, the encounter dates and the gravity-assist partners, with respect to the resulting Δv requirements. It can be seen that similar gravity-assist encounter dates were utilised. The most commonly used group of Earth encounters is again around 56,470 MJD. Two trajectories utilise a gravity assist around 56,125 MJD (one year earlier) and a few use a gravity assist around 56,850 MJD (one year later). The best results are produced by using the encounter around 56,470 MJD (as it is the case in Phases 1 and 2). The Mars is used as a gravity-assist partner only three times. These three trajectories were computed during the benchmark run, which did not use any constraints.

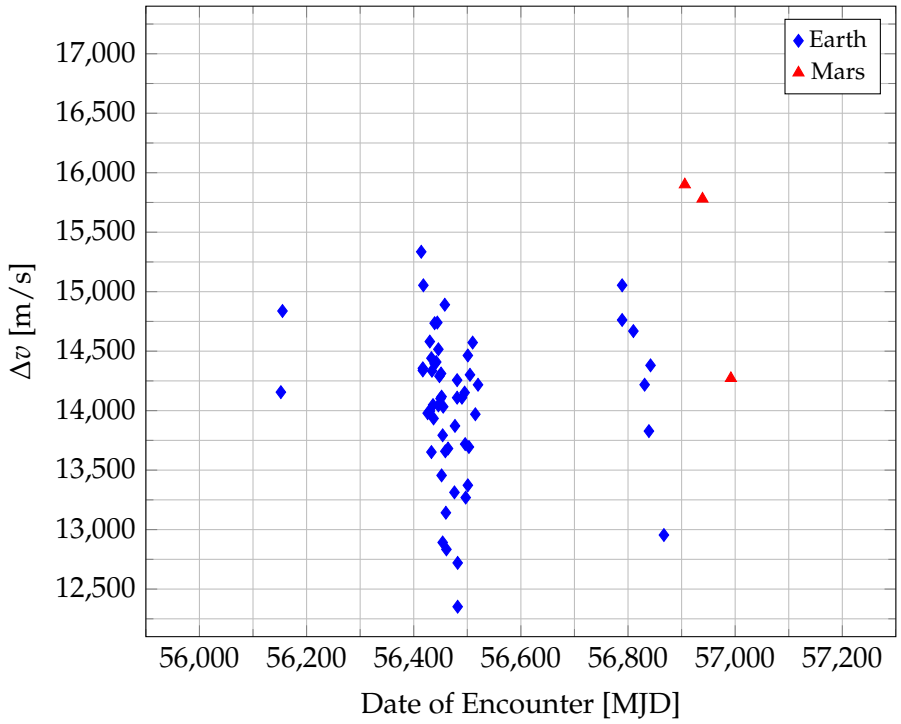

FIGURE 4.14: *Date of encounter and gravity-assist partner, with the respective Δv requirement, for the 2-leg missions of Phase 3.*

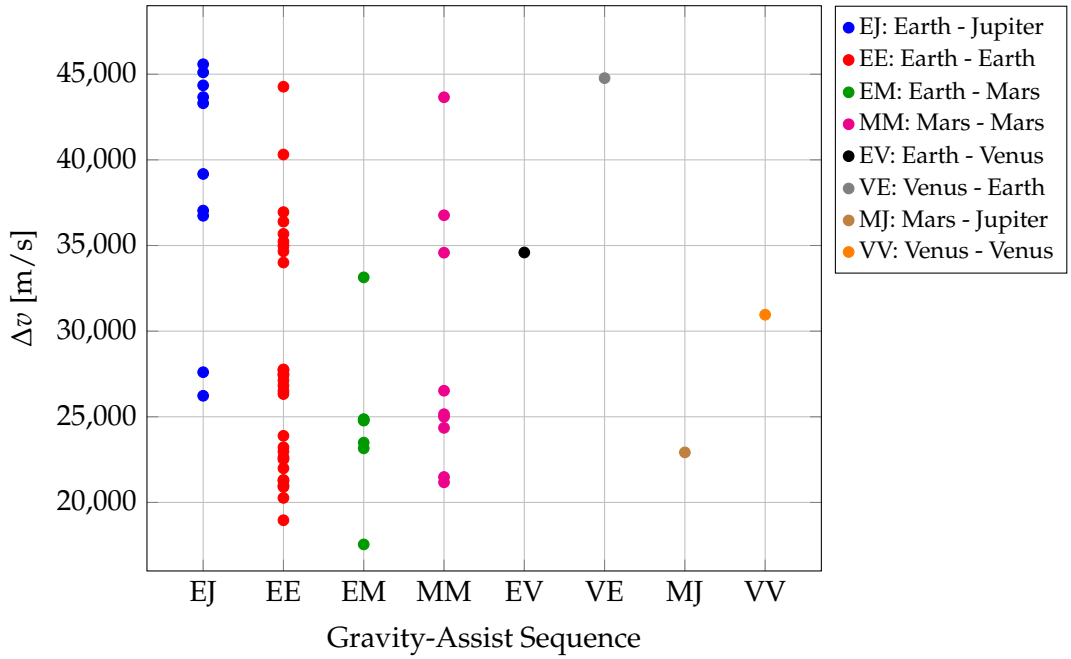

FIGURE 4.15: *Gravity-assist sequences which are used in the 3-leg trajectories of Phase 3.*

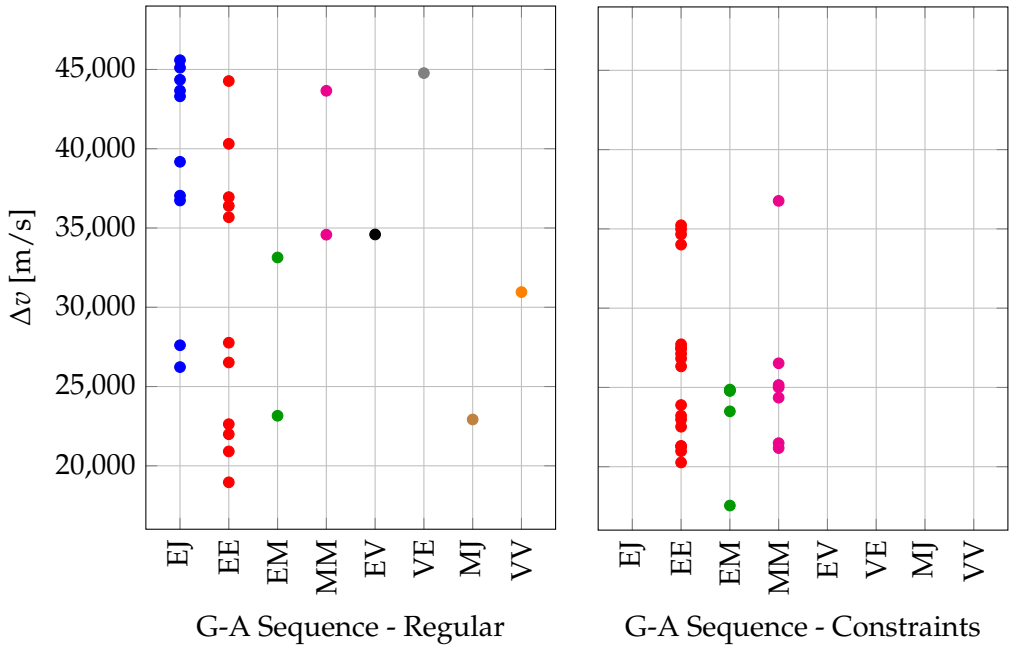

FIGURE 4.16: *Gravity-assist sequences which are used in the 3-leg trajectories of Phase 3, either with the regular method (left side) or by utilising the constraints (right side).*

To examine the gravity-assist partner sequences used by the 3-leg trajectories, Figure 4.15 depicts the eight different gravity-assist sequences which were found, in respect to the resulting Δv requirement. Most of the trajectories utilise a Earth - Earth (EE) gravity-assist

sequence, which enables very good, but also very bad solutions. Similar results are produced by using a Mars - Mars sequence (MM). The groups EJ, EV, VE, VV, for the most part, do not enable good results. Using a Earth - Mars gravity-assist sequence (EM) is, on the other hand, very beneficial for the resulting Δv requirement. Furthermore, the application of a Mars - Jupiter sequence (MJ) is also enabling the calculation of efficient trajectories, based on the one trajectory which was found.

To be able to compare which of these gravity-assist sequences were found by which search method setting, Figure 4.16 depicts a separated version of Figure 4.15. The left side shows the sequences used by the regular method and the right side shows which ones were used while applying the constraints. The application of the constraints drastically reduces the amount of different gravity-assist sequences. Only Earth - Earth, Earth - Mars, and Mars - Mars sequences are used.

Trajectory Examples

Figure 4.17 displays the best 2-leg trajectory found during Phase 3 and by that all testing phases. It requires just 12,351 m/s Δv and the flight path is similar to the best solution of Phase 2, depicted in Figure 4.10. The only differences are the slightly earlier launch date (56,299 MJD opposed to 56,335 MJD), gravity assist date at Earth (56,482 MJD opposed to 56,506 MJD) and a reduced flight time to 2,665 days opposed to 2,850 days. The shorter flight time can be explained by regarding the two different approach trajectories before the arrival at Jupiter. The spacecraft exceeds Jupiter's orbital path by a smaller amount than in the Phase 2 solution, accomplishing a more direct insertion into the aimed for orbit. As can be seen in the data, this trajectory is more efficient and less time consuming.

The best 3-leg trajectory found during Phase 3 is illustrated in Figure 4.18. With a Δv requirement of 17,549 m/s, this trajectory nearly reached the 1-leg trajectory benchmark. The first gravity assist is conduced at Earth after a flight time of 118 days and the second manoeuvre is performed at Mars after an additional flight time of 516 days. After the second encounter the spacecraft proceeds towards Jupiter on a helical-shaped trajectory.

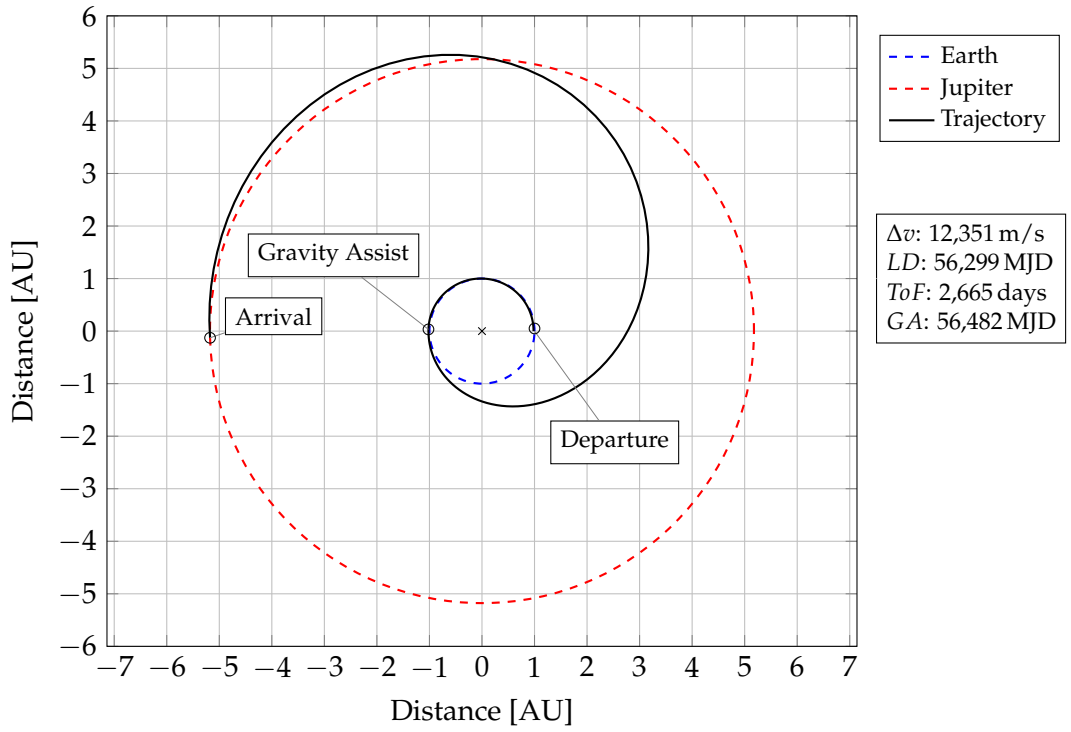

FIGURE 4.17: *Best 2-leg trajectory found during Phase 3. This solution represents the most optimal solution found during all three testing phases.*

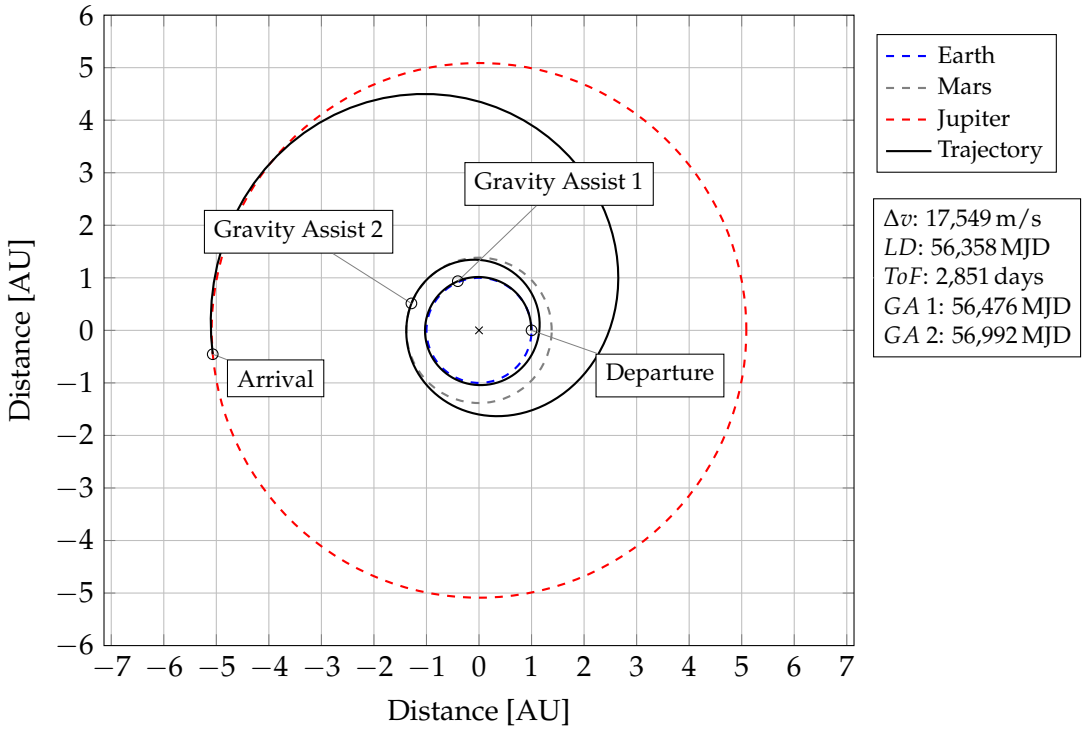

FIGURE 4.18: *Best 3-leg trajectory solution found during Phase 3. This solution represents the best 3-leg trajectory produced during all three testing phases.*

5 Discussion

In the following chapter the objectives of this thesis, outlined in Chapter 1.2, are analysed and discussed. At first the results of the testing phases are discussed, beginning with the assumptions made prior to Phase 1. Afterwards the problems emerged during Phase 1 are illustrated, followed by the insights gained by evaluating Phases 2 and 3. Subsequent, the gravity-assist partners and the encounter dates are examined. The usefulness of an application of the constraints is discussed and concluded afterwards. In the last section of this Chapter preliminary further investigations of different mission profiles are described.

5.1 Discussion of the Simulation Results

5.1.1 Assumptions

The design process of Phase 1 was based on two assumptions concerning the properties of the modified search method (see Figure 5.1). The first assumption was that small changes of the parameters d_{peri} and d_{velo} have great influences on the resulting trajectories and that values bigger than 10 % are not beneficial, because the behaviour of the search method is believed to match the unchanged version.

As can be seen in the Figures 4.1, 4.2 and 4.8 this is only in part supported by the observations. During Phases 1 and 2 it proofed to be very disadvantageous to set both parameters to 0 %. This is plausible, because by trying to utilise the single best possible gravity-assist manoeuvre, the search space gets strictly limited and it might be the case that better solutions get discarded in the first place. For instance, it might be possible that the spacecraft obtains too much velocity from the gravity assist and a large deceleration manoeuvre is necessary to reach the target. Or it is conceivable that two gravity assist, each yielding less energy, might enable a more efficient trajectory. Gaining the maximum amount of energy with one particular fly-by, does not implicate that the resulting trajectory is optimal.

In contrast to the assumption, the actual optimal value of d_{peri} lies in the range of 20 % (Phase 2) to 50 % (Phase 1). The optimal value of d_{velo} is, as later discussed, not as obvious as for d_{peri}, but better results are likewise computed when setting the parameter to values in the similar range. In accordance to the first part of the assumption, the results are becoming drastically better if d_{peri} is just slightly higher than 0 %, but the second part, that values greater than 10 % do not enable better results, could not be verified.

The second assumption was that allowing gravity-assist partners which are more than two positions away from the current body has negative effects on the resulting calculation. By

allowing gravity-assists to reach out too far away from the current body, it is assumed that the search space gets unrestricted, negating the inducement to apply the *PP* constraint in the first place. This assumption is not meant to be generally applicable and just specific to the tested Earth - Jupiter trajectory.

During the conducted testing phases, this assumption was just indirectly tested, as the *PP* constraint was never set to values greater than 2. But the assumption is supported by the fact, that the parameter setting of *forwards:1 / backwards:1* proofed to produce particularly better results than *forwards:2 / backwards:1* or *forwards:2 / backwards:2*. This outcome indicates that, at least for this particular mission profile, setting the *PP* constraint parameter to values greater than 2 should not enable better results. Greater values just widen the search space of possible gravity-assist partners. In this case, setting *forwards:3* for instance, would entail that the first gravity assist manoeuvre could be executed at Saturn. Allowing such manoeuvres for a trajectory to Jupiter would most likely be disadvantageous either in terms of Δv requirement or mission flight time.

It is thus appropriate to expect that setting the parameters to values higher than 2 does not enable the computation of better trajectories. For generalisable statements which parameter settings should be used for different mission profiles (e.g. to the outer rim of the solar system), more testing needs to be done. But initially it could be reasonable to set the partner pool constraint in a way that the first gravity-assist encounter is not allowed to be conducted directly at the end body (except for trajectories to Mars and Venus, because these planets are only one step away from Earth). In the case of an Earth - Jupiter trajectory, because these planets are just two steps away from each other within the solar system, limiting the possible next gravity-assist partner to the planet one step further in each direction of flight seems

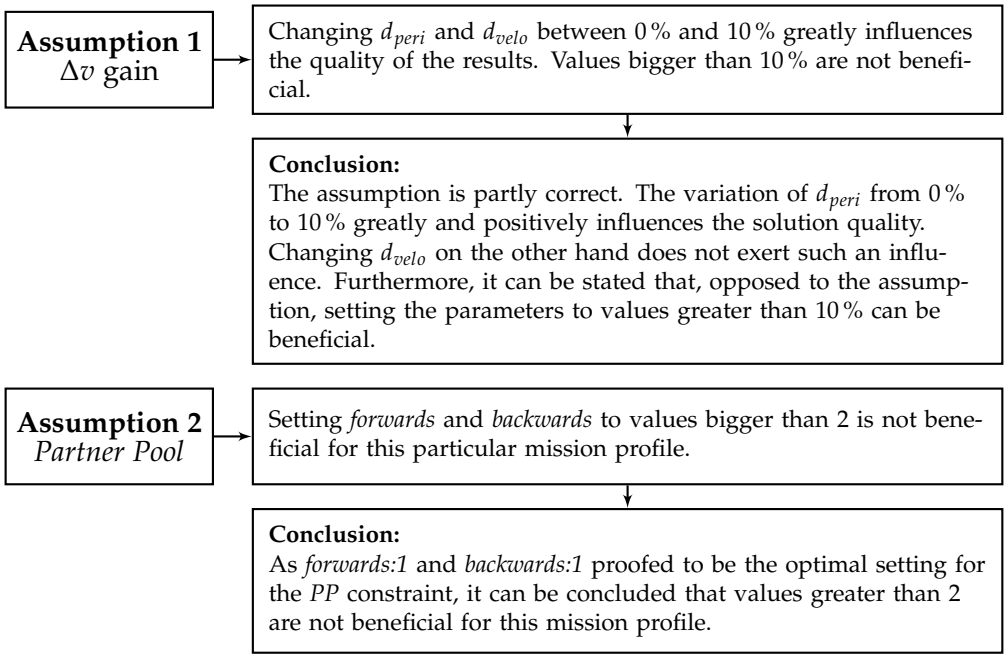

FIGURE 5.1: *Conclusions of the assumptions made prior to Phase 1.*

to be an advantageous approach. For computing e.g. an Earth - Neptune trajectory, it might be beneficial to set the parameters to allow gravity-assist manoeuvres which are four steps in direction (one fewer step than a direct path from Earth to Neptune) and one, or maybe two, steps against the direction of flight.

Without a priori knowledge about the to be expected behaviour, and the goal of Phase 1 to acquire this knowledge, founding the design of the first testing phase on these assumptions made sense. As it turned out, the first assumptions was just partially supported by the observations, but the acquired understanding about the influences of the two constraints on the resulting trajectories, lead to a more established design process in the second testing phase.

5.1.2 Phase 1

Problems

Phase 1 inherited the drawback of being the first testing phase and thus was required to be based on the discussed assumptions concerning the optimisation behaviour. Furthermore, the chosen settings, like the hyperbolic excess velocities at start and end ($v_{\infty,start}$, $v_{\infty,end}$) and the low number of generations were not ideal to find the best solutions. This is why Phase 1 involved a few problems and difficulties which will be further examined in this section.

One of the problems of Phase 1 originated in the not anticipated behaviour when d_{peri} is set to 0 %. During the runs number 8 to 12 the parameter d_{velo} was increased from 1 % to 50 %, while d_{peri} was set to 0 %. All these runs produced increasingly bad results for the 2-leg and

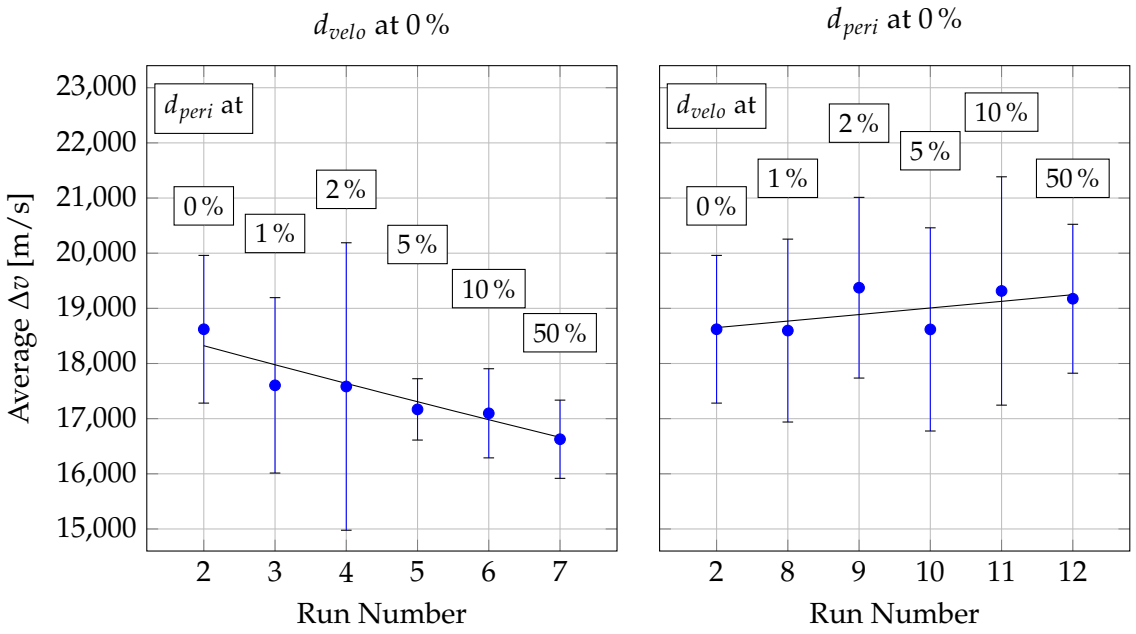

FIGURE 5.2: *Comparison between of the averagely required Δv during Phase 1, while increasing d_{peri} (left side) or d_{velo} (right side) from 0 % to 50 %*

3-leg missions alike. This effect is more obvious for the 2-leg missions though. Taking the run number 2 as starting point and placing the runs 3 to 7 (d_{peri}), respectively 8 to 12 (d_{velo}) behind it, the changing Δv requirement can be clearly observed. This connection is depicted in Figure 5.2. The black lines are laid through the average values and show the trend how the solutions get better while increasing d_{peri}, and worse by increasing d_{velo}.

The primary source of these changing Δv requirements is most likely based on the setting of d_{peri} and not influenced by increasing d_{velo}. Keeping d_{peri} constantly at 0 % during the runs number 8 to 12, probably prohibits the calculation of better solutions, as the search space gets strictly limited to a few different possible gravity-assist manoeuvres. It could have been expected that greater values for d_{velo} widen the search space in a manner that more different gravity assist become possible and better solutions could be found. But this expected outcome could not be observed. Thus, higher values of d_{velo} do not have a positive influence on the solutions.

The data gathered during Phase 2 (see Figure 4.8) supports this claim. Increasing d_{velo} from 10 % to 100 % during the runs 13 to 22 of Phase 2, does not have a consistent influence on the quality of the results. As said in Chapter 4.2, the average values and standard deviations differ only in a small range, which could be due to statistical effects, as just 10 calculations were executed per run. More details on this topic will be provided in Chapter 5.1.3.

The next problem which occurred during Phase 1 is that the evolutionary search algorithm was set to optimise the solutions over the course of just 200 generations. This limit was reached by 85 of the 210 2-leg trajectories (around 40 %) and by 25 of the 3-leg trajectories (around 12 %). Hitting this boundary indicates that the particular solution could have been further optimised.

This circumstance is unfavourably, but by assessing over how many generations each trajectory was optimised and by regarding the resulting Δv requirement, no distinct correlation between the quality of the solution and the number of generations can be found. Solutions which reached the generation limit are varying in the same range of possible Δv requirements as the other solutions. Accordingly, it can be stated that the overall bad solutions of Phase 1 are very likely not originating from the low number of generations.

Presumably, the main source of the poor solution quality is the definition of values for the hyperbolic excess velocities at start and end of the mission, which might be the most severe problem of Phase 1. Comparing the 1-leg trajectories of all three testing phases, it can be seen that the trajectories of Phase 1 require roughly the same amount of Δv as the 1-leg trajectories of Phases 2 and 3. The initial and terminal hyperbolic excess velocities thus did not enable the calculation of better solutions. Additionally, the produced 1-leg trajectories variate in a small range, because setting values for $v_{\infty,start}$ and $v_{\infty,end}$ introduces random real-values variables into the system, making it impossible to consistently find the same 1-leg trajectory.

The 2-leg and 3-leg missions, on the other hand, are generally requiring significantly more Δv than in Phases 2 and 3. It is possible that the initial velocity vector influences the flight path of the spacecraft in a way, that returning to Earth to conduct a gravity assist gets more

complicated. As later discussed (see Chapter 5.1.5) conducting a gravity assist at Earth enables the best 2-leg trajectories of the whole testing phases. Due to the possible restraint of finding a trajectory which utilises this beneficial gravity-assist manoeuvre, most of the 2-leg trajectories of Phase 1 might be forced to conduct a disadvantageous fly-by at Mars, resulting in higher Δv requirements. It is thus safe to argue that the generally higher Δv requirements of the 2-leg and 3-leg trajectories of Phase 1 are originating from the poor choice of the values of $v_{\infty,start}$ and $v_{\infty,end}$.

Lessons Learned

Taking the overall goal of Phase 1 into account, to get a first overview about the possible effects of an application of the constraints, these mentioned problems are not substantial. Although the solutions produced during this phase are not optimal, the implications of using the constraints can be assessed. Accordingly, the following statements about the constraint parameter settings can be derived from the data:

- Using the constraints has an impact on the results. Depending on the specific settings, this influence can be positive or negative.
- More beneficial parameter settings enable the calculation of trajectories which require less Δv in average, with a lower standard deviation (up to 7.8 % less Δv with 73 % smaller σ for 2-leg missions and 28.8 % less Δv with 57.2 % smaller σ for 3-leg missions).
- Different settings impede the calculation of solutions which are better than the 1-leg mission benchmark, which results in increased Δv requirements (up to 13.2 % more Δv with 74.4 % higher σ for 2-leg missions and 40.6 % more Δv with 17.3 % higher σ for 3-leg missions).
- Changing d_{peri} to values greater than 0 % has a large and positive influence on the solutions.
- Changing d_{velo} to values greater than 0 % has a small and mostly negative influence on the solutions.
- The best Δv-gain constraint settings found during this phase are: around 50 % for d_{peri} and around 0 % to 1 % for d_{velo}.
- Utilising the *PP* constraint has also the potential to increase or decrease the quality of the solutions.
- The best solutions are calculated by setting this constraint to *forwards:1*, *backwards:1*. With these settings the trajectories require on average 1.1 % less Δv with 61.2 % smaller σ for 2-leg missions and 7.2 % less Δv with 40.6 % smaller σ for 3-leg missions.
- Constraint parameter settings which produce good results for 2-leg missions, are also very likely to produce good results for 3-leg missions, see below.

Correlation Between 2-leg and 3-leg Solutions

The correlation between the quality of the 2-leg and 3-leg solutions is best visualised by normalising the average Δv requirements of the 3-leg missions in respect to the 2-leg missions. This normalisation is done by dividing the average Δv of the benchmark run number 1 of the 3-leg missions by the benchmark run of the 2-leg missions. The resulting ratio is 2.3877, see Eq. (5.1). By dividing every average Δv of the 3-leg missions by this ratio and generating the percentage wise deviation of the benchmark, the graph depicted in Figure 5.3 is resulting.

$$\begin{aligned} \text{Run 1, 2-leg Missions} \qquad & \overline{\Delta v} : 17{,}779\,\text{m/s} \\ \text{Run 1, 3-leg Missions} \qquad & \overline{\Delta v} : 42{,}451\,\text{m/s} \\ & \frac{42{,}451}{17{,}779} \approx 2.3877 \end{aligned} \tag{5.1}$$

The benchmark run number 1 is displayed as a black mark and the dashed line. The average normalised deviation, of the Δv requirements of the 2-leg and 3-leg missions in percent, are depicted as blue and red marks. It can be seen that the quality of the results changes in a similar way for the 2-leg and 3-leg missions alike. It is thus reasonable to conclude that parameter settings which produce good results for 2-leg missions are most likely producing good results for 3-leg missions as well.

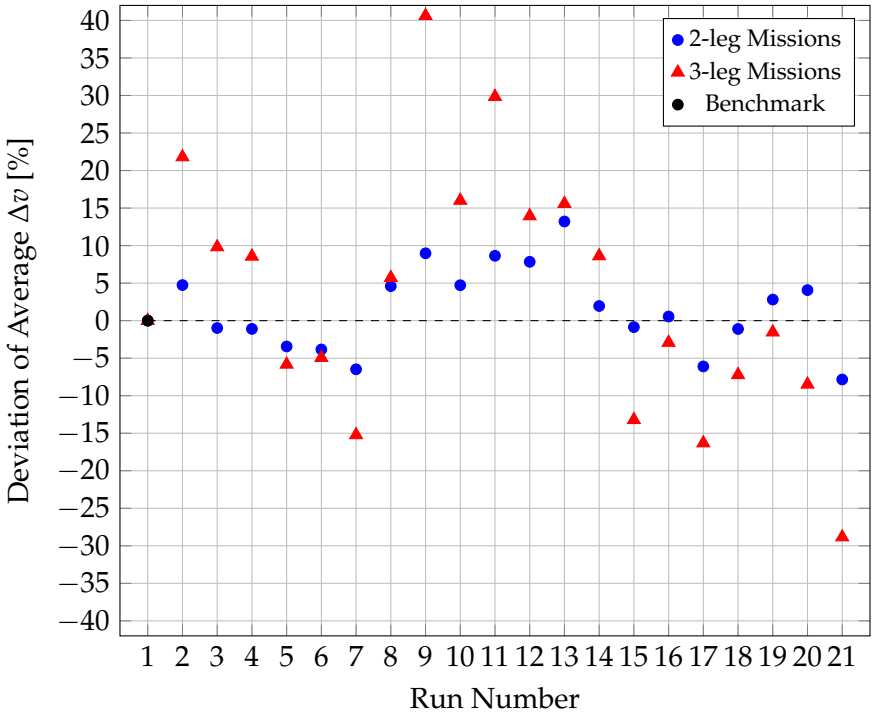

FIGURE 5.3: *Average Δv requirements of 2-leg and 3-leg missions of Phase 1. Normalised in respect to the benchmark run number 1 in which no constraints were used. The Y-axis depicts the deviation in percent from the benchmark.*

5.1.3 Phase 2

The insights gained by the first phase of testing were directly incorporated in the design process of Phase 2. Accordingly, the velocities at start and end of the mission ($v_{\infty,start}$, $v_{\infty,end}$) were set to 0 m/s, the maximum number of generations was set to 1,000 and based on the correlation between 2-leg and 3-leg trajectories, only 2-leg trajectories were computed to reduce the computing time requirements. Furthermore, the exploration of the Δv gain parameter settings was done in a sequential manner, in which the first step was to determine the best value for d_{peri} and afterwards examine the different settings for d_{velo} based on this value. Taking in account that d_{peri} should have a significantly larger influence on the solution quality, which was observed during Phase 1, this approach is reasonable to better asses the impact of changing d_{velo}.

Alteration of d_{velo}

To examine this impact, Figure 5.4 depicts the averagely required Δv and standard deviation of the corresponding trial runs. During these runs, d_{peri} is set to 20 % and d_{velo} is increased from 0 % to 100 %. As mentioned in Chapter 4.2, the influence of changing d_{velo} is relatively small. Most of the average values are inside a scope of $\pm$ 1.5 % in respect to run number 4. Based on the relatively low amount of calculations per run, this deviation is most likely based on random variations of the solution quality. Nevertheless, it can be observed that the

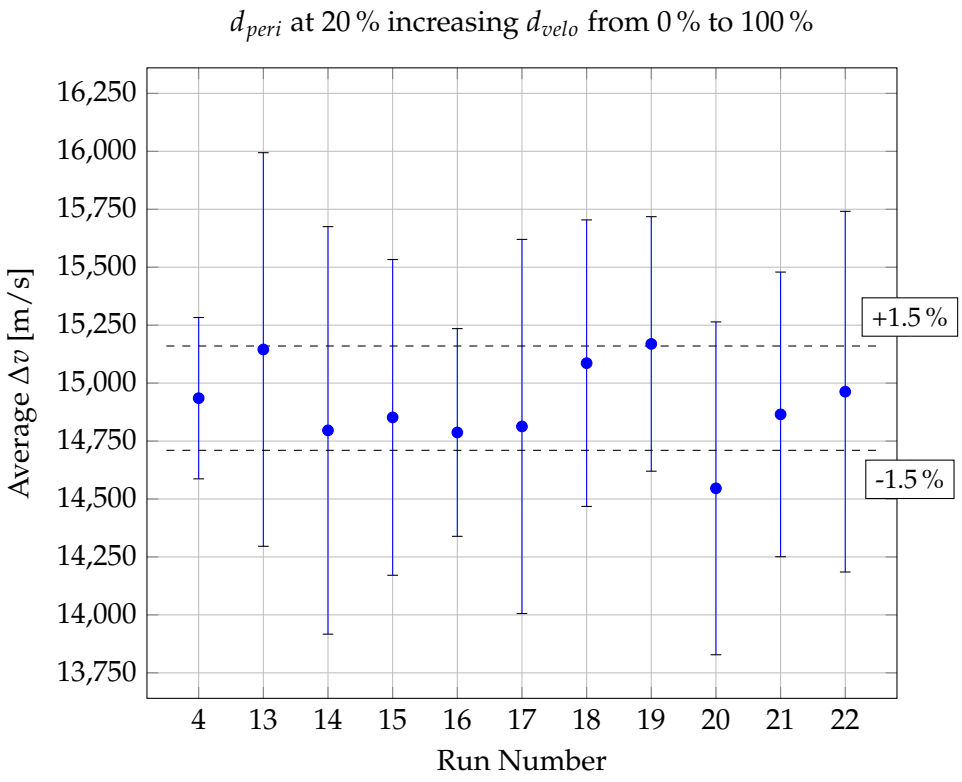

FIGURE 5.4: *Increasing d_{velo} from 0 % to 100 % in 10 % steps, while setting d_{peri} to 20 % during Phase 2. Most of the average values are located in a range $\pm$ 1.5 % away from the average Δv produced during run number 4.*

standard deviation of the runs 13 to 22 is bigger than the standard deviation of the solutions of run number 4. This indicates that increasing d_{velo} might have a small negative influence on the deviation of the solutions.

Considering how the gravity assist is calculated based on the constraint parameters, this outcome is plausible (see Figure 3.2). The first step is to randomly define the pericenter distance in a specified range (utilising d_{peri}) away from the closest possible distance, which generally defines how much Δv can be gained with a gravity-assist manoeuvre. The maximum possible Δv gain is achieved, as specified in Chapter 3.2, by setting the hyperbolic excess velocity of the encounter to be equally to the circular orbit velocity of the given pericenter distance. Setting d_{velo} to a non-zero value thus entails a variation of the hyperbolic excess velocity within the specified range around this maximum. This second step, in which the actual hyperbolic excess velocity is randomly chosen, encompasses a significantly smaller range of possible outcomes than the definition of the pericenter distance in the first place. Accordingly, choosing the pericenter distance roughly appropriates the possible Δv gain and the fine tuning of the hyperbolic excess velocity in the second step accurately determines the actual Δv gain of the manoeuvre.

It is thereby reasonable to confirm the observation of Phase 1, that the parameter d_{peri} has a greater influence on the resulting trajectories. Furthermore, the setting of both parameters to values around 20 % should be sufficient to produce good results in most cases. This is considered to enable a good balance between choosing a high-quality gravity-assist manoeuvre and not limiting the search space too much.

Assessment of the Partner Pool Constraint

To examine the impact of the *PP* constraint, the average Δv requirements and deviations of the associated runs of Phase 2 are shown in Figure 5.5. The graph on the left side depicts the solutions produced by just using the *PP* constraint, i.e. by deactivating the Δv-gain constraint. Run number 1 serves as the benchmark, in which the *PP* constraint is also deactivated. On the right side, the four runs are displayed in which the Δv-gain constraint is applied (with d_{peri}: 20 % and d_{velo}: 40 %) and the *PP* constraint utilises the same parameter settings as before. Run number 15 is the benchmark in this case.

As can be seen in the graphs, setting the *PP* constraint to *forwards:1* and *backwards:1* (runs 26 and 23) positively influences the quality of the solutions in a similar way in both cases. Setting the parameters to *forwards:2* and *backwards:1* has a negative influence (runs 27 and 24). The influence of setting the parameters to *forwards:2* and *backwards:2* on the other hand is ambiguous (runs 28 and 25).

Is the Δv-gain constraint deactivated, the averagely required Δv is decreased, while the standard deviation is increased. Thus, by using this setting, solutions are produced in a wider range, with greater deviation. If the the Δv-gain constraint is applied, the standard deviation is mainly just increased by a small factor, while the average Δv remains constant (refer to Table 4.2 for the exact values). While using both constraints, setting the *PP* constraint to *forwards:1* and *backwards:1* is therefore a more promising approach.

Connecting Δv Gain and PP Constraint

The most important result of examining the graphs in Figure 5.5 is that the influence of the two constraints on the solution quality is independent from another. This means that the influence of the Δv-gain constraint and the influence of the *PP* constraint add up to magnify the overall influence on the results. The influence of just using the *PP* constraint can be assessed in the graph on the left hand side. The influence of the Δv-gain constraint (i.e. significantly lower standard deviation and average Δv requirements which are a little bit lower), is present in all solutions on the right side.

It can be observed that the basic alterations of the Δv requirement and the respective standard deviation on the left side are mostly similar to the alterations displayed on the right side. This means that the influence of each constraint can be regarded separately and it can be stated that the two constraints are independently from another affecting the quality of the solutions. Disregarding however, that the solutions of run number 28 require less Δv in average. As this run entails a higher standard deviation, the lower average value is assumed to be originating from this variance. A wider spread of the solutions should implicate a more distorted average value.

Conclusion of Phase 2

In conclusion, Phase 2 enabled a better assessment of the implications of different constraint parameter settings as Phase 1. This is mainly due to the elimination of the optimisation problems encountered during Phase 1. Besides the drawbacks of Phase 1, the observations and conclusions regarding the behaviour of the constrained search method, could be mostly

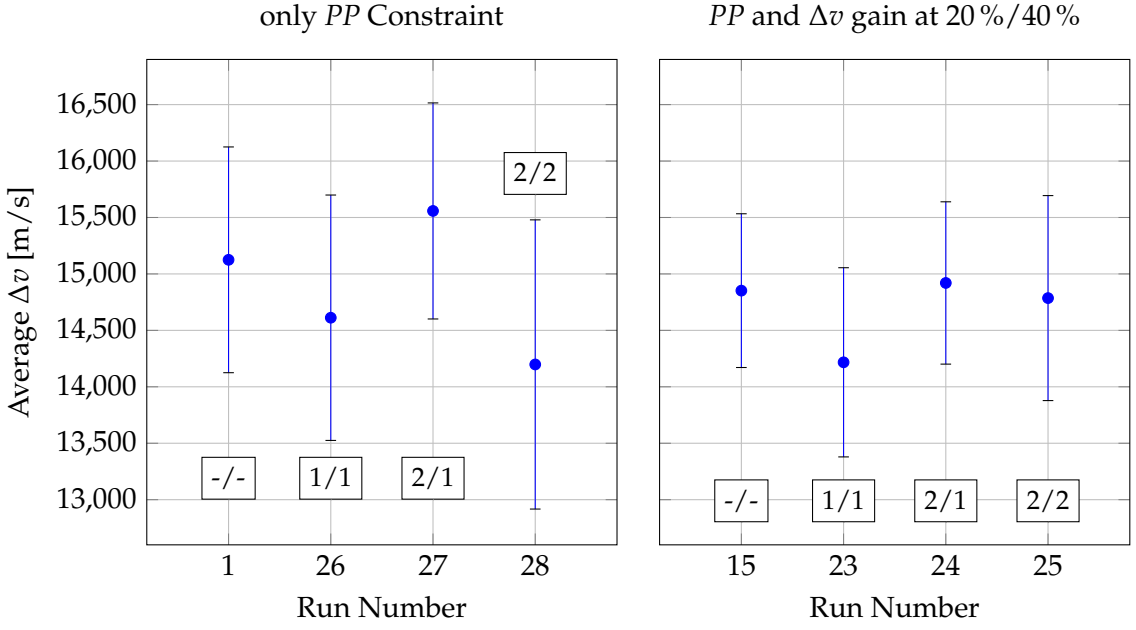

FIGURE 5.5: *Comparison between applying solely the PP constraint (left side), and using PP and Δv-gain constraint together (right side). The Δv-gain constraint utilises the values 20 % for d_{peri} and 40 % for d_{velo}. Data gathered during Phase 2.*

confirmed or refined by evaluating Phase 2.

- d_{peri} greatly influences the results. The best setting found during Phase 2 is 20 %.
- d_{velo} has a smaller influence on the solutions and should be set to values between 0 % to 40 %.
- The *PP* constraint enables better results when set to *forwards:1* and *backwards:1* for this particular mission profile.
- The combination of both constraints enables the calculation of better solutions, as the effect of each particular constraint is independent from the other.
- Even while using more beneficial search method settings (no initial and terminal hyperbolic excess velocities $v_{\infty,start}$, $v_{\infty,end}$ and 1,000 generations), the application of the constraints enables a further improvement of the solutions.

5.1.4 Phase 3

The third phase of testing was designed to thoroughly investigate the influence of both constraints using one particular parameter set. d_{peri} and d_{velo} were both set to 20 % and the *PP* constraint was set to *forwards:1* and *backwards:1*. In distinction to the other phases, the population size was increased to 200 members, which should enable the calculation of overall better trajectories, according to tests conducted by Maiwald, see Chapter 2.4.2. The goal is to investigate whether the constraints are able to further improve the solution quality.

2-leg Missions

As can be seen in Chapter 4.3 this further improvement could be achieved. As for the 2-leg missions, the utilisation of the constraints entailed a notable reduction of the variation of the solutions (39.5 % smaller standard deviation), which means that, on average, a smaller variety of different solutions was found. The average Δv requirement was not reduced by much however. This is most likely due to the (near) optimality of the solutions in the first place. If there is just a small amount of trajectories possible that require less Δv, an optimisation of the search space through utilisation of the constraints cannot necessarily generate better trajectories. Nevertheless, this drastic reduction of the deviation of the solutions is a great result.

3-leg Missions

The situation is different for the 3-leg trajectories. It can be assumed that these trajectories are far away from optimality by requiring more than 34,000 m/s Δv in average (see Table 4.3). This is why the constraints should be able to greatly influence the solution quality. As the data shows, this is the case (see Figure 4.13). The averagely required Δv was decreased by 23.8 % and the standard deviation by 35.1 %. This is a remarkable improvement to the regularly calculated 3-leg trajectories. The utilisation of two sequential gravity-assist manoeuvres enables the constraints to genuinely enhance the overall Δv requirements. This behaviour could already be observed during Phase 1, where the average Δv was reduced up to 28.8 % with a 57.2 % smaller standard deviation for the best 3-leg trajectories.

Gravity-Assist Sequences

The gravity-assist partners and encounter dates of the 2-leg missions of all testing phases will be further discussed within the next section. Beyond that, the gravity-assist sequences of the 3-leg missions of Phase 3 show interesting results as well. All sequences which were found during this phase are depicted in Figures 4.15 and 4.16 within Chapter 4.3. The separated version of the graph, were the sequences found by the regular method are depicted on the left side and the ones produced by using the constraints are shown on the right side, illustrates the effect the *PP* constraint has on the selection of the gravity-assist partner.

The complete first sequence group (EJ: Earth - Jupiter) is not used at all by the constrained method, because a gravity assist at Jupiter is not allowed after an encounter with Earth. The specified parameter settings of the constraint only permit a jump from one planet to the next in two consecutive fly-bys. Please note: the number of planets which need to be skipped after executing the last gravity assist and heading to the target planet are not restricted by the constraint.

The second group of sequences (EE: Earth - Earth) was often used by the regular search method and produced a great variety of good and bad solutions (ranging from 19,000 m/s to 44,000 m/s). While applying the constraints, this sequence was also extensively used, but with better results in average (from 20,000 m/s to 36,000 m/s). The sequences EM and MM on the other hand were just used four times by the regular method and did not produce exceptionally good results. But with the constraints, these sequences were found five (EM) and seven (MM) times with notably better resulting trajectories. One of the Earth - Mars sequences even nearly exceeded the benchmark of the 1-leg trajectory (16,034 m/s) with requiring only 17,549 m/s. The sequence groups EV, VE, MJ, VV were not used by the constrained search method, although the Mars - Jupiter sequence seems to enable good results, with a Δv requirement of 22,922 m/s. Maybe this sequence would have been used if more calculations had been conducted.

Conclusion of Phase 3

Regarding the results of Phase 3 and Phase 1, it can be concluded, that the constraints have a greater impact on trajectories which utilise more gravity-assist manoeuvres. Furthermore, the constraints reduce the amount of different gravity-assist sequences that are used in 3-leg missions. This result is plausible, as the constraints exclusively influence the selection and calculation of gravity-assist manoeuvres. For trajectories with even more gravity assists this conclusion has to be further tested, which is why a short preliminary investigation of different mission profiles was conduced. These stripped-down testing series are presented in Chapter 5.3.

5.1.5 Considerations About Gravity-Assist Partners and Encounter Dates

To further examine the gravity-assist partners and encounter dates, the 2-leg trajectories of all three phases are gathered in Figure 5.6. The encounter dates and resulting Δv requirements of all 550 trajectories are accumulated in this graph. The red marks represent the

trajectories calculated during Phase 1, the black marks depict the trajectories of Phase 2 and the blue marks the trajectories of Phase 3. The diamond shapes depict Earth and the triangles Mars as gravity-assist partner. The launch windows of the three phases are illustrated as well. During Phase 1, the window spans from 56,000 MJD to 56,100 MJD. The other two phases used a launch window from 56,000 MJD to 56,360 MJD. All together, four different main encounter groups at Earth and two at Mars can be distinguished. These six encounter groups are listed in the graph and delimited by black vertical lines. However, a few minor exceptions exist which do not fit into one of the defined groups.

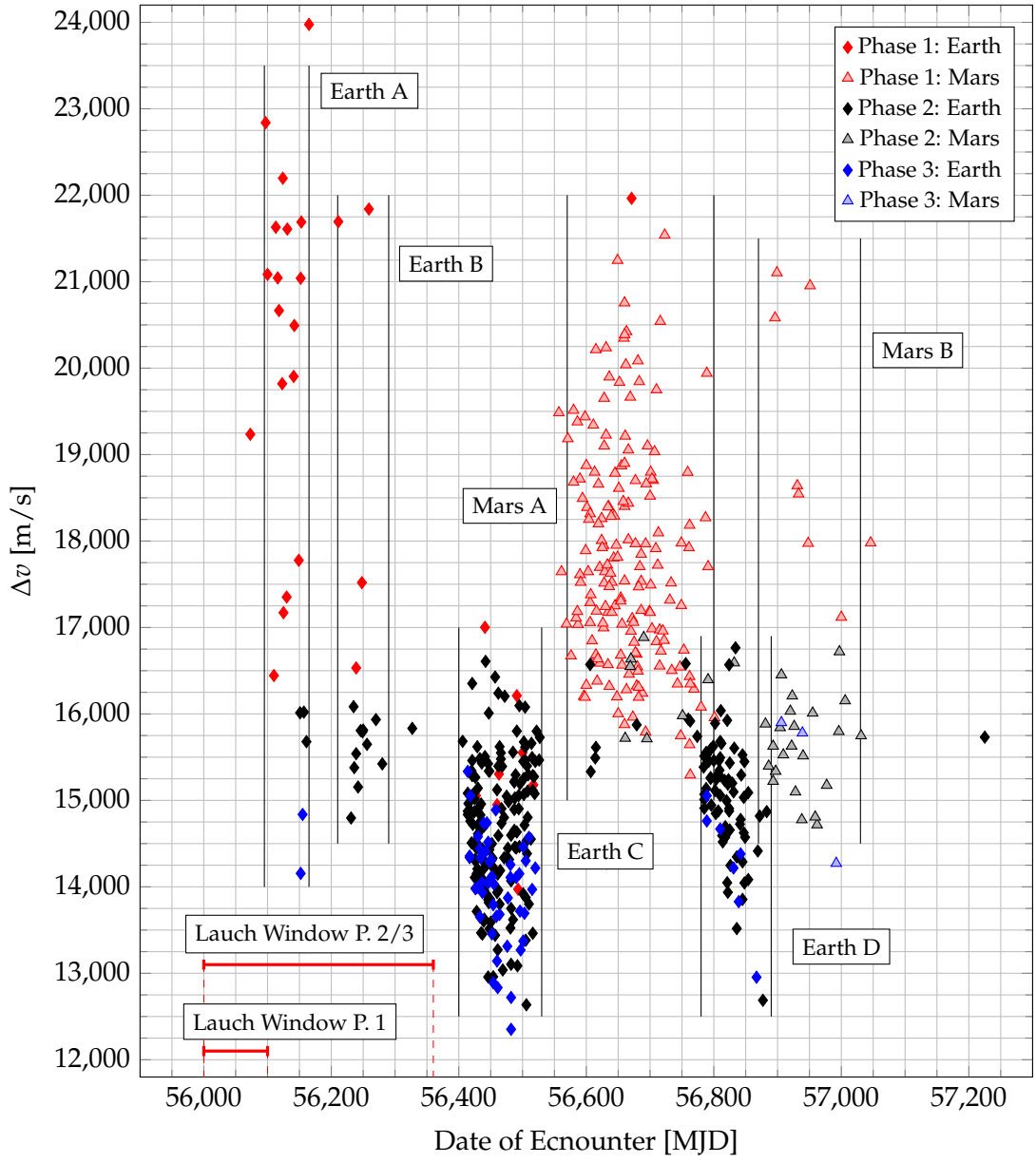

FIGURE 5.6: *Encounter dates and fly-by bodies of all 550 calculated 2-leg trajectories in respect to the overall Δv requirement of the trajectory. Differenced by testing phase and encounter body. Six main groups of encounters can be defined, in which most of the performed gravity assists can be allocated.*

Earth Encounter Groups

The first encounter group at Earth (*Earth A*, around 56,125 MJD) was found during all three phases, but most of the calculated trajectories origin from Phase 1. The group is very close to the launch window of Phase 1 and right in the middle of the launch window of Phases 2 and 3. This might influence why this particular encounter was not often used during Phases 2 and 3. As nearly half of the allowed launch dates are at a later time, most of the trajectories might not be able to use this encounter in the first place. In general, utilising a group *Earth A* encounter, does not enable the calculation of exceptionally good trajectories. Instead the found trajectories are located in a broad range, from around 14,000 m/s to well above 24,000 m/s. None of the other encounter groups show such a wide deviation of the solution quality.

The second group of encounters, *Earth B*, is located around 56,250 MJD and was used just a few times during Phases 1 and 2. This might indicate that this encounter group is rather sub-optimal, as it was never used during Phase 3, which is assumed to utilise the most optimal trajectories.

The next two encounter groups at Earth (*Earth C* and *Earth D* at 56,470 MJD and 56,850 MJD) are used most frequently during Phases 2 and 3 and enable the calculation of the best trajectories of the whole testing phases. The group *Earth C* seems to be especially beneficial, as it was used during all three phases and enabled a great variety of really good solutions. The best of these trajectories, found during Phase 3, requiring just about 12,300 m/s. The worst trajectory of this group, calculated during Phase 1, requires 17,000 m/s of Δv, which means that the encounters of this group are very close together and most of the found solutions exceed the 16,000 m/s benchmark of the 1-leg trajectories.

This indicates that a local optimum of the solution space is located in vicinity of 56,470 MJD. Based on the data this encounter might even be the overall global optimum, enabling the most efficient trajectories from Earth to Jupiter. The encounter group *Earth D*, which is located around one year later than *Earth C*, was also used often, especially during Phase 2. This observation suggests that this encounter group also comprises a local optimum which enables good trajectories, but is not as beneficial as *Earth C*.

Mars Encounter Groups

The first group of Mars encounters (*Mars A*) was mostly used during Phase 1 and generally does not enable good trajectories. Just a few of the trajectories utilising this particular encounter could exceed the 1-leg benchmark. The significantly frequent usage of this encounter during Phase 1, might cohere with the general flaws in the optimality of the solutions and the definition of initial and terminal hyperbolic excess velocities. The initial velocity leading away from Earth could encourage the utilisation of a gravity assist in the *Mars A* group, which does not enable exceptionally good results. The optimiser might got stuck at this encounter with small chances to redirect the trajectory to a more beneficial encounter in the *Earth C* or *Earth D* groups.

The second group of Mars encounters *Mars B*, on the other hand, was used during all three

phases. The trajectories in this group that were found during Phase 2 and 3 did perform relatively well, with Δv requirements between 14,000 m/s and 17,000 m/s. In contrast to the solutions of Phase 1, which require more Δv. This particular encounter was used relatively often during Phases 2 and 3, which indicates that the local optimum of the Mars encounters might be located in the area around 56,950 MJD.

Summary

To summarise, the graph depicts were the local optima of the solution space can be found and that a greater optimality of the trajectories entails an accumulation of the solutions in the *Earth C* encounter group. The correlation between optimality of the solution and the selection of an *Earth C* encounter can be assessed by regarding the total number of *Earth C* encounters within every testing phase.

Phase 1 produced the least optimal solutions, and just 9 of the trajectories (4.3 %) utilise this particular encounter. Phase 2 produced considerably better results and altogether, 155 of the trajectories (55.4 %) are located in the *Earth C* group. In the best performing run of Phase 2 (run number 23) 8 of the 10 trajectories (80 %) are located in this group. During Phase 3 the most optimal solutions were calculated. Of the 30 trajectories which were produced by utilisation of the constraints, 26 are located in this encounter group (86.7 %). The four remaining solutions are situated in the group *Earth D*.

The trajectories of Phase 3 located in the *Earth A* and *Mars B* encounter groups, were exclusively found by the regular method. In total, while *not* using the constraints, 22 of the 30 trajectories (73.3 %) are utilising the *Earth C* encounter. It can be thus stated, that more optimal solutions tend to be found in the *Earth C* encounter group. This result confirms the observation made during preliminary simulations done by Maiwald [1], see Chapter 2.4.2. Furthermore, as the application of the constraints is driving the search method to utilise this beneficial gravity-assist encounter, it can be derived that the constraints produce more optimal solutions.

5.2 Improvement of the Convergence

To summarise, the application of the constraints beneficially influences the identification of the global optimum and the convergence of the solutions to this optimum. For 2-leg trajectories, utilising the constraints can drastically reduce the standard deviation of multiple solutions. The average Δv requirement, on the other hand, could generally just be decreased by a few percent. This means that the constraints work as intended, since they were specifically designed to find better fly-by manoeuvres and disregard the less advantageous ones. Using the constraints further directs the optimiser to conduct more profitable gravity-assist manoeuvres, as explained above. It is a great result that the application of the constraints thus increases the chances to find good solutions.

Additionally can be observed that the constraints have a notably bigger impact on the calculation of trajectories with more gravity-assist manoeuvres. The produced 3-leg trajectories

show a great reduction in the average Δv requirements and significantly smaller standard deviation. The variety of different 3-leg gravity-assist sequences (found in Phase 3) were drastically reduced to sequences which are more beneficial for the mission profile.

However, it has to be considered that by choosing the wrong parameter settings, these benefits can be discarded and trajectories which require much more Δv can be produced. To gain knowledge about the most beneficial parameter settings for a given mission profile (i.e. start and end body, launch date and flight time), it is essential to initially conduct a pre-evaluation of different parameter settings. These tests should just use 2-leg missions, with a reduced amount of population members, to limit the time requirements of the computation. Parameter settings which are found to be beneficial during this initial evaluation are very likely to produce better results for trajectories with more gravity assists and which are computed by utilising more population members.

To further investigate the possible impacts of the constraints, different mission profiles have to be examined. In future investigations about beneficial parameter settings for various missions, the two constraints need to be regarded separately.

The most beneficial parameter for the *PP* constraint are most likely severely dependent on the respective start and end body, as the limitation of the next possible gravity-assist partner should not be too strict, to prohibit the spacecraft to actually reach the target planet.

The Δv-gain constraint, on the other hand, might possibly possess a globally optimal parameter setting for all different kind of mission profiles. The constraint is designed to harvest the most energy of each gravity-assist manoeuvre. The parameter settings which were found to be most beneficial during the tests in this thesis, might also be advantageous in different scenarios. The feature of sticking to gravity assists of higher quality has the potential to be commonly useful with a specific parameter set, without the requirement of determining these parameters for every new missions profile.

5.3 Preliminary Further Investigations

The thorough examination of the constraint's implications realised in the context of this thesis focusses on one particular mission leading from Earth to Jupiter. Investigating different mission profiles in a similar, detailed manner would go beyond the feasible scope. However, three additional, short simulation series were conducted to examine how the constraints affect trajectories to Mercury, Mars and Saturn.

The three tests are structured in a similar manner as Phase 3. One run is executed with the regular search method and one run while applying the constraints. The parameters are the same as in Phase 3, accordingly d_{peri} and d_{velo} are set to 20 % and the *PP* constraint is defined as *forwards:1* and *backwards:1* for the trajectories to Mercury and Mars, and to *forwards:2* and *backwards:2* for the trajectory to Saturn. The higher *PP* parameters for the mission to Saturn are chosen, as explained above, to prevent that the first gravity assist can be conducted at

TABLE 5.1: *Starting conditions of the additional calculated missions to Mercury, Mars and Saturn.*

Parameter	**Mercury**	**Mars**	**Saturn**
Start Body	Earth	Earth	Earth
End Body	Mercury	Mars	Saturn
Launch Date	58,300 MJD	58,150 MJD	57,700 MJD
Launch Window	360 days	150 days	360 days
Maximum Flight Time	2,000 days	1,000 days	4,500 days
Minimum Flight Time	500 days	200 days	2,000 days
Maximum Revolutions	6	4	8
$v_{\infty,start}$	0 m/s	0 m/s	0 m/s
$v_{\infty,end}$	0 m/s	0 m/s	0 m/s
Maximum Number of Gravity Assists	3	3	3
Population Size	100	100	100
Maximum Number of Generations	1,000	1,000	1,000

the target planet and to prevent that the search space is too strictly limited for this more complex trajectory.

Each run is calculated five times, to limit the calculation effort on one hand, and, on the other hand, to gather data which is better statistically evaluable than one single calculation per setting would have been. Furthermore, each mission utilises up to three gravity-assist manoeuvres, resulting in 1-leg, 2-leg, 3-leg and 4-leg trajectories. The calculation of 4-leg trajectories is done to investigate whether the observed beneficial influence of the constraints also transfers to trajectories with more gravity assists. The evolutionary search algorithm is set to work with a population size of 100 members. The launch dates, launch windows and flight times are more or less arbitrarily chosen to match Hohmann-transfer windows to the respective target bodies. The main search method settings are listed in Table 5.1.

The goals of these additional simulations are to assess the impact of applying the constraints with the given parameter settings on different mission profiles. As stated above, the Δv-gain constraint parameters found to be beneficial for the Earth - Jupiter trajectory are believed to be also advantageous when applied to other trajectories. This assumption is tested in a small scale during the following simulations.

5.3.1 Mercury

The first additional investigated mission is starting at Earth and ending at Mercury. In total, ten solutions were produced, five using the regular search method and five applying the constraints. Every solution comprises of a 1-leg, 2-leg, 3-leg and 4-leg trajectory and for each trajectory type, the average Δv requirement and standard deviation σ are calculated. The resulting values are differentiated based on the application of the constraints. The resulting values and the graphical representation are illustrated in Figure 5.7.

The 1-leg trajectory benchmark requires 18,636 m/s of Δv. As before, the search method was able to always find the same 1-leg trajectory. The calculated 2-leg trajectories are in close proximity to this benchmark. Two of the five trajectories produced by the regular search method exceeded this limit by requiring 17,918 m/s and 17,994 m/s. While applying the constraint, four trajectories were found which require less than the benchmark (18,069 m/s, 18,512 m/s, 18,211 m/s and 17,963 m/s). This results in an average Δv requirement which is decreased by 3.5 % and a standard deviation which is 35.1 % smaller, when the constraints are applied.

The same can be observed for the 3-leg and 4-leg trajectories. As for the 3-leg missions, the average Δv requirement drops by 19.8 % and the standard deviation by 83.8 % when utilising the constraints. The 4-leg missions show a 23.7 % reduction in the average Δv, while the standard deviation is increased by 11.7 %.

For all types of trajectories can be observed that the application of the constraints reduces the average Δv requirement. The standard deviation decreases for the 2-leg and 3-leg missions but increases for the 4-leg missions. It shall be noted that these result are based on just ten solutions in total, which is why they have to be taken cautiously. It could be the case that these findings do not hold if more calculations had been performed. Nevertheless, the overall trend can be observed, that utilising the constraints with the given parameter set does beneficially influence the solutions.

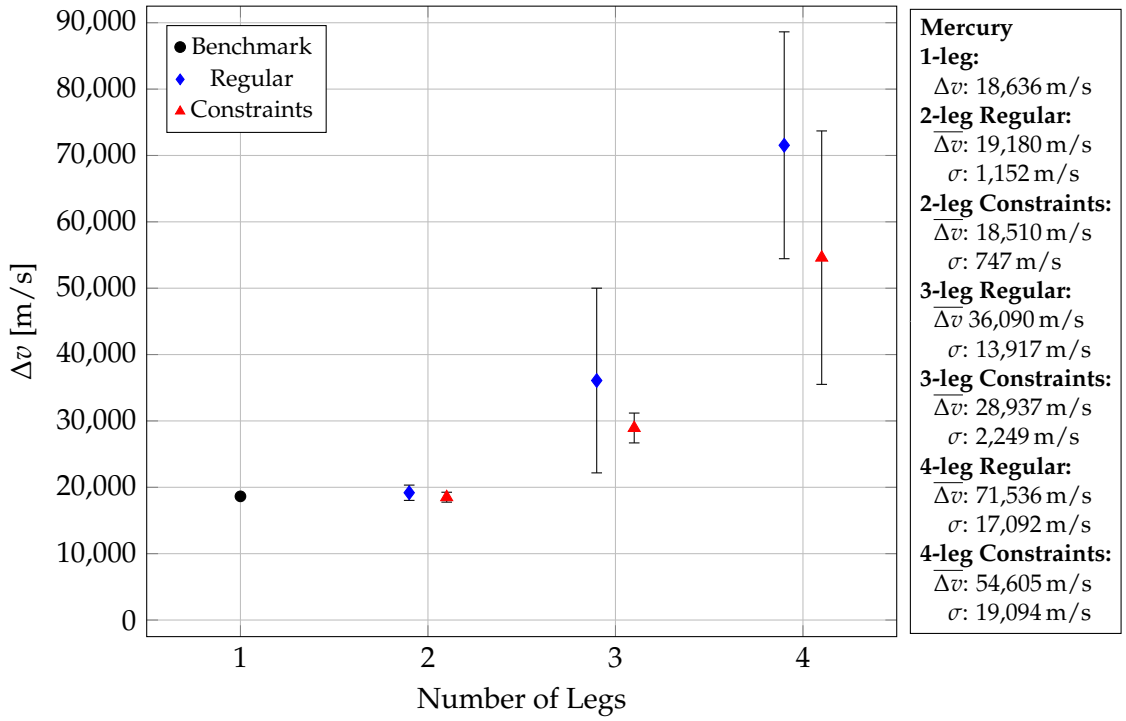

FIGURE 5.7: *Results of the trajectories leading to Mercury. The blue marks depict the solutions produced by the regular search method, the red marks the solutions of the constrained method. The average values and standard deviations of the runs, each consisting of five solutions, are displayed to the right.*

5.3.2 Mars

The results of the calculated trajectories leading from Earth to Mars are shown in Figure 5.8. The 1-leg trajectory requires 5,584 m/s of Δv and three of the ten 2-leg trajectories have exceeded this benchmark by requiring 5,224 m/s (regular search method) and 5,237 m/s, 4,955 m/s (constrained search method).

Generally can be observed that the application of the constraints reduces the averagely needed Δv for each number of gravity assists. The 2-leg trajectories require in average 10.4 % less Δv. The 3-leg trajectories register a drop of 21.3 % and the 4-leg trajectories 4.1 %. But the standard deviation of the 2-leg and 3-leg trajectories are both increasing by around 5 % when the constraints are applied. The 4-leg trajectories however show a decreasing standard deviation of 31.5 %. The trend seen for the missions to Mercury, that the constraints do positively influence the solution's quality, can thus be also observed for the missions to Mars.

However, a distinct characteristic of the trajectories to Mars is the wide spread between the Δv requirements of the 2-leg and 3-leg trajectories. The 2-leg trajectories are in close proximity to the 1-leg solution and the 3-leg missions require around five times as much Δv as the 2-leg trajectories. For the trajectories to Mercury, this difference is less than two times the Δv requirement. This indicates that missions to Mars do not benefit as much from conducting more than one gravity assist. The calculation of solutions incorporating more than one fly-by manoeuvre are significantly more difficult for a trajectory from Earth to Mars as opposed to a trajectory targeting Mercury.

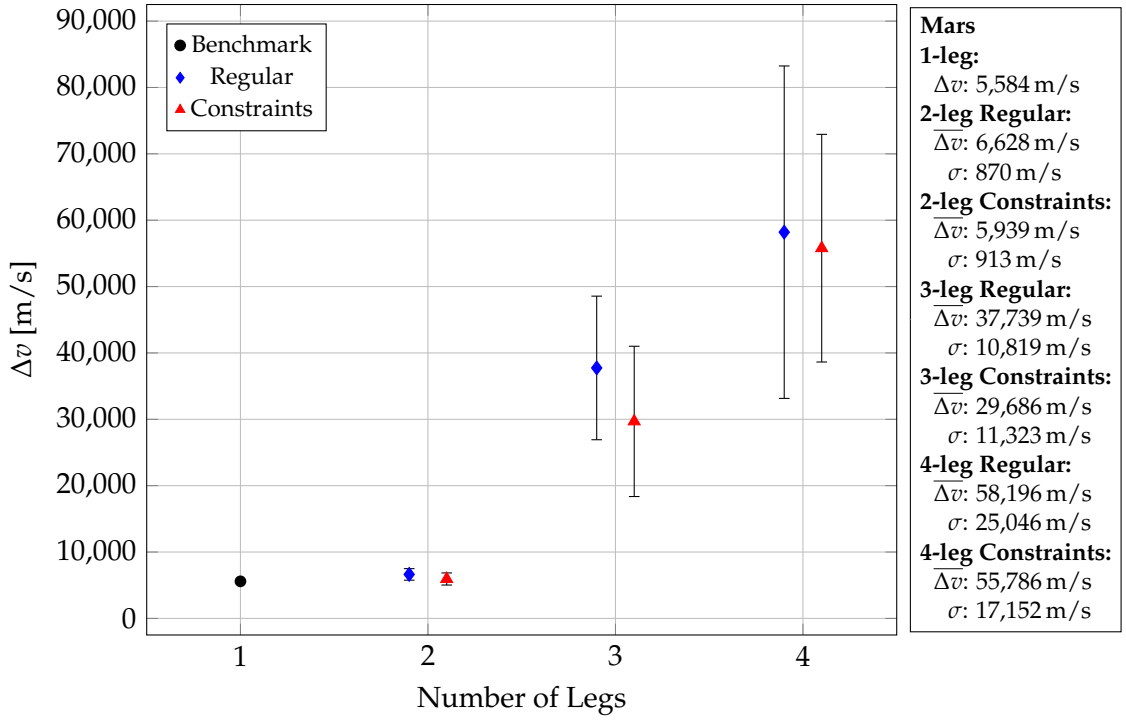

FIGURE 5.8: *Results of the trajectories leading to Mars.*

5.3.3 Saturn

The results produced for the Earth - Saturn mission are displayed in Figure 5.9. The calculated 1-leg trajectory requires 19,034 m/s of Δv. Four 2-leg solutions exceeded this benchmark, three of which were calculated with the regular search method (requiring 17,657 m/s, 18,822 m/s and 18,867 m/s) and one solution was produced while applying the constraints (requiring 18,487 m/s).

This results in an 1 % higher average Δv requirement for the 2-leg trajectories while applying the constraints. However, the standard deviation drops by 56.6 %, as the solutions are significantly closer together. The 3-leg trajectories show a reduction of the average Δv by 35.3 % and a 51.5 % lower standard deviation. The 4-leg solutions require 20.1 % less Δv with a 71.8 % higher standard deviation.

Of the ten 4-leg trajectories, five were found which did not converge to an optimum. These trajectories require between 800,000 m/s and 5,000,000 m/s of Δv. Two non-converging solutions were found while using the regular search method and three while utilising the constraints. To avoid data corruption, these trajectories are discarded for this evaluation.

The benefits of applying the constraints, observed for the Mercury and Mars missions, are identifiable for the Earth - Saturn solutions as well. However, the most distinct difference between using the regular and the constrained search method are found for the 3-leg trajectories. This greater influence of the constraints in reducing the Δv requirement and standard deviation of the 3-leg missions, can also be observed in the trajectories targeting Mercury and Mars.

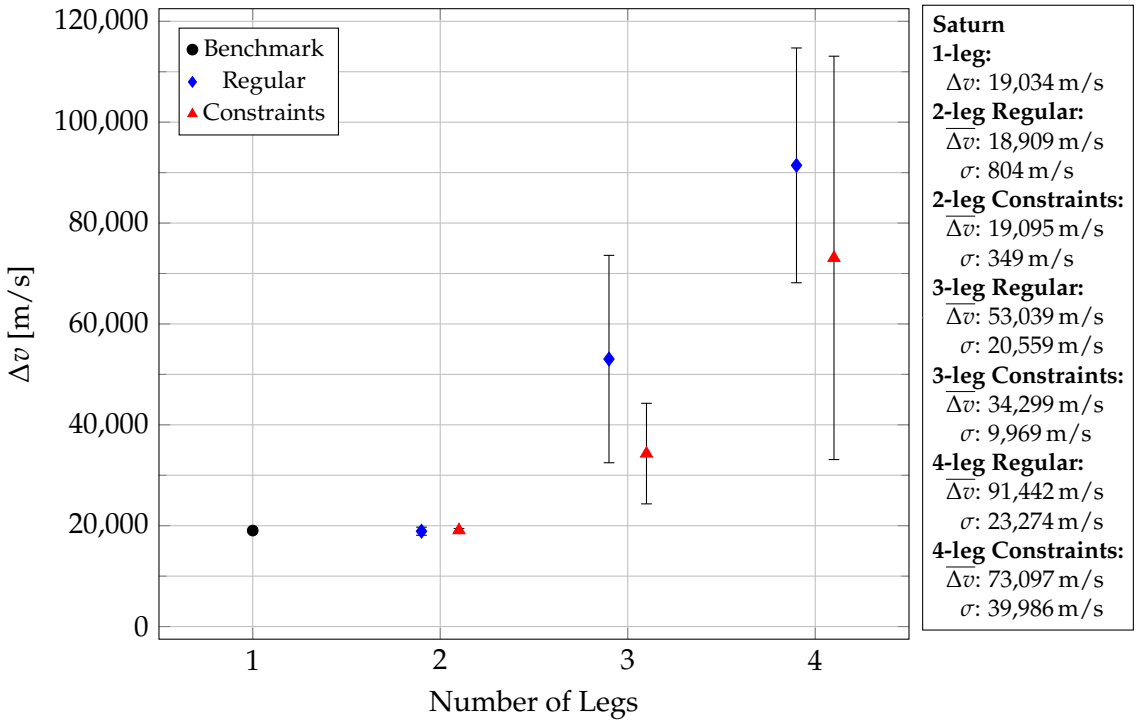

FIGURE 5.9: *Results of the trajectories leading to Saturn.*

5.3.4 Conclusions of the Further Investigations

TABLE 5.2: *Best trajectories of all three additional missions. The Δv requirement and the flight time (ToF) of each trajectory is listed. The solutions produced by application of the constraints are* ***emphasised****.*

	Mercury		**Mars**		**Saturn**	
Legs	Δv in m/s	*ToF*	Δv in m/s	*ToF*	Δv in m/s	*ToF*
1	18,636	563	5,584	533	19,034	4,012
2	17,918	659	***4,955***	***783***	17,657	4,432
3	***26,219***	***789***	***12,951***	***993***	***24,235***	***3,919***
4	***30,234***	***1,455***	30,170	748	***44,823***	***4,397***

Table 5.2 lists the best trajectories for each number of legs produced during the additional simulations. For every trajectory the required Δv and flight time are provided. The six solutions calculated by application of the constraints are emphasised.

It can be seen that the best 2-leg trajectories targeting Mercury and Saturn are calculated by the regular search method. The best 2-leg trajectory to Mars is in turn provided by applying the constraints. For the 3-leg trajectories this is different, as all of the best trajectories are found by using the constrained search method. The best 4-leg trajectories leading to Mercury and Saturn are also found by utilisation of the constraints. The best 4-leg solution for a mission to Mars was found by the regular search method.

These findings support the claim that the constraints have a greater impact on trajectories with more gravity-assist manoeuvres, as most of the best trajectories incorporating more than one fly-by were found by application of the constraints.

Besides that, it can be seen that the 4-leg trajectories require significantly more Δv than the 3-leg trajectories, which in turn require significantly more than the 2-leg missions. Whether the increasing Δv requirement resembles an exponential or a linear growth cannot be clearly assessed, based on the small sample size. Nevertheless, it can be reasoned that the calculation gets linearly or exponentially more complicated for every added trajectory leg.

In summary, these preliminary investigations of different mission profiles showed that the applied constraints, with the given parameter set, generally enabled the calculation of better solutions than the regular search method. This is a great result as it indicates that the constraints might posses globally useful parameter settings which are viable for different mission profiles. However, these preliminary simulation series need to be extended to a similar scale as the investigated Earth - Jupiter trajectories to confirm these observations.

6 Final Remarks

6.1 Open Issues and Outlook

The application of the constraints can have a great impact on the solution's quality. However, during the evaluation of the conducted simulation series some questions remained unanswered. These still open issues and ideas for further investigations are presented in this chapter.

Premature Termination of the Optimisation Process
Currently, the optimisation process is either stopped when the maximum amount of generations is reached, or as soon as over five subsequent generations no better solution is found in the whole population. After five generations without improving the fitness of the solutions, it is assumed that an optimum got located. Most of the 2-leg solutions, produced during the three testing phases, stopped after around 100 to 250 generations. The optimisation of the 3-leg trajectories was generally stopped around 50 generations earlier.

It might be the case that the solution quality can be enhanced by allowing the optimisation process to proceed over more than five generations without improving the fitness of the solutions. It is fair to assume that, especially for the more complex trajectories with three legs, the optimisation is terminated too soon to enable a sufficient exploration of the search space. However, this question remains open and must be further examined in subsequent investigations.

Dynamic Adaptation of the Partner Pool Constraint
The *PP* constraint has on its own a beneficial influence on the quality of the produced solutions. The limitation of the search space of possible next gravity-assist partners thus works as intended. One method to possibly further improve this constraint is to dynamically adapt the allowed subsequent partner planets as the missions proceeds and the spacecraft gets closer to the target.

For example, a mission with three gravity-assist manoeuvres from Earth to Saturn is regarded, where the *PP* constraint is set to *forwards:2* and *backwards:2*, as it has been done in the additional testing series. The gravity-assist sequence is assumed to be Mars - Jupiter - Saturn, where a gravity assist at Saturn is performed before actually reaching the targeted orbit around it. For each fly-by manoeuvre, the next possible partner is located in a static range around the current planet ($\pm$ 2 jumps in this case), regardless of the fact if the planet is already the actual target.

It might be beneficial to dynamically reduce the range in which the next gravity-assist partner can be chosen, if the spacecraft is already in close proximity to the target. In the case of the example, after the second gravity assist which is performed at Jupiter, the setting of the *PP* constraint in its current form, allows the last gravity assist to happen at all planets between Earth and Uranus. If the range is dynamically reduced to just allow gravity assists at Jupiter and Saturn for this last manoeuvre, this might possibly further improve the calculated solution quality, as gravity-assist sequences which are very likely suboptimal get discarded. This extension of the *PP* constraint and the evaluation of the usefulness have to be done in further investigations.

Globally Viable Constraint Parameter
The evaluation of the three additional simulation series in Chapter 5.3, indicated that the chosen constraint parameter settings positively influenced the solution's quality of these different missions. As mentioned in the above chapter, these findings have to be confirmed by conducting a systematically evaluation of different parameter settings and mission profiles. The sample size of five calculations per setting is too small to reliably validate the usefulness of this particular parameter set for the respective mission profile.

To assess whether globally viable constraint parameter settings do exist, further extensive investigations need to be conducted.

6.2 Conclusion

The goal of this thesis is to implement the proposed constraints into the present search method GOLT and to conduct simulations to demonstrate their usefulness. In conclusion can be stated, that these goals are successfully completed. The implemented constraints work in the intended way and can greatly influence the quality of the solutions. In regard of improving the simple to use low-thrust gravity-assist sequencing method, the work done within this thesis contributes to further optimise the search method's performance.

Through the introduction of the constraints, the user is enabled to adjust the settings in a broader range, to better match the specific mission profile. However, the additional input values entail that the process of determining the right settings increases in its complexity. As wrong parameter settings of the constraints might negatively influence the results, preliminary evaluations have to be conducted to define which settings enable the best solutions for a given mission profile. But the possible benefits of using the constraints with the right parameter settings are remarkable.

Bibliography

[1] V. Maiwald. "A New Method for Optimization of Low-Thrust Gravity-Assist Sequences". In: *CEAS Space Journal* (2017).

[2] S. Ulamec and N. Hanowski. "Weltraumastronomie und Planetenmissionen". In: *Handbuch der Raumfahrttechnik*. Ed. by W. Ley, K. Wittmann, and W. Hallmann. Vol. 4. München: Carl Hanser Verlag, 2011. Chap. 7.4, pp. 553–569.

[3] M. Moltenbrey. *Dawn of Small Worlds: Dwarf Planets, Asteriods, Comets*. Springer, 2016.

[4] K. Brieß. "Systementwurf und Systemintegration". In: *Handbuch der Raumfahrttechnik*. Ed. by W. Ley, K. Wittmann, and W. Hallmann. Vol. 4. München: Carl Hanser Verlag, 2011. Chap. 8.2, pp. 632–641.

[5] D. J. Salt. "Space Operations for a NewSpace Era". In: *SpaceOps Conference*. Apr. 2010.

[6] J. Lassmann and M. H. Obersteiner. "Gesamtsysteme". In: *Handbuch der Raumfahrttechnik*. Ed. by W. Ley, K. Wittmann, and W. Hallmann. Vol. 4. München: Carl Hanser Verlag, 2011. Chap. 3.1, pp. 132–149.

[7] W. Ley and F. Huber. "Raumfahrzeug-Subsysteme". In: *Handbuch der Raumfahrttechnik*. Ed. by W. Ley, K. Wittmann, and W. Hallmann. Vol. 4. München: Carl Hanser Verlag, 2011. Chap. 4, pp. 219–220.

[8] H. D. Schmitz. "Satellitenantriebssysteme". In: *Handbuch der Raumfahrttechnik*. Ed. by W. Ley, K. Wittmann, and W. Hallmann. Vol. 4. München: Carl Hanser Verlag, 2011. Chap. 4.4, pp. 313–339.

[9] E. Y. Choueiri. "New Dawn for Electric Rockets". In: *Scientific American* 300 (Feb. 2009), pp. 58–65.

[10] G. A. Rauwolf and V. L. Coverstone-Carroll. "Near-Optimal Low-Thrust Orbit Transfers Generated by a Genetic Algorithm". In: *Journal Of Spacecraft And Rockets* 33.6 (Nov. 1996).

[11] J. Sims and S. Flanagan. "Preliminary Design of Low-Thrust Interplanetary Missions". In: *NASA Technical Report* (1997).

[12] D. Shortt. "Gravity-Assist". In: *The Planetary Society Blog* (27. Sep. 2013).

[13] T. Crain et al. "Interplanetary Flyby Mission Optimization Using a Hybrid Global-Local Search Method". In: *Journal Of Spacecraft And Rockets* 37.4 (July 2000).

[14] Z. Zhao et al. "Low Thrust Gravity Assist Optimization Based on Hybrid Method". In: *IEEE fifth International Conference on Advanced Computational Intelligence (ICACI)*. Nanjing, Jiangsu, China, Oct. 2012.

[15] T. T. McConaghy et al. "Design and Optimization of Low-Thrust Trajectories with Gravity Assists". In: *Journal Of Spacecraft And Rockets* 40.3 (May 2003), pp. 380–387.

[16] B. Wall and B. Conway. "Shape-Based Approach to Low-Thrust Rendezvous Trajectory Design". In: *Journal of Guidance, Control, And Dynamics* 32.1 (Jan. 2009).

[17] A. Petropoulols and J. Longuski. "Shape-Based Algorithm for Automated Design of Low-Thrust, Gravity-Assist Trajectories". In: *Journal Of Spacecraft And Rockets* 41.5 (Sept. 2004).

[18] J. E. Prussing and B. A. Conway. *Orbital Mechanics*. 2nd ed. Oxford University Press, 2012.

[19] G. R. Hintz. *Orbital Mechanics and Astrodynamics - Techniques and Tools for Space Missions*. Springer, 2015.

[20] V. Maiwald. "About Combining Tisserand Graph Gravity-Assist Sequencing with Low-Thrust Trajectory Optimization". In: *6th International Conference on Astrodynamics Tools and Techniques (ICATT)*. Darmstadt, Germany, Mar. 2016.

[21] W. A. Hoskins et al. "30 Years of Electric Propulsion Flight Experience at Aerojet Rocketdyne". In: *33st International Electric Propulsion Conference*. George Washington University, USA, Oct. 2013.

[22] "Timeline and Trajectory of Dawn Spacecraft". In: *NASA - JPL, URL: http://dawn.jpl.nasa.gov/mission/timeline_trajectory.html* (2013).

[23] C. H. Yam, F. Biscani, and D. Izzo. "Global Ooptimization of Low-Thrust Trajectories via Impulsive Delta-V Transcription". In: *27th International Symposium on Space technology and science, ISTS paper*. Tsukuba, Japan, July 2009.

[24] N. J. Strange and J. M. Longuski. "Graphical Method for Gravity-Assist Trajectory Desing". In: *Journal Of Spacecraft And Rockets* 39.1 (2002).

[25] M. Minovitch. "The invention that opened the solar system to exploration". In: *Planetary And Space Science* 58 (Mar. 2010), pp. 885–892.

[26] R. A. Broucke. "The Celestial Mechanics Of Gravity Assist". In: *AIAA 88-4220* (Aug. 1988).

[27] J. Miller and C. Weeks. "Application of Tisserand's Criterion to the Design of Gravity Assist Trajectories". In: *American Institute of Aeronautics and Astronautics*. Aug. 2001.

[28] V. Maiwald. "Applicability of Tisserand's Criterion for Optimization of Gravity-Assist Sequences for Low-Thrust Missions". In: *66th International Astronautical Congress (IAC)*. Jerusalem, Israel, Sept. 2015.

[29] A. F. Heaton et al. "Automated Design of the Europa Orbiter Tour". In: *Journal Of Spacecraft And Rockets* 37.1 (2002), pp. 17–22.

[30] E. A. Rinderle. *Galileo User's Guide, Mission Design System, Satellite Tour Analysis and Design Subsystem*. Rept. JPL D-263. Pasadena, CA, July 1986.

[31] D. Ashlock. *Evolutionary Computation for Modeling and Optimization*. Springer, New York, NY, 2006.

[32] B. Dachwald. "Low-Thrust Trajectory Optimization and Interplanetary Mission Analysis Using Evolutionary Neurocontrol". PhD thesis. Universität der Bundeswehr München, 2004.

[33] D. Shiffman. *The Nature of Code*. 2012.

[34] K. Weicker. *Evolutionäre Algorithmen*. Vol. 2. Teubner, Wiesbaden, 2007.

[35] R. Storn and K. Price. "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces". In: *Journal of Global Optimization* 11 (1997), pp. 341–359.

[36] K. V. Price, R. M. Storn, and J. A. Lampinen. *Differential Evolution - A Practical Approach to Global Optimization*. Springer, 2005.

[37] E. Minisci, M. Vasile, and M. Locatelli. "Analysis of Some Global Optimization Algorithmus for Space Trajectory Design". In: *Journal Of Spacecraft And Rockets* 47.2 (Mar. 2010).

A Settings

Within this appendix, the complete settings file is presented and explained. Furthermore, the settings of the three testing phases are listed below.

Contents of the Settings File

General Settings

start_body	Planet at which the trajectory origins
end_body	Planet at which the trajectory ends
r_min	Minimum allowable distance from the barycentre in AU
launch_date	Date of launch in MJD
launch_window	Window of launch in days
flight_time_max	Maximum flight time in days
flight_time_min	Minimum flight time in days
accuracy	Relative difference between the set and the calculated flight time
max_thrust	Maximum allowed thrust acceleration in m/s^2
nrev_max	Maximum number of revolutions around the barycentre
step_no	Number of steps, every trajectory leg is approximated by
fitness_weight	Weight of Δv over flight time in calculation of the fitness
v_inf_start	Initial hyperbolic excess velocity in m/s
v_inf_end	Hyperbolic excess velocity at arrival at the target planet in m/s
N_GA_max	Maximum number of gravity assists
Debug_Mode	The progress of the calculation is displayed via the console if this option is *true*. Setting it to *false* prohibits this, leading to an increasing performance of around 45 %.

Differential Evolution Algorithm Settings

N_pop	Size of the population (recommended: at least 10 times the number of control variables $(3 + N\_GA\_max) \cdot 10$)
Gen_max	Maximum number of generations
Crossover_Constant	Value of the crossover constant Cr, between 0 and 1
Diff_Weight	Value of the differential weight W, between 0 and 2

Constraint Settings

delta_v_gain	*true* activates the Δv-gain constraint
diff_pericenter	Value of d_{peri} between 0 and 100
diff_V_inf	Value of d_{velo} between 0 and 100
partner_pool	*true* activates the Partner Pool constraint
forwards_diff	Allowed jumps in direction of flight
backwards_diff	Allowed jumps against the direction of flight

Setting Files of the Testing Phases

Parameter	**Phase 1**	**Phase 2**	**Phase 3**
start_body	Earth	Earth	Earth
end_body	Jupiter	Jupiter	Jupiter
r_min	0.25	0.25	0.25
launch_date	56,000	56,000	56,000
launch_window	100	360	360
flight_time_max	3,000	3,000	3,000
flight_time_min	1,000	2,000	2,000
accuracy	0.000,1	0.000,1	0.000,1
max_thrust	0.000,15	0.000,4	0.000,4
nrev_max	3	4	4
step_no	100	100	100
fitness_weight	60,000	20,000	20,000
v_inf_start	1,000	0	0
v_inf_end	50	0	0
N_GA_max	2	1	2
Debug_Mode	false	false	false
N_pop	100	100	200
Gen_max	200	1,000	1,000
Crossover_Constant	0.75	0.75	0.75
Diff_Weight	1	1	1
delta_v_gain	true/false	true/false	true/false
diff_pericenter	0 to 50	0 to 100	20
diff_V_inf	0 to 50	0 to 100	20
partner_pool	true/false	true/false	true/false
forwards_diff	1 to 2	1 to 2	1
backwards_diff	1 to 2	1 to 2	1

B Specification Test Computer

As a reference point to bring the computation time of the different testing phases into context, the specifications of the test system are listed below. As mentioned in the respective chapters, the calculations per trajectory either took 4 hours (Phase 1), 2 hours (Phase 2) or 12 hours to 15 hours (Phase 3). As the program does not provide multi-threading compatibility one instance of the program can be executed per CPU core without performance losses. The most important specification to assess the computation time is the single-core performance of the CPU. To provide a benchmark, the Geekbench 4 Single-Core Score of the test system is listed additionally.

CPU	Intel Core i5-3550 @ 3.30 GHz (4 logical processors) (Geekbench 4 Single-Core Score: 3790)
RAM	8 GB DDR3 SDRAM 666 MHz
OS	Windows 10 Pro 64-bit
Mainboard	Gigabyte H77-D3H
GPU	Radeon HD 7850