Automated Multiple Gravity-Assist Sequence Optimisation

An intelligent parallel-computing methodology

Sean Cowan

Delft University of Technology

TU Delft

Image on titlepage created with DALL-E OpenAI.

Automated Multiple Gravity-Assist Sequence Optimisation

An intelligent parallel-computing methodology

by

Sean Cowan

to obtain the degree of

Master of Science

in Aerospace Engineering

at Delft University of Technology,

to be defended publicly on Wednesday June 28th, 2023 at 15:00.

Student number:	4687493	
Project duration:	18/07/2022 - 28/06/2023	
Thesis committee:	Dr.ir. E. Mooij	Committee chair
	Ir. R. Noomen,	Supervisor
	Dr.ir. E. van Kampen	External examinator

TU Delft

Preface

I want to start by thanking Ron, my daily supervisor, for the many meetings that guided me along this journey. I thank the other members of my graduation committee, Erwin and Erik-Jan, for taking the time to witness my graduation and assess my report. A special thanks is also due to my secret agent in the faculty, I would not be here without him. In this past year, I have sat in many places, but only a few will remain in my memory. Room 2.38, I will never forget the spaceship-like humming of the ventilation. Many thanks to Vassili for the many fruitful and fun conversations. The library silent room is decidedly not the most inviting box I have worked in, but it treated me well. I thank Irmak for the countless coffees and refreshing breaks upon the library hill on warm summer days – and mostly cold rainy ones. I thank all my loved ones for, well, their love, but also all their support that has allowed me to be where I am today; I will be forever grateful.

It has been a busy, interesting, and eventful period: alongside the thesis project, I participated in various other activities. First and foremost, I worked as a TA for Tudat since November 2022, translating into 24 weeks of work at an average of eight hours per week. Second, I attended the Clean Space Training Course 2023 organised by ESA; this is a week-long training course on the lifecycle design of satellites. Third, I have taken three vacation weeks throughout the thesis process, which were helpful to get my mind off things and return with a fresh perspective. Besides these activities, I have participated in numerous seminars and other multi-day events: I attended a COMET-ORB seminar, the Solaris Research Technology Day, the Delftse Bedrijvendagen and countless graduations. I also started a job at Ecosmic, a start-up originating from this faculty, which has delayed the graduation date by more than a month.

To all, I wish you a pleasant read.

Sean Cowan
Delft, June 2023

Summary

While the space industry expands rapidly and space exploration becomes ever more relevant, this thesis aims to automate the design of space missions. This work focuses on optimising Multiple Gravity-Assist (MGA) interplanetary transfers using low-thrust propulsion technology. In particular, during the preliminary design phase of space missions, where all candidate MGA sequences are still considered, the combinatorial complexity is large and current optimisation approaches require extensive experience and can take days to simulate. Therefore, a novel optimisation approach is proposed and developed that uses the hodographic-shaping low-thrust trajectory representation together with a combination of tree-search methods to automate the optimisation of MGA sequences. The approach includes novel figures of merit as well as parallelisation concepts to increase the robustness and accelerate the convergence.

The approach developed in this thesis is called Recursive Target Body Approach (RTBA). This approach uses nested loops, where the inner loop is the Low-Thrust Trajectory Optimisation (LTTO), and the outer loop is the Multiple Gravity-Assist Sequence Optimisation (MGASO). The LTTO process is performed using a genetic algorithm applied to hodographic shaping combined with a simplified patched conics approach. An extensive tuning process is performed to improve the quality of the LTTO and recreate a selection of MGA trajectories from literature: the genetic algorithm parameters, the bounds of the optimisation problem, and the hodographic-shaping parameters are tuned. Tuning improved the quality of the MGA trajectories substantially and as a result, a robust LTTO could be ensured. Parallelisation is introduced in the form of a topology of optimisation processes and a local optimisation step is performed using the Nelder-Mead Simplex algorithm.

The outer loop is the main focus of development: a recursive set of steps is executed for each layer of Gravity-Assist (GA) manoeuvres in a combinatorial tree. A set of MGA sequences is created per available GA candidate, a topology is created to link parallel-computed processes and enable the migration of individuals, and the optimisation is run. After each recursion, the optimised sequences are represented by the fitness of each GA candidate. This fitness is defined by the minimum and mean of the ΔV of MGA sequences. The best GA candidate is subsequently added to a pseudo sequence representing the best GA candidates at each recursion, after which the next recursion begins. This pseudo sequence is extended after each recursion and should converge toward the optimal MGA sequence within the definition of the problem. An Earth-Jupiter transfer with a maximum of three gravity assists is considered as a test problem.

To ensure the quality of the sequence optimisation, verification and validation steps are considered. The hodographic-shaping method is verified for single legs as well as for MGA trajectories, and the genetic algorithm is verified. A complete validation is not possible due to the lack of data. However, by using software features that have been validated, the method can also be considered partially validated. The results consist of a ranking of the optimality of MGA sequences as well as the optimal pseudo sequence. It is found that the RTBA performs well in ranking MGA sequences. The ΔV values found in LTTO verification runs are similar to those in the sequence ranking, indicating accurate MGA sequence fitness values. Moreover, a distinct group of highly fit MGA sequences is consistently found that can be passed to a higher-fidelity method. The pseudo sequence converges to MGA sequences that are theoretically expected to be the best. However, a strong dependence is found on the number of Mercury and less so Venus transfers evaluated during the optimisation of the test problem, thereby limiting the robustness of the pseudo sequence. In conclusion, the RTBA can automatically and reliably be used for the preliminary optimisation of low-thrust MGA trajectories. Yet, due to a lack of references, a full verification of the RTBA remains to be done and various improvements are proposed as future recommendations.

Contents

Nomenclature

Abbreviations

Abbreviation	Definition
ACO	Ant Colony Optimisation
AI	Artificial Intelligence
BGA	Binary Genetic Algorithm
BSS	Beam Search Strategy
CPU	Central Processing Unit
DSM	Deep Space Manoeuvre
DSMP	Dynamic-Size Multiple Populations
EOM	Equations of Motion
GA	Gravity Assist
GACO	Extended Ant Colony Optimisation
GIL	Global Interpreter Lock
GIM	Generalised Island Model
GPM	Gauss Pseudospectral Method
GPU	Graphics Processing Unit
GTO	Geostationary Transfer Orbit
GTOP	Global Trajectory Optimisation Problem
HGGA	Hidden-Genes Genetic Algorithm
HOCP	Hybrid Optimal Control Problem
ICRF	International Celestial Reference Frame
IHS	Improved Harmony Search
RPS	Recursive Pseudo Sequence
LTT	Low-Thrust Trajectory
LTTO	Low-Thrust Trajectory Optimisation
MBH	Monotonic Basin Hopping
MCTS	Monte-Carlo Tree Search
MGA	Multiple Gravity Assist
MGASO	Multiple Gravity-Assist Sequence Optimisation
MHACO	Multi-Objective Hypervolume-based Ant Colony Optimisation
MINLP	Mixed-Integer Non-Linear Program
NLP	Non-Linear Program
NSGA2	Non-dominated Sorting Genetic Algorithm 2
PaGMO	Parallel Global Multiobjective Optimiser
PSO	Particle Swarm Optimisation
PTB	Pre-defined Target Body
PyGMO	Python Parallel Global Multiobjective Optimiser
RTBA	Recursive Target Body Approach
SGA	Simple Genetic Algorithm
SOI	Sphere Of Influence

SSB	Solar System Barycenter
TNW	Tangential, Normal, and angular momentum frame
ToF	Time of Flight
Tudat	TU Delft Astrodynamics Toolbox
UCT	Upper-Confidence bounds for Trees
UDA	User-Defined Algorithm
UDI	User-Defined Island
UDP	User-Defined Problem
VSDS	Variable-Size Design Space
2BP	Two-Body Problem

Symbols

Symbol	Definition	Unit
	Roman	
a	Semi-major axis	[m]
A	Set of islands representing a sequence	[-]
c_{eff}	Effective exhaust velocity	[m·s^{-1}]
nc	Number of CPUs	[-]
ct	CPU time	[s]
C	Total combinatorial complexity	[-]
e	Eccentricity	[-]
f	Generic function	[-]
f	Thrust acceleration	[m·s^{-2}]
f_s	Fitness value of sequence	[m·s^{-1}]
f_X	Fitness value of PTB	[m·s^{-1}]
$\vec{f}$	Perturbation acceleration	[m·s^{-2}]
fpc	Free parameter count	[-]
F	Function value	[-]
$\vec{F}$	Perturbation force	[kg·m·s^{-2}]
g	Generic function	[-]
gc	Generation count	[-]
G	Universal gravitational constant	[6.6743·10^{-11} m^3·kg^{-1}·s^{-2}]
$\vec{h}_{pl}$	Angular momentum vector	[kg·m^2·s^{-1}]
i	Index of a body	[-]
$\hat{i}$	Unit vector of x-axis in local frame	[-]
ips	Islands per sequence	[-]
I_{sp}	Specific impulse	[s]
j	Index of a body	[-]
$\hat{j}$	Unit vector of y-axis in local frame	[-]
k	Index of a body	[-]
$\hat{k}$	Unit vector of z-axis in local frame	[-]
K	Hodographic shaping coefficients	[-]
m	Mass of a body	[kg]
m	Order of base functions	[-]

m	Number of possible GA candidates	[-]
m	Integer dimension	[-]
m_f	Delivery mass	[kg]
m_s	Mass of the spacecraft	[kg]
mng	Maximum number of GAs	[-]
M	Mass of the attracting body	[kg]
M	Current mass of the propelling body	[kg]
M_0	Mass of the propelling body at t_0	[kg]
n	Maximum number of GAs remaining	[-]
n	Number of mass points	[-]
n	Order of base functions	[-]
n	Total dimension	[-]
N	Number of revolutions	[-]
P	Boundary condition on position	[m]
$\tilde{P}$	Pseudo period	[s]
q	Fraction of recursion step	[-]
Q	Total fraction evaluated	[-]
r	Radius	[m]
r_a	Apocenter radius	[m]
r_p	Pericenter radius	[m]
$\vec{r}$	Position vector	[m]
rt	Run time	[s]
spp	Number of sequences per GA candidate	[-]
sr	Number of sequence recursions	[-]
S	Set of sequences	[-]
t	Time	[s]
t_0	Initial time	[s]
tbo	Transfer body order	[-]
u	Dependent variable for base functions	various
$\hat{u}$	Unit vector for x-axis in TNW frame	[-]
v	Base functions for hodographic shaping	[m·s^{-1}]
$\hat{v}$	Unit vector for y-axis in TNW frame	[-]
V	Velocity	[m·s^{-1}]
V	Boundary condition on velocity	[m·s^{-1}]
$\vec{V}_{pl}$	Planet velocity vector	[m·s^{-1}]
$\vec{V}_r$	Radial velocity vector	[m·s^{-1}]
$\vec{V}_z$	Axial velocity vector	[m·s^{-1}]
$\vec{V}_{\infty,in}$	Incoming hyperbolic velocity vector	[m·s^{-1}]
$\vec{V}_{\infty,out}$	Outgoing hyperbolic velocity vector	[m·s^{-1}]
$\vec{V}_\theta$	Normal velocity vector	[m·s^{-1}]
$\vec{V}^h_{in}$	Incoming heliocentric velocity vector	[m·s^{-1}]
$\vec{V}^h_{out}$	Outgoing heliocentric velocity vector	[m·s^{-1}]
ΔV	Velocity change	[m·s^{-1}]
$\hat{w}$	Unit vector for z-axis in TNW frame	[-]
x	Position component	[m]
$\vec{x}$	State vector	various
y	Position component	[m]

z	Position component	[m]
	Greek	
α	Alphabetic character pair	[-]
β	Orbit orientation angle	[rad]
δ	Deflection angle	[rad]
η	CPU efficiency	[-]
μ	Gravitational parameter	[$m^3 \cdot s^{-2}$]
θ	Polar coordinate	[rad]
θ	In-plane angle	[rad]
ϕ	Out-of-plane angle	[rad]
ψ	Transfer angle	[rad]
	Notation	
$\dot{\square}$	First derivative w.r.t. time t	
$\ddot{\square}$	Second derivative w.r.t. time t	
$\vec{\square}$	Vector	
$\hat{\square}$	Unit vector	
	Super- and Subscript	
$\square_{dep}$	Departure condition	
$\square_{arr}$	Arrival condition	
$\square_{g}$	A GA-specific quantity	
$\square^{h}$	Heliocentric representation	
$\square_{i}$	A leg-specific quantity	
$\square_{in}$	Incoming condition	
$\square_{out}$	Outgoing condition	
$\square_{pl}$	Relating to a planet	
$\square_{\infty}$	Hyperbolic condition	
$\square_{k}$	Index for a recursion level	
$\square_{p}$	Index equal to the number of islands	
$\square_{cart}$	Cartesian state	
$\square_{cyl}$	Cylindrical state	

1 Introduction

As of 2023, large-scale interplanetary space missions are reaching the periphery of the Solar System. These missions are crucial for gaining insight into the origins of the Solar System and the potential for life beyond Earth. However, these missions are also extremely challenging: their duration, cost, and complexity require a tremendous amount of engineering design and technological innovation. Trajectory design and propulsion technology play a considerable role in this challenge. The state-of-the-art chemical propulsion systems cannot provide enough thrust to reach these distant places without assistance. To address this, Gravity Assist (GA) manoeuvres have been discovered and efficient low-thrust technologies have been developed to help a spacecraft gain enough energy to reach its target. These developments allow for more design freedom but constitute a formidable optimisation problem, which is the basis for this thesis.

To design Multiple Gravity-Assist (MGA) sequences, engineers started by using Tisserand graphs and experience-based initial guesses. A more modern and desirable approach is to solve this problem through computational optimisation and automation. Optimisation algorithms have been developed and applied to the problem of MGA sequencing using low-thrust transfers with mixed results: current optimisation methods have high run times and few verification options. As for the low-thrust technologies, numerous successful missions in the past decade have demonstrated and are currently demonstrating the potential of low-thrust engines with the likes of BepiColombo and DART. This potential is high due to the large exhaust velocities as compared to high-thrust alternatives, resulting in more efficient thrusting in terms of propellant mass usage.

The existing approaches to fully automate the MGA sequencing process in combination with low-thrust propulsion require extensive run times due to the combinatorial complexity of the problem in addition to the multi-dimensional low-thrust trajectory optimisation problem. Progress can be made on the performance of this particular aspect of trajectory optimisation problems. Specifically, the sequence optimisation is generally not tuned properly to the fidelity of the low-thrust trajectory optimisation (LTTO), and as a result, computational time is wasted. To reduce this, a novel approach is proposed that employs various optimisation techniques and parallelisation paradigms.

In particular, a nested-loop approach is chosen for the optimisation of low-thrust MGA trajectories. The inner loop performs an LTTO for specific MGA sequences. The outer loop explores the combinatorial search space and optimises the MGA sequence. A tree-search method is proposed that combines aspects of various methods to increase convergence, named the Recursive Target Body Approach (RTBA). The Generalised Island Model (GIM) is applied in a unique way to allow for higher statistical confidence in the quality of a single sequence by evaluating it multiple times in parallel. This increase in confidence and convergence can be realised with the hodographic-shaping method, which is expected to provide a suitable fidelity level for the accuracy required to rank optimal MGA sequences. Furthermore, a novel quantification of sequence optimality is chosen, to allow for a more informed decision.

Having established the problem and aim of this thesis, a research question is defined:

How can the MGA sequencing problem using low-thrust trajectories be tackled with an automated parallel-computing optimisation approach for preliminary trajectory design?

Several accompanying sub-questions are defined to break the overarching question into more processable blocks:

- How should low-thrust trajectories be represented to guarantee the accurate assessment of the performance of an MGA sequence?
- How can parallelisation be used to assist the optimisation process?
- What metric should be used for the quality of a sequence to increase the robustness of the sequence optimisation?
- How effective is the developed methodology in optimising a low-thrust MGA sequencing problem?

This thesis starts with this introduction, being the first chapter of this report. It is split into six parts. The fundamentals of the topics covered are presented in Part I. This part starts with Chapter 2 which lays out the relevant previous research in this field. A few key concepts are explained such as an introduction to graph theory. This chapter is followed by Chapter 3 which lays the astrodynamical groundwork for describing spacecraft motion as well as the modelling of gravity-assist manoeuvres. Finally, Chapter 4 discusses low-thrust propulsion as a technology and the hodographic-shaping method as a trajectory model. The next part, Part II, discusses the inner loop of the optimisation and contains two chapters: Chapter 5 describes the structure of the low-thrust optimisation problem and Chapter 6 discusses the tuning process of the optimisation problem. After the inner loop, the outer loop is developed in Part III. The two chapters are analogous to the last two chapters, defining the outer-loop optimisation problem in Chapter 7 with the development of the RTBA approach on the one hand, and the pruning and tuning of the sequence optimisation implementation in Chapter 8 on the other hand. Subsequently, the results of this approach are presented in Part IV, consisting of the verification and validation of the various models in Chapter 9 and the results in Chapter 10. Finally, in Part V, conclusions are drawn in Chapter 11 and recommendations for future work are made in Chapter 12. In addition, Part VI includes numerous appendices.

Part I

Fundamentals

2

Heritage and computational tools

In this chapter, the heritage is discussed as well as the tools that are typically used. In particular, the heritage of the MGASO is considered first, after which the software and hardware tools are determined and the usage of parallelisation is introduced.

2.1. MGA sequence optimisation heritage

In this section, an introduction is given to the various approaches used for solving the MGASO problem together with relevant literature and theory.

2.1.1. Introduction

A plethora of MGASO methods exist, some of which are more relevant than others. This subsection introduces MGA sequencing and a selection of the relevant literature.

Literature exists in abundance on MGA sequencing that is experience-based [Crain et al. 2000; Debban et al. 2002] – one part of which relies on Tisserand graphs [Strange and Longuski 2002]. Adaptations have been made to extend MGA sequencing to low-thrust optimisation applications [Maiwald 2017]. The slow and experience-based basis for MGA sequence design led to the development of an automated design approach. For this, there are two distinct optimisations; the LTTO, which optimises the trajectory for a single MGA sequence, and the MGASO, which optimises the MGA sequences. These two optimisations lead to two types of top-level optimisation methods: single-loop and nested-loop. The loop refers to the optimisation loop. The first type combines the LTTO with the MGASO into a single loop. The second type – known as nested-loop – splits the two optimisation processes into nested loops, where for each sequence a corresponding inner LTTO loop is executed. For each of these approaches, various algorithms can be applied to solve the optimisation: stochastic algorithms, deterministic algorithms, and tree-search methods. This latter option mainly focuses on integer or discrete optimisation, which is also very relevant to this thesis and is discussed further in Section 2.1.4.

The literature available on LTTO and MGASO is far wider than the two approaches mentioned before, so the selection of relevant literature has to be constrained. First, this thesis only considers low-thrust propulsion, discussed in Chapter 4, meaning that the literature on high-thrust propulsion is less relevant. However, the combinatorial part of the problem remains identical and in some cases may be applicable. Second, many approaches exist that look at low-thrust trajectory optimisation, though most of these techniques aim at a different fidelity level – an overview of which is provided in Figure 2.1. The techniques are either not accurate enough and employ singular Lambert arcs, or are too accurate: numerically propagating the low-thrust trajectory while taking account of many perturbations. This thesis considers a specific fidelity level for the low-thrust trajectories depicted in Figure 2.1 below the red circle by the 'Shape-based Methods'. It should be noted that the fidelity level is only an indication.

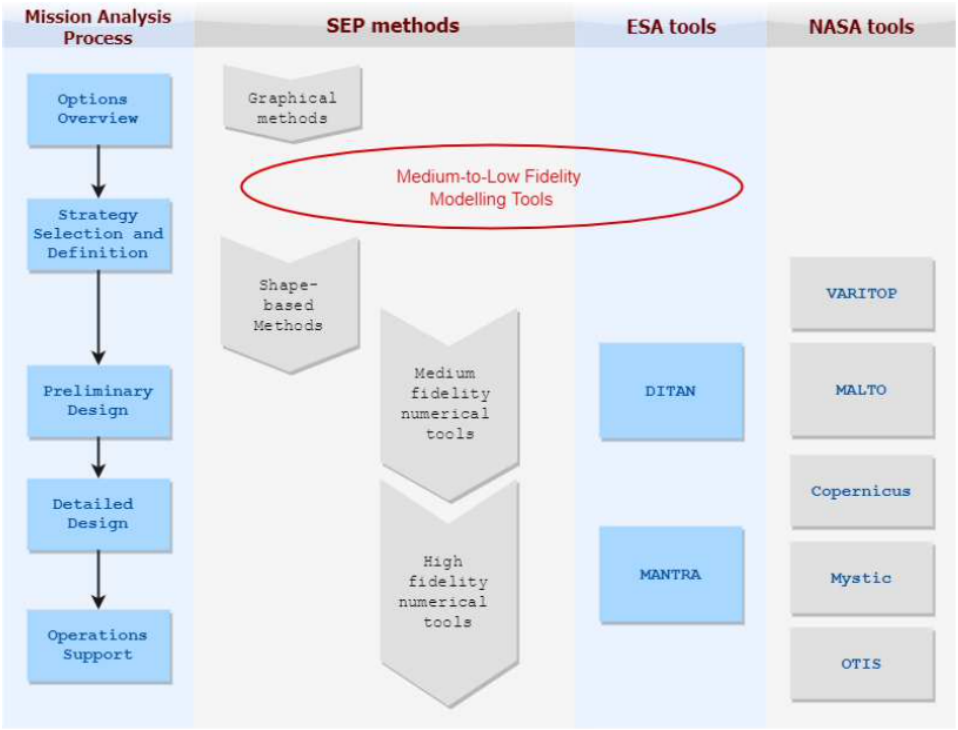


Figure 2.1: Overview of different fidelity levels [Galletti 2017].

Third, some papers use Artificial Intelligence (AI) [Dachwald 2004; Carnelli et al. 2009; Dachwald and Ohndorf 2019; Kranen 2019], but due to the training data requirements, the unavailability of training data for MGASO for learning, and a steeper learning curve compared to conventional optimisation algorithms, this genre of optimisation is also not considered in this thesis. Fourth, some algorithms consider multi-asteroid rendezvous missions [Zhang et al. 2015; Massari and Wittig 2015]. The theory behind these tasks is relevant, though the results themselves are not considered as this thesis only discusses a simplified combinatorial space limited to planets as possible GA targets. Last but not least, some papers use multiple optimisation approaches in sequential phases of the optimisation [Morante et al. 2019; Bellome et al. 2021; Ueda and Ogawa 2021], which are referred to as multi-fidelity methods. These papers implement a multi-step method to approximate low-order solutions followed by a high-accuracy optimisation process. The two separate fidelity levels used in multi-fidelity methods often do not correspond with the fidelity level required for the objective of the optimisation in this thesis: they often consist of graphical or fully analytical methods combined with high-fidelity numerical tools. Therefore they are not discussed further.

To summarise, the remaining relevant scopes of literature are single-loop and nested-loop optimisation, and by extension also tree-search methods. These scopes are discussed in the next subsections.

2.1.2. Single-loop optimisation

This subsection shortly discusses the single-loop approach, its pros and cons, as well as examples of its implementation in literature.

Single-loop optimisation methods use one design space – and one design variable vector for the optimisation of MGA sequences. This approach comes with some advantages, the most prominent of which is that there is no nested looping: a basic time-complexity paradigm that cannot always be avoided. This property allows for much simpler code. The design variable vector, consequently, includes the encoded MGA sequence as well as the variables that are required by the LTTO process. This has a substantial disadvantage as well: the design variable vector does not only include both continuous and integer variables but is also extensive and includes many dependencies. The optimal Time of Flight (ToF) variables depend on the number of revolutions, the flyby body, and shaping parameters that were chosen in that specific run, for example. In addition, the number of flyby's and with that the number of

design variables also vary. These dependencies make it challenging for any optimisation algorithm to converge. Furthermore, not all optimisation methods are compatible with Mixed-Integer Non-Linear Programming (MINLP). For example, a gradient-based optimisation method cannot readily optimise any categorical variables: the gradient from planet Earth to planet Mars as a flyby candidate is nonsensical. This constrains the choice of an optimisation algorithm.

As mentioned before, the design variable vector combines the categorical space of MGA sequences with that of continuous variables related to the definition of several low-thrust legs. This leads to two challenges. On the one hand, the design variable vector, and thus the design space, is variable in size – in literature referred to as a Variable-Size Design Space (VSDS): the MGA sequence determines the number of ToF and revolution variables, among others. On the other hand, the MGA sequences have to be encoded into discrete values. Moreover, based on the assumption that PyGMO is used as an optimisation software due to its various advantages, which does not have the option for categorical variables at the time of writing, this implementation would be highly impractical. The choice of software is discussed in Section 2.2.2.

Several encoding methods have been implemented to solve the VSDS problem: the Hidden-Genes Genetic Algorithm (HGGA) [Gad and Abdelkhalik 2011], Null Genes [Englander et al. 2012], Leading Bit [Chilan and Conway 2013], and Dynamic-Size Multiple Populations (DSMP) [Abdelkhalik and Gad 2012]. The HGGA uses an extra 'hidden' string of binary digits – one per variable – to determine if the variable is 'activated'. The Null Genes method uses a single digit for each planet, with extra digits assigned to 'Null', which corresponds to no gravity assist. The Leading Bit method defines a binary bit that determines the start of the relevant length of the binary string that represents the MGA sequence, enabling the definition of various input combinations. The DSMP splits a population into multiple sub-populations, where the sub-populations have a fixed design variable vector length. This method, developed by [Abdelkhalik and Gad 2012], is not particularly novel in its approach to solve the VSDS problem: it allows for multiple populations. A follow-up paper [Nyew et al. 2015] expands to a more general case that can match more complex problems. In [Cowan 2022], the null genes encoding technique – shown in Table 2.1 as an example – was determined to be the best out of these options.

Table 2.1: Integer codes for target bodies [Englander et al. 2012].

Integer code	Body
0	Null
1	Mercury
2	Venus
3	Earth
4	Mars
5	Jupiter
6	Saturn
7	Uranus
8	Neptune
9-15	Null

Tree-search methods are theoretically applicable to single-loop optimisation problems, though the physically continuous quantities would have to be discretised. This conversion brings other disadvantages, such as rounding which introduces a bias and therefore unreliable results. All in all, the single-loop approach is capable of finding an optimal MGA sequence. However due to the dependencies, the expected need for an extensive tuning process, and the deciphering of the dependencies, the nested-loop approach is expected to be more reliable for convergence and robustness of results.

2.1.3. Nested-loop optimisation

The other main type of optimisation considered is the nested-loop approach. This subsection discusses this approach analogously to the previous subsection, with some theoretical elaboration as well as some

references that are useful for the approach developed in this thesis.

The nested-loop optimisation approach splits the optimisation into continuous and discrete variables. This split is intuitive for the MGASO application because the quality of an inner-loop LTTO can be summarised by a single quantity, which then defines the fitness of an MGASO individual. In this thesis, however, due to the use of the hodographic-shaping method and its requirement for the definition of the number of revolutions, the inner loop concerning the LTTO also includes an integer quantity and is therefore still an MINLP problem. One could add the number of revolutions to the outer loop, though this obfuscates the distinction between the LTTO and the MGASO problems. Therefore, the outer loop is still subject to the VSDS and accompanying constraints if implemented using evolutionary algorithms, whereas the inner loop has a constant-size design space. Another disadvantage of nested-loop approaches is that the nature of the optimisation with nested loops can result in extensive run times. Numerous optimisation methods are used for both the outer and inner loops in literature, a few of which are presented below.

Typically, a genetic algorithm is used for the outer loop method [Englander et al. 2012; Chilan and Conway 2013; Englander and Conway 2017]. This is due to its relative simplicity and the vast collection of literature available. The inner loop of these papers has been implemented using either Monotonic Basin Hopping (MBH) or Particle Swarm Optimisation (PSO).

In particular, [Bellome et al. 2020] implements the Tisserand Criterion without low-thrust corrections. An enumerative approach is then used to determine reachable bodies. The inner loop uses PSO. The patched-conics approach is used with the zero Sphere Of Influence (SOI) assumption, Lambert arcs, and Deep Space Manoeuvres (DSM's). [Bellome et al. 2021] implements these results into a software package called ASTRA, which adds an extra high-fidelity step.

[Englander et al. 2012; Englander and Conway 2017] are cited frequently. As mentioned before, these papers use a binary Genetic Algorithm (BGA) for the outer-loop optimisation using 'Null genes' to solve the VSDS problem. The former only implements high thrust, and the inner-loop approach is therefore not relevant. The latter expands the capabilities to low-thrust applications and chooses MBH for this optimisation. Lambert arcs were determined to be too inaccurate, and therefore Sims-Flanagan transcription was chosen. Sims-Flanagan is a method for approximating low-thrust trajectories by dividing a transfer into several segments connected at nodes, where at each node a DSM is executed. More nodes result in a finer thrust profile, which can more accurately represent a low-thrust transfer. The combinatorial complexity in [Englander and Conway 2017] is somewhat comparable to the complexity considered in this thesis and low thrust is used. Therefore, the run times are representative: optimisations for Pluto rendezvous missions and BepiColombo mission simulations took up to 67 hours. [Chilan and Conway 2013] also uses a BGA for the outer-loop optimisation, however, the optimisation is focused on the determination of thrusting events and revolutions, rather than MGA sequences. To solve the VSDS, 'Leading bits' were used. A mission scenario is considered with both low and high thrust.

Interestingly, [Ceriotti and Vasile 2010] uses an Ant Colony Optimisation (ACO) for the outer loop and a tree-search method for the inner loop. ACO has a variant that is capable of integer programming. Specifically, the Beam Search Strategy (BSS) is implemented as the inner-loop tree-search method; the BSS is discussed further in the next subsection. The choice of a stochastic method for the integer programming part is not intuitive, although the results show that it is indeed possible. While the tree-search method for the inner MINLP also seems counter-intuitive, the paper considers high thrust only, which already reduces the number of continuous variables significantly. If only a few continuous variables are left the discretisation can still be reasonable and is similar to an approach that would perform a grid search on various continuous variables. The departure date is a common example of this discretisation.

In summary, the nested-loop approach is chosen due to its expected reliability and convergence despite its potentially higher run-time. As a small verification for this choice, [Ellison 2018; Morante et al. 2021] make a theoretical comparison of the single- and nested-loop approach and state that the nested-loop variant is the better choice for the same reasons as discussed above. [Zhang et al. 2015] numerically compares both approaches and finds that a mixed-code genetic algorithm does perform better than the two-level genetic algorithm, however, this result focuses on asteroid targeting, which results in a substantially different scale of integer and continuous variable counts, making the preference not as relevant. No other comparison has been found during this research project. The next subsection

discusses one specific type of outer-loop optimisation: the tree-search method.

2.1.4. Tree search algorithms

This subsection discusses tree-search methods, applied to the outer loop of the nested-loop optimisation approach. Tree-search methods do not use gradient information, but also do not rely on chance to discover the optimum, making them theoretically ideal for categorical optimisation problems such as MGA sequencing.

For the combinatorial space, a 'tree' or 'graph' can be defined – in this thesis 'tree' is used. There are a plethora of strategies that can be used to traverse these trees and make decisions, which generally fall in the category of graph theory. The focus of this thesis is not to reinvent the wheel regarding graph theory, but rather dissect the relevant aspects to help define a strategy that can be applied best to the low-thrust MGA sequencing optimisation problem. A few of the relevant terms are defined to give context to what is meant by a tree, after which some references are discussed that chose various tree-search strategies.

Introduction to Graph Theory

A tree has two building blocks: vertices and edges. A vertex is a point and an edge is a connection of two vertices. In the context of MGA sequencing, the terms nodes and legs are more common, where a node is equivalent to a celestial body and a leg is a transfer trajectory between two bodies. The traversal of a tree is the process of linking vertices and edges that are connected until some termination criterion has been met. One full traversal is called a roll-out of a tree. Any sequence of vertices of any length with connected edges is called a branch.

A tree can be directed or undirected. This characteristic concerns the vertices of a tree. The edges of a tree can have a direction – and therefore can only be traversed in one direction – or be undirected in that the tree can be traversed in both directions. Another characteristic of a tree is whether it is cyclic or acyclic. This characteristic is only valid in the context of directed graphs, as an undirected graph inevitably allows for a two-vertex cycle roll-out. A cyclic tree can have vertices that form a loop with however many edges in between. An acyclic tree never contains a loop. Trees can also be weighted. The vertices can be weighted, the edges can be weighted, neither can be, or both can be, depending on the application. If only the edges are weighted, the result is a travelling salesman problem, for example. A general example of a tree can be seen in Figure 2.2.

There are vertices and edges at various levels, which build up gradually as the tree is traversed. Figure 2.2 shows an absolute view of an entire tree. A relative view is also useful, where the root of a relative tree can be set to any particular vertex of the absolute tree. This difference is useful to visualise the recursive nature of the methodology developed in this thesis. In addition, a few terms are useful to know:

- Root: The root of a tree is the vertex that has no parent vertex.
- Branch: Any singular sequence of vertices and connected edges of any length.
- Leaf: A vertex that is at the end of a branch.
- Roll-out: One traversal of the tree until and including a leaf vertex.

Additional nomenclature is introduced related to the relative view mentioned above:

- Parent Vertex: The parent vertex is the predecessor vertex that is directly linked to the vertex at hand.
- Child Vertex: The child vertex is the descendant vertex that is directly linked to the vertex at hand.
- Sibling Vertex: The sibling vertex is any vertex that shares the same parent vertex as the vertex at hand.

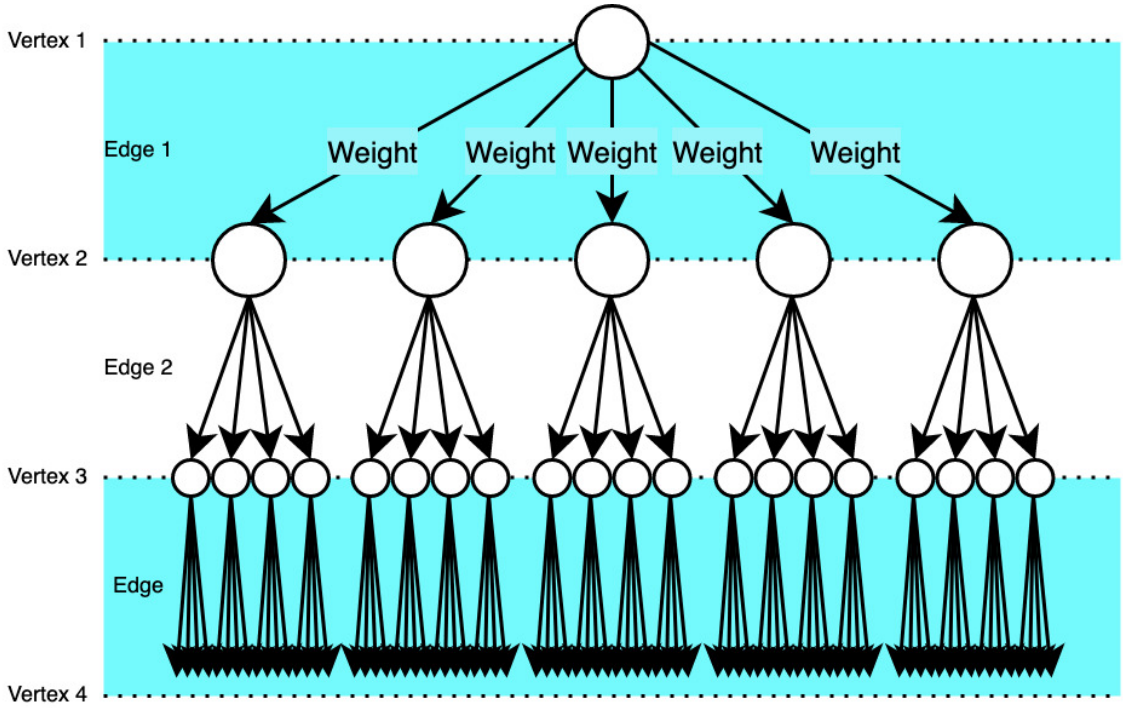


Figure 2.2: Directed acyclic weighted tree for 3 levels.

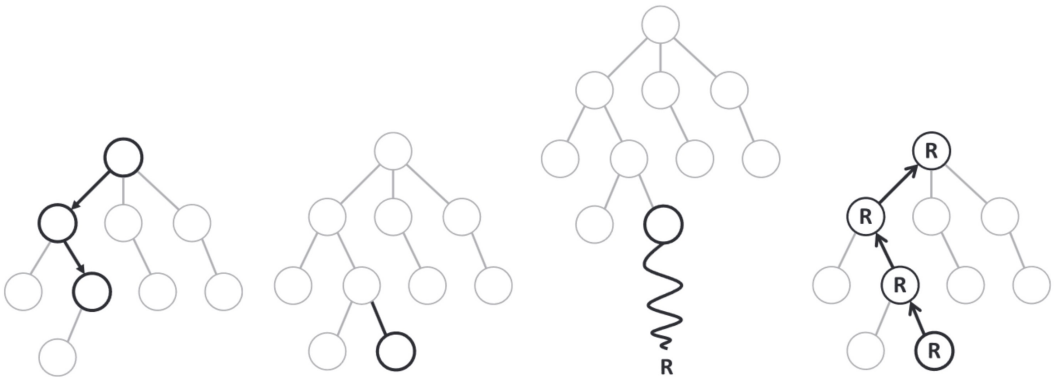
Tree-search methods used for MGA sequencing
A selection of tree-search methods is presented that has been applied to MGASO. These methods help shape the methodology developed in this thesis.

[Ellison 2018] implements the BSS for low-thrust trajectories. BSS is a best-first approach, or a greedy approach, in that it prunes away all the non-optimal branches and relies on its best guess. The implementation uses multi-threading with OpenMP to increase performance and the low-thrust trajectories are modelled using Sims-Flanagan. Other papers such as [Ceriotti and Vasile 2010; Bellome et al. 2021] have also used the BSS, albeit for different use cases. The BSS is described as a suitable option for RAM-limited simulations and is relatively simple. The BSS is not the focus of [Ellison 2018], and the choice for BSS should therefore not be presumed to be optimal. In terms of run time and problem type the BSS is a suitable candidate.

The tree-search method developed by [Vasile et al. 2015] is inspired by the Physarum organism for optimising MGA-DSM missions. The method consists of two main parts: decision network exploration and decision network growth. There is also a restart condition to prevent stagnation of the optimiser. The exploration part is similar to ACO, where the strength of a direction depends on a quantity that has traversed it. The growth of the tree is based on 'virtual agents' that traverse the tree and either move to the next existing node or create a new one.

[Hennes and Izzo 2015] implements Monte-Carlo Tree Search (MCTS) – presented more thoroughly by [James et al. 2017] – for MGA-DSM missions, rather than low-thrust missions. Specifically, Cassini-Huygens and Rosetta are used as test cases. The algorithm consists of four steps as seen in Figure 2.3: selection, expansion, simulation, and back-propagation. MCTS is a method that is free of pruning and uses random sampling to explore the tree. The selection strategy deploys a law of selection to tackle the exploration-exploitation dilemma of tree search methods, which is called Upper-Confidence Bounds for Trees (UCT) in this paper. The exploration-exploitation law determines how to descend through the tree at each iteration until a leaf node is reached. A leaf node is a node that does not have any child nodes. Once a leaf node is reached, an extra node is attached – called expansion – and a Monte-Carlo analysis is conducted on the descending tree, named simulation. After this, the back-propagation phase updates all the parent nodes with the respective fitness to influence the next generation of exploration-exploitation decision-making. In the end, a sequence will return the highest

fitness values based on the expected return at each node. Furthermore, [Hennes and Izzo 2015] adds the departure date and ToF as a grid to the combinatorial space, making it computationally expensive. This is to allow for the optimisation to converge properly without adding too many design variables, as mentioned previously. Finally, [Hennes and Izzo 2015] also considers a greedy policy, which performs better than the UCT or 'flat' selection policy, though besides a mathematical definition of the selection policy being called 'greedy', it does not explicitly state whether the MCTS also only utilises one iteration – which would correspond to a greedy approach. The run times are significantly lower which would indicate that the assumption of one iteration is correct.



(a) Selection. **(b)** Expansion. **(c)** Simulation. **(d)** Backpropagation.

Figure 2.3: Monte-Carlo tree search steps [James et al. 2017].

Following the MCTS from [Hennes and Izzo 2015], [Fan et al. 2022] developed a 2MCTS algorithm for MGA low-thrust missions. This paper utilises Bezier shapes to model low-thrust trajectories. This paper would have been the perfect verification case that could also be used for the performance assessment. Unfortunately, however, it is not explicitly stated what sequences come out of the optimisation, only how long it took to obtain the optimum. After contacting the authors, the only extra information obtained is that Juno-Jupiter-Jupiter is the optimal sequence. This means that the paper does not only consider planets as GA targets but also potentially other bodies that are not explicitly mentioned. The 2MCTS derivative of MCTS tries to improve upon three distinct disadvantages of MCTS. First, the balance parameter is difficult to determine. The balance parameter relates to the variables present in the selection step from Figure 2.3. Second, feasible results are only seen when the branches are fully developed. Only once the tree is fully expanded to leaf nodes are any actual trajectories formed. Third, only horizontal expansions are used to expand the search tree. To remedy these deficiencies, a vertical expansion step is added. The information after any random simulation is not discarded but stored – making the algorithm more memory intensive. This addition also allows the method to obtain feasible trajectories at every iteration as the results for the entire roll-out are stored. Do note that, for the majority of this thesis, [Fan et al. 2022] had not been published yet. As a side note, tree-search methods, and MCTS specifically are also applied to many other topics, not only to MGA sequencing. For example, it can be applied to the automation of vehicles and the decision-making processes that have to take uncertainties into account [Stegmaier et al. 2022].

This concludes the heritage of MGASO. The tree-search methods applied to low-thrust MGA sequencing are limited, but it is clear that tree-search methods have potential. Besides the heritage that presents options for algorithms and models, a software and hardware tool have to be chosen for implementing and testing the approach developed in this thesis, which is discussed next.

2.2. Tools for optimisation

In this section, hardware tools are discussed that are fitting for the constraints of this thesis. In addition, a selection of software tools is presented that are used in literature to execute an LTTO. Based on the outcome, a programming language is chosen to implement the MGASO.

2.2.1. Hardware options

This subsection focuses on the hardware tools that are used to perform all the simulations. A few constraints are considered for the decision: the time is limited by the duration of the thesis and there is no budget available for an MSc thesis at Delft University of Technology. The former determines the time period in which simulations can be performed, and thus also the efficiency that is required when running many simulations. A personal computer is used for developing the software and executing basic test runs. For the bulk of the simulations, a free high-performance computing source is desired. DelftBlue is a supercomputer of Delft University of Technology that is free to use for students once approved. The Dutch National Supercomputer Snellius, provided by SURF, is also free of charge and has high computing abilities. However, the application process is more extensive, and the support is remote. Therefore, DelftBlue is chosen. The two hardware systems used are shortly introduced.

The personal computer is a Macbook Pro 2017 model with a 3.1 GHz Quad-Core Intel Core i7 processor. There are four physical Central Processing Units (CPUs) and four additional virtual cores, amounting to eight logical cores in total. In this case, that means that eight threads can be simultaneously evaluated. However, because they share memory, the performance does not necessarily increase if there are more logical cores. The DelftBlue supercomputer has 218 nodes. Each node has two Intel XEON E5-6248R 3.0Ghz CPUs with 24 cores. Due to the shared nature of the supercomputer, the computational resources are limited and there are maxima for how much can be used at any given moment. For a direct comparison of the performance of these two hardware systems, see Appendix A. DelftBlue also has high-memory nodes and Graphics Processing Unit (GPU) nodes, but these are not used in this thesis. The constraints of DelftBlue are shortly listed below.

- A job, run, or simulation has a maximum run time of 24 hours.
- Eight jobs can be run concurrently.
- 48 CPUs is the maximum allowable number of CPUs for a single task.

The queuing system prioritises smaller jobs, so a shorter run time with fewer CPUs will run quicker than a larger job. As will be seen later on in various stages of this thesis, these constraints influence the choices that are made regarding the robustness and type of results. The constraints themselves are not hardware constraints, but software constraints, which raises the question of what software package to use.

2.2.2. Software tool inventorisation

This subsection looks into the software counterpart that is used for this thesis. Many software packages do some form of LTTO. A few of the most relevant ones are summarised in Table 2.2. The MGASO is not included in this, as the approach used for this optimisation is developed from scratch.

Table 2.2: LTTO software [Cowan 2022].

Name	Ref	Institute
DITAN	[Vasile et al. 2002]	ESA
Copernicus	[Johnson et al. 2003]	Univ. of Texas and NASA
GALLOP	[McConaghy et al. 2003]	Purdue Univ.
STOUR-LTGA	[Petropoulos and Longuski 2004]	Purdue Univ. and JPL
IMAGO	[De Pascale and Vasile 2006]	-
InTrance-GA	[Carnelli et al. 2009]	DLR
jTOP	[Campagnola et al. 2015]	Univ. of Tokyo and JAXA
EMTG	[Englander et al. 2015]	NASA and Illinois Univ.
MODHOC	[Ricciardi and Vasile 2018]	Strathclyde Univ.
MOLTO	[Morante et al. 2019]	-
LInX	[Ozimek et al. 2019]	John Hopkins Univ.
PaGMO	[Biscani and Izzo 2020]	ESA
Tudat	[Dirkx et al. 2022]	TU Delft

To assist in the decision for the choice of software package, several requirements and wishes are defined to distinguish the most suitable software packages. These criteria are presented in Tables 2.3 and 2.4. The requirements are general for any software that is chosen, whereas the wishes are split into three categories: General, Low-Thrust Trajectory (LTT) specific, and Optimisation specific.

Table 2.3: Software-related requirements.

Requirement ID	Definition
REQ001	The collection of software used shall be able to model low-thrust trajectories and perform a user-defined optimisation.
REQ002	The software shall be accessible for the entire duration of the thesis.
REQ003	The software shall be documented properly, such that it can potentially be applied and customized within two months.
REQ004	The software usage shall be free of charge.

Table 2.4: Software-related list of wishes split into three categories.

Category	Wish
General	• Open-source availability • Recent release version • Developed in Python or C/C++
LTT	• Availability of the Two-Body Problem (2BP), or potential to model it • Capability to model three dimensions, or potential to model it • Includes the hodographic-shaping method
Optimisation	• Availability of heuristic and hybrid methods • Possibility to perform multi-objective optimisation • Possibility to optimise mixed-integer problems • Includes the GIM

Based on the requirements above, only EMTG, LInX, Parallel Global Multiobjective Optimizer (PaGMO), and TU Delft Astrodynamics Toolbox (Tudat) remain as possible solutions. Tudat is chosen for a number of reasons, where the wishes in Table 2.4 are the deciding factor. Most importantly, the existence of shape-based methods, but also the author's familiarity with Tudat and the consequent increase in development speed are decisive aspects. In addition, promoting Tudat as a multi-functional open-source astrodynamics toolbox is a separate goal of the author and Delft University of Technology. PaGMO is chosen as optimisation software due to its compatibility with the problem – including mixed-integer and parallel capabilities – and with Tudat. Moreover, due to the affinity of the author with Python, PyGMO – a Python wrapper for PaGMO – is used for the optimisation process. In the next section, the usage of PyGMO and parallelisation is explained in more detail.

2.3. Parallelisation

Parallelisation is used in various ways for the methodology, but the actual programmatic implementation is done with a single framework: PyGMO. This section is a side-step that elaborates on the relevance of parallelisation in this methodology and presents several key concepts used in this thesis.

2.3.1. Choice for parallelisation

As was shortly mentioned in the introduction, MGA sequencing with low thrust is an extraordinarily complex optimisation problem. The run times for a preliminary optimal solution are long, and a remedy for that is to use parallelisation. Its usage does not by definition make the approach and optimisation more computationally efficient – although it can if the parallel information is used in a particular way. Parallelisation does decrease the run times of computationally intensive tasks and therefore allows for

more iterations even if not used more efficiently. Another reason to opt for the use of parallelisation is that it can increase the robustness of the results: parallel processes can exchange information that would otherwise be lost or inefficient if executed serially. This is explained more in-depth in Sections 2.3.3 and 6.3.5.

The choice for using parallelisation is a trade-off, as a set of parallel tasks requires more time to initialise than a sequence of tasks. Computational resources such as memory and processor time must be managed and the tasks must be scheduled to run in a safe yet efficient way. It can readily be decided that, due to the complexity of the optimisation problem and the long, repetitive computations, parallelisation is indeed worthwhile for the application in this thesis. On a separate note, the added value that is gained from using parallelisation is dependent on the hardware and software that is used – which was discussed in the previous section. Nevertheless, the gain should be noticed in almost any computing system. The implementation of parallelisation in this thesis is also designed such that it uses an available and non-exhaustive amount of resources depending on the system it is executed on. In the future, one could add an automated resource estimation based on the input parameters of the optimisation.

2.3.2. Parallelisation concepts

In this thesis, multiprocessing CPU parallelisation is used. The main reason is the already existing frameworks within PyGMO. Multi-processing is a form of parallelisation in which separate tasks are appointed to separate processors. Separate processes are thus running on independent CPUs, with independent memory management. This type of memory management does mean that any optimisation can only be executed on a processor level, and not on a thread level. Parallelisation using GPUs was also considered and is discussed in more detail in [Cowan 2022], however, it requires longer to set up and is therefore not worthwhile for the time frame of this thesis. Multithreading is also a concept that is useful for complex optimisation tasks, which could be considered in future work, however because Python is used as a programming language, the Global Interpreter Lock (GIL) becomes an issue. The GIL forces the ownership of a Python interpreter to be limited to a single thread. Circumvention of this lock is possible, but requires more development work which is not worthwhile analogously to the GPU consideration; multi-threading is therefore not considered.

In the realm of multiprocessing parallelisation, a number of concepts are relevant to present here. One characteristic of a parallel task, that is crucial to the exchange of information during the optimisation, is synchronicity; this characteristic is split into asynchronous and synchronous parallelisation. The concept is depicted in Figure 2.4 below. Asynchronous parallelisation effectively means that processes do not have to wait for one another to continue computing. In other words, an optimisation algorithm can continue its optimisation after it sends individuals to another optimisation process with no lead time. This type of parallelisation is more efficient, but can not always be applied. PyGMO allows for asynchronous parallelisation.

A unique and novel aspect of this thesis is the usage of the Generalised Island Model (GIM) [Izzo et al. 2012]. The island model is inspired by the geographical island. Generally, islands are grouped together and form so-called archipelagos. The island then represents a single optimisation process, and the archipelago represents the complete optimisation problem. The GIM has thus far been used with a selection of optimisation algorithms: each island solves the same optimisation problem but with a different algorithm. The purpose of this distribution is to magnify the positive aspects of some algorithms and suppress the negative aspects of others. Throughout the optimisation, highly fit individuals can be exchanged, thereby accelerating the convergence of algorithms that are performing less well. This thesis applies the GIM differently: sequences with various GA targets are distributed over the islands instead. The specific distribution of GA targets across the islands is explained in Section 7.1.

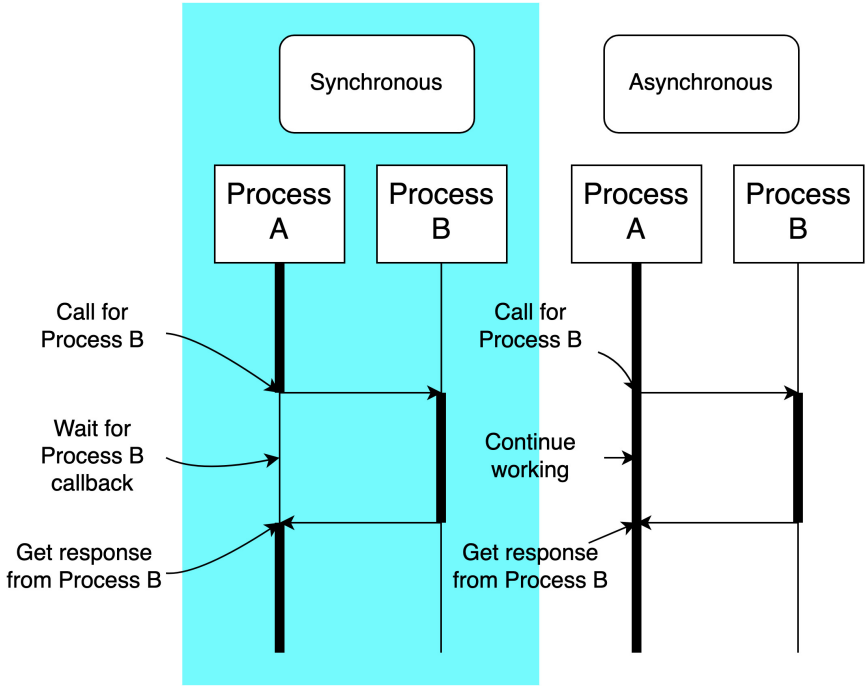


Figure 2.4: Asynchronous vs Synchronous processing.

2.3.3. Terminology for PyGMO

As mentioned before, this thesis considers PyGMO and reference will be made to PyGMO terminology in various places. This subsection introduces the relevant terminology from PyGMO.

The 'pg.problem' class is explained first. This problem is a class that defines the optimisation problem – an LTTO problem in this thesis. The 'pg.problem' class includes the calculation of the fitness of every individual, it defines what parameters are continuous or integer, and it defines the bounds of the problem. Almost all PyGMO objects can be constructed from scratch, though PyGMO has default objects; these objects are named 'User-Defined x', where x is the object that you are customising. A User-Defined Problem (UDP) is developed for the LTTO problem. This problem class is given to a 'pg.island' class together with a 'pg.algorithm' object as an attribute. The 'pg.island' was explained previously and the 'pg.algorithm' defines the User-Defined Algorithm (UDA) that is used to solve the UDP. A default multi-processing island is used, 'pg.mp_island', and the 'pg.sga' algorithm is used – where 'sga' is the Simple Genetic Algorithm (SGA), discussed in Section 5.3. Various problems with various definitions can be given to each User-Defined Island (UDI). Once all islands have been defined, they are connected by a 'pg.archipelago', where migration is possible through a 'pg.topology'.

The 'pg.topology' is a User-Defined Topology (UDT) class that connects various islands. A connection is defined by a probability of an individual migrating from one population to another. The connection is defined by a source and target island, and also a migration probability. The most basic example of a topology is the 'Null' topology, which has no connections, and therefore the islands function as separate parallel processes. This can be fully customised to only allow for the migration of individuals between specific islands with pre-defined probabilities. To run the optimisation on all islands in parallel, the 'pg.archipelago' is used, which was also discussed previously. This class has a method that manages the parallel evolution of each island while taking the topology into account.

Now that this terminology is clear together with the necessary context of the parallelisation, the astrodynamics related to this optimisation problem can be presented.

3

Orbital mechanics

In this chapter, the orbital mechanics related to the problem of this thesis are concisely discussed. Specifically, reference descriptions necessary to describe a trajectory, the two-body problem needed to model the dynamics, and the concept of gravity assists are explained.

3.1. Reference description

A good understanding and proper definition of reference frames and coordinate systems are essential for representing a trajectory. The various frames and coordinate systems that are used in this work are discussed in the next two subsections.

3.1.1. Reference frames

In this subsection, the notion of a reference frame is explained, along with frame characteristics, the relevance of reference frames, and what reference frame is used.

To comprehensively describe any position or velocity, a reference is required as well as a method for the quantification of space. A reference is any point other than the point of interest that can be used as an origin. From said origin a vector can be formed toward the point of interest, which by definition consists of and only of a magnitude and direction. A reference frame is a set of three orthogonal vectors coinciding in an origin. In the next few paragraphs, the most suitable reference frame is found by discussing the crucial properties of reference frames and their effect on the behaviour of the dynamical model.

To begin with, the global convention is to define the reference frame as a right-handed system, defined by $\hat{i} \times \hat{j} = \hat{k}$ where $\hat{i}$, $\hat{j}$, $\hat{k}$ are the unit vectors of any axes. A few origins are considered: the Sun, the Solar System Barycenter (SSB), and rarely a planet or Lagrange point. SSB is chosen here because the formulation is physically intuitive in the context of interplanetary trajectories. To keep the formulation of the Equations Of Motion (EOM) as simple as possible, the discussion on reference frames is limited to those that are inertial, which is defined as [Wakker 2015]:

"An inertial reference frame is a reference frame with respect to which a particle remains at rest or in uniform rectilinear motion if no resultant force acts upon that particle."

Using reference frames constrained by this definition, the EOM do not include d'Alembert forces and are therefore less complicated. Reference frames can rotate, which can be useful for missions that analyse the position of an object relative to the surface of a celestial body. In this thesis, however, a non-rotating frame results in more intuitive results. Regarding the orientation of the frame, for general applications, the orientation is typically defined relative to the equatorial or ecliptic plane. Moreover, the plane as it was defined at a specific point in time; generally the astronomical epoch J2000 is taken as reference, resulting in the J2000 and ECLIPJ2000 inertial reference frames, for the equatorial and ecliptic reference frame, respectively. The positive X-direction points towards the vernal equinox as seen in Section 3.1.1.

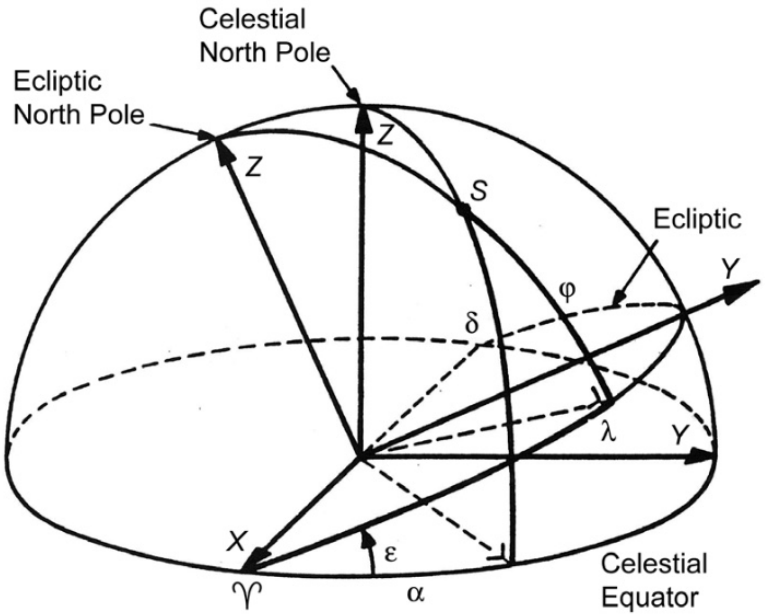


Figure 3.1: Equatorial and ecliptic system of coordinates [Wakker 2015].

For orders of magnitude higher precision, the so-called International Celestial Reference Frame (ICRF) can be used. The orientation for this frame is based on the radiation received from active galactic nuclei as reference points; the positions of active galactic nuclei only change over a significantly longer time period compared to the vernal equinox. As this thesis concerns preliminary trajectory optimisation, the J2000 reference frame suffices. Due to the interplanetary nature of these trajectories and tradition, an ecliptic reference frame is chosen, which leaves only the ECLIPJ2000 reference frame. One additional advantage is that Tudat has already implemented this reference frame.

Besides the main description of a trajectory, as a patched-conic approximation is used, two separate reference frames are needed to describe the GA dynamics. Hodographic shaping, as it is implemented in Tudat, uses the TNW frame, and a local frame. Using this implementation is well suited for the mathematical description of the patched-conic approximation – discussed in Section 3.3. In particular, the simplification as described in [Cowan 2022] is appropriate because the planetary velocity vector coincides with an axis of the reference frame, making descriptions simpler. However, the description is not perfect for physical interpretation, discussed in Section 3.3. The TNW abbreviation stands for tangential, normal, and angular momentum where the W refers to ω, as can be seen in Figure 3.2.

In Figure 3.2, $\vec{V}_{pl}$ is the planet velocity vector, $\vec{h}_{pl}$ is the angular momentum vector of the planet, $\vec{V}_{\infty,in}$ is the incoming hyperbolic velocity vector of the spacecraft, θ is the in-plane angle, ϕ is the out-of-plane angle, and [$\hat{u}$, $\hat{v}$, $\hat{w}$] are the unit vectors in the TNW frame. The third axis is constructed using the cross-product of the velocity and angular momentum vector, which is typical for the TNW frame. This frame introduces θ and ϕ of the incoming velocity vector, discussed later in Section 3.3.

The local frame is needed to define a third and separate quantity, known as the orbit orientation angle β, discussed in Section 3.3. This frame, seen in Figure 3.3, is defined using $\vec{V}_{\infty,in}$, and $\vec{V}_{pl}$ from Figure 3.2.

In Figure 3.3, $\vec{V}_{\infty,out}$ is the outgoing heliocentric velocity vector, δ is the deflection angle, and $\hat{i}$, $\hat{j}$, $\hat{k}$ are the unit vectors for the local frame. These variables become clear in Section 3.3. In summary, the ECLIPJ2000 frame is used with the SSB as the origin for the transfer trajectories, and the TNW frame is used for the GA dynamics. In the next subsection, the realisation of this reference frame is explained in the form of coordinate systems.

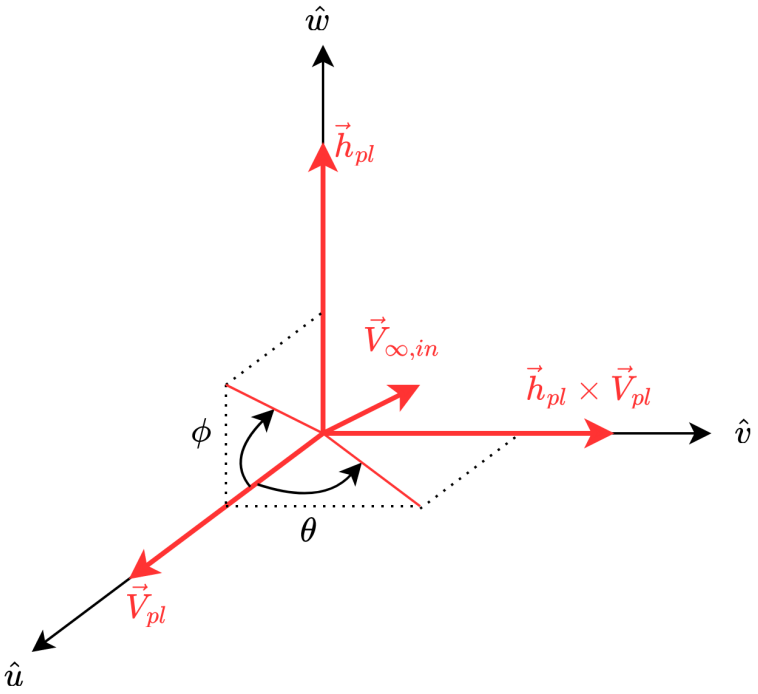


Figure 3.2: TNW frame as implemented in Tudat.

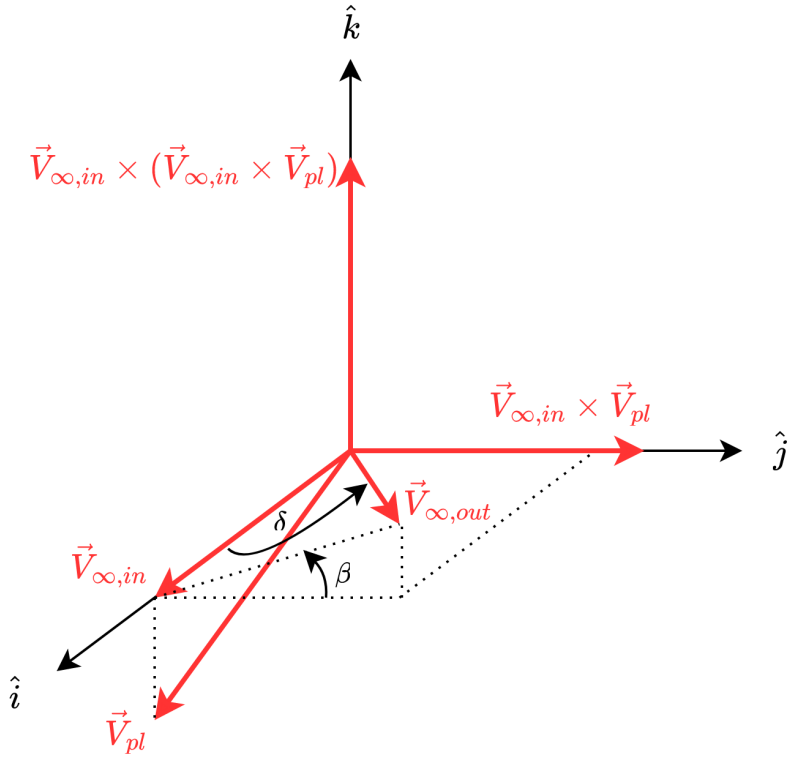


Figure 3.3: Local frame as defined in Tudat.

3.1.2. Coordinate systems

Coordinate systems are intrinsically linked to reference frames. To quantify any distance in a reference frame, a formulation is needed that quantifies the position relative to the origin. Cartesian, cylindrical,

and spherical coordinates are among the most straightforward and often-used systems. Hodographic shaping uses a cylindrical coordinate system, which shall therefore be presented explicitly. In addition, as Tudat converts all coordinates into Cartesian coordinates for various computational steps, the Cartesian coordinate system is relevant and is also discussed.

Cartesian coordinates, see Equation (3.1) for the state, represent the most straightforward system: each coordinate is a measure of distance in one of three orthogonal directions. This set of coordinates is robust as there are no singularities that can lead to faulty calculations. However, objects in astrodynamics generally rotate around the origin, resulting in fluctuations of the coordinate values. This means that the numerical accuracy is limited and the results are not always easy to interpret. While Cartesian coordinates use distances for each coordinate, cylindrical coordinates use one angular quantity, θ, as seen in Equation (3.2). The velocity corresponding to the angular quantity is defined by using $r\dot{\theta}$ as seen in Equation (3.2). The conversion between Cartesian and cylindrical coordinates is found in Equations (3.1) and (3.2).

$$\begin{cases} x = r\cos\theta \\ y = r\sin\theta \\ z = z \\ \dot{x} = V_r\cos\theta - V_\theta\sin\theta \\ \dot{y} = V_r\sin\theta + V_\theta\cos\theta \\ \dot{z} = V_z \end{cases} \text{, where } \vec{x}_{cyl} = \begin{bmatrix} r \\ \theta \\ z \\ V_r \\ V_\theta \\ V_z \end{bmatrix} \tag{3.1}$$

$$\begin{cases} r = x^2 + y^2 \\ \tan\theta = \frac{y}{x} \\ z = z \\ V_r = \dot{r} = \frac{x\dot{x}+y\dot{y}}{\sqrt{x^2+y^2}} \\ V_\theta = r\dot{\theta} = \frac{x\dot{y}-y\dot{x}}{\sqrt{x^2+y^2}} \\ V_z = \dot{z} \end{cases} \text{, where } \vec{x}_{cart} = \begin{bmatrix} x \\ y \\ z \\ \dot{x} \\ \dot{y} \\ \dot{z} \end{bmatrix} \tag{3.2}$$

The velocity components are split into V_r, V_θ, and V_z, which correspond to the radial, normal, and axial directions, respectively. These definitions are useful for transfer trajectories; due to the general direction of the velocity vector, the coordinate oscillations are less pronounced. Other descriptions of motion in space are possible, such as orbital element sets. However, these are not as useful for interplanetary transfers: the position and velocity components are not explicit and independent, resulting in more complex descriptions of the position and velocity. Specifically, because hodographic shaping is used, analytical derivatives and integrals are computed which result in simpler expressions if cylindrical coordinates are used.

3.2. Two-body problem

In the previous section, the reference description is explained, however, the dynamics that are described with the reference description are yet to be discussed. This section explains the two-body problem from a mathematical perspective to introduce the dynamical model that is simulated.

As finding the globally optimal low-thrust trajectory with the highest precision is not the objective, many perturbations in the dynamical system of the spacecraft are irrelevant. As was found in [Cowan 2022], most interplanetary missions spend the vast majority of time gravitationally close to only one celestial body, rather than two. To this end, the orbital dynamics that define the motion of the spacecraft can be based on the two-body problem, rather than the three-body problem.

The ECLIPJ2000 frame was chosen in the previous section for the description of points. The derivation of the two-body problem starts from the definition of the reference frame – in particular, an inertial, non-rotating reference frame seen in Figure 3.4.

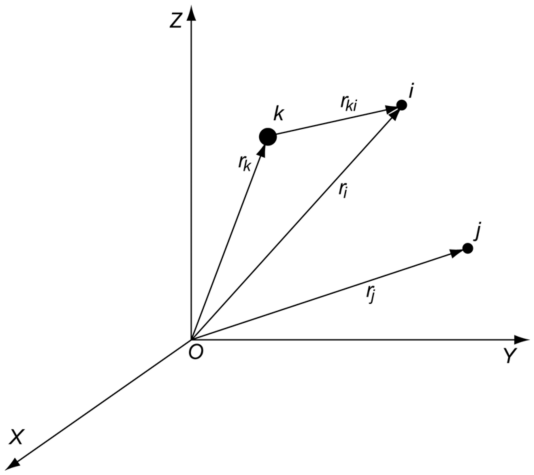


Figure 3.4: Description of n mass points in an inertial non-rotating reference frame [Wakker 2015].

With this reference, the position of mass points can be described and hence also the dynamical interaction. Starting with the mutual gravitational attraction between n mass points, seen in Equation (3.3) [Wakker 2015].

$$\frac{d^2\bar{r}_i}{dt^2} = -G\frac{m_i + m_k}{r_i^3}\bar{r}_i + G\sum_{j\neq i,k} m_j\left(\frac{\bar{r}_j - \bar{r}_i}{r_{ij}^3} - \frac{\bar{r}_j}{r_j^3}\right) \tag{3.3}$$

Here, $\vec{r}$ is a position vector, G is the universal gravitational constant, m is the mass of a body, and $[i, j, k]$ are the indices of a body. Equation (3.3) can be simplified because only two bodies are regarded. A favourable consequence of the description of only two bodies is that only one body has to be described relative to the other; the spacecraft is described relative to the attracting body. The gravitational force therefore only acts in one dimension – the radial direction. All assumptions that can be made with the two-body problem are listed below.

- Negligible mass of the spacecraft compared to the attracting body
- An inertial coordinate system is used
- Both bodies are point masses
- No other forces act on the system except for the mutual gravitational force

The equation is then reduced to Equation (3.4) [Wakker 2015].

$$\ddot{\vec{r}} = -G\frac{M}{r^3}\vec{r} \tag{3.4}$$

Here, M is the mass of the attracting body. The fourth assumption here is only made in the context of this derivation. As low-thrust propulsion is used for this thesis, a constant thrust force acts on the spacecraft. The thrust has a significant influence on the dynamics, thereby requiring an extra term for the thrust acceleration in the equations. The addition of this thrust acceleration results in Equation (3.5) [Wakker 2015] below which is the basis for solving the hodographic-shaping method.

$$\ddot{\vec{r}} = -G\frac{M}{r^3}\vec{r} + \vec{f} \tag{3.5}$$

In Equation (3.5), $\vec{f}$ is any perturbing acceleration – defined as $\vec{f} = \frac{\vec{F}}{m_s}$, where $\vec{F}$ is the perturbing force and m_s is the mass of the spacecraft.

As a short addition to the optimisation of any mass-related quantity: the hodographic-shaping method – discussed in Section 4.2 – returns the thrust acceleration history and state history, and by using the Tsiolkovsky equation, as seen in Equation (3.6) [Turner 2008], the delivery mass can be deducted. The use of Equation (3.6) is further explained in Section 4.2.

$$\Delta V = c_{eff} \ln \frac{M_0}{M} \tag{3.6}$$

where M_0 is the mass of the rocket at t_0, and M is the current mass of the rocket. c_{eff} is the effective exhaust velocity. This is useful for understanding the objective definition in Section 5.4.

In the next section, the GA and its use for the optimisation problem in this thesis are explained.

3.3. Gravity assist

In this section, the GA manoeuvre is discussed from a mathematical and theoretical point of view.

GAs are relevant as ever because the outer planets of the Solar System have been shown to be scientifically intriguing. The traditional use of chemical propulsion to provide the necessary ΔV is not feasible when traversing large parts of the Solar System.

A GA, flyby, or less often named swingby, is a manoeuvre that uses the momentum of celestial bodies – generally planets – to gain heliocentric energy. In the context of the two-body problem, GAs are modelled with the patched-conics approach. Patched conics refers to a conic section which describes a trajectory in a two-body system adhering to Kepler's laws. The patched-conic method strings multiple trajectories together, where the separation between conic sections is defined by the SOI of the GA planet. This is done to model the effect of a GA manoeuvre by alternating between heliocentric and planetocentric segments.

Because the Sun is the main gravitational body for a large portion of any interplanetary trajectory, switching and patching multiple conic sections together is unnecessary. Rather, an approach is chosen that simplifies the number of legs that have to be modelled. This choice is made based on the functioning of the hodographic-shaping method, where each extra leg adds to the computational time required to evaluate the whole trajectory. This simplified patched-conic method only models legs between point masses of planets, by effectively decreasing the size of the SOI to approach zero. The effect of the GA is calculated using the mathematical tools from the patched-conic approach, only the planetocentric leg of each GA is reduced to an instantaneous change in the velocity vector of the spacecraft.

In Figure 3.5, the change in the velocity vector can be seen as a consequence of a GA. The nomenclature and derivation of the quantities necessary to define a GA use the formulation by [Conway 2010], which is presented below.

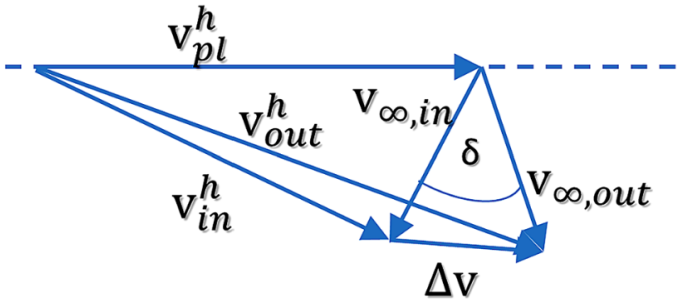


Figure 3.5: GA velocity vectors in both frames with turning angle [Maiwald 2017].

The incoming hyperbolic velocity – defined in a planetocentric frame – can be determined by subtracting the two vectors as seen in Figure 3.5, shown in Equation (3.7).

$$\vec{V}_{\infty,in} = \vec{V}_{in}^h - \vec{V}_{pl}^h \tag{3.7}$$

Here, $\vec{V}_{\infty,in}$ is the incoming hyperbolic velocity vector. However, the incoming heliocentric velocity vector is unknown, and it is therefore chosen to define the incoming hyperbolic velocity vector using three quantities [Musegaas 2013]: the norm of the velocity, θ, and ϕ, defined previously. Using Figure 3.2, an expression can be formed of the incoming hyperbolic velocity vector, seen in Equation (3.8).

$$\vec{V}_{\infty,in} = |\vec{V}_{\infty,in}|(\cos(\phi)\cos(\theta)\hat{u} + \cos(\phi)\sin(\theta)\hat{v} + \sin(\phi)\hat{w}) \tag{3.8}$$

For a GA, a few parameters need to be determined: the pericenter radius of the GA is defined beforehand, the gravitational parameter is that of the GA body, and the eccentricity then follows based on the previously found incoming hyperbolic velocity and these two quantities, defined by Equation (3.9).

$$e = 1 + \frac{r_p}{\mu|\vec{V}_{\infty,in}|^2} \tag{3.9}$$

In Equation (3.9), e is the eccentricity, r_p is the planetocentric hyperbolic pericenter radius, and μ is the gravitational parameter of the GA body. The deflection angle, δ, from Figure 3.5 then follows as shown in Equation (3.10), which determines the angle between the incoming and outgoing hyperbolic velocity vector.

$$\delta = 2\arcsin(\frac{1}{e}) \tag{3.10}$$

In addition to the deflection angle, an extra quantity is required to fully define the nature of the GA manoeuvre. The outgoing hyperbolic velocity vector is constructed relative to the incoming hyperbolic velocity vector using not only δ but also β, as can be seen in Equation (3.11) and Figure 3.3.

$$\hat{V}_{\infty,out} = \cos(\delta)\hat{i} + \cos(\beta)\sin(\delta)\hat{j} + \sin(\beta)\sin(\delta)\hat{k} \tag{3.11}$$

In Equation (3.11), $\hat{V}_{\infty,out}$ is the unit vector for the outgoing hyperbolic velocity vector, β is the plane orientation, $\hat{i} = \frac{\vec{V}_{\infty,in}}{V_{\infty,in}}$, $\hat{j} = \frac{\hat{i}\wedge\vec{V}^h_{pl}}{|\hat{i}\wedge\vec{V}^h_{pl}|}$, and $\hat{k} = \hat{i}\wedge\hat{j}$. To get the heliocentric outgoing velocity vector, this outgoing hyperbolic velocity unit vector is multiplied by the incoming hyperbolic velocity and subsequently added to the heliocentric planetary velocity vector, as seen in Equations (3.12) and (3.13).

$$\vec{V}_{\infty,out} = |V_{\infty,in}|\hat{V}_{\infty,out} \tag{3.12}$$

$$\vec{V}^h_{out} = \vec{V}^h_{pl} + \vec{V}_{\infty,out} \tag{3.13}$$

The specific implementation of this model in Tudat is specified by [Musegaas 2013]. The interpretation of these quantities is not the most intuitive, as β is defined in a frame that is dependent on the other quantities, such as the incoming heliocentric velocity vector. The definition used in Tudat is consistent, however, in future work a different formulation may be explored as this would ease the analysis of the physical interpretation of the quantities. As an addition, during the tuning process in Section 5.2, configurations are defined that add various components to this description. One of these quantities is the Oberth ΔV, which is an impulsive thrust that is applied at the periapsis. The next chapter discusses the hodographic-shaping method, which functions as the low-thrust and trajectory model.

4

Low-thrust propulsion

Low-thrust technologies are currently used in multiple interplanetary missions due to their efficiency. This potential is recognised and thus chosen as the thrust type for the transfers in this thesis. This chapter consists of a discussion on the low-thrust technologies that enable the type of mission considered in this thesis, followed by a discussion of the model chosen to represent the low-thrust MGA trajectories.

4.1. Low-thrust technologies

In this section, low-thrust technology is concisely presented to give context to the choice of low-thrust modelling for MGA trajectories.

Low-thrust propulsion is characterised by its high exhaust velocity (c_{eff}) compared to that of chemical propulsion, leading to more efficient propellant-mass usage. Low-thrust propulsion systems – also known as electric propulsion or ion propulsion – are mostly based on the usage of electric energy to heat up particles or electromagnetic fields to accelerate charged particles to high velocities.

It should be mentioned that low-thrust propulsion systems do not include the power source or exhaust medium. The power source can be either solar radiation or a nuclear fission reactor. Solar radiation is useful because it does not require any fuel and has an 'unlimited' source of power. However, using solar radiation also come with a few constraints, among which is the requirement that the solar array has to be in the line of sight of the Sun. Furthermore, the further away the mission is from the Sun, the lower the power output, which limits its use cases to missions to the outer planets. In this thesis, the power source is not taken into consideration.

Different types of low-thrust propulsion systems exist: electrothermal, electrostatic, and electromagnetic. Electrothermal propulsion is based on using electric energy to heat up a gas, forcing that gas to be expelled from a nozzle. Electrostatic propulsion ionises propellant particles, after which an electric field accelerates the charged particles out of a nozzle. Electromagnetic propulsion uses the Lorentz acceleration to accelerate plasma.

As mentioned in Chapter 1, many low-thrust missions have been executed in the recent past, some of which are presented below with properties of their propulsion system.

In Table 4.1, it can be seen that the exhaust velocity ranges between 16 and 42 km/s, whereas for traditional chemical propulsion systems, the exhaust velocity caps at 6 km/s [Wakker 2015]. The maximum thrust of all presented thrusters is in the order of mN, which is orders of magnitude smaller than the thrust of chemical boosters used for impulsive shot manoeuvering. When observing the total thrust time of chemical thrusters compared to electric propulsion systems, the duration of operation of each thrust type is indicative of the total amount of ΔV that is produced. Thrust times for low-thrust propulsion systems can be in the order of years. Therefore, the total ΔV produced by low-thrust propulsion systems is generally higher per kg of propellant mass.

It has been determined that low-thrust propulsion systems have potential and will be considered in

Table 4.1: Previous and current low-thrust missions with their characteristics [Tsuda et al. 2013; Wakker 2015; Casteren and Novara 2011] in chronological order of launch date.

Mission	Organiser(s)	Launch date	# Thrusters	Maximum thrust [mN]	Exhaust velocity [km/s]
Deep Space 1	NASA	24/10/1998	1	92	30
Hayabusa 1	JAXA	09/05/2003	4	8	31
SMART-1	ESA	27/09/2003	1	68	16
Dawn	NASA	27/09/2007	3	92	30
Hayabusa 2	JAXA	03/12/2014	4	9	27
BepiColombo	ESA & JAXA	20/10/2018	2	290	42
DART	NASA & ESA	24/11/2021	1	92	30

The Dawn and DART thrusters are the same as those of the Deep Space 1 spacecraft. The values are rounded to the nearest integer. The maximum thrust is per thruster. The exhaust velocities are sometimes calculated from the specific impulse, depending on what literature provides.

this thesis, but how to model a low-thrust trajectory is still undefined. The method that enables the representation of low-thrust trajectories is discussed next.

4.2. Hodographic shaping

As mentioned before, the hodographic-shaping method is chosen as low-thrust trajectory representation. In this section, the reasoning behind this choice is discussed as well as the methodology itself.

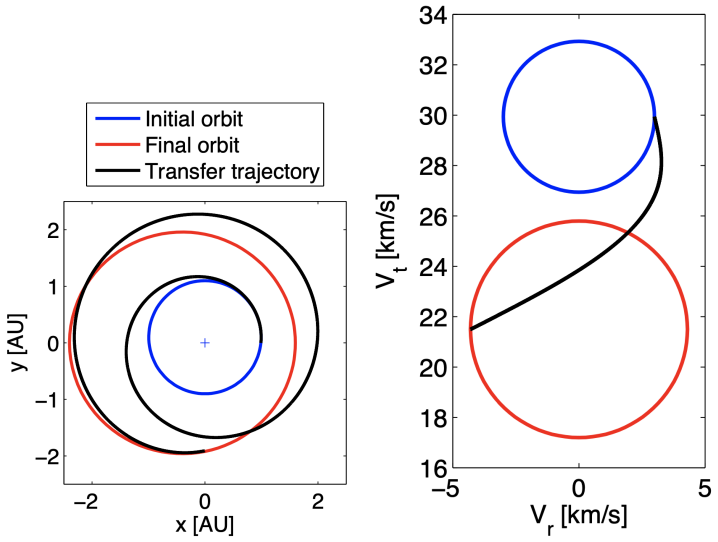
In search of a method to model low-thrust trajectories, hodographic shaping was chosen due to its wide application range, robustness, and low implementation effort [Gondelach and Noomen 2015]. Hodographic shaping is one of the shaping methods. Shaping methods, or shape-based methods, are mostly analytical procedures that describe a low-thrust trajectory by making assumptions about its shape, and inferring the dynamics that would lead to such a shape. Other examples of shaping methods are exponential sinusoids [Petropoulos and Longuski 2004], inverse polynomial [Wall and Conway 2009], spherical shaping [Novak and Vasile 2011], and Fourier series [Abdelkhalik and Gad 2012].

Hodographic shaping is based on the velocity hodograph. The velocity hodograph is a graph that plots two velocity components against one another. In Figure 4.1b, the radial velocity is plotted against the tangential velocity. This velocity profile corresponds to the trajectory, seen on the left in Figure 4.1a.

The shapes can be constructed as a function of time or of polar angle. Due to the physically real derivative of the time-based method, and its slightly more simple implementation, the time-based variant is chosen. [Gondelach and Noomen 2015] recommends a heliocentric or SSB-centered reference frame, which is in accordance with the reasoning from Section 3.1. As previously alluded to, the TNW reference frame is used. Moreover, the hodographic-shaping method uses cylindrical coordinates because it only uses one angular quantity, and is less sensitive to fluctuations if multiple revolutions are present.

The velocity-hodograph shapes are described mathematically; a velocity profile needs to be defined for all axes in a cylindrical coordinate system – presented in Section 3.1.2. The shapes can often be described well by simple mathematical building blocks. The most basic collection of mathematical building blocks used to create velocity shapes is shown in Table 4.2. The building blocks have to satisfy one condition: they have to be analytically differentiable and integrable.

In Table 4.2, v is the base function, u is the dependent variable – t in this case, but otherwise θ, and m and n are orders of the base functions. A radial, normal, and axial velocity function is created using three base functions per axis, shown in Table 4.3. Three base functions are the minimum number required because velocity functions have to satisfy three boundary conditions each, derived in Appendix B.1. This requirement means that there are three coefficients per axis. Velocity functions can be added to improve the agility of the method to find lower ΔV solutions, consequently, extra coefficients are added



(a) Trajectory **(b)** Velocity hodograph

Figure 4.1: Example trajectory with corresponding velocity hodograph [Gondelach and Noomen 2015].

Table 4.2: Base functions v [Gondelach and Noomen 2015].

Base function	$v(u)$	Base function	$v(u)$
Constant	1	Power times sine	$u^n \sin(2\pi m u)$
Power	u^n	Power times cosine	$u^n \cos(2\pi m u)$
Exponential	e^{nu}	Exponential times sine	$e^{nu} \sin(2\pi m u)$
Sine	$\sin(2\pi n u)$	Exponential times cosine	$e^{nu} \cos(2\pi m u)$
Cosine	$\cos(2\pi n u)$		

that are design variables during the LTTO. [Gondelach and Noomen 2015] recommends certain base functions to use for the three velocity base functions, but also in case free coefficients are used, resulting in more base functions. The recommended base functions for an Earth-Mars transfer are shown in Table 4.3.

Table 4.3: Recommended base functions for an EM transfer [Gondelach and Noomen 2015].

Type of function	Axis	Name and equation		
Base	R + N	**C Pow Pow2**		
		c_1	$+c_2 t$	$+c_3 t^2$
	A	**CosR5 P3CosR5 P3SinR5**		
		$c_1 \cos(2\pi t(N+0.5))$	$+c_2 t^3 \cos(2\pi t(N+0.5))$	$+c_3 t^3 \sin(2\pi t(N+0.5))$
Additional	R + N	**PSin05 PCos05**		
		$c_4 t \sin(0.5t\pi)$	$+c_5 t \cos(0.5t\pi)$	
	A	**P4CosR5 P4SinR5**		
		$c_4 t^4 \cos(2t\pi(N+0.5))$	$+c_5 t^4 \sin(2t\pi(N+0.5))$	

R is radial, N is normal, and A is axial. C is Constant. Pow or P is Power. Px or Powx where x is the exponent. CosR5 where Cos is a cosine. R refers to the factor N. R5 refers to (N+0.5); 05 refers to the factor 0.5

These recommended functions are used in this thesis. The quality of the results depends on the base functions chosen. One difference that is expected to affect potential results significantly is whether the

transfer is to an outer or inner planet. The base functions that are recommended for an Earth-Mercury transfer, for example, are almost identical, with two exponents that are increased by two in the axial component. This increase in only the axial component is caused by the relatively high inclination of Mercury at more than seven degrees relative to the ecliptic. These shaping functions were tested, however, no immediate improvement was found.

Now that the velocity functions have been fully determined, to create the trajectory and calculate the required ΔV, several steps are necessary. From the velocity functions, the analytical integrals are calculated to have an expression for the position. The derivatives are calculated to determine the inertial acceleration. Using Equation (3.5), the thrust acceleration can be derived. The thrust acceleration is then integrated over time to produce the total ΔV required for that trajectory. For the derivation of the coefficients that follow from the boundary conditions, see Appendix B.1. A concise list of steps is provided in Appendix B.2.

This sums up the theory behind the hodographic-shaping method. The next part, Part II, continues with the low-thrust topic and specifically the optimisation process.

Part II

Low-Thrust Trajectory Optimisation

5

LTTO setup

This part concerns the inner loop of the optimisation problem, where the low-thrust trajectories of a specific MGA sequence are to be optimised such that the quality of the sequence can be compared to other sequences with sufficient confidence. In this chapter, the LTTO problem is presented. Specifically, the structure of the optimisation is laid out, the parameters and variables are presented, and the objective function is defined.

5.1. Optimisation structure

This section shortly describes the definition of the LTTO problem and the top-level structure.

The LTTO problem is defined as a Hybrid Optimal Control Problem (HOCP), which is translated into a Mixed-Integer Non-Linear Program (MINLP) using direct transcription. A HOCP is a general formulation for a problem including a collection of continuous and discrete variables that minimise an objective function. Direct transcription is a transformation based on an approximation of a parametrised state and control input vector. Simply put, the state and control vectors have to be described somehow. Generally, this description involves making several assumptions, resulting in the representation being only an approximation of reality. The direct transcription itself then transforms the formulation from an HOCP into an MINLP with the aforementioned approximation. An MINLP can be seen as a block that takes in a number of parameters and design variables, and returns a collection of the fitness values. The problem is hybrid – or mixed-integer, because the inputs of a hodographic-shaping leg include both continuous and discrete variables. Running separate optimisations for each discrete option as a grid becomes significantly less efficient with larger combinatorial complexity. The problem is non-linear because of the complex dynamics that need to be modelled; the gravitational acceleration in a two-body system is the most straightforward example.

To visualise the elements of the MINLP that form the LTTO problem, a top-level block diagram is shown in Figure 5.1. Several parameters are defined to characterise the LTTO problem, as well as design variables with their corresponding bounds – these quantities are defined in the next section. Some bounds are dependent on parameters or other design variables for their definition, introducing further complexity. The LTTO block consists of an optimisation algorithm combined with termination conditions. The termination conditions specify whether the number of generations has been evaluated. Additional conditions can be implemented such as termination if the champion fitness value does not improve more than x% within y generations. This termination condition is not implemented because the topology is used, which needs a certain minimum number of generations to have any substantial effect. Moreover, the effect that the topology has should be equal across all simulations for better reproducibility. A further constraint could be the maximum thrust acceleration of the low-thrust engines, however, this constraint is not implemented because it may unnecessarily inhibit the optimisation, by removing individuals with high potential but a slightly too high thrust value. This constraint could be added in future work. The various inputs are passed into a PyGMO-compatible problem class; the PyGMO classes were described in Section 2.3.3. This problem class is used to create populations, which are subsequently passed to a

PyGMO archipelago class together with the algorithm and input parameters. The output consists of a dictionary of champion fitness and design variable vectors.

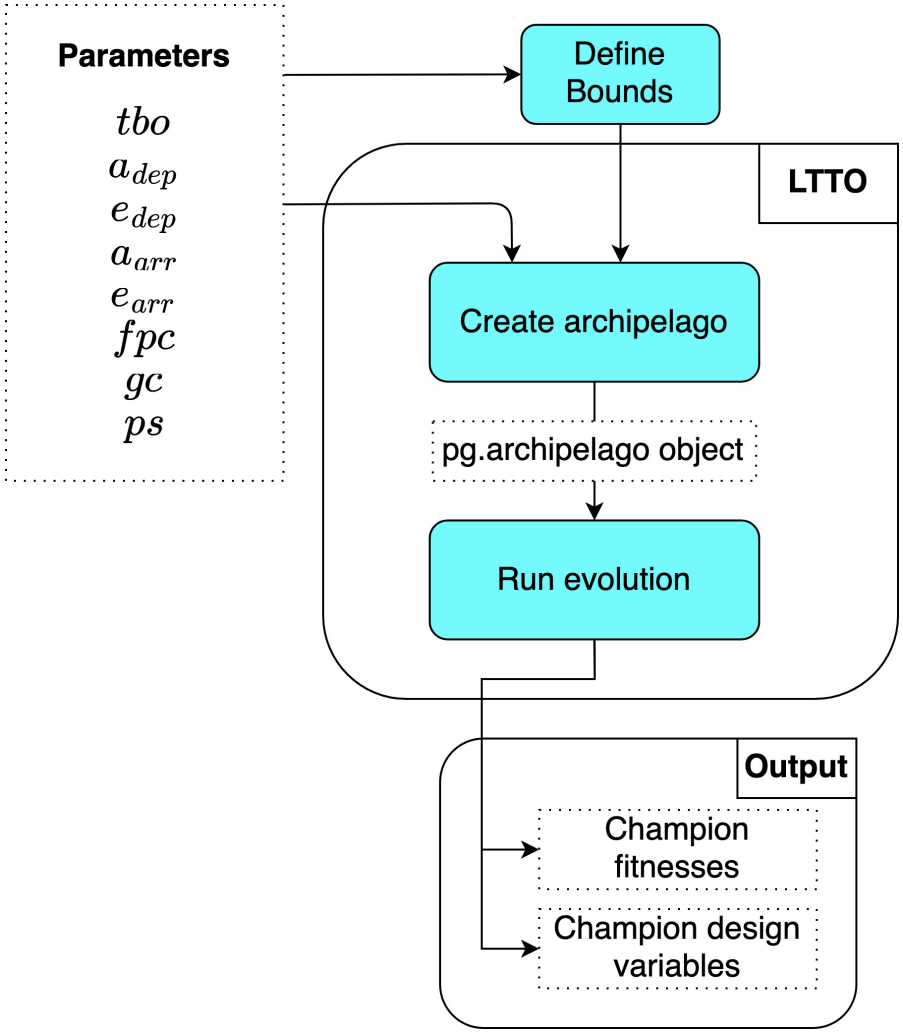


Figure 5.1: Top-level block diagram of the LTTO process.

In Figure 5.1, tbo is the transfer body order, a_{dep} and a_{arr} are the departure and arrival semi-major axes, respectively, e_{dep} and e_{arr} are the departure and arrival eccentricities, respectively. The other Keplerian quantities are taken from the ephemerides database of the departure or arrival body. Furthermore, fpc is the free parameter count, gc is the generation count, and ps is the population size. These parameters are defined in the next section.

5.2. Parameters and design variables

Now that the general structure has been established, parameters and design variables are defined that constitute the inputs to the LTTO. The difference between parameters and design variables is whether the quantities are fixed as is the case for the parameters, or whether a certain range is allowed and the quantities are determined uniquely for each individual in a population.

5.2.1. Parameters

To start with the parameters, the transfer body order is required, which defines the sequence of planets that the spacecraft will perform a GA on. This parameter is given as input from the MGASO optimisation, which is discussed in Part III. The departure semi-major axis and eccentricity and arrival semi-major axis and eccentricity are defined to determine the initial and final state, respectively. The number

of free coefficients is necessary as a parameter to tune several shaping functions that are used to create hodographic-shaping legs. For the optimisation algorithm specifically, the generation count and population size are necessary. All these quantities are summarised in Table 5.1. In addition, if the objective would be a mass-related quantity, then the initial mass and specific impulse would be needed to determine the objective.

Table 5.1: Parameter list for LTTO optimisation.

Parameter Type	Parameter Name	Variable	Unit
General	Transfer Body Order	tbo	-
Mass-related Objective	Initial mass	M_0	kg
	Specific Impulse	I_{sp}	s
Hodographic Shaping	Departure Semi-major Axis	a_{dep}	m
	Departure Eccentricity	e_{dep}	-
	Arrival Semi-major Axis	a_{arr}	m
	Arrival Eccentricity	e_{arr}	-
	Number of free coefficients	fpc	-
Algorithm	Generation count	gc	-
	Population size	ps	-

In the previous section, it was stated that some bounds are dependent on some parameters. The transfer body order determines the number of legs and GAs, and specifically its length. The ToF, number of hodographic-shaping coefficients, and revolutions are all dependent on the number of legs, and the design variable vector will therefore change based on the transfer body order length. The number of GAs is also directly linked to the transfer body order. For each additional GA, an additional variable for each design variable in the 'GA specific' column in Table 5.2 is added. The number of free coefficients determines how many shaping functions are used per axis. Incrementing the number of free coefficients by one means that all three axes need one extra coefficient for the extra base function. As a result, three extra design variables are added to the design space.

5.2.2. Design variables

A number of design variables are input to the optimisation, which are summarised in Table 5.2. Various configurations of these variables – which variables are added – exist, and are discussed later in this subsection.

Several variables are general for the entire trajectory. The departure epoch is crucial for finding optima because the relative phase angle between planets at any given time almost exclusively will provide sub-optimal conditions from which to transfer – the notion of what sub-optimal means is defined by the objective, which is discussed in the next subsection. Therefore, a departure date bound is given that should include at least one optimal relative phase configuration. The departure velocity and arrival velocity are also relevant for mission design; the departure velocity depends greatly on the launch vehicle used and the amount of ΔV that can be provided by that launch vehicle. The arrival velocity needs to be defined, as sometimes the velocity has to be zero in the case of a rendezvous. The exact specification of the departure and arrival state is not necessary. The two angular quantities for both departure and arrival are assumed to be such that the state coincides with the body that it departs from. If the spacecraft departs from Earth, and the departure semi-major axis and eccentricity are set to ∞ and 0 respectively, then the excess hyperbolic velocity vector, v_∞, is 0. The initial state then coincides with that of Earth.

For each leg in the MGA sequence, a ToF range must be defined, which is logical because extremely small ToF values guarantee a high ΔV to reach the target on time, and extremely high ToF values are undesirable from a practical mission standpoint. In addition, long ToF values that exceed the Hohmann transfer time can cause higher thrust accelerations to arrive later at the target. The free coefficients have to be bound as well, based on the findings of [Gondelach and Noomen 2015]. Very large free coefficient

magnitudes result in physically impossible trajectories. The number of revolutions is an essential part of low-thrust trajectory design: a low-thrust trajectory may need multiple revolutions to increase the orbital energy to one that coincides with that of an outer planet, for example.

Depending on the configuration, which is discussed shortly, a different number of unique design variables are needed for each GA in an MGA trajectory. The incoming velocity and GA altitude are defined for each GA manoeuvre; the incoming velocity is the incoming hyperbolic velocity magnitude. Both variables define part of the dynamics of a GA, as defined by [Conway 2010]. If the incoming velocity is too high, the gravity assist body does not 'pull' on the spacecraft long enough and thus will only have a marginal effect on the outgoing hyperbolic velocity vector, while too low of an incoming velocity will capture the spacecraft. The specific choices made with regard to the bounds of these variables are discussed in Chapter 6.

Table 5.2: Design variables for LTTO optimisation.

Design Variable Type	Design Variable Name	Variable	Unit	Configuration
General	Departure Date	t_{dep}	s	I, II, III, IV, V
	Arrival Velocity	v_{arr}	m/s	I, II, III, IV, V
	Departure Velocity	v_{dep}	m/s	I, II, III, IV, V
	Departure In-plane Angle	θ_{dep}	rad	III, V
	Departure Out-of-plane Angle	ϕ_{dep}	rad	III, V
	Arrival In-plane Angle	θ_{arr}	rad	III, V
	Arrival Out-of-plane	ϕ_{arr}	rad	III, V
Leg specific	ToF	tof_i	s	I, II, III, IV, V
	Free Velocity Coefficients	$(c_{10})_i, ..., (c_j)_i$	-	I, II, III, IV, V
	Revolutions	rev_i	-	I, II, III, IV, V
GA specific	Incoming Velocity	v_g	m/s	I, II, III, IV, V
	GA altitude	$r_{p,g}$	rad	I, II, III, IV, V
	Oberth Delta V	ΔV_g	m/s	II
	Orbit orientation angle	β_g	rad	IV, V
	GA In-plane angle	θ_g	rad	IV, V
	GA Out-of-plane angle	ϕ_g	rad	IV, V

Subscripts g and i refer to parameters that have multiple variables equal to the number of GAs or legs, respectively.

The rightmost column in Table 5.2 indicates what variables are included in what configuration. For reference, the configurations are named:

- Configuration I: Basic configuration with necessary variables.
- Configuration II: Configuration I with the addition of an Oberth ΔV.
- Configuration III: Configuration I with the addition of departure and arrival angles.
- Configuration IV: Configuration I with the addition of GA angles.
- Configuration V: Configuration III with the addition of GA angles.

The 'Oberth Delta V' variable can be added to allow for Oberth manoeuvres, in addition to the low-thrust trajectory. While it seems strange to add this if one wants to avoid using both low-thrust and impulsive-thrust engines, its implementation was deemed necessary and is discussed in Chapter 6. Four further variables are defined to customise the departure and arrival velocity vector direction: θ and ϕ for both the departure and arrival velocity. These angles are defined in a TNW frame, as explained in Sections 3.1.1 and 3.3. A velocity vector that coincides with that of the departure and arrival body is most often the case in literature, however, this is not always the case for mission design. Some missions use launch vehicles that can bring the spacecraft into a specific orbit, such as a Geostationary Transfer Orbit (GTO), where the resulting velocity vector differs from Earth. Three variables are defined that fully

define each GA: β, θ and ϕ. To recap, the first angle is an angle that indicates how the spacecraft flies around the body relative to the incoming hyperbolic velocity vector. The other two angles are analogous to the departure and arrival angles but for each GA partner. These angles are separate because they have different requirements: it is easy to fix the departure and arrival angles with this implementation. In addition, a configuration is created with only the GA angles without the departure and arrival angles, as these are fixed for some reference problems. More configurations are possible but are not relevant to the tuning results presented in Chapter 6. These design variables have to be optimised using an optimisation algorithm. What algorithm is chosen and why is discussed in the next section.

5.3. Optimisation algorithm

The choice for an algorithm is based on the availability of mixed-integer capabilities for both objective definitions using the PyGMO framework. This thesis only considers single-objective optimisation due to time constraints, however, future work could include multi-objective optimisation, which would then use an algorithm comparable to the single-objective optimisation. The single-objective algorithms that are capable of solving mixed-integer problems are limited to the Extended Ant Colony Optimisation (GACO), the Improved Harmony Search (IHS), and the SGA. The genetic algorithms are more regularly used in literature which eases the verification of results, and helps in finding initial guesses for any tuning process. Separately, GACO and Multi-Objective Hypervolume-based ACO (MHACO) are not capable of having a design variable vector where the size of the domain of an integer variable is zero. Additionally, GACO requires a minimum population size of 63, which constrains the tuning process for the LTTO. IHS does not have a multi-objective counterpart, which would result in inconsistent use of a type of optimiser. Regarding the multi-objective optimisation, the multi-objective mixed-integer optimisation algorithms are limited to Non-dominated Sorting Genetic Algorithm (NSGA2) and MHACO. MHACO, similarly to GACO, does not allow for the size of the domain of an integer variable to be zero either. NSGA2 does not allow the size of the domain of any design variable to be zero. Yet, for consistency regarding the interpretation of population size and generation count, NSGA2 is more suitable. It can be concluded that the SGA is best used for single-objective optimisation. The NSGA2 algorithm remains a recommendation for the consistency with multi-objective extensions of this thesis.

5.4. Objective formulation

The final step of the LTTO process, as was shown in Figure 5.1, is the objective function evaluation. While ΔV is the most common objective for trajectory optimisation, a number of alternatives can be used as well, summarised in Table 5.3.

Table 5.3: Parameter list for LTTO optimisation.

Objective name	Objective definition	Unit
Delta V	ΔV	m/s
Delivery mass	$-m_f$	kg
Delivery mass fraction	$-\frac{m_f}{m_0}$	-
Propellant mass	$m_0 - m_f$	kg
Propellant mass fraction	$\frac{1-m_f}{m_0}$	-

In Table 5.3, m_f is the delivery mass. Depending on the reference paper, different objectives – or combinations thereof – are used. The hodographic-shaping method returns the thrust acceleration history when solving the EOM based on the velocity profile. To reiterate, this means that the delivery mass can be approximated using the Tsiolkovsky equation as defined in Chapter 3 by integrating the mass over time. The extra integration has a profound effect on the run time of the optimisation. This is taken into account when determining what objectives to use in the optimisation, and by extension also the reference problem definitions used for the optimisation – both the LTTO verification and the MGASO optimisation. Because the algorithms in PyGMO define the optimisation problems as minimisation problems, a minus sign is added to the delivery mass objective. Besides the delivery mass, the propellant mass can also be considered, which is similar in its definition. In addition, the delivery and propellant

mass can both be expressed as a fraction, which normalises the value; normalisation reduces the objective function to a value between zero and one.

In the next chapter, the values and bounds of the parameters and design variables in the LTTO, respectively, are investigated through an extensive tuning process.

6

LTTO tuning

This chapter lays out the process of tuning the LTTO, such that it can create scientifically relevant results – scientifically relevant meaning verifiable, but also consistent and therefore robust. The tuning methodology is discussed and an extensive tuning process is conducted on each reference problem. To conclude this chapter, the final configuration is presented, which is used for the MGASO discussed in Part III.

6.1. Tuning methodology

To shortly recap, three reference problems are presented as test cases, after which a trade-off is made of what parameters and design variables need tuning, and the structure is presented with which the LTTO is tuned.

6.1.1. Test cases

Before the tuning can be performed, test cases must be chosen that represent the complexity of this problem. Any test case shall exclusively use low-thrust propulsion as the source of thrust. It has to be well-defined and reproducible, such that the physical interpretation of the trajectory is clear. Ideally, the test cases shall use a shape-based method to ensure a comparable fidelity level. An alternative method such as Sims Flanagan could work as well but is not pursued here.

As a side note, for the duration of this thesis, specific MGA sequences will be described by a character string. Each planet has a character as a reference, defined below. If a particular problem is being discussed with any departure and arrival planet, then the transfer is denoted by the respective departure and arrival planet name. For instance, an Earth-Saturn transfer may be considered, where EMS is a particular sequence indicating an Earth-Mars-Saturn transfer. Here, Earth-Saturn is used for the transfer type and EMS is used to indicate a particular transfer. This notation prevents confusion between a reference to a transfer problem and a specific transfer sequence within that problem.

- Y : Mercury
- V : Venus
- E : Earth
- M : Mars
- J : Jupiter
- S : Saturn
- U : Uranus
- N : Neptune

As a test case, a simple EM and EY transfer from [Gondelach and Noomen 2015] is an option, because the optimisation bounds are defined well and the global optimum is known. However, the design space

for this problem is rather small compared to those of the reference problems required for this thesis. MGA sequences have multiple legs which have a profound effect on the design variable vector, as discussed in Chapter 5: the departure and arrival velocity are not always trivial, the incoming velocity and GA altitude dimensions are added, and the number of ToF and revolution variables is increased as well. This means that the quality of a tuned EM or EY transfer from [Gondelach and Noomen 2015] is not representative of the actual optimality of the LTTO problem. Instead, the maximum number of GAs in the test cases must be comparable to those used in the relevant literature. Furthermore, keeping the MGASO in mind, it is essential that the reference problems only consider planets as possible GA candidates. This constraint also aims to limit the design space complexity. Several test cases were found and tested, as is discussed in the next three subsubsections.

Earth-Jupiter with coasting

[Morante et al. 2019] is chosen because it performs a sequence optimisation, which can also be used in Part IV. Moreover, the problem is well-defined and a shape-based method is used: generalised logarithmic spirals. Specifically, the Earth-Jupiter transfer is studied and can be used as it is an interplanetary transfer, rather than an asteroid rendezvous, for example. Yet, due to the multi-objective nature of the problem in combination with the inclusion of coasting arcs, the LTTO problem as implemented in this thesis is not expected to be able to exactly reproduce objective values found by [Morante et al. 2019]. Nevertheless, some testing is done to test these hypotheses. The low-thrust sequence optimisation papers are limited in number. [Englander and Conway 2017] is the main other candidate, but its trajectory also implements coasting arcs and uses Sims-Flanagan, rather than a shape-based method. Therefore, [Morante et al. 2019] is preferred. The problem is defined in Table 6.1.

Table 6.1: Problem bounds of Earth-Jupiter transfer from [Morante et al. 2019].

Design Variable	Bound/Value	Unit
Departure Date	[62136, 62865]	MJD
V_{dep}	2000	m/s
θ_{dep}	0	rad
ϕ_{dep}	0	rad
V_{arr}	[0, 7000]	m/s
θ_{arr}	0	rad
ϕ_{arr}	0	rad
ToF	[100, 1500]	days
Revolutions	0	-

The tinted cells represent variables that are not explicitly given in the reference papers, but are inferred or assumed from the results of the reference papers.

It should be noted that the departure date unit is calculated slightly differently in Tudat: all dates are in JD2000, whereas most papers give date units in MJD2000 or MJD. Therefore, there is half a day difference between the day given and what is implemented into Tudat. This difference is minor, but worth noting.

Earth-Jupiter without coasting

Coasting is not expected to be verifiable, and consequently, a reference paper is sought that does not include coasting. [Fan et al. 2021] is a well-suited candidate: it uses Bezier shapes, has no coasting arcs, and focuses on interplanetary sequences. Specifically, the process of minimising ΔV by adding GAs is presented, which provides a suitable platform for verification. This paper does not optimise the sequences, rather it analyses a selection of sequences, making it less ideal for the performance assessment in Part IV. The problem definition is given in Table 6.2. This paper compares the performance of the Bezier shaping method with the Finite Fourier Series (FFS) shaping method and a local refinement step using the Gauss Pseudospectral Method (GPM).

Table 6.2: Problem bounds of Earth-Jupiter transfer from [Fan et al. 2021].

Design Variable	Bound/Value	Unit
Departure Date	[61400, 63400]	MJD
V_{dep}	0	m/s
θ_{dep}	0	rad
ϕ_{dep}	0	rad
V_{arr}	0	m/s
θ_{arr}	0	rad
ϕ_{arr}	0	rad
ToF	[100, 5000]	days
Revolutions	[0, 4]	-

The tinted cells represent variables that are not explicitly given in the reference papers, but are inferred or assumed from the results of the reference papers.

Earth-Neptune without coasting

As is shown later in Section 6.4, different shape-based methods find different values for optimal transfers. For first-order methods, therefore, it is a grey area as to what sequence is the true optimum. [Fan et al. 2021] may find a more optimal result when adding GAs, but this does not guarantee that hodographic shaping should produce the same ranking of optimal sequences. In fact, it is seen later in Section 6.3 that hodographic shaping does not find the same ranking of optimal sequences. Therefore, an extra MGA sequence problem is created, inspired by [Novak and Vasile 2011], to show that adding GAs specifically with hodographic shaping can lead to an improved result consistently. In particular, an Earth-Neptune transfer is defined as shown in Table 6.3, and various GA bodies are added to the sequence to investigate potential ΔV improvements.

Table 6.3: Problem bounds of Earth-Neptune transfer inspired by [Novak and Vasile 2011].

Design Variable	Bound/Value	Unit
Departure Date	[60544, 61744]	MJD
V_{dep}	[0, 3500]	m/s
θ_{dep}	0	rad
ϕ_{dep}	0	rad
V_{arr}	[0, 50]	m/s
θ_{arr}	0	rad
ϕ_{arr}	0	rad
ToF	[5000, 80000]	days
Revolutions	[0, 4]	-

The tinted cells represent variables that are assumed from the results presented in the reference paper.

The departure date bound is a subset of the design space used in [Novak and Vasile 2011], the bounds chosen include two local optima. Conducting a grid search for the other departure dates is not useful, as the relative difference in quality between sequences is the quantity of interest, which is expected to be the comparable for different departure date windows. The goal for this test case is not to recreate results found by the paper, but rather to have an independent comparison of various MGA sequences. Next, the quantities that will be tuned are presented.

6.1.2. Tuning structure

This subsection presents the structure of the tuning process. For instance, the steps covering different aspects that need to be tuned or the resolution of a grid search. To clarify what tuning exactly means:

several inputs to a system are varied to determine the effect on the output and to find the optimal inputs for the optimisation. Part of the tuning process is discovering whether the desired accuracy can be achieved when tuned. For LTTO, sufficient objective accuracy within a reasonable run time is essential for the robustness of the MGA sequence optimisation. Concretely, a robust approach is one that finds consistent and verifiable objective values. Due to the limited time available for this thesis, not all relevant quantities can be tuned.

Relevance of a tuning process

To test whether results even have to be tuned, two runs are compared from [Fan et al. 2021]: an EJ and an EMJ transfer. If a relatively simple GA transfer performs worse than a direct transfer – where literature finds a better performance – then tuning is needed to converge to more optimal results. This step assumes that the optimal objective values of the reference problems are correct.

An initial comparison is made using [Fan et al. 2021]. In Table 6.4, it can be seen that the ΔV values for the direct transfer are similar. The difference in ΔV is 0.5%, which is a negligible difference for low-fidelity methods. Therefore, this result verifies the accuracy of a single-leg low-thrust arc compared to another shape-based method. The verification of the hodographic-shaping method compared to numerical results can be found in Section 9.1.1. The EMJ transfer, contrary to the EJ transfer, shows a 16 km/s higher ΔV. This result is not accurate whatsoever, and therefore a tuning process is required for MGA transfers.

Table 6.4: Final comparison of minimum ΔV for the EJ and EMJ transfer.

Source	Transfer	
	EJ [km/s]	EMJ [km/s]
[Fan et al. 2021] (Bezier shaping)	17.81	15.18
[Fan et al. 2021] (FFS)	17.90	16.98
This thesis (Hodographic shaping)	17.78	33.03

Relevance of tuning various aspects

The relevance of a quantity – parameter or variable – for tuning is ultimately defined by its effect on the objective of the optimisation process. This effect often comes to fruition when a quantity affects the complexity of the design space. A larger or more complex design space makes it harder for any optimisation algorithm to converge to an optimum. Adding an extra design variable scales the design complexity exponentially, whereas an increase in a bound of a design variable only scales the design space linearly. Therefore, preventing the addition of an extra variable is more impactful than tuning a specific bound.

First, the CPU time and quality of a single LTTO run are determined predominantly by the inputs to the optimisation algorithm itself: the population size, ps, and generation count, gc. Each generation needs to evaluate the entire population, thus the computational complexity is approximately $ps \cdot gc$, given a constant selection of quantities. These parameters have a time complexity of O(n), and may therefore be detrimental to the run-time. This thesis strives not only for robust results but also for a performance increase from a mission design point of view. Therefore, the run time and computational complexity are important. Consequently, these two quantities are tested first in Section 6.3.1. Specifically, a grid search is conducted in Section 6.3.1. Of course, to ultimately determine performance, run time is not used as this is hardware and software dependent. Second, the configurations presented in Section 5.2.2 are investigated. The optimal configuration also determines the number of bounds that need tuning, which is discussed next. Third, tuning bounds – as defined in Table 5.2 – is an example of a group of factors that has an effect on the design space. A departure date bound that is twice as large will result in an analogous increase in the size of the design space, which can be expected to affect the performance of the optimisation process. A fine grid search of all these lower and upper bounds is not feasible, though a selection of bounds can be made based on astrodynamics knowledge and bounds found in the test case reference papers.

Fourth, another factor for the quality of the LTTO is the number of free coefficients. Note, that this is different from the fpc parameter. The complexity linked to the number of free coefficients is also

dependent on the number of legs and the dimension. The number of free coefficients is defined as $fpc \cdot$ dimension $\cdot$ no_of_legs. This means that for every extra leg up to six additional coefficients are added (in the case that $fpc = 2$ and three dimensions are considered). This has a significant effect on the design complexity, and therefore the number of coefficients must be tuned. It was found by [Gondelach and Noomen 2015] that having more than two free coefficients has a negligible effect on accuracy, while requiring significantly more computation time. Consequently, three options are investigated in Section 6.3.4: zero, one, or two free coefficients. Fifth, as multiple islands will be used to strengthen the statistical robustness of the results, a topology can be used for further improvement – introduced in Section 2.3.3. This topology can take islands that have converged to a sub-optimum, and pass new individuals to them which has a profound effect on the distribution of optimal objective values for multiple islands, and therefore also on the optimality of a given sequence. This is touched on in Section 6.3.5. Sixth, the option for a local optimisation is explored using a gradient-based method to see if the results can be forced to a more optimal solution. This addition is discussed in Section 6.3.7. Last but not least, the optimisation algorithm can be tuned as well. The SGA algorithm, discussed in Section 5.3, has numerous options for configuration on top of ps and gc. The three operators that determine the subsequent population can all be changed in various ways. For instance, the mutation operator has a type, and a probability, both of which may have noticeable effects on the variety of the population, and ultimately on the distribution of the objective values of each island. There are too many parameters to tune, so the importance of each parameter of each operator is determined, and a grid search is done in two iterations to determine the best performance.

Before the tuning is performed, there are a few constraints that influenced the decision-making for the simulations – discussed in Section 2.2.1. Most importantly, as DelftBlue is used, a time limit of 24 hours is imposed on each simulation. This limits the vastness of the simulations, and therefore certain tuning steps have to be changed accordingly. The next three sections present the tuning results for the various test cases shown in Section 6.1.1.

6.2. Earth-Jupiter with coasting

This section discusses findings regarding the Earth-Jupiter transfer as defined by [Morante et al. 2019]. As was touched on previously, [Morante et al. 2019] uses coasting and implements a multi-objective optimisation. However, it is still worth testing whether the resulting objective values are at all comparable to those found by [Morante et al. 2019].

[Morante et al. 2019] uses two steps: the first step is based on a shaping method that uses logarithmic spirals and the second step performs a local, single-objective optimisation method. It is therefore best to compare the results found in this section with the first step – using logarithmic spirals, as this step is closer to the fidelity level of the optimisation in this thesis. A few transfers are deemed close to optimal or Pareto-optimal by [Morante et al. 2019]: EJ, EVJ, EMJ, EVEJ, EVEMJ. The Pareto-optimal trajectories found for these sequences are shown in Figure 6.1. The EVEMJ transfer is tested as this is the most complex sequence and immediately gives an indication of the proximity to a verifiable result. Note that these sequences are optimal only within the bounds of the problem; other departure dates would almost certainly show different optimal sequences.

Because [Morante et al. 2019] uses the propellant mass fraction as one of the objectives – ToF being the other one – an additional integration is required, which was discussed in Section 5.4. This results in a slow optimisation, which further limits the complexity of the problem that can be simulated. Optimisations with 300 generations, 300 individuals, and eight different seeds for the initial population – more precisely on eight different islands – were run. The NSGA2 algorithm was used and the objectives are defined using Table 5.3. In the interest of time, only 'Configuration I' was used. These configurations affect the results, as will be shown in Section 6.3.2, but it is expected that they would not have shown significant improvement because of the fundamental differences in the model used by [Morante et al. 2019] – coasting in particular.

For EVEMJ, shown in Table 6.5, the smallest propellant mass fraction was 0.96 with a ToF value of 3.766 years. The propellant mass fraction is almost 10 times larger than was found by [Morante et al. 2019]. This result is substantially different and it can therefore not be seen as a useful result. The reason for this difference could be the coasting arcs, but it could also be the complexity of the problem

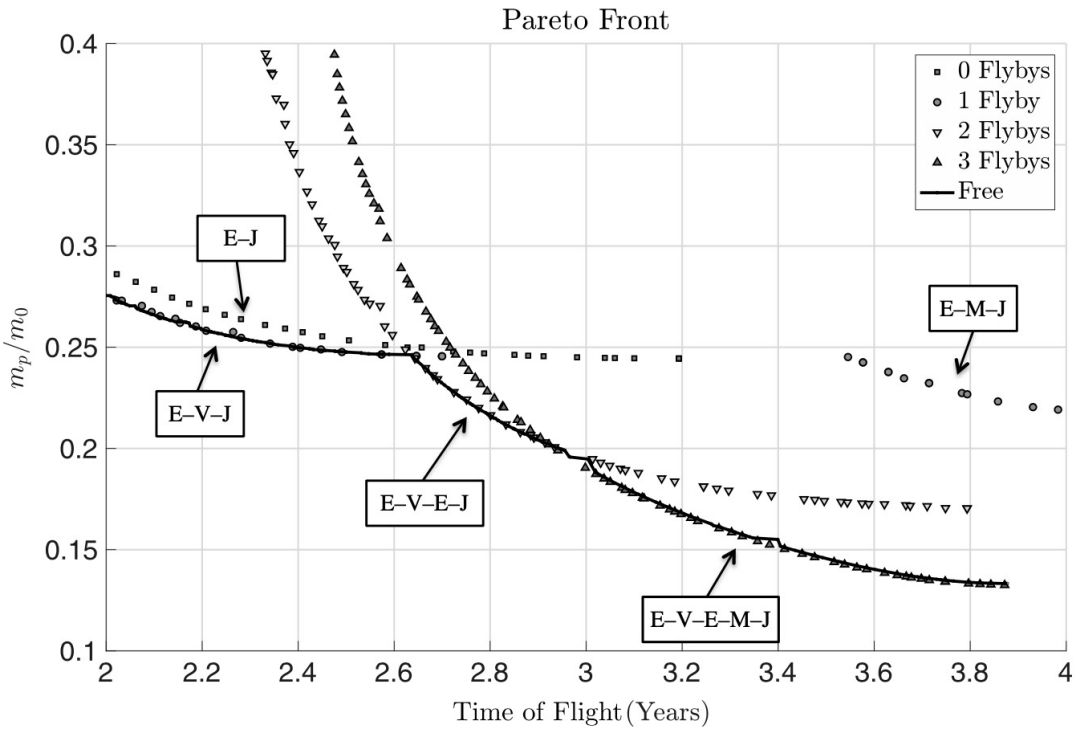


Figure 6.1: Pareto-optimal solutions for EJ, EVJ, EMJ, EVEJ, and EVEMJ transfer found by [Morante et al. 2019].

caused by the number of GAs. Subsequently, to check the second possibility, the EVEMJ trajectory was dissected into its separate legs, testing EV, EVE, EVEM, respectively to see how swiftly the propellant mass value deteriorated. The resulting average propellant mass fractions and ToF for eight different initial population seeds are shown in Table 6.5 for EVEMJ and separate legs.

Table 6.5: Average propellant mass fraction and ToF for EVEMJ partial sequences.

Transfer body order [-]	Average propellant mass fraction [-]	Average ToF [days]
EV	0.218	145
EVE	0.887	172
EVEM	0.965	607
EVEMJ	0.970	1101

To ensure that the propellant mass fraction is accurate, tests were run with various values for the number of integration points. This parameter represents the number of time steps that is used in the integration of the thrust acceleration and mass to determine the ΔV and propellant mass fraction, respectively. As is shown in [Gondelach and Noomen 2015], from about 25 points onwards there is only a slight increase in accuracy for the ΔV values. Because the ΔV calculations are mathematically linked to the mass fraction, it is expected that this number of points also suffices for accurately estimating the mass fraction using Tsiolkovsky as defined in Equation (3.6).

As a side note, multi-revolution trajectories are not considered in [Morante et al. 2019]. The multi-objective algorithm NSGA2, as discussed in Section 5.3, is not capable of modelling only zero revolution transfers. A feature is added that artificially forces the number of full revolutions to be zero. This implementation is inefficient as the revolution variables for each leg are not removed, but have no effect. This may have affected the convergence of the results, but it is unlikely that this is causing the results from Table 6.5. An alternative is to create another configuration, besides those mentioned in Section 5.2, though the zero revolution case is rather unique for low-thrust MGA applications, and is therefore not

worth implementing.

Additional tests were done where various parameters were constrained to limit the design space. The departure date was both fixed to a single day – equal to the optimal departure date found by [Morante et al. 2019] and expanded to include a wider range of departure dates. The latter is expected to produce better results because the coasting feature results in significantly different ToF values and departure dates as compared to the reference solution. However, both of these options showed no improvement. A few tests were conducted with higher generation counts and population sizes, as well as zero, one, and two free coefficient counts, all of which also showed no improvement in the propellant mass fraction found.

It can therefore be concluded that this reference problem can not be readily reproduced with the models and implementation used in this thesis. Next, the other Earth-Jupiter case is considered that did not include coasting.

6.3. Earth-Jupiter without coasting

This section shows the tuning results applied to the Earth-Jupiter transfer from [Fan et al. 2021]. This problem is expected to provide better results because it is single-objective and has no coasting arcs, in contrast to [Morante et al. 2019]. In particular, an EJ, EMJ, EEMJ, and EEEMJ trajectory are analysed. For verification, the results of the EEEMJ trajectory are presented for the same reasons that the EVEMJ transfer was chosen in the previous section: the LTTO needs to be able to model the longest possible sequence with enough accuracy that it can be readily compared to shorter sequences. More results are produced for the other sequences and various side steps during this section, which are given in Appendix C. This section is rather long, so it is broken down into many steps.

6.3.1. Population size and generation count

As was established in Section 6.1.2, changing the population size and generation count is crucial for the accuracy, but also for the computation time. In Section 6.3.6, other parameters that are expected to have a more nuanced effect on the LTTO accuracy are also investigated. A coarse grid search of these two parameters is performed: the population size is analysed at 100, 600, and 1200, and the generation count is analysed at 30, 100, and 300. The resolution is a trade-off of computational time and quality of the results; it is expected that this resolution is enough to determine a sufficiently optimal combination. These parameters are not independent of the rest of the tuning process: the population size and generation count affect the tuning of the other optimisation parameters presented in Section 6.3.6, but there is not enough time to investigate these dependencies.

Several choices have to be made regarding the settings of the other parameters and variables, as these have not been tuned yet. In general, a large design space complexity is desired for these tests as this maximises the difference in the quality of results, thereby giving a more clear recommendation for the optimal set of parameters. For this, the configuration is defined accordingly, summarised in Table 6.6. To start with the configuration, 'Configuration V' is used. This is the configuration with the most parameters, thus maximising the design complexity. Do note that the Oberth manoeuvre variable is not included, because [Fan et al. 2021] does not include them. The EEEMJ transfer is evaluated as it has the largest design space complexity. As will be discussed in Section 6.3.4, one free parameter for the EEEMJ transfer is faster, without a decrease in results. In this case, the increase in the free parameter count complicates the design space dramatically and it is expected that no population size or generation count would produce proper results. The other parameters, besides ps and gc, are defaults discussed previously. As for the departure date, the 2000-day window that is investigated in [Fan et al. 2021] is large, but not necessary for this grid search. The design space needs to include a region with and without the expected optimum, but should otherwise be made as small as possible to decrease the occurrence of any sub-optimal wells. An interval was chosen of 400 days because this is the smallest window that includes very high and very low fitness values – assuming the optima are spread out regularly and are of the same width. This assumption can readily be made because of the relatively constant synodic periods of the various planets. The synodic period is defined as [Wakker 2015]:

"The synodic period is the time that passes between two consecutive transits of the satellite through a certain meridian (Section 11.1) on the Earth's surface. In interplanetary spaceflight, the concept of a

synodic period is also in use. There, it indicates the time it takes the Sun, the Earth and another planet to return to their original relative positions (Section 18.7).”

This definition is used for further discussion regarding the relevance of synodic periods. In this case, the choice for 400 days is driven by the synodic period of Earth-Jupiter. The remaining design variables are set to defaults discussed in Section 6.3.3.

Table 6.6: Configuration for the ps and gc grid search.

(a) Parameters

Parameter Name	Value	Unit
tbo	EEEMJ	-
a_{dep}	∞	m
e_{dep}	0	-
a_{arr}	∞	m
e_{arr}	0	-
fpc	2	-
gc	TBD	-
ps	TBD	-

(b) Design Variables

Design Variable Name	Bound/Value	Unit
Departure Date	[61672, 62072]	MJD
V_{dep}	0	m/s
θ_{dep}	0	rad
ϕ_{dep}	0	rad
V_{arr}	0	m/s
θ_{arr}	0	rad
ϕ_{arr}	0	rad
ToF	[100, 4500]	days
Free Coefficients	$[-10^4, 10^4]$	-
Revolutions	[0, 2]	-
$\vec{V}_{\infty,in}$	[0, 5000]	m/s
r_p	$[2 \cdot 10^5, 10^9]$	m
β_g	$[0, 2\pi]$	rad
θ_g	$[0, 2\pi]$	rad
ϕ_g	$[-\frac{\pi}{4}, \frac{\pi}{4}]$	rad

The configuration includes the departure and arrival conditions as parameters with no possible values other than zero. As will be seen in Section 6.3.2, this does not make a different performance-wise. With this configuration in mind, the simulations are run and the results are found in Table 6.7. Four islands were used initially – islands perform the same optimisation, however, the simulations did not consistently find similar ΔV values. These results are therefore not robust or consistent. Of course, small ps and gc values inhibit the algorithm from finding similar optima, but even with the larger generation counts and population sizes the results were not consistent. Consequently, it is concluded that using not only the minimum value but also the mean value can help in determining the fitness of a specific configuration. The mean value in this case becomes less sensitive with more islands. It was therefore decided to use 24 islands. The number 24 is based on the maximum number of CPUs that DelftBlue seems to regard as a high-priority list. Higher CPU counts would wait in the queue for significantly longer. Therefore, the grid search is conducted with 24 islands, of which the results are summarised in Table 6.7.

Table 6.7: Grid search results of EEEMJ transfer for ps and gc parameters with 24 islands.

Minimum ΔV [km/s]			Generation count		
			30	100	300
Population size	100	Min	36.552	23.964	19.848
		Mean	63.823	39.393	33.720
	600	Min	25.376	18.239	18.986
		Mean	38.155	24.357	22.410
	1200	Min	24.279	18.753	17.469
		Mean	30.692	22.551	20.569

In Table 6.7, it can be seen that both the mean and minimum ΔV values improve with higher population sizes and generation counts. There is one exception: for a population size of 600 with 100 and 300 generations, the minimum value increases. This exception can be attributed to chance: an island coincidentally converged to a lower optimum. This is confirmed by the fact that the mean value for these two runs does follow the expected trend of improvement. This result exemplifies the use of the mean to determine the fitness of an island. From a population size of 100 to 600, there is approximately a 40% improvement across all generations, and from 600 to 1200 individuals the improvement ranges from 20% for 30 generations to about a 5% improvement for 100 and 300 generations. This comes at a computational cost, for each increase in ps and gc, there is an approximate increase of 600% in computational time. These trends and improvements are mirrored by the increase in generation count: a 40% improvement from 30 to 100, and a 5-20% improvement from 100 to 300 generations. To minimise the computational complexity while maintaining sufficient accuracy, a trade-off should be made, but because this subsection investigates whether the results from [Fan et al. 2021] are reproducible, a higher accuracy is preferred. The only constraint is the 24-hour maximum set by DelftBlue. For the sake of the tuning process, to see if the results can be reproduced a generation count of 300 is chosen and a population size of 1200. As an additional reason, the topology – which allows for the migration of individuals – is found to only be useful with higher generation counts. The CPU time of the most resource-intensive combination of generation count and population size is 80 minutes, which allows for many iterations within a short time frame because eight concurrent jobs are allowed as mentioned in Section 2.2.1.

The ΔV values found by [Fan et al. 2021] are shown in Table 6.8. This table is used throughout, to verify the quality of the results. It can be seen that the ΔV values found in Table 6.7 for high population sizes and generation counts are still more than three km/s higher than those found by [Fan et al. 2021] using the Bezier shapes. This shows that further tuning is required to converge to the reference objective values.

Table 6.8: Comparison of the Bezier shape-based method, the FFS shape-based method and the Gauss Pseudospectral Method (GPM) [Fan et al. 2021].

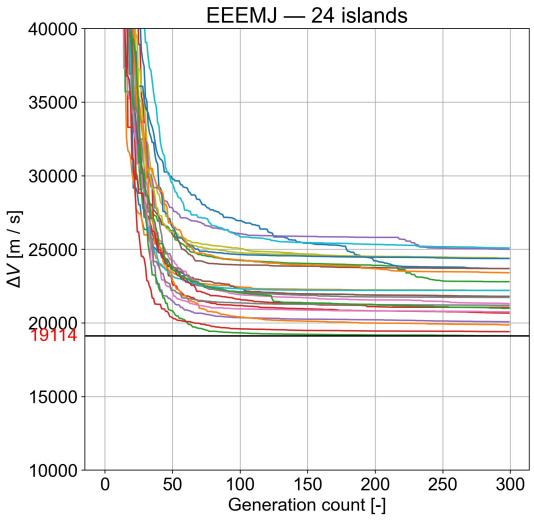
Transfer process	Launch date [MJD]	ΔV [km/s]		
		Bezier	FFS	GPM
No-gravity-assist: EJ	62654	17.81	17.90	17.49
One-gravity-assist: EMJ	61872	15.18	16.98	14.83
Two-gravity-assist: EEMJ	61452	14.95	16.30	14.85
Three-gravity-assist: EEEMJ	61872	14.39	15.92	14.32

Now that the computational complexity has roughly been fixed and the need for further tuning has been established, the various configurations are tested to see what parameters should be added to the design space to improve the accuracy without increasing the computational complexity unnecessarily.

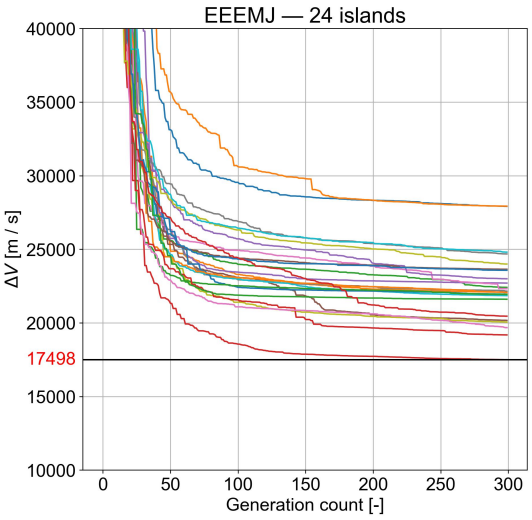
6.3.2. Configurations

A key part of reproducing results is making sure that the optimum is actually inside the design space. This means that all the variables that make up a trajectory have to be fixed, or added as a design variable. Multiple configurations were therefore defined in Section 5.2, and this subsection presents the results of various configurations. EEEMJ is chosen for these simulations because the added parameters have more opportunity to improve the sequence.

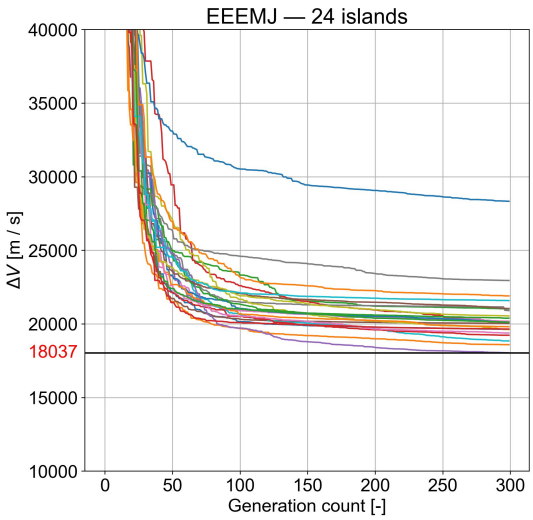
Regarding 'Configuration II', while [Fan et al. 2021] does not allow for any Oberth manoeuvres, a short test was conducted to check for an improvement when adding such manoeuvres. The addition showed no particular improvement, and 'Configuration II' was therefore not further considered. For 'Configuration III', as the departure and arrival conditions are fixed, these quantities do not need to be added as this would unnecessarily complicate the design space. 'Configuration IV' is tested as the GA-related angles are expected to be essential for finding optimal values. As an additional test, to see the negative effects of increasing the design complexity with no theoretical benefit, 'Configuration V' is tested as well. The ΔV values per generation per island are shown in Figure 6.2 for the various configurations. Each colour represents an island that converges in parallel.



(a) 'Configuration I' configuration



(b) 'Configuration IV' configuration



(c) 'Configuration V' configuration

Figure 6.2: ΔV per generation with various LTTO configurations for an EEEMJ transfer with 24 islands.

Omitting a few outliers at the lower and higher end of the ΔV spectrum in Figures 6.2b and 6.2c that

can be filtered out, the spread of the islands is comparable – read similar standard deviations. These outliers, particularly those in the higher ΔV region, can be mitigated with a topology, which is discussed in Section 6.3.5. In Figure 6.2, it can be seen that there is a significant improvement from 'Configuration I' in Figure 6.2a to 'Configuration IV' and 'Configuration V' in Figures 6.2b and 6.2c regarding minimum ΔV. The lower minimum ΔV with these configurations was observed with multiple runs and different seeds. These configurations can therefore be concluded to be more robust. This result is also plausible because a trajectory is tested with three GAs and with 'Configuration I' the spacecraft is forced to approach each GA planet from the same direction in the TNW reference frame. Moreover, the spacecraft flies around the same side of the planet because β is fixed in the local frame. The hodographic-shaping legs therefore converge to shapes that fulfil that constraint and in almost all cases these angles are not optimal at zero – zero is the default.

Surprisingly, when comparing 'Configuration IV' to 'Configuration V', no consistent difference is observed when adding the departure and arrival quantities that are forced to be zero. The minimum ΔV is higher in 'Configuration V', but the mean is lower. In runs with other seeds this trend was no longer observed. Therefore, because other test cases may require the definition of these angles, 'Configuration V' is henceforth used unless specified otherwise. The fact that the minimum and mean ΔV are different depending on the seed does indicate a lack of robustness, which indicates that further tuning is required. In addition, the individual quality of the results instead of the relative performance of the various configurations gives an indication of the current performance relative to the results from [Fan et al. 2021]. The EEEMJ transfer converges to a minimum observed ΔV of 17.5 km/s, which is still multiple km/s too high. This leads to the next tuning step: the bounds of the design variables.

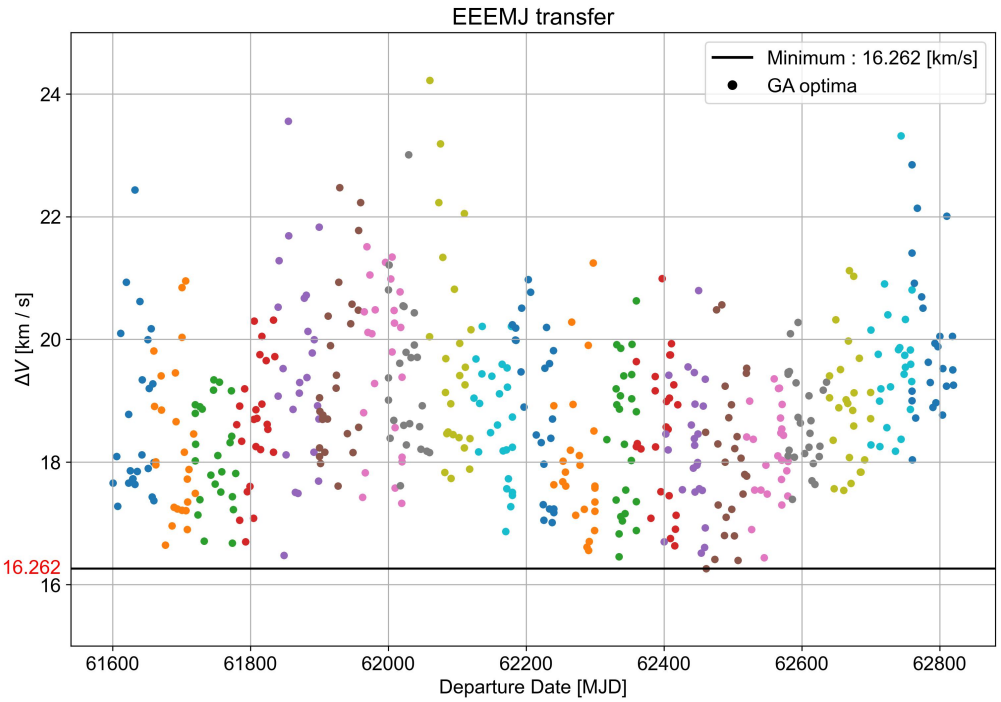
6.3.3. Design space exploration

As previously alluded to, the bounds of the design variables should be subject to some analysis: a random choice can have a detrimental effect on the accuracy. A very large bound may mean that the algorithm does not converge and only finds sub-optimal values, but too small of a bound may exclude the true optimum. In this thesis, only static bounds are considered. A short analysis of dynamic bounds is discussed in Appendix E, but this remains as a recommendation for future work. For the determination of the various variables, 'Configuration V' is used. Note that the tuning of these values is case specific, so for other test cases new tuning may have to be done for certain bounds.

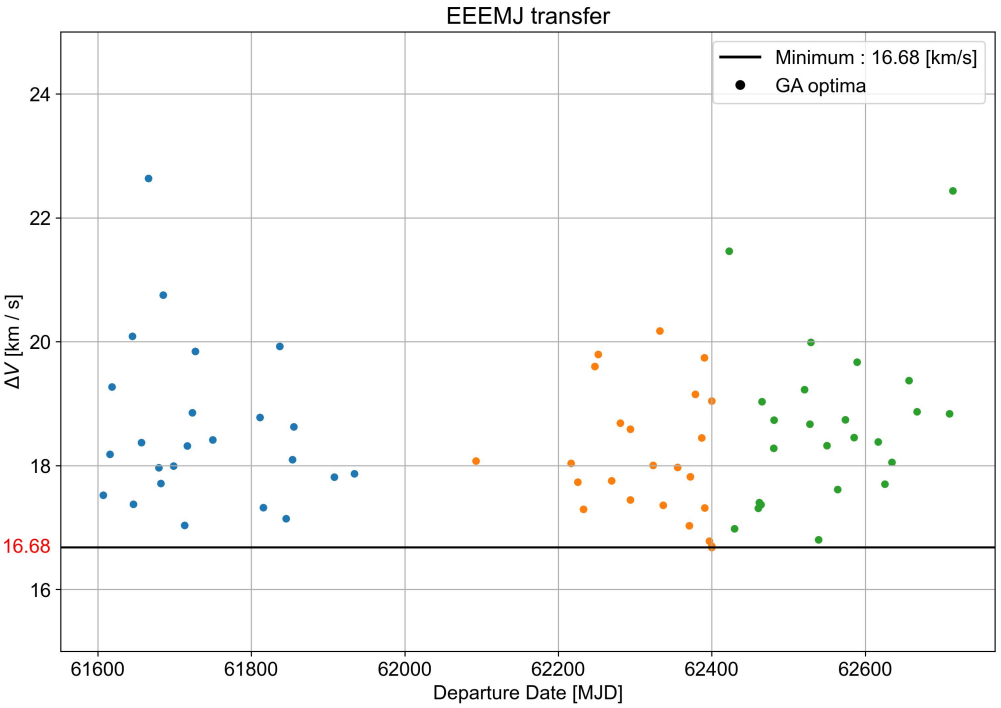
Before reading on, as this subsection is rather long, a summary of the results from tuning these bounds can be found at the end of this subsection.

Departure date

First, the simulations are defined that are run in this step of the tuning process. To start with the general parameters as seen in Table 5.2, specific departure date bounds are often a requirement for a mission, and would therefore not be subject to any tuning. However, if no specification is given in the reference paper, a departure date bound does need determination. This variable determines to a significant extent the optimal sequence – as will be seen shortly because the departure date determines the phasing of the celestial bodies at the time of departure. [Fan et al. 2021] presents results as a grid search over departure date values, so performing a grid search with the implementation in this thesis can act as verification of the results. Two departure date windows are tested thoroughly. A grid search of optimisations over a 1200-day departure date window, with either 60-day bounds or 400-day bounds. The 1200-day window is chosen because it includes at least one synodic period, and it is divisible by the two bound values. Results for both bounds are shown in Figure 6.3. For this simulation, the ToF bounds are set manually per GA based on the ToF values found in [Fan et al. 2021] with an extra buffer, which is defined in Table 6.9. The other parameters and bounds are set to the defaults defined in Table 6.6.



(a) Grid search from 61400-62800 [MJD] with 60-day optimisation bound.



(b) Grid search from 61400-62800 [MJD] with 400-day optimisation bound.

Figure 6.3: Comparison between 60- and 400-day bounds for departure date grid search of EEEMJ transfer.

In Figure 6.3, various colours can be seen that are linked to successive departure date intervals. 24 optima are plotted per departure date interval because 24 islands are used. The absolute minimum is also shown for the entire window. There is a clear periodicity in Figure 6.3a and a less pronounced periodicity in Figure 6.3b. The period is not completely clear but is approximately 750 days. This period is logical because the relative phasing of planets determines to a large extent the optimality of the sequence, even with the revolutions and ToF values being flexible. In these sequences, there are EE,

EM, and MJ transfer legs. These transfer legs all have respective synodic periods. The synodic period of Earth-Mars is about 780 days, which is close to the 750-day estimate. This result is confirmed by the results found in [Fan et al. 2021] and can also be observed for other transfers (EMJ and EEEMJ). However, the EEMJ transfer does not overlap, likely due to a difference in phasing caused by the singular EE transfer. Not enough information is provided by [Fan et al. 2021] to verify this expectation. The Earth-Mars synodic period is the longest of all the transfer legs and is thus the limiting factor which is why it is the most pronounced in the results. The same can be applied to the EJ transfer, where the Earth-Jupiter synodic period is around 399 days and the observed periodic optimality is also around 400 days. The synodic period is approximate because it is an average value over longer time periods. The results for the other transfers can be found in Appendix C.1. The actual phasing of these periods is almost identical, with a difference of approximately 50 days. The difference is hard to determine concretely because the lowest ΔV regions are 200 days wide. Moreover, the number of points that are found are not shown to be exhaustive in this thesis nor in [Fan et al. 2021]. The small approximate shift that can be observed can be attributed to small model differences, such as ephemerides. Unfortunately, the reference paper does not state the source of the ephemerides, and therefore this cannot be verified.

For the quantitative analysis, the minimum ΔV and the difference in the various optima are crucial. Only the 60 day bound is analysed, followed by the analysis of the 400-day bound and the comparison of the two bounds. The minimum ΔV values at every 60-day interval over the entire window are around 2 km/s higher than the results from [Fan et al. 2021], shown in Figure 6.4. This indicates that further tuning needs to be done or that the LTTO implementation can not readily reproduce values of this type of optimisation problem. Some reasons for this increase in ΔV are possible. On the one hand, other bounds may not include the optimum that is found in Figure 6.4. On the other hand, the simulations maybe did not converge to the global optima, but rather to a local one – it has been verified that the islands all indeed converged. The difference between the lowest ΔV candidates – if one were to fit a curve through them – is considered. The results show a difference of up to 1.5 km/s depending on the departure date observed, whereas [Fan et al. 2021] finds a difference of approximately 6 km/s. The differences in the spread of optimal ΔV values can most likely be attributed to the low-thrust trajectory model. The Bezier shape implementation has an explicit maximum thrust acceleration value, whereas hodographic shaping does not – the reason for this was explained in Section 5.1. The Bezier shapes, as a result, have less freedom to make up for the worse phasing of the planets and therefore find relatively higher ΔV values than the minimum, as compared to the difference in Figure 6.3a.

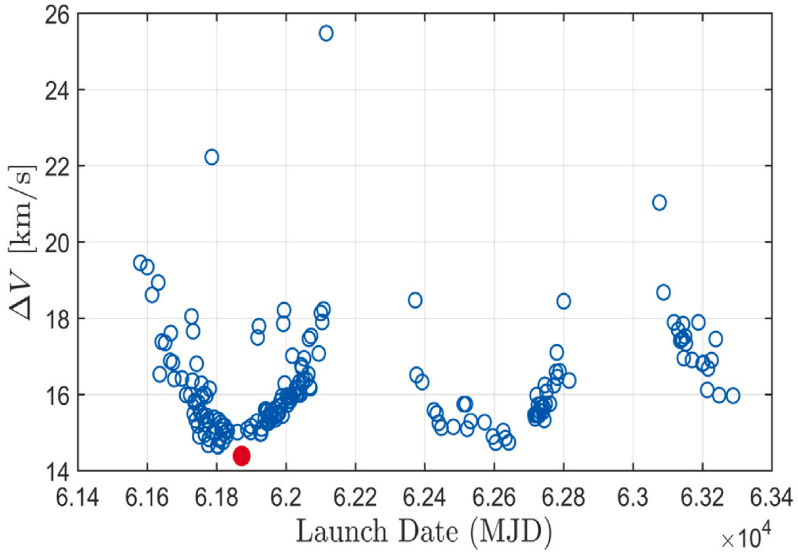


Figure 6.4: Grid search from 61400-63400 [MJD] for EEEMJ transfer. Figure 12 from [Fan et al. 2021].

In Figure 6.3b, there are significantly fewer data points because the number of points is constrained by the number of islands and the number of intervals. Consequently, as mentioned before, the trends observed in the 60-day bound results are less pronounced in the 400-day bound results. The minimum

ΔV value is 400 m/s higher, which is very competitive considering the 60-day bound minimum is 2 km/s higher than the Bezier shape minimum. The difference in lowest ΔV values across the entire window is roughly the same as with the 60-day intervals, although this quantity is statistically less accurate. The fact that the optimum converges to around the same epoch, together with the similar periodicity and trends in optimality, shows that the 400-day bound results are physically relevant, but less robust. It is in any case possible to converge to ΔV values that are close to the minimum of the 60-day bound results. The 60-day bound is still the better option for tuning and verification purposes, despite the six times higher computational cost. For optimisation, the trade-off is different because a six times longer computation time for an increase of 400 m/s may not be worthwhile for a first-order method. For higher fidelities, a 400 m/s increase would be more important.

Compared to the results in Figure 6.3, the results found by [Fan et al. 2021] in Figure 6.4 show an exhaustive enumeration of the valid trajectories found, rather than only the optimal one per optimisation per interval. The Bezier shape-based method that is employed has more constraints that lead to a smaller number of valid results, however, the results are competitive nevertheless. The difference in valid results can also be affected by the computational resources used.

It can be concluded that the robustness of the results and optimality of the sequence are dependent on the size of the departure date interval. More importantly, the departure date itself has an enormous effect on the optimality of the sequence. The quality of the results decreases with an increase in the departure date bound, as one would expect. Having tested only two interval sizes, the ideal departure date interval can not be determined, though it can be concluded that the optimum lies somewhere between the two tested values. When using this problem definition henceforth, a departure date of [61842, 61902] is used, because all four transfers have a local optimum around this departure date – these local optima can be observed in Appendix C.1.

Departure and arrival state
The departure and arrival velocities are often also fixed, as is also the case in [Fan et al. 2021], however, that is not always the case. Pre-defined velocity conditions could include a rendezvous or a flyby as an intermediate step before or after analysing a trajectory in a different frame – for instance, the Jovian system. The model used by [Fan et al. 2021] states that the spacecraft departs and arrives with a velocity vector equivalent to that of the departure and arrival body, respectively, indicating that V_∞ is 0. As the reference papers that may be used for the performance assessment also assume these conditions, the tuning of these parameters is not necessary for this work.

Time of flight
Some leg-specific variables must be investigated as well. The ToF is more challenging: ToF values differ depending on the departure and arrival body. An Earth-Earth transfer can be orders of magnitude shorter than a direct Earth-Jupiter transfer.

The results from the simulation from Section 6.3.1 with 300 generations and a population size of 1200 can be seen in Figure 6.5. As an initial guess, the minimum and maximum values were taken from [Fan et al. 2021] for the lower and upper bound, respectively: [100, 4500] days. It can be seen in Figure 6.5 that the EE legs are limited to values between roughly 100 and 800 days. This means that the lower bound may be too high and the optimisation wants to converge to a value lower than the lower bound for multiple islands. The lower bound of the EE leg is subsequently updated to 20 days, which is verified to be wide enough to include all optima. For the EM leg, the values are between 200 and 1600 days and for the MJ leg the values are between 1500 and 3500 days. Therefore, the values for these legs can be considered reasonable. The ToF values per leg are summarised in Table 6.9. These values are used in the simulations henceforth unless specified otherwise. Again, these values are specific to this transfer and are not universally applicable. To give a feeling of the size of these windows, each value is scaled relative to an EM Hohmann transfer which spans approximately 259 days.

These ToF bounds are of course also dependent on revolution count and to a lesser extent some other variables. However, Figure 6.5 includes 24 islands that have converged to various revolution counts, and even with the different revolution counts, the ToF values are limited to those specified before. These manual ToF bounds are therefore a safe guess. More iterations could be performed but the difference is not expected to be substantial. With the addition of these manual bounds, there is

no observable difference in optimal ΔV values, so the relaxation of these bounds is possible without sacrificing performance.

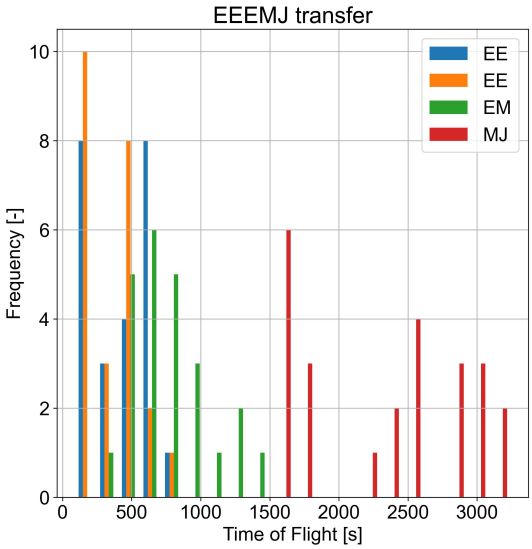


Figure 6.5: Histogram of the ToF for various legs for an EEEMJ transfer with 24 islands.

Table 6.9: Manual ToF bounds for EEEMJ transfer legs.

Leg	ToF bound [days]	Scaled ToF bound [-]
EE	[20, 1000]	[0.077, 3.861]
EM	[200, 1500]	[0.772, 5.792]
MJ	[1000, 4000]	[3.861, 15.444]

The scaled ToF bound is given by the fraction of the ToF bound divided by the EM Hohmann transfer time, which is 259 days.

Revolutions

The number of full revolutions is determined based on the maximum number of full revolutions that are used in [Fan et al. 2021], which is two. The other parameters and bounds are sourced from the problem definition from Section 6.3.1, with the addition of the manual ToF bounds. The revolution counts found in this simulation can be seen in Figure 6.6.

The revolution count rarely converged to two revolutions. Moreover, there is no significant difference in the spread of the number of revolutions per leg, with the exception that the MJ leg has no two revolution islands, and the EM leg has significantly more two revolution optima than the EE transfers. This trend indicates that the bounds do not have to be extended. As a further check, the number of revolutions found in the departure date grid search can also be analysed per departure date window. When observing the seven intervals of 60 days for the departure date grid search in Figure 6.3a, the number of islands with two revolutions per interval was [0, 0, 0, 0, 2, 3, 7] islands. There is a distinct trend in these results: the number of islands with two revolution solutions increases for later departure dates, which implies that – for the EEEMJ transfer with this departure date window – the phasing of the planets is such that higher revolution counts are beneficial for later departure dates within that window. Due to time constraints, further analysis was not possible. The presence of this trend does mean that, depending on the window, it is possible that more than two revolutions must be permitted. For tuning and verification purposes the original values are considered.

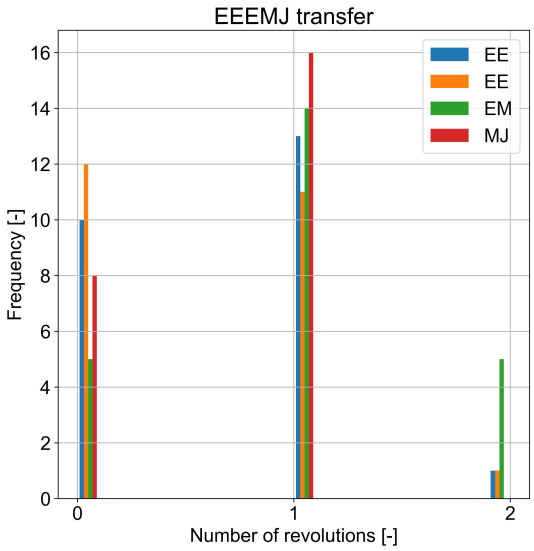
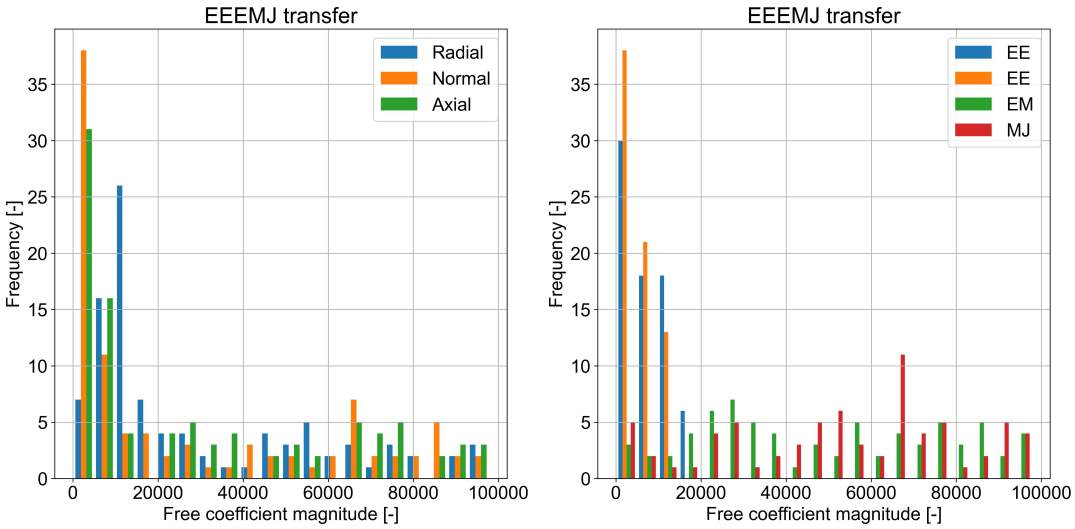


Figure 6.6: Histogram of the full revolution count for various legs for an EEEMJ transfer with 24 islands.

Free coefficients

As the final leg-related quantity, the free coefficient bounds are considered. The range of free coefficient values is also essential for producing accurate shapes that use low ΔV values. Small bounds, similar to the previous variables, may exclude the optimum from the design space. Very wide bounds often produce coefficients that lead to invalid incoming or outgoing velocities for the GA manoeuvres or negative heliocentric radius values during transfer legs. [Gondelach and Noomen 2015] explains the reasoning behind these phenomena that occur with hodographic shaping. The implementation in this thesis sets the objective value of invalid individuals to 10^{16} – thereby filtering them out of the population within a few generations – as they are physically impossible. Individuals with a high potential are filtered out after one generation, so more nuanced penalty functions could have been investigated, but this option is reserved for cases where all coefficient ranges resulted in these invalid individuals. The simulations use the same configuration as previously.



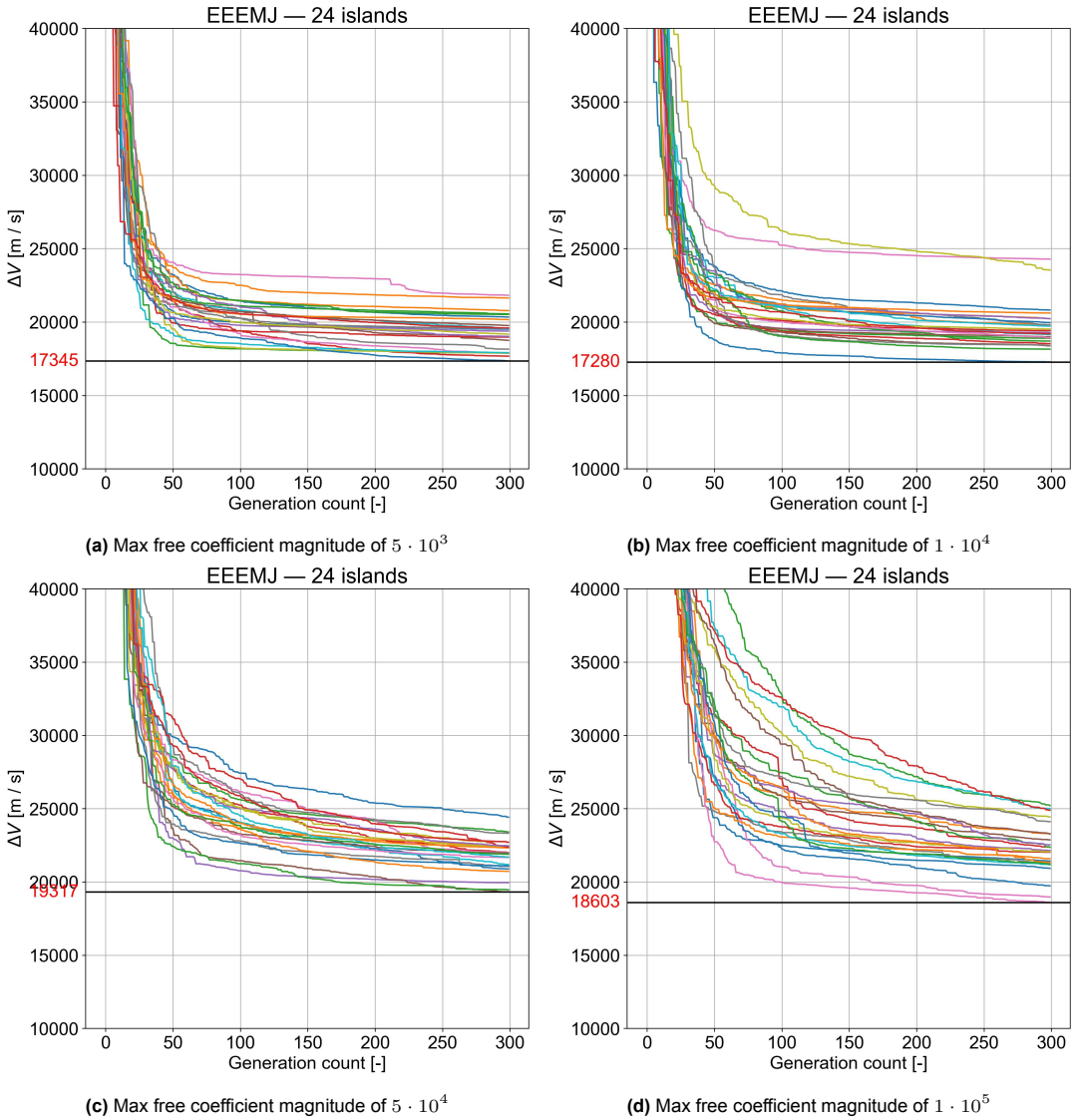
(a) Histogram sorted by axis.

(b) Histogram sorted by leg.

Figure 6.7: Histograms with 1e5 free coefficient magnitudes sorted per leg and per axis for two EEEMJ transfers with 24 islands.

A coarse grid search is conducted with various bounds to check what results in the lowest ΔV. Specifically, the free coefficient magnitudes are set to four values: 5e3, 1e4, 5e4, and 1e5. The coefficients

can be both positive and negative. A histogram is plotted in Figure 6.7 using the largest bound as a limit to see to what values the islands converge to. The largest bound gives the most insight into what values coefficients converge to, however, the increased bound does affect the complexity of the optimisation which can affect the convergence. Though similar to before, it is not expected that this has a detrimental effect on the convergence, and therefore the results are still useful. The ΔV values per generation per island are presented in Figure 6.8 to see how various bounds affect the optimality. Figure 6.7 is split into two sub-figures, sorted by leg and by axis. This distinction provides further information on the convergence of the free coefficient values and the potential effect of their bounds.



(a) Max free coefficient magnitude of $5 \cdot 10^3$

(b) Max free coefficient magnitude of $1 \cdot 10^4$

(c) Max free coefficient magnitude of $5 \cdot 10^4$

(d) Max free coefficient magnitude of $1 \cdot 10^5$

Figure 6.8: ΔV values per generation for various free coefficient bounds of an EEEMJ transfer with 24 islands.

As can be seen in Figure 6.7, the far majority of all coefficients across all islands converge to values below 15000. The other results are evenly spread out over the bound, with one exception. The EE legs do not have coefficients higher than 20000, shown in Figure 6.7b. This can be explained by the fact that EE legs can often be described by relatively simple shapes with low low-thrust acceleration values. Moreover, a smaller thrust acceleration means a smaller change in velocity and therefore smaller coefficient values are necessary to represent the velocity hodograph. In Figure 6.7a, while there is some discrepancy between the specific values of the axes, the differences do not show any particular trend. The evenly spread results indicate that the shaping functions for all three axes require similar free coefficient bounds. If the shaping functions of one particular axis could not readily represent the shape

properly, the free coefficients would converge to higher values, which would be visible as an anomaly. Regarding the validity of results with the higher free coefficient magnitudes, coefficients larger than $\approx 2 \cdot 10^4$ returned predominantly invalid trajectories, which is solved by the filtering mechanism explained previously. Such an invalid result consequently means that the optimisation uses that individual for one generation and applies the genetic algorithm operations ineffectively, leading to a slower convergence.

In Figure 6.8, the convergence of various bounds can be compared by looking at the slope of the islands per generation. With increasing bounds, an increase in slope can be observed on average across all generations. This trend is most visible in Figures 6.8c and 6.8d. The result is plausible, as a higher bound results in a more complex design space and therefore has more difficulty converging – assuming an equal number of generations and population size. Additionally, the minimum ΔV gets consistently worse, which is explained by the same lack of convergence. As only bounds under $1 \cdot 10^4$ converge properly and result in the lowest ΔV, combined with the previously made observations that most islands converge to coefficient magnitudes of below $1 \cdot 10^4$, it is therefore concluded that the bounds shall be kept at $1 \cdot 10^4$. This conclusion is specific to the EEEMJ transfer, and other transfers may vary. Especially for transfer legs towards the Sun, it is expected that the free coefficient values may change, following the analysis of [Gondelach and Noomen 2015] that found substantially different optimal free coefficient values for a transfer leg toward Mercury. A gradient-based method is shown by [Gondelach and Noomen 2015] to work well for optimising free coefficients efficiently, which could be employed as an additional and separate optimisation for each leg. Due to time constraints, no further investigation is possible.

In conclusion, the bounds for the free coefficient magnitudes remain unchanged at $1 \cdot 10^4$. The next subsections discuss GA-specific bounds.

Incoming velocity

The remaining parameters are related to the GAs themselves, starting with the incoming velocity. Before reading on, for the final results of the tuning of these bounds, the reader can skip forward to the end of this subsection. To refresh, the incoming velocity is the incoming hyperbolic velocity vector magnitude with respect to the GA planet. From a theoretical point of view, as the incoming velocity is a small bound compared to the free coefficient values for example, it will converge more quickly and is therefore not the bottleneck for convergence. Of course, it does contribute to the general design complexity and with that the convergence. While the bound interval is small in size, a smaller change in the bound may have a larger effect on the outcome, compared to for instance the GA altitude. The incoming velocity has a direct impact on the amount of thrust that needs to be given to meet the velocity boundary conditions, a one km/s difference can also translate to one km/s extra ΔV – approximately. The GA altitude, however, has a smaller effect, which is discussed in the next subsubsection: if the GA altitude were to differ by one km, the effect that has on the final ΔV will not be as impactful. The optimisation problem is defined analogously to the previous subsubsections. The histogram with the frequency of various incoming velocity values, sorted by GA, is shown in Figure 6.9.

It can be seen in Figure 6.9 that the values range from almost 0 m/s to almost 4000 m/s, which covers most of the search space that was defined as [0, 5000] m/s. Looking at the astrodynamics behind this quantity, explained in Section 3.3, it is known that the smaller the incoming velocity is, the longer the GA body gravitationally attracts the spacecraft, and the larger the deflection angle will be. It seems then that the algorithm converges to a situation where larger deflection angles are optimal. This can be explained by the fact that lower incoming velocities can result in higher changes in heliocentric velocity, which may lead to lower thrust values. If the bound allows for very low incoming velocity values, there is more freedom for the GAs to play a larger role in decreasing the ΔV. Looking at Figure 6.9, the incoming velocity magnitude differs significantly per GA body, which is not surprising: in the case of the EEEMJ transfer, the first two Earth GAs – combined with the low-thrust propulsion – will generally not have generated enough ΔV to have high incoming velocities of multiple km/s. The Mars GA follows after two Earth GAs, and therefore already has more heliocentric velocity, which explains the higher incoming velocity optima found. As the incoming velocity bounds differ greatly depending on the GA target and the position in the sequence, dynamic bounds would be useful in decreasing the design space. An introduction to this is given in Appendix E, but it is not implemented in this thesis due to time constraints. The initial bound was [0, 5000] m/s, which was an accurate initial estimate, and therefore is used henceforth as well.

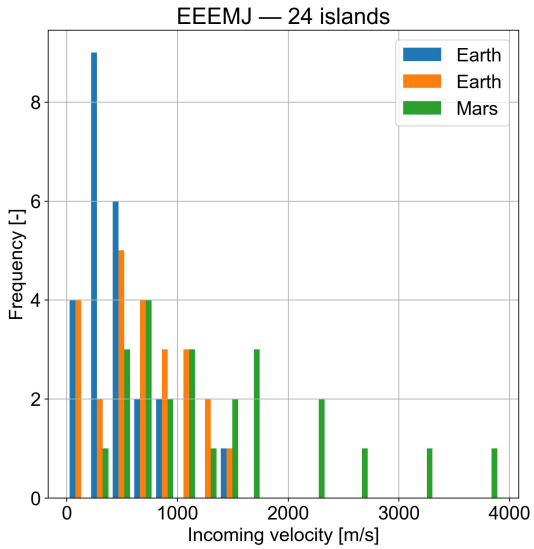
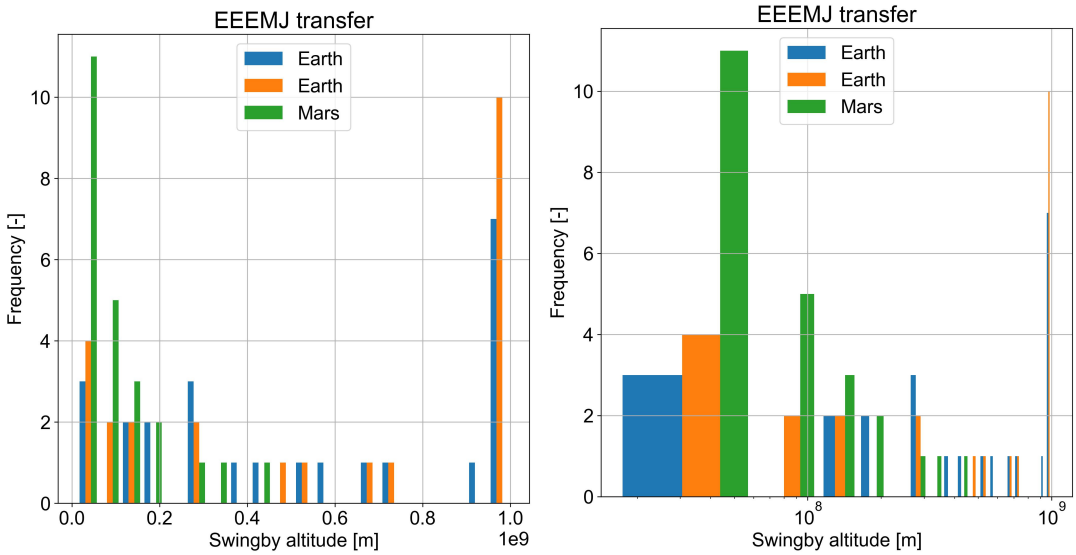


Figure 6.9: Histograms of incoming velocity and GA altitude for each GA of EEEMJ transfer with 24 islands.

GA altitude

The GA altitude is naturally a wide interval, as the physical distance at which one can fly by a planet comprises many orders of magnitude. Based on GAs of past missions and other reference papers, generally, the flyby altitude is no closer than 200 km from the surface, therefore the lower bound is set to 200 km. An initial guess for the upper bound is set to 1,000,000 km, based on the very small two-body point mass gravitational acceleration that can be achieved when performing a GA at this altitude. In Figure 6.10, the GA altitude optima per island are shown. The figure consists of two sub-figures, which show the same data with different x-axis scales. The x-axis in Figure 6.10b is scaled logarithmically to visualise the minimum GA altitude observed. As is mostly the case, the same problem definition is used as in the previous subsubsections.



(a) Linear scale.

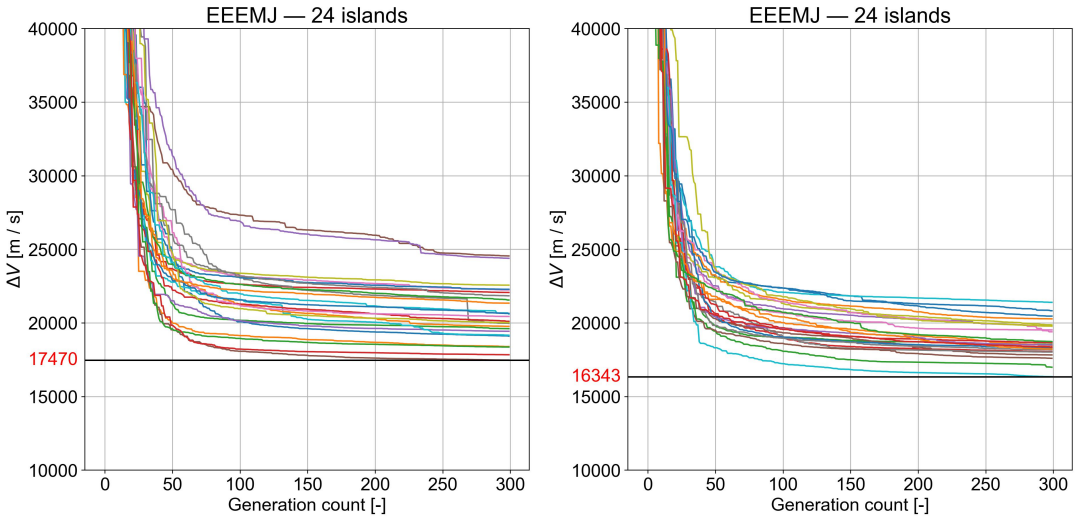
(b) Logarithmic x-axis.

Figure 6.10: Histograms of GA altitude per GA of EEEMJ transfer with 24 islands.

As seen in Figure 6.10b, the data is grouped into two general regions: one is in the order of magnitude of 10^7 m, and the other is located at the maximum bound. Specifically, the minimum GA altitude found was 12,487 km, and the maximum altitude is 999,796 km. The group at the upper bound only consists of the

Earth GAs, shown in Figure 6.10a. Many islands approach the upper bound, which indicates that the upper bound is too small and that for certain optima – for the Earth GA in particular – involving certain revolution counts, higher GA altitudes are desired. Consequently, a few upper-bound extensions are considered, and discussed shortly. The rest of the Earth GAs converge mostly to lower GA altitudes in the 10^7 m order of magnitude, with small exceptions spread out between the lower and upper groups of optima. The very high GA altitudes are interesting because a higher value results in a smaller heliocentric change in angular momentum, which means that certain solutions converge to optima that minimise the effect of the GA. In certain cases, where the relative phasing is aligned in a specific way, this is plausible.

The lower group specifically is dominated by the Mars GA. The maximum Mars GA optimum converges to $2 \cdot 10^8$ m, shown in Figure 6.10. A lower altitude means a higher gravitational attraction, and therefore a larger difference in heliocentric angular momentum. This result, together with Figure 6.9, shows more clearly that the Mars GA is the one that provides most of the ΔV from the GAs in the EEEMJ transfer. In future research, the link between these bounds can be investigated. Specifically, the GA altitude might be correlated for certain GAs with certain incoming velocity values and revolution count values.



(a) Histogram of GA altitude with [200, 1e6] km bound.

(b) Histogram of GA altitude with [200, 5e6] km bound.

(c) Histogram of GA altitude with [200, 5e7] km bound.

(d) Histogram of GA altitude with [200, 1e8] km bound.

Figure 6.11: Histograms of various GA altitudes for an EEEMJ transfer with 24 islands.

For further tuning of this bound, a coarse selection of three extra upper bounds is chosen and compared

in Figure 6.11. Upper bounds of [1e9, 5e9, 5e10, 1e11] m are compared. Looking at Figure 6.11b, the minimum ΔV decreases by more than 1 km/s when increasing the bound by a factor of five. However, the best optimum found in Figure 6.11b is an outlier and the second best optimum is already $\approx$ 600 m/s higher, so the minimum value does not show any consistent improvement. There are no high outliers, compared to the smaller bound in Figure 6.11a, which is expected to be a coincidence. For more reliable results, multiple seeds should be run, however, due to time constraints, this is not done. When analysing the maximum GA altitude found by the increased bound, there is still a group that converges to the upper bound, shown in Table 6.10. A further increase in the upper bound by a factor of 10 – seen in Figure 6.11c – shows an increase in minimum ΔV, however, there is no increase in the mean or standard deviation. This result indicates that the islands perform similarly despite the increased maximum bound. For every maximum bound value, the maximum GA altitude found across the optima of all islands approaches the maximum bound. It seems, therefore, that irrespective of the bound some islands will converge to the largest number possible. The GA model does not take into account that it is physically infeasible: it is a design variable that can be set to any value. When increasing the maximum bound further, the mean and standard deviation of the ΔV values across all islands increases again.

Table 6.10: Statistics on various GA altitude bounds.

GA Altitude Bound [km]	Min [km]	Max [km]	Mean [km]	Std [km]
$1{\cdot}10^6$	$1.248{\cdot}10^4$	$9.998{\cdot}10^5$	$4.033{\cdot}10^5$	$3.821{\cdot}10^5$
$5{\cdot}10^6$	$3.548{\cdot}10^3$	$4.992{\cdot}10^6$	$1.245{\cdot}10^6$	$1.570{\cdot}10^6$
$5{\cdot}10^7$	$1.144{\cdot}10^3$	$4.982{\cdot}10^7$	$1.531{\cdot}10^7$	$1.937{\cdot}10^7$
$1{\cdot}10^8$	$1.511{\cdot}10^4$	$9.953{\cdot}10^7$	$2.794{\cdot}10^7$	$3.751{\cdot}10^7$

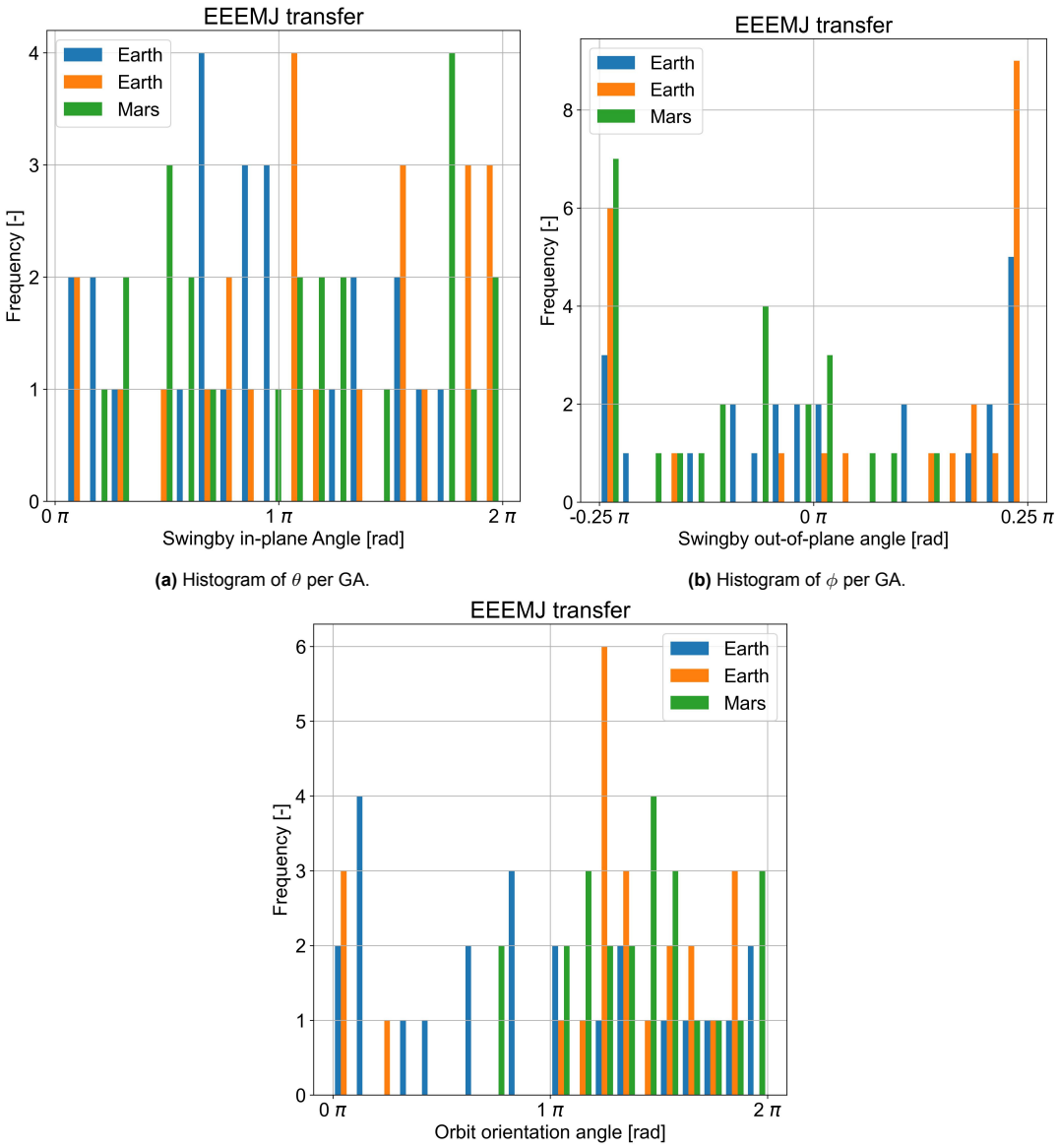
In conclusion, the bound of 5e10 m is chosen for further simulations, as the convergence does not observably decrease, whereas it has been shown that the optima all lie within the bound. It should be mentioned that these variables are not independent of one another, meaning that it is theoretically possible that this tuning process is still leaving out the global optimum. In addition, this is specific to the EEEMJ case, though it is expected for an Earth-Jupiter transfer that these values will not differ such that they will lie far outside the chosen bound. Due to time constraints, these areas are not investigated further.

GA specific angles

Finally, the in-plane (θ), out-of-plane (ϕ), and orbit orientation (β) angle bounds are considered. To shortly recap, θ and ϕ define the incoming hyperbolic velocity vector relative to the planet velocity vector in a TNW frame. β defines the outgoing velocity vector direction relative to a local frame defined by the incoming hyperbolic velocity vector and planet velocity vector. The mathematical definition can be found in Section 3.3.

These quantities are angular and thus are limited to a small bound. Though, a small change in these quantities can theoretically have a major effect on the optimality, depending on the value of the other quantities related to the dynamics of a GA. For example, if the incoming velocity is low and the GA altitude is small, then the GA will have a larger effect. Consequently, a poor choice for the direction of the outgoing velocity vector would immediately result in a substantial decrease in the optimality of that particular individual.

Like previous bounds, the histograms are presented in Figure 6.12. The simulations are those defined in Section 6.3.1, which includes the entire possible bound for θ and β, and a broad bound for ϕ. Specifically, ϕ is limited to $[-\frac{\pi}{4}, \frac{\pi}{4}]$, which is based on the expectation that due to the proximity of all planets to the ecliptic, ϕ would not need to differ greatly from the planet velocity vector in the out-of-plane direction. However, it is seen in Figure 6.12b that the values converge mostly to the boundaries of the bound. This observation is confirmed by the minimum and maximum value approaching $|\frac{\pi}{4}|$. The ϕ bound should therefore be extended to the full bound, which should not have too large of an effect on the design complexity due to its small size, despite the relative increase of 100%. There is a difference between the Earth and Mars GAs: in Figure 6.12b the Mars GA converges to lower ϕ values, though due to time constraints, this is not further investigated.



(a) Histogram of θ per GA.

(b) Histogram of ϕ per GA.

(c) Histogram of β per GA.

Figure 6.12: Histograms of GA specific angles for an EEEMJ transfer with 24 islands.

Regarding the distribution of θ values in Figure 6.12a, there is an even spread of Earth GAs. This is somewhat plausible, as the GAs occur shortly after departure, where the velocity vector is already quite close to that of Earth, and depending on how the hodographic-shaping function chooses the velocity profile, the optimal θ may lie anywhere in the 2π range. The difficulty with this parameter is that it must be defined manually, per definition, but intuitively a large fraction of the bound is sub-optimal based on any given trajectory. The optimal θ also depends on the number of revolutions – which affects the amount of time the spacecraft can accelerate, but also the ToF and departure date. These dependencies further obfuscate the expected values of θ and are exemplified by the similar results between Earth and Mars GAs.

As for the last quantity, in Figure 6.12c, it can be seen that there is a spread across the entire possible bound, though there is a skew to values below π for the first Earth GA and values above π for the second Earth GA and Mars GA. The analysis of this quantity is challenging, as the physical interpretation is only relative to the local frame defined by the other two angular quantities. As mentioned in Section 3.3, this formulation could be changed in future work so the physical interpretation is more straightforward, for example by including the deflection angle as a parameter.

If $\beta < \pi$, then the spacecraft will turn further away from the planet velocity vector. This can physically be interpreted by the spacecraft flying by on the inside of the planet relative to the Sun. If $\beta > \pi$, then the spacecraft turns towards the planet velocity vector, which means that the spacecraft flies by on the outside of the planet relative to the Sun. The assumption is made that δ is smaller than twice the size of the angle between the incoming velocity vector and the planet velocity vector. Otherwise, the spacecraft flies by on the inside even if $\beta > \pi$. Flying by on the outside increases the angular momentum relative to the Sun, whereas flying on the inside decreases it. This holds for the assumption that θ is between 0 to $\frac{\pi}{2}$ and $\frac{3\pi}{2}$ to 2π, for the other values, the opposite is true. In the EEEMJ case, for the second and third GA, the distribution is skewed towards values larger than π. In this case, it is plausible that the velocity has increased enough that the spacecraft should fly by the outside of the respective planet. Only this is not confirmed by θ, as no skew is observed there. Another explanation for the lack of a clear trend may again be the many dependencies. For example, two different departure dates with different revolution counts will logically result in different relative phases of the GA bodies, and the angle from which one approaches the planet will then be different.

In conclusion, while some trends can be found, this is particular to the EEEMJ case, and to ensure that the MGASO does not prune away optimal points, the entire bounds are kept. Further investigation is not possible due to time constraints, but it would be useful to map out the link between these three quantities more to see their effect on the optimality of the trajectory.

Final bound configuration
After this extensive tuning process for the bounds of the 'Earth-Jupiter without coasting' reference problem, the findings are summarised in Table 6.11.

Table 6.11: Complete problem definition of EJ transfer using Table 6.2.

Design Variable	Bound/Value	Unit
Departure Date Bound	[61842, 61902]	days
V_{dep}	0	m/s
θ_{dep}	0	rad
ϕ_{dep}	0	rad
V_{arr}	0	m/s
θ_{arr}	0	rad
ϕ_{arr}	0	rad
Manual ToF	[[20, 20, 200, 1500],[1000, 1000, 1500, 4000]	days
Revolutions	[0, 2]	-
Free Coefficients	$[-10^4, 10^4]$	-
$\vec{V}_{\infty,in}$	[0, 5000]	m/s
r_p	$[2 \cdot 10^5, 5 \cdot 10^{10}]$	m
β_g	$[0, 2\pi]$	rad
θ_g	$[0, 2\pi]$	rad
ϕ_g	$[-\frac{\pi}{2}, \frac{\pi}{2}]$	rad

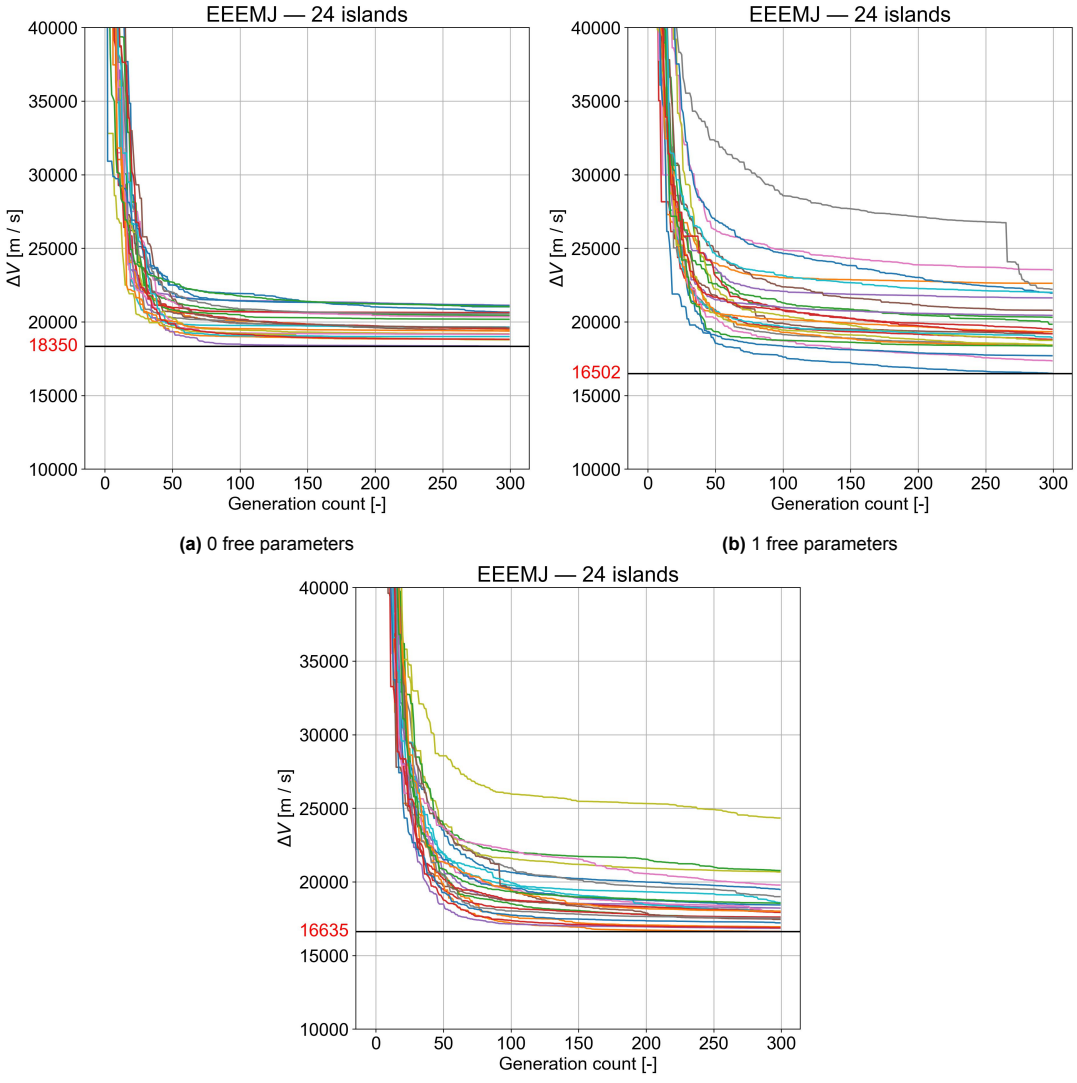
6.3.4. Free coefficient count

The tuning process is not complete yet, as the ΔV values have not improved significantly. This subsection presents the findings of varying the free parameter count (fpc).

[Gondelach and Noomen 2015] has shown that more than two free coefficients do not result in a useful improvement in accuracy. More than two free coefficients lead to significantly blown-up computation times. For this reason, tests are done with zero, one, and two free coefficients. Additionally, a comparison is made of various transfer sequence lengths for two free parameters. fpc is tested on all trajectories from [Fan et al. 2021] because the free coefficient has a profound effect on the accuracy and complexity of the result: the accuracy differs when legs are added or subtracted. The results for these two comparisons

are shown below in Figure 6.13, and Figure 6.14. The simulation uses the final configuration as defined in Table 6.11.

In Figure 6.13, the EEEMJ transfer is shown with the various fpc values. It can be seen that the configuration with zero free parameters in Figure 6.13a converges swiftly compared to one and two free parameters. The best ΔV values are found after roughly 150 generations. However, the minimum ΔV is almost 2 km/s higher than the simulations with one and two free parameters, and assuming the spread of results for the one and two free parameter cases can be improved, it can be concluded that the zero free parameter case does not include the global optimum – or close to it – in its design space. The one free parameter case in Figure 6.13b shows the lowest overall ΔV, but with a higher mean value of 19.756 km/s, compared to the 18.431 km/s of the two free parameter case. The downside of the two free parameter case is the computation time, which is found to be roughly twice as high for this simulation – it depends on the number of legs, for example. The differences in minimum and mean values of these 24 island simulations stay consistent with different seeds.



(a) 0 free parameters

(b) 1 free parameters

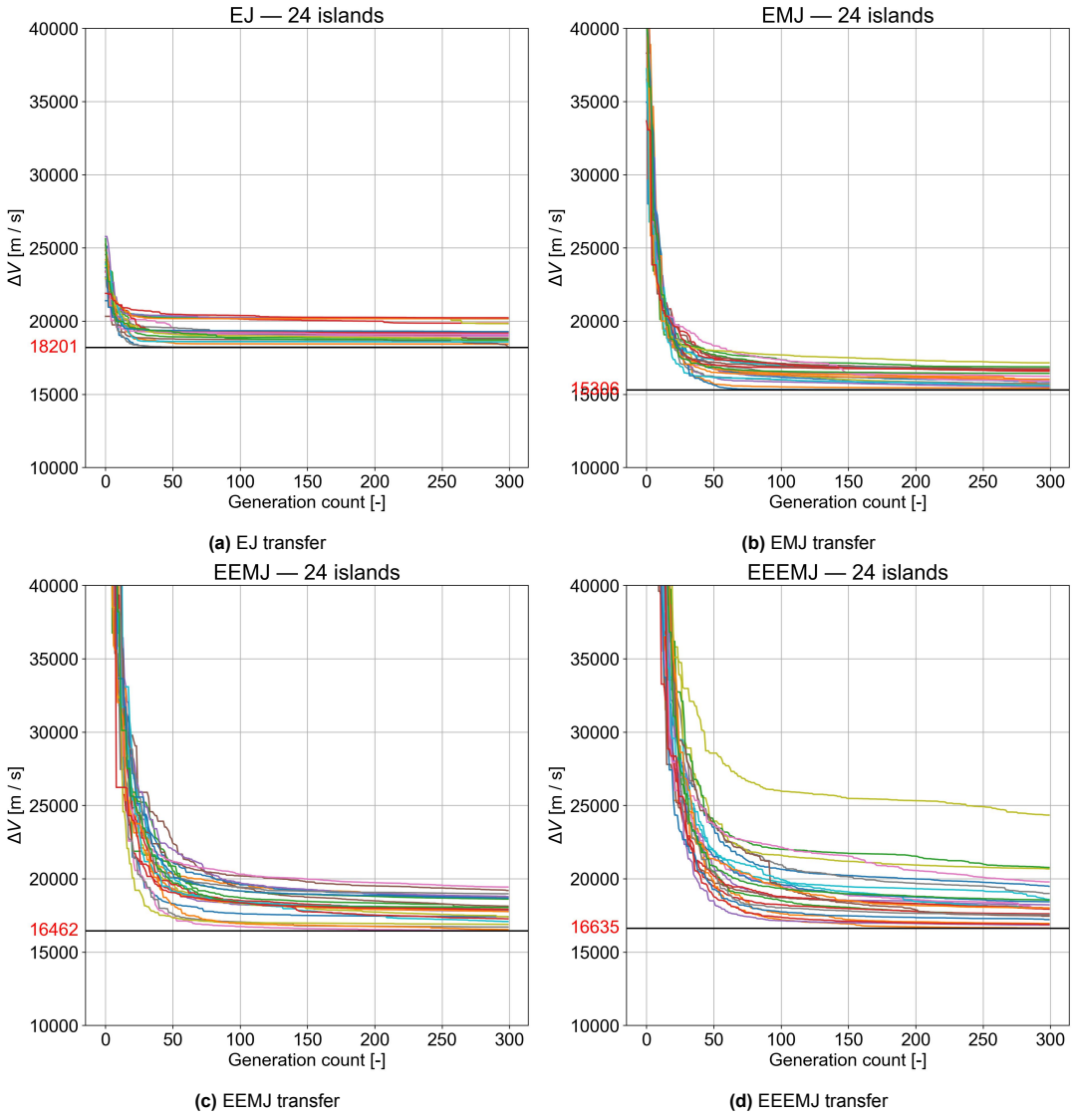
(c) 2 free parameters

Figure 6.13: ΔV per generation comparison between 0, 1, and 2 free parameters for the EEEMJ transfer.

A separate aspect is an observation that specific islands keep decreasing per generation – shown in all the sub-figures of Figure 6.13 by a consistent downward slope. An extreme case can be seen in Figure 6.13b, where the ΔV value of one island decreases abruptly by almost 5 km/s after 250

generations. This is expected to be an unlikely case. The mean and standard deviation of the one free parameter case are higher, however, it is expected that this disadvantage can be corrected using a topology, presented in Section 2.3.3 and tuned and implemented in Section 6.3.5. The advantage of the one free parameter case is the significant increase in computational performance, which would allow for robustness sourced from other input parameters with the constraint of a 24-hour run time using DelftBlue – the constraints of DelftBlue are given in Section 2.2.1.

The EEEMJ trajectory is only one sequence, and the robustness of results can differ significantly as the design complexity scales exponentially with added legs. Therefore, all four sequences found in [Fan et al. 2021] are compared with two free parameters and are shown in Figure 6.14.



(a) EJ transfer

(b) EMJ transfer

(c) EEMJ transfer

(d) EEEMJ transfer

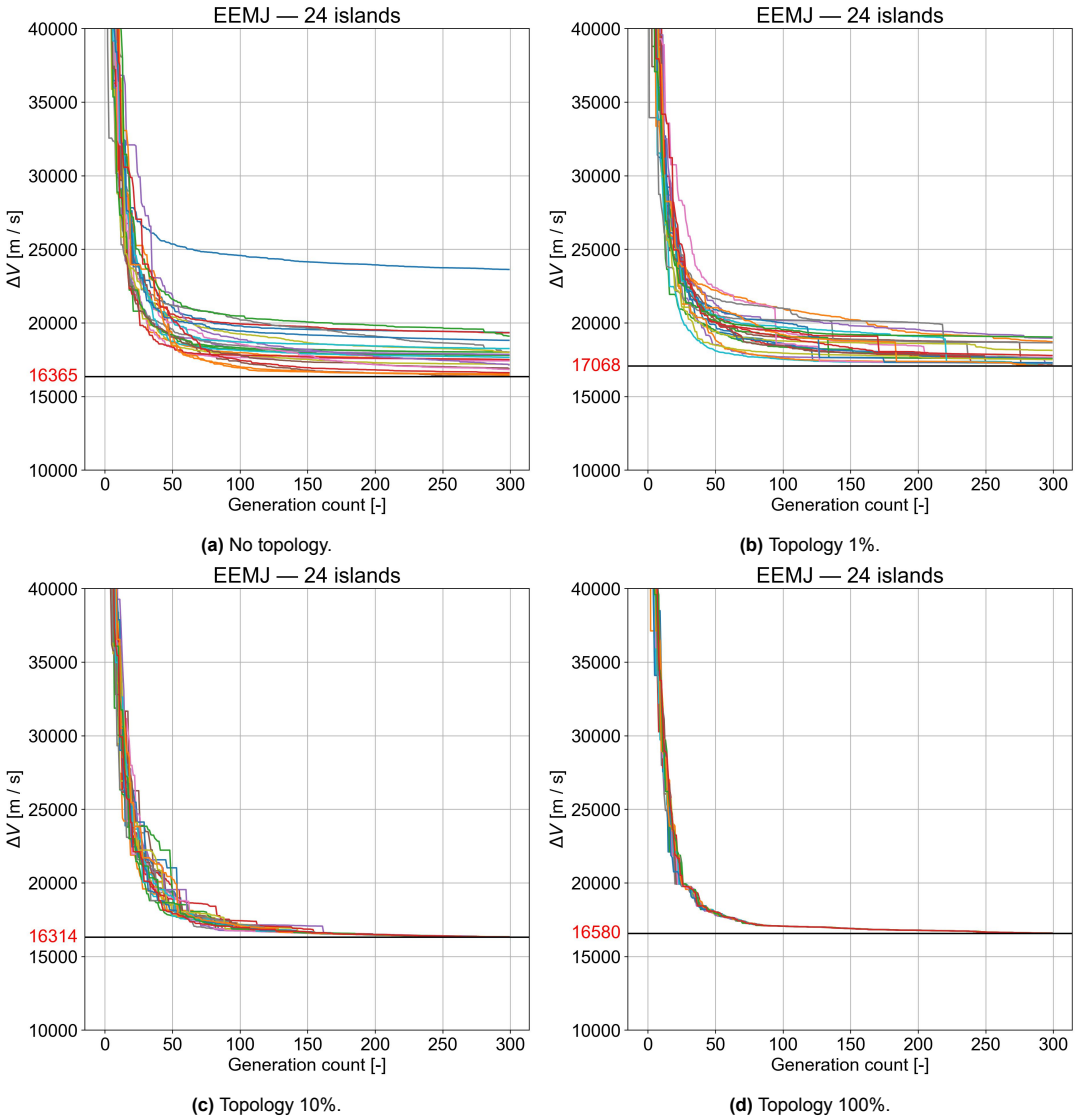
Figure 6.14: ΔV per generation comparison of each transfer with 2 free parameters.

The EJ direct transfer in Figure 6.14a converges within 50 generations and sub-optimal islands show some improvement. The convergence then steadily decreases with the length of the sequence, with the EEEMJ sequence converging the least. So, as it was previously shown that the one free parameter case would be sufficient for up to three GAs, the cases with fewer GAs are also going to suffice with only one free parameter. The one and two free parameter cases are both valid, and the two free parameter case is chosen unless the run-time constraint is met. This conclusion is only valid if a topology is implemented to reduce the mean and standard deviation. The next subsection discusses precisely that,

the introduction and tuning of the topology for a single sequence across multiple islands to reduce the mean and standard deviation.

6.3.5. Investigation of topology

The results in the tuning process so far have shown a relatively large spread in the final results of the islands. This is because the stochastic nature of the genetic algorithm leads to islands converging into different (local) optima. The spread is in the order of multiple km/s, where an optimal outlier could define the quality of the sequence if only the minimum is used. Therefore, the results are not expected to be consistent, and by extension also not robust. To remedy this, an additional 'operator' can be used for a single LTTO: the topology. The topology was shortly touched on previously, and to refresh its function: a topology is a network of connections between the various islands of an archipelago enabling the migration of individuals between islands according to certain migration laws. This allows for the spread – read standard deviation and mean – to decrease systematically.



(a) No topology.

(b) Topology 1%.

(c) Topology 10%.

(d) Topology 100%.

Figure 6.15: ΔV per generation for various topology probabilities.

The topology can be defined in various ways, and depending on the type several parameters can be specified to fully define the nature of the migration between islands. All islands that are run for the LTTO are identical – this changes for the MGASO when multiple islands are dedicated to various sequences. A custom topology may be necessary, but this is discussed in Part III. The fact that all islands evaluate

the same sequence in this tuning process means that the migration can be undirected, as there is no straightforward way to know in which direction to migrate individuals. As a result, the 'pg.fully_connected' class is chosen, in which all islands are connected to all other islands in the archipelago. This type of topology allows for the definition of two parameters: the desired number of vertices and the weight of all the edges. The number of vertices, however, is meaningless as a 'fully_connected' topology is used. The weight of the edges is crucial here as this determines the probability that a champion from one island will migrate to another island. A selection is made of various weights that cover the entire design space because a small difference in topology is expected to only have a nuanced and unobservable difference in the final result and to have a broad perspective on the effects. The different weights are 0.01, 0.1, and 1, and they are run for an EEMJ transfer. Do note that this is not the EEEMJ case, but the choice between a three-leg and four-leg transfer, in this case, is not destructive because Figure 6.15 compares the relative performance of various probabilities, assuming that the solution with no topology has outliers, which has been shown to be true.

It can be seen that Figure 6.15a is analogous to previous benchmark runs that have a spread of a few km/s. There are no sudden drops in fitness values due to the migration between islands. The other figures – seen in Figures 6.15b, 6.15c and 6.15d do show these sudden drops. It is visually and numerically confirmed that the islands between two sequential generations both obtain the identical lowest minimum ΔV. The migrations, as expected, happen at random iterations. The frequency of these drops increases with higher migration probabilities, which is also plausible. Regarding the final result, the progression through higher probabilities of migration improves the results in terms of the spread of the various islands: the spread decreases and the islands converge partially or fully to identical optima. The minimum value does not decrease linearly with an increase in migration probability. No topology, together with the 10% migration rate run found the lowest ΔV. This trend is unexpected, though it can be explained. The migration can be an advantage when one island is stuck in a high ΔV local optimum; the island can receive a migrated champion individual and start searching in another part of the design space. However, the migration can also be a disadvantage if an island prematurely receives individuals, thereby preventing that island from exploring with its own random selection of individuals. Knowing this, one would expect an initial decrease in minimum ΔV with an increase in topology migration probability followed by an increase in minimum ΔV. The reason for the difference between the expectation and the results in Figure 6.15 is that the seed may have a large effect on the outcome. Further investigation is recommended but not feasible due to time constraints. In terms of the final minimum ΔV, the 10% probability topology is best. Although these results do not provide a definitive answer to what probability is best, it is not necessary for this thesis. The original purpose was to filter out the outliers and increase efficiency, not to remove any difference between the mean and the minimum. The information in the 'mean' quantity is lost if the migration probability is too high, as it provides useful insights into the optimality of a large group of local optima. Separately, in Figure 6.15a, it can be observed that within the first 100 generations, there are various islands that overtake other islands in minimal ΔV. Consequently, the topology mechanism could be turned on after a certain number of generations to prevent this premature jumping, which is discussed later as a further recommendation.

It can be concluded that the topology works properly, and can be useful for influencing the convergence of high ΔV islands. For tuning purposes, a 10% probability is henceforth used, however, for the results, this probability may be changed. The next section discusses the tuning of the optimisation algorithm parameters in continuation of the population size and generation count.

6.3.6. Optimisation algorithm parameters

This subsection presents the tuning results of the parameters defining the optimisation algorithm. Specifically, the parameters besides the population size and generation count in Section 6.3.1 are tuned. These consist of mutation probability, mutation type, and cross-over probability. The tuning consists of two phases: first a general grid search of all parameters and their individual effects were performed – shown in Table 6.12 – and second a more focused tuning based on the previous findings was conducted – shown in Table 6.13. The findings for the second iteration are presented in this section in Figures 6.16 and C.10. The first phase can be found in Appendix C.

To summarise, the results from Phase 1 consisted of varying the mutation probability, mutation type, and cross-over probability. It was concluded that the mutation type and probability had to be further

Table 6.12: Phase 1 grid search setup for optimisation parameters.

Experiment name	mutation probability	mutation type	crossover probability
Benchmark	0.02	polynomial	0.9
m0.08_mut-poly_cr0.9	0.08	polynomial	0.9
m0.16_mut-poly_cr0.9	0.16	polynomial	0.9
m0.02_mut-gaus_cr0.9	0.02	Gaussian	0.9
m0.02_mut-uni_cr0.9	0.02	uniform	0.9
m0.02_mut-poly_cr0.5	0.02	polynomial	0.5
m0.02_mut-poly_cr0.2	0.02	polynomial	0.2

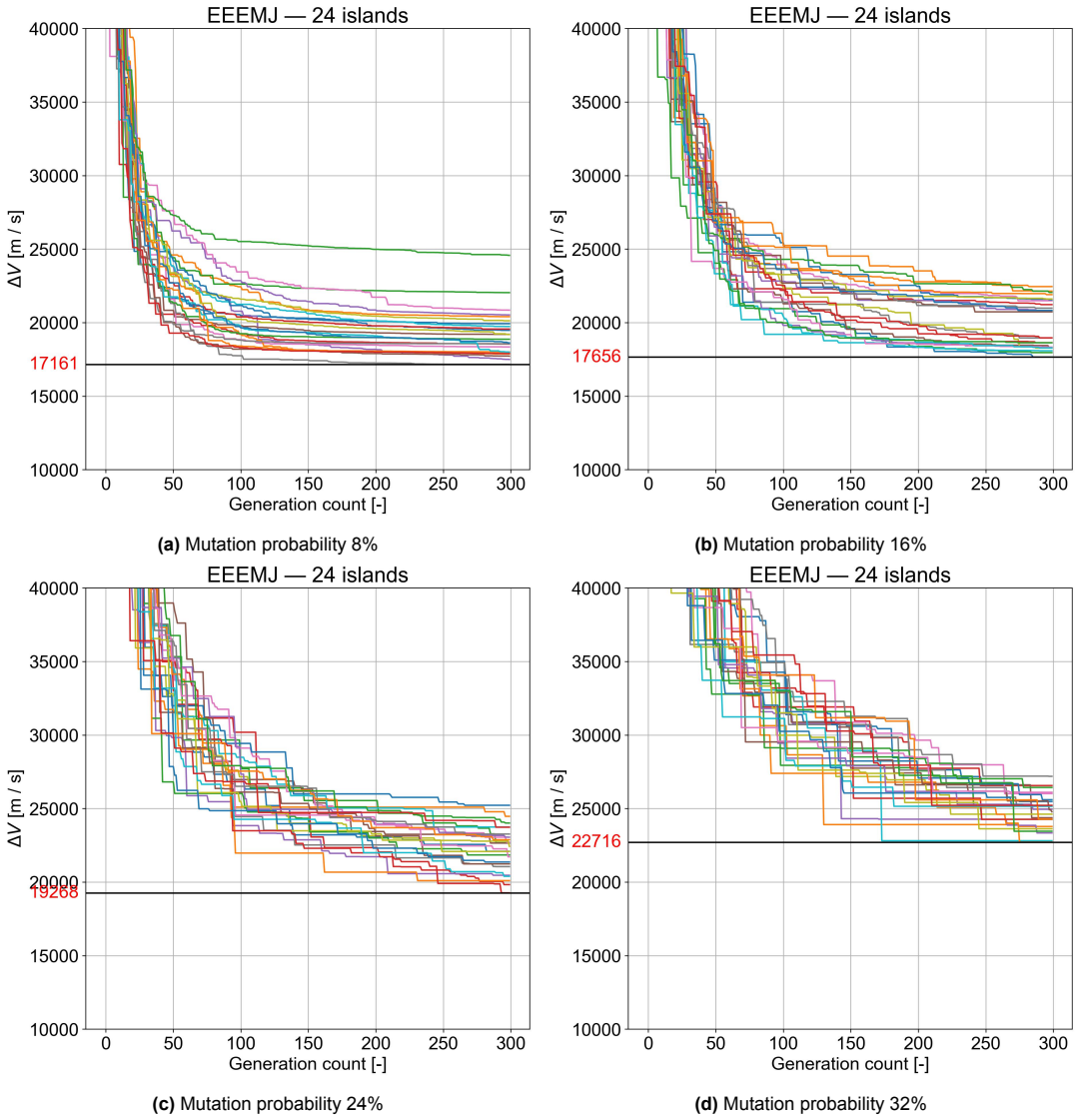
investigated, and in particular their inter-dependence. An increase in mutation probability showed an improvement in all relevant statistics, therefore even higher values need to be tested. The Gaussian mutation type shows a lower standard deviation, but otherwise slightly worse results in terms of minimum and mean ΔV, whereas the uniform mutation type shows a large drop in minimum ΔV. This decrease is found by an outlier, and with multiple seeds this drop was less pronounced, however, with the inclusion of the topology this outlier is useful. A small decrease in the cross-over probability showed no significant difference, whereas a large decrease from 0.9 to 0.2 led to higher minimum, mean, and standard deviation values for ΔV. It was therefore concluded that the default cross-over probability of 0.9 was best. In phase 2, a small grid is created for testing both Gaussian and uniform mutation for four different mutation probabilities, shown in Table 6.13.

Table 6.13: Phase 2 grid search setup for optimisation parameters.

Experiment name	mutation probability	mutation type
phase2_m0.08_mut-gaus	0.08	Gaussian
phase2_m0.16_mut-gaus	0.16	Gaussian
phase2_m0.24_mut-gaus	0.24	Gaussian
phase2_m0.32_mut-gaus	0.32	Gaussian
phase2_m0.08_mut-uni	0.08	uniform
phase2_m0.16_mut-uni	0.16	uniform
phase2_m0.24_mut-uni	0.24	uniform
phase2_m0.32_mut-uni	0.32	uniform

In Figure 6.16, the correct functioning of the parameters can be verified because the number of observed mutations increases as the mutation probability increases. However, the minimum ΔV increases, which indicates a decrease in optimality with mutation probabilities higher than 8%. This trend can be observed for both the Gaussian and the uniform mutation type. This result, combined with the increased performance when increasing the mutation probabillity from 0 to 8%, leads to the conclusion that for the EEEMJ sequence in an Earth-Jupiter transfer, an 8% mutation probability is best. The mean and standard deviation of the runs for both mutation types barely decrease with an increase in mutation probability. In addition, Figure 6.16a has one or two outliers, which is less pronounced in the uniform runs. The outliers, as was seen in the previous subsection, can be filtered out with the topology that is added to the final configuration. Between the Gaussian and uniform mutation type, the Gaussian result has a slightly lower minimum and the mean value is also lower excluding the islands that would be filtered out. Therefore, the Gaussian distribution shows a slight performance increase and will therefore be used for the remainder of the simulations in this thesis. As a short side note, multiple seeds were tested as well, and this provided no significant differences that would impact the nature of or conclusions drawn from the results.

One interesting observation is that Figure 6.16b shows two groups of optima with a gap in ΔV. This can either be explained by coincidence, or a specific design variable in a certain range was found by roughly half of the islands. It is expected to be coincidental, as this behaviour is only observed for a



(a) Mutation probability 8%

(b) Mutation probability 16%

(c) Mutation probability 24%

(d) Mutation probability 32%

Figure 6.16: ΔV per generation for Gaussian mutation with wider mutation ranges.

specific mutation probaility and is not observed in any other runs.

6.3.7. Local optimisation

This subsection discusses the addition of a local optimisation procedure at the end of the SGA to force the population to a local optimum. This is done in a computationally cheap and efficient way by using a deterministic approach, rather than having a stochastic optimisation process endlessly scour the design space for any improvement.

With the large selection of results in the previous subsections, it can be said that the simulations after 300 generations have generally converged to some local optimum. This is because of the functioning of the GA, where most of the cross-over operations and elitism will at some point remain unchanged. The only exploration that still happens is due to mutation, and while this quantity was increased from the default in Section 6.3.6, it does not result in significant improvements in later stages of the optimisation. This convergence, however, is no guarantee that there is not a lower ΔV solution in the vicinity, which leads to the need for local deterministic optimisation.

Generally, local optimisation methods include methods that do not rely on probability to search the design space and determine the individuals of the next generation. Rather, gradient-based methods are often used. These methods use the gradient information to force convergence to a local optimum.

For the local optimisation, the Nelder-Mead Simplex solver is used. This solver is recommended by [Gondelach and Noomen 2015] for hodographic-shaping legs – although the design variable vector in this thesis includes substantially more variables. The solver is capable of solving a mixed-integer problem, which is an essential property. The algorithm regards the mixed-integer variables as discrete, and it can take the gradient of the discrete variables. Categorical variables are not a valid input for the integer optimisation in PyGMO, and they have to be translated, using null genes, for example, which was discussed in Section 2.1.2. The Nelder-Mead method is a numerical method that defines a simplex – a polytope consisting of n+1 vertices in an n-dimensional space. The objective function is evaluated at each vertex and the result is extrapolated to form a prediction for the next simplex. To check if the algorithm improves the results, the local optimisation is applied to the departure date grid search from Section 6.3.3. The results of this local optimisation are shown in Figure 6.17 as an addition to Figure 6.3a.

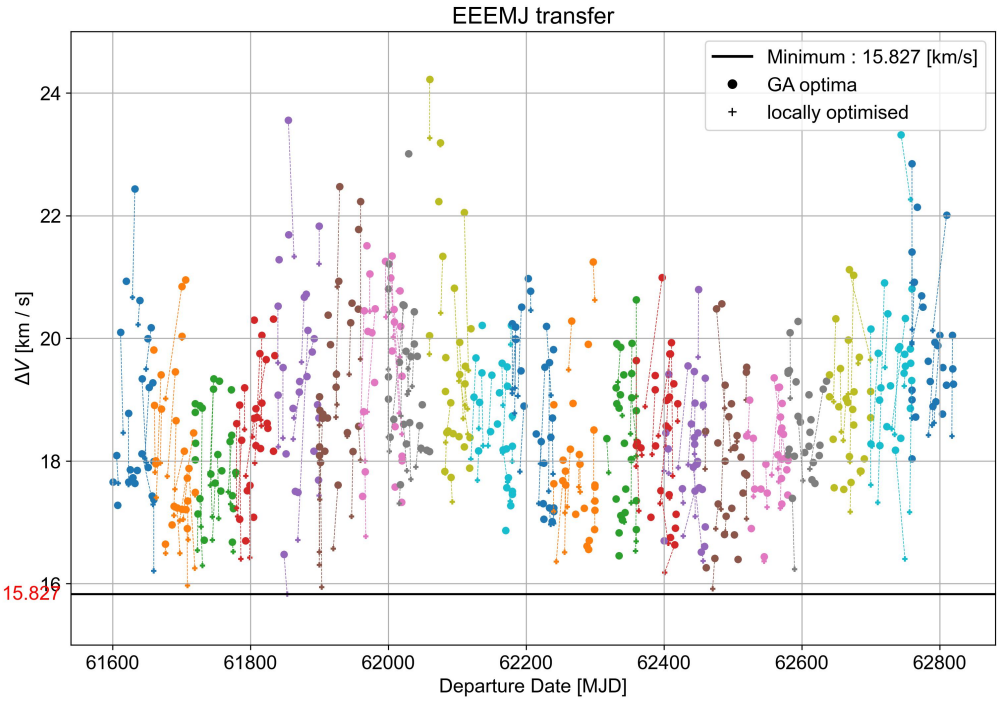


Figure 6.17: Grid search from 61400-62800 MJD with additional local optimisation step.

For a significant portion of the optima found per island per 60-day interval, a decrease in ΔV is realised across all relevant statistics that have been used throughout the tuning process – mainly the minimum and mean. This decrease is not present for all islands, which is to be expected as the GA in that case has indeed completely converged to the local optimum. There is no difference in the performance of the local optimisation across the departure date window, which is to be expected as the fidelity at each departure date is the same. There is also no observable trend found in the improvement of the local optimisation across the whole ΔV range. One could expect this trend, as lower optima are relatively closer to the global optimum, and therefore have less room to improve. The absence of this trend indicates that more improvement is possible or that the complexity of the optimisation problem is too high for the current implementation to optimise. A separate visualisation is shown in Figure C.14.

In general, the global minimum ΔV value has decreased from 16.261 km/s to 15.827 km/s. This decrease is substantial, especially when putting the result into perspective with the desired results from Table 6.8. The FFS, for example, finds 15.92 km/s which is comparable to the result found by the LTTO in this thesis. The Bezier shape applied to the EEEMJ transfer in [Fan et al. 2021] still performs considerably better with 14.39 km/s. The difference in ΔV that remains does not make the results of this thesis invalid: the crucial part is the robustness of the results by themselves, and relative to other runs within the same problem definition. If the results are consistent across multiple runs, and the same difference in quality can be observed between various sequences, then the LTTO can be

used for the MGASO. Ultimately, there are only a few sequences that will have comparable ΔV values, and these should be distinguishable from the other sub-optimal sequences. In addition, excluding the local optimisation simplifies the optimisation process, and decreases the computational complexity slightly as well: the Nelder-Mead Simplex typically converges for this optimisation problem after 23 iterations which amount to approximately 1100 function evaluations per island – assuming 48 design variables from the EEEEMJ transfer used throughout this chapter. This is only a 0.3% increase in function evaluations compared to the $3.6 \cdot 10^5$ function evaluations that an LTTO optimisation process performs assuming 300 generations and a population size of 1200. However, this increase may change depending on the problem formulation. In conclusion, the local optimisation is not expected to be crucial for obtaining robust results, so for the results it is kept as a backup in case the ΔV values seem to not have converged at all.

A challenge that remains after this tuning process is that the statistics of the four sequences investigated by [Fan et al. 2021] do not follow the same trend as the results found by the FFS and the Bezier shapes. Where both other shaping methods see a consistent improvement when adding a GA as defined in [Fan et al. 2021], the LTTO process defined here finds that EMJ, EEMJ, and EEEMJ all have similar ΔV values. While this may be a more accurate result than the FFS and Bezier shape methods, it is also possible that the optimality is harder to recreate with longer sequences. To separately verify that a longer sequence can indeed provide lower ΔV values, the Earth-Neptune case is considered, which is discussed next in Section 6.4.

6.4. Earth-Neptune transfer

To show for a separate, unverified example that the ΔV values do not necessarily increase with an increased MGA sequence length, an Earth-Neptune transfer is considered. Subsequently, various GA targets are added to the direct transfer and their relative performance is analysed. The problem definition, as shown in Table 6.3, is inspired by [Novak and Vasile 2011]: an Earth-Neptune rendezvous mission is recreated within specified departure date bounds. For reference, the results for the EN transfer found by [Novak and Vasile 2011] are shown in Figure 6.18.

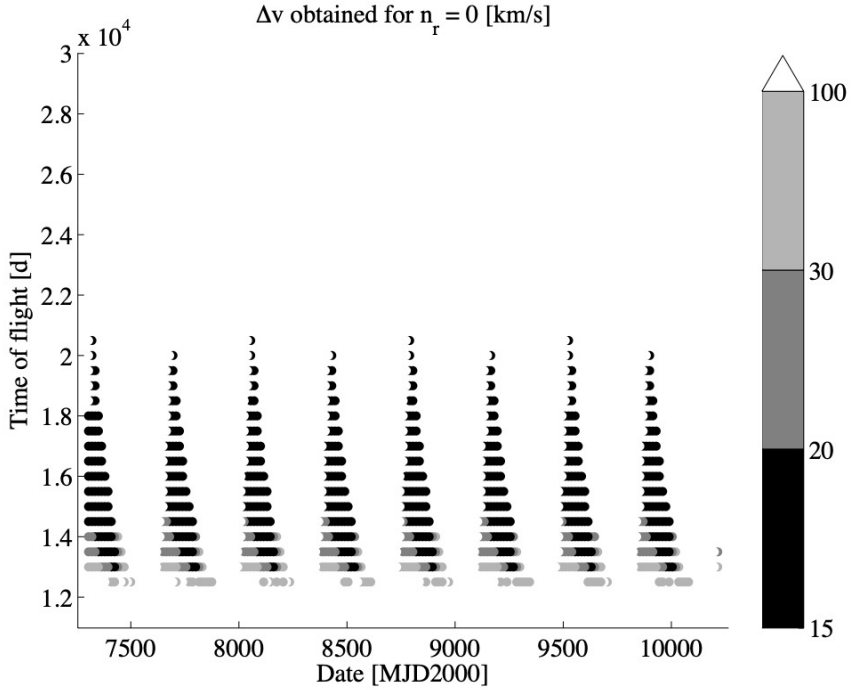


Figure 6.18: ΔV values as a function of ToF and departure date [Novak and Vasile 2011].

Gradually, similar to [Fan et al. 2021], the sequence is increased in length to test if additional GA

manoeuvres decrease the ΔV consistently. In this section, no tuning is done due to time constraints. Consequently, the results are not an accurate representation of the full capabilities of hodographic shaping, rather they are an indication of the relative quality of results. A grid search is conducted with two departure date windows of 1200 days to cover the entire region shown in Figure 6.18. The topology mechanism is not used to check the spread of the results without artificially changing them. A local optimisation is performed for the sake of verification. The results of all of these sequences are summarised in Table 6.14. The grid search can be found in Appendix C.5.

Table 6.14: Statistics on various Earth Neptune transfer sequences for two departure date windows.

Departure date interval [MJD2000]	Earth-Neptune sequences [-]	Min [km/s]	Max [km/s]	Mean [km/s]	Std [km/s]
7800 - 9000	EN	73.711	73.846	73.752	0.038
	EJN	25.342	27.243	26.531	0.433
	EMJN	22.541	32.395	27.556	2.177
	EEMJN	23.878	29.992	25.625	1.560
	EEEMJN	25.128	37.434	28.649	2.807
9000 - 10200	EN	73.614	73.696	73.683	0.017
	EJN	26.564	35.161	28.228	2.074
	EMJN	24.665	34.672	27.222	2.736
	EEMJN	23.786	32.528	27.254	2.119
	EEEMJN	23.430	39.124	26.426	2.965

Before the quantitative analysis, it should be mentioned that in the grid search figures in Appendix C.5, the same periodic optimality based on the synodic period is found, which shows that the theoretical expectations of periodic optimality are consistent.

In Table 6.14, it can be seen that the EN direct transfer converges to about 73.7 km/s – with a standard deviation of only 38 m/s. This value is immensely high compared to the results found by the spherical shaping method in [Novak and Vasile 2011]. It is expected that this is partially due to the shaping functions that are used for this thesis which are based on an Earth-Mars transfer. The Earth-Neptune case has considerably different ToF and velocity profile characteristics due to the difference in scale. Besides the shaping functions, the other parameters and bounds are also untuned, which can result in considerable differences, as is also seen in Section 6.3.1. The ToF of the EN transfer all converged to 5000 days, which is the minimum bound for ToF. Tuning would significantly reduce the ΔV, but it is not necessary for the goal of this section. To reiterate, this section aims to see the effect of adding GAs to a different, comparable transfer problem. When adding a Jupiter GA, the minimum and mean ΔV decreases by 48 and 47 km/s for the two windows, respectively. This represents more than a 50% decrease in ΔV. For the 7800-9000 departure date interval, the addition of extra GAs beyond the EMJN transfer does not lead to an improvement in the ΔV value. For the 9000-10200 departure date interval the extra addition does lead to a further improvement. Thus, depending on the departure date bound used, the transfers either steadily improve when adding more GAs in both the minimum and the mean ΔV with each added GA, or flatten out as is the case for the Earth-Jupiter transfer in the previous section. It can be concluded that the addition of extra GAs beyond the limit of three GAs, as was the case in the previous section, can lead to an improvement in the results, despite the increase in design complexity and the accompanying difficulty with converging. This concludes the LTTO tuning process; the next section shortly summarises the findings from this chapter.

6.5. Conclusions

This chapter has contained an extensive process, so this section aims to summarise the outcome of the various tuning aspects. In particular, the insights, the final parameters, and recommendations are summarised, as well as the aspects that remain to be investigated in future work.

As was stated in Section 6.1.2, the untuned results were not robust or optimal. A single addition of a GA dramatically increased the ΔV found. Subsequently, three papers were used as a basis for the tuning process: [Morante et al. 2019; Fan et al. 2021; Novak and Vasile 2011]. [Fan et al. 2021] was found to be the most applicable and therefore represented the majority of the tuning steps.

From the first paper [Morante et al. 2019], it was found that the presence of coasting in combination with the multi-objective nature of the problem did not lead to results that were useful: the propellant mass fraction – being the objective in [Morante et al. 2019] – increases significantly with additional GAs. Therefore, this paper was deemed not adequate for tuning the implementation in this thesis. Almost all the tuning was conducted using the problem definition from [Fan et al. 2021]. In particular, the population count and generation count, the bounds of all design variable types, and the free coefficient count were investigated. Furthermore, the addition of a topology was tested, and finally, a local optimisation was performed. The final configuration of all these aspects is summarised in Tables 6.15 and 6.16. These values are used in the MGASO tuning and results in Parts III and IV. A key observation from tuning the optimisation problem as defined in [Fan et al. 2021] was the apparent increase in ΔV with a larger MGA sequence. To test this, a final reference problem – from [Novak and Vasile 2011] – was used. The goal was to double-check whether adding GAs was the cause of the increased ΔV values. It was found that adding GAs does not have to result in an increase in ΔV caused by an increase in design complexity. Rather it depends significantly on the departure date. This is a positive discovery as it indicates that the design complexity is not the constraint in terms of finding highly optimal values. Below the general parameters are presented that can be used for any reference problem as well as the Earth-Jupiter transfer-specific parameters that are used henceforth as a test case. Do note that these parameters are not optimal, but they have been shown to provide results that are verified – discussed in Chapter 9 – and robust.

Table 6.15: Tuned problem bounds for MGASO.

Parameter/Feature	Value
ps	1200
gc	300
Configuration	IV
fpc	2
Topology probability	0.01

Table 6.16: Tuned bounds for Earth-Jupiter transfer.

Design Variable	Bound/Value	Unit
Departure Date	[61842, 61902]	MJD
V_{dep}	0	m/s
V_{arr}	0	m/s
Incoming velocity	[0, 5000]	m/s
GA altitude	$[2 \cdot 10^5, 5 \cdot 10^{10}]$	m
β	$[0, 2\pi]$	rad
θ_g	$[0, 2\pi]$	rad
ϕ_g	$[-\frac{\pi}{2}, \frac{\pi}{2}]$	rad
Free coefficients	$[-3 \cdot 10^4, 3 \cdot 10^4]$	-
Number of revolutions	[0, 2]	-

To concretely summarise how the main tuning process improved the quality of the results, the difference in ΔV is tabulated from before and after the tuning process in Table 6.17. The values from before tuning stem from Table 6.4 and the values from after tuning are shown in Section 9.1.3.

Table 6.17: Initial comparison of minimum ΔV for the EJ and EMJ transfer from [Fan et al. 2021].

	Transfer	
	EJ [km/s]	EMJ [km/s]
Before tuning	17.78	33.03
After tuning	17.48	15.30

It is immediately evident that the MGA transfer has improved significantly. The EJ transfer only improves slightly, which is also expected as there is less freedom due to the substantially smaller design space. The EMJ transfer has improved by more than 100% and has converged to ΔV values that are competitive with literature – discussed explicitly in Section 9.1.3. In general, the LTTO is found to be capable of reproducing ΔV values from a reference problem to a high degree of accuracy in a robust and consistent way. Beyond the findings of the tuning process, numerous recommendations remain that would further improve the robustness of the LTTO as it is defined and implemented in this thesis. In short, the bounds should be further investigated as for most tuning aspects only one iteration was possible; the tuning was not performed to optimise the accuracy as well as the computational efficiency. For example, the size of the population or the number of generations could be decreased to reduce run time, while still conserving enough robustness to have a reliable MGASO. Furthermore, the potential expansion to applications that include coasting arcs [Morante et al. 2019; Moreno Gonzalez 2020] or multi-objective optimisation. The multi-objective optimisation capabilities were implemented and can be accessed via GitHub[1], however, there was no such reference paper that fit the problem well. Last but not least, the addition of dynamic bounds should be investigated to further generalise the inputs to the LTTO. These aspects are discussed more elaborately in Chapter 12. This concludes the LTTO part of this thesis, the next part addresses the MGASO, and uses the LTTO to perform an optimisation that can provide optimal MGA sequences.

[1] https://github.com/sbcowan/ThesisCode

Part III

MGA Sequence Optimisation

7

MGASO setup

This part consists of a development process similar to the LTTO, however, it considers the outer loop of the optimisation, which will use the LTTO as defined in the previous part as one aspect of the overall optimisation of MGA low-thrust sequences. In this chapter, the MGASO is set up. Specifically, the RTBA is presented, after which the algorithm is defined concretely. The RTBA is a novel approach developed in this thesis.

7.1. RTBA introduction

This section discusses the RTBA that is developed to optimise the MGA sequence of any given low-thrust interplanetary mission in a robust way with competitive run times.

7.1.1. Tree-search problem statement

Achieving run times that are competitive with other research is a core challenge: in all the papers concerning tree-search methods discussed in Section 2.1 that consider low-thrust propulsion specifically, there is still a high computational load that is required, and consequently, long run times are observed. In some cases, the run time is reduced, yet no paper has verified the optimality of its findings using tree-search methods for the MGASO. The main contenders for performant tree-search methods that solve the MGA sequencing problem consider almost exclusively high-thrust cases: [Hennes and Izzo 2015] and [Ellison 2018]. To shortly recap, the former implemented the BSS which is a computationally effective tool with the risk of a lack of robustness. The latter implements MCTS, which includes multiple iterations and therefore has a more thorough search with robust results. Each paper quantifies its performance differently, making a comparison of their computational performance and the robustness of their results unreliable. This raises the question of how effective a tree-search method would be on low-thrust MGA sequencing applications. It is expected that the advantages of both the BSS and the MCTS can be combined to produce data that is robust enough to ensure the correct pruning of possibilities while remaining greedy in nature. The robustness of the data after one iteration is determined by checking whether the same result is found for various seeds (various selections of sequences that are evaluated). In future work, this should also be verified with other MGASO algorithms.

With a systematic approach such as a tree-search method, combined with the relatively novel and promising hodographic shaping method, it is expected that a robust ranking of optimal low-thrust MGA sequences can be produced that requires significantly less computational time.

7.1.2. Tree formulation

In the context of sequence optimisation, a combinatorial space can be defined. All possible MGA sequences are part of this space. This thesis also defines the combinatorial space as a tree and is solved using a tree-search method. Before diving into the specifics of the approach, a reminder that the basics and context of tree-search methods (and graph theory) have been discussed in Section 2.1.4.

The tree is chosen to be directed, acyclic, and unweighted. Firstly, the tree is directed so that the

complete combinatorial enumeration is visualised and each unique sequence can be defined by an edge at each level in the tree. Secondly, the tree is acyclic, because constructing a cyclic tree for MGA sequencing would mean that an EMEMJ transfer, for example, would cycle back and therefore the ΔV cost at each EM leg would be identical. Rather than having a conditional tree, where edges have certain values depending on the number of traversals or length of the roll-out, one just adds extra edges and vertices. The disadvantage of this approach is that the tree has more legs and nodes, however, it does not affect the actual combinatorial complexity of the problem to be optimised. Finally, the tree is unweighted, because the ΔV value of the legs and nodes depends on the previous and subsequent legs. Each sequence is unique and therefore the values at each specific node are not constant, which makes weighting the graph non-sensical. Each roll-out of the tree is assigned a value, namely the ΔV of that sequence. An example combinatorial tree is given in Figure 7.1 for an EJ transfer with zero, one, or two possible GAs.

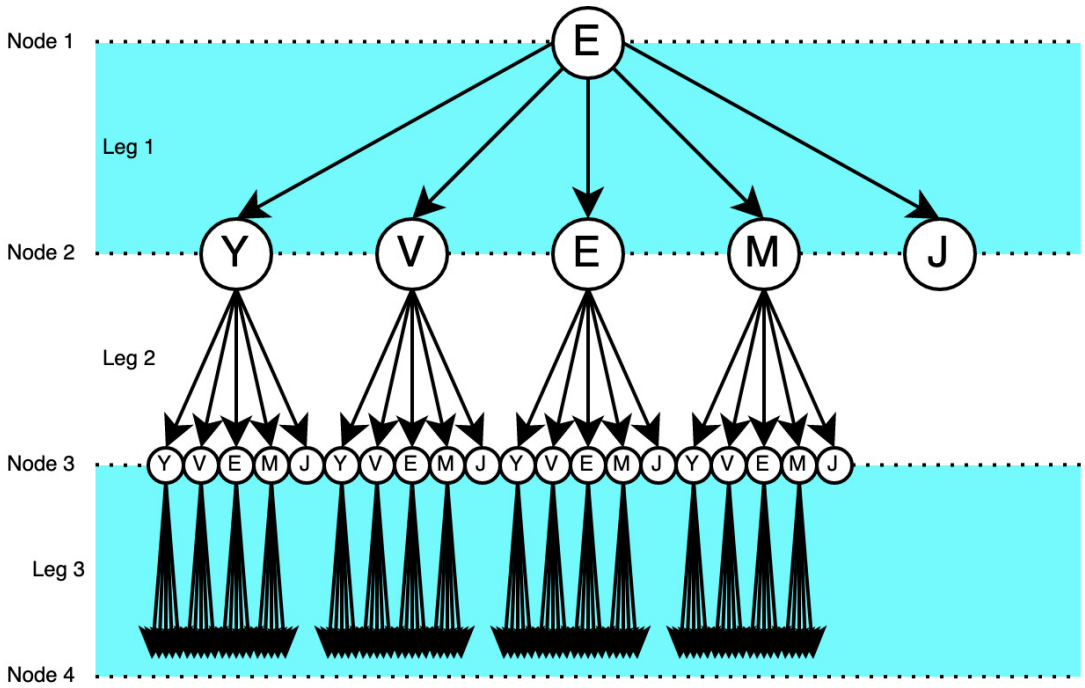


Figure 7.1: A three-level tree for an Earth-Jupiter transfer.

This depiction is useful for understanding the structure behind the tree-search method from a top-level view. The steps in the methodology are presented next.

7.1.3. Top-level approach

As previously alluded to, the RTBA is a greedy MCTS derivative applied to low-thrust MGA sequencing. Low-thrust is crucial, as the sensitivity, robustness, and performance of various tree-search methods are very much dependent on their application. The representation of low-thrust trajectories involves significantly more parameters than any high-thrust equivalent, resulting in the complex optimisation problem considered in this thesis.

The RTBA consists of a few steps in every recursion:

1. Monte Carlo search
2. LTTO optimisation
3. Recursive Pseudo Sequence (RPS)
4. Repeat step 2

General characteristics
First, some general comments are given after which the aforementioned steps are explained one by one. The RTBA is recursive in that it performs the same combination of steps over and over again until several criteria have been met. The tree that is used becomes smaller with every recursion. Each branch of the tree can be seen as a new tree that the approach can be applied to. The RTBA is greedy in that the branches that have not been evaluated after one recursion are pruned for the rest of the MGASO. The RTBA is a derivative of MCTS because sequences that are evaluated in each recursion are chosen at random, within the confines of the problem formulation. To clearly explain this step, the RPS and Pre-defined Target Body (PTB) terms are defined. The RPS is the portion of the MGA sequence that is fixed by the various recursions. The RPS is initially empty, and after each recursion a body is appended to this sub-sequence. The PTB is the list of possible GA candidates that are pre-determined to be traversed at each level. This process is defined by either the number of sequences per possible GA candidate or the fraction of the total combinatorial space seen from that branch.

MCTS search
The first step in every recursion is the Monte-Carlo search step. A number of sequences are picked at random that will be evaluated in every recursion. Each sequence has multiple islands that are conducting the same optimisation to ensure robust results as shown in Section 6.3.1. The number of islands per sequence is determined based on the objective of the simulation and the DelftBlue constraints. The sequences are not fully random: at each level, to ensure that each PTB has the same level of robustness, the number of sequences is evenly distributed over all PTBs. This constraint is only necessary for smaller combinatorial problems, as the nature of Monte Carlo is that it converges to a certain distribution with sufficient evaluations. The quantity that defines the extent of the Monte-Carlo search step is the fraction of the combinatorial space at each recursion level, denoted by q, which is defined in the next section. The fraction of the combinatorial space translates into a number of sequences per PTB. These sequences are passed to the archipelago.

Single recursion archipelago
This step takes all the sequences that need to be evaluated and performs an LTTO on them. The LTTO has been tuned extensively in Part II. The islands for every sequence are contained in one LTTO consisting of a single 'pg.archipelago' object. This step outputs statistics and relevant quantities of each sequence, which is passed to the next step where the optimal PTB is determined. There are many ways to define the optimal PTB. Because the RTBA is greedy, it is crucial that this metric is robust during every recursion. As it was determined in Part II that the mean, minimum, and standard deviation by themselves are not expected to sufficiently characterise the fitness of any sequence, a novel approach is taken: a linear combination of the minimum and mean ΔV is used to allow for a more informed decision of the optimal PTB. The standard deviation does not directly correlate with the optimality of the ΔV of a given sequence and is therefore not considered. The skew and kurtosis of the ΔV values are also neglected for the same reason. The linear combination of minimum and mean is used on two different levels. For the first level, the minimum and mean of all the islands that have evaluated one sequence are summarised by a fitness value f_s, defined by a proportion denoted as fp, which defines the relative importance of the minimum versus the mean of the islands. These quantities are defined in Section 7.2.1. The second level relates to the RPS expansion which is discussed next.

RPS expansion
The concept of a linear combination is also applied to the fitness of the PTB candidates. All the evaluated sequences are grouped according to the PTB. The minimum and mean f_s values of the sequences of each PTB are used to form another fitness, called f_X, defined by the proportion fpi. This proportional quantity is also crucial for a robust decision-making process, as there may be anomalies: a specific sequence with a target GA may be highly optimal according to one island, however, the real optimum may lie in a different branch with more GAs. A clear example is EMJ, which is found to be highly fit according to the problem definition in Part II. The more optimal trajectories are EEMJ and EEEMJ according to [Fan et al. 2021], therefore the RTBA may wrongly decide that Mars is indeed the best PTB, and subsequently append that to the RPS. This example assumes that EEMJ and EEEMJ are indeed more optimal than EMJ. The f_X and fpi quantities are defined in Section 7.2.1. In short, the RPS is a pseudo-sequence that encapsulates the perceived optimum GA target at each 'slot' in the multiple gravity-assist trajectory.

The process explained above concludes one recursion. The same procedure is repeated with an updated RPS. Importantly, the RTBA uses recursion until there are no branches left. However, in the end, a ranking is given of all the sequences that were found, so the final optimal sequence is not guaranteed to have a sequence length equal to the number of allowed GAs. In the next section, the RTBA is fully described with an explanation of the actual functions at all stages of the approach.

7.2. Structure of the RTBA

This section gives a more concrete overview of the optimisation, by discussing the relevant parameters and variables, as well as the structure of the code and the parallelisation that is present.

7.2.1. Parameter definition

This subsection discusses what parameters are input to the RTBA and what quantities are used as metrics for the assessment of the MGASO. On top of the parameters that have been defined before in Sections 5.2 and 6.3.5, several new quantities are defined following their introduction in Section 7.1.3.

These parameters together with the new quantities are shown in Table 7.1. The previously defined parameters are tinted; these were all summarised in Table 5.2. The maximum number of GAs, departure planet, and arrival planet are problem dependent. The departure and arrival planets are fixed for a specific problem. The maximum number of GAs is needed to define the depth of the tree. In theory, each node has the same child nodes, this maximum introduces a termination condition. As will be seen in Section 8.2, this parameter is set to three. While that may seem small compared to the number of GAs – or flybys more specifically – that contemporary space missions achieve, this thesis only considers a single target for an interplanetary trajectory. A mission like JUICE will transfer to Jupiter before performing some 30 flybys within the Jovian system. Contemporary space missions perform a large number of flybys as there can be numerous scientific goals related to a flyby. For future work that would include more bodies, constraints such as a larger number of PTBs may be added.

Table 7.1: Relevant parameters for MGASO definition.

Parameter name	Parameter description	Unit
bounds	Collection of design variable bounds for LTTO	various
fpc	Number of free parameters	-
gc	Generation count	-
ps	Population size	-
tw	Probability of migration in the topology	-
mng	Maximum number of GAs	-
p_{dep}	Departure planet	-
p_{arr}	Arrival planet	-
ips	Islands evaluated per sequence	-
q_k	Fraction of combinatorial complexity per recursion	-
fp	Fitness proportion of each island	-
fpi	Fitness proportion of each PTB	-

The index k refers to the recursion count.

The ips parameter determines the number of islands per sequence. This value influences the robustness of each sequence, as a larger confidence is obtained in the minimum and mean. The downside is that more CPUs are needed to evaluate each island. This value was maximised to 24 in Chapter 6 for verification purposes. Time did not allow for the tuning of the number of islands necessary to ensure a robust result. 14 islands are used henceforth because of the maximum run-time constraint set by DelftBlue; this choice is explained further in Section 8.2.2. ips remains constant for all sequences evaluated, though this could be changed in future work.

Before defining the remainder of the parameters, a few other quantities should be defined for clarity. The combinatorial complexity of any tree is defined as the number of possible combinations, given by

Equation (7.1). This definition can be used at any recursion level for every possible tree and sub-tree.

$$C = \sum_{i=0}^{n} m^i \tag{7.1}$$

In Equation (7.1), C is the number of combinations, m is the number of possible GA candidates, and n is the maximum number of GAs remaining. For the 'Earth-Jupiter with coasting' problem from Section 6.1.1, three GAs are allowed, and there are eight possible planets, resulting in 585 unique sequences in the first recursion – so for the full tree. This quantity is rather large if one would have to calculate every single sequence with multiple islands, and this number can therefore be brought back drastically, which is discussed in Section 8.1.1. With the complexity defined, the proportion of the combinatorial space that is evaluated for each tree or sub-tree can be set, which brings us to q.

q is the fraction of combinations evaluated relative to the complexity C. q is defined for a single recursion. In this thesis, q is constant across the recursion levels; tuning the fraction levels per recursion is not feasible within the run-time constraints. Overall the recursions, a total fraction is defined called Q. In each recursion an additional 50% of the remaining combinatorial space is evaluated, resulting in a value of Q that is higher than q. In other words, if q is 0.5 for example, then the total fraction of sequences evaluated after multiple recursions will be larger than 0.5. The choice for q as a metric, as opposed to a fraction for the total combinatorial complexity evaluated, is because this definition results in an equivalent relative robustness level for each recursion. Choosing only a single total combinatorial fraction leaves the fraction at each recursion level undefined. When discussing results, the total complexity will be mentioned. Besides q, the last two parameters – fp and fpi – relate to the output quantities of the optimisation, which requires a few more definitions.

Given a sequence A, there is a corresponding discrete set of islands defined as $A = \{A_1, A_2, .., A_{ips}\}$. For every recursion, there exists also a continuous alphabetic set of sequences defined as $S = \{A, B, .., Z, AA, AB, .., AZ, BA, BB, .., \alpha\}$ – with α being any given alphabetic character sequence equal to the final sequence. The fitness of a sequence is then defined by Equation (7.2).

$$f_A = min\{\Delta V_{A_1}, \Delta V_{A_2}, .., \Delta V_{A_{ips}}\} \cdot fp + \frac{\Delta V_{A_1} + \Delta V_{A_2} + .. + \Delta V_{A_{ips}}}{|A|} \cdot (1 - fp) \tag{7.2}$$

Here, f is the fitness, $min()$ finds the minimum value in a set, and $|A|$ gives the length of a set of islands. The sequences in set S are sorted according to their PTB, giving sub-sets of each PTB. The subsets of an Earth-Jupiter transfer that can fly by Mercury, Venus, Earth, and Mars, is denoted by $S = \{S_Y, S_V, S_E, S_M\}$, where $S_Y = \{A, B, .., r\}$, for example. Here, Y, V, E, and M are the PTBs as defined in Section 6.1.1, and r is the final MGA sequence that flies by a given PTB; Mercury in this case. With the fitness of a sequence defined, the fitness of a PTB is then given by Equation (7.3).

$$f_X = min\{f_A, f_B, .., f_r\} \cdot fpi + \frac{f_A + f_B + .. + f_r}{|S_X|} \cdot (1 - fpi) \tag{7.3}$$

Here, X is any particular PTB. The lowest fitness from the available PTBs in each recursion is appended to the RPS. Contrary to the LTTO, the MGASO is not solved with a conventional optimisation algorithm, and therefore there are no design variables. The only variable is the sequence string. To better explain how all the steps and variables of the RTBA fit together, a simplified case is considered in Example 7.1.

Example: 7.1 – Earth-Jupiter transfer with two GAs

To exemplify the functioning of the RTBA, a simplified case is introduced using an Earth-Jupiter transfer with two GAs, where only Earth and Mars are possible GA candidates, one recursion is performed, and half of the combinatorial space is explored. The combinatorial space of this problem is visualised by the tree shown in Figure 7.2.

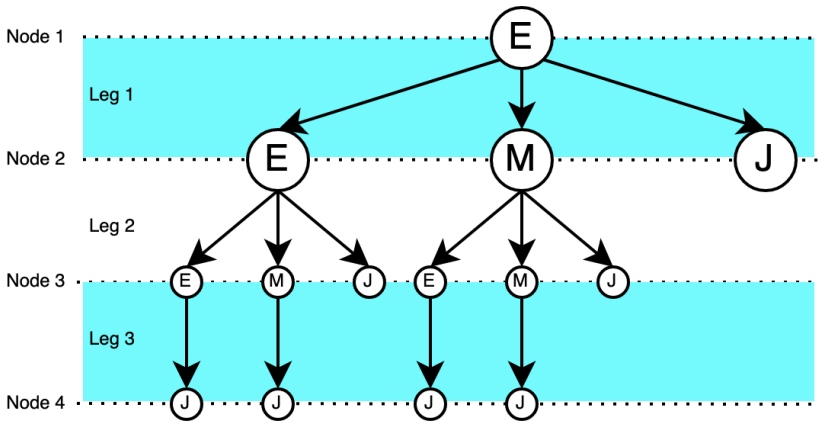


Figure 7.2: Graph of Earth-Jupiter transfer with two GAs.

The combinatorial complexity C is seven using Equation (7.1), where m is two and n is two as mentioned above. Furthermore, as half of the combinatorial space is explored, q is 0.5, and because there is only one recursion, Q is also 0.5. A random selection of sequences is then made. If the size of the combinatorial space is uneven as is the case here, then one sequence is added, resulting in four sequences that need to be evaluated. The direct transfer – in this case EJ – is set to always be evaluated in the first recursion. EEEJ, EEJ, and EMEJ are randomly selected. 24 islands are appointed to each sequence and the optimisation is performed, resulting in a number of ΔV statistics in Table 7.2.

Table 7.2: Statistics on EJ, EEEJ, EEJ, and EMEJ transfer.

Sequence [-]	Minimum ΔV [km/s]	Mean ΔV [km/s]	f_s [km/s]
EJ	19.032	19.568	19.032
EEEJ	18.207	21.262	18.207
EEJ	18.810	19.548	18.810
EMEJ	29.117	31.459	29.117

To evaluate the fitnesses, proportional quantities were defined. For simplicity, fp and fpi are set to 1.0. Using Equation (7.2), the f_s is calculated and shown in Table 7.2. Subsequently, these f_s values are grouped by the PTBs, namely 'E' and 'M', and are used as input for Equation (7.3), resulting in f_X shown in Table 7.3.

Table 7.3: Relevant quantities per PTB.

PTB [-]	Relevant f_s [km/s]		f_X [km/s]		Best PTB [-]
Earth	18.207 & 18.810	>	18.207	>	X
Mars	29.117		29.117		

The lowest f_X value is considered the best PTB of that recursion – because $fpi = 1.0$. From Table 7.3, it can be concluded that Earth is the best GA for this recursion, and therefore 'E' is appended to the RPS. This is followed by another selection of random sequences within the branch of 'E' starting from Node 2 in Figure 7.2, which marks the start of the next recursion. This example aims to illustrate the process of the RTBA and its complexities. In the next subsection, the algorithm itself is presented to give an overview of how the approach is structured in the code.

7.2.2. Sequence optimisation structure

This subsection presents the various blocks of the algorithm, from a system point of view.

The general setup of the MGASO can be seen in Figure 7.3 and is discussed here step by step. To initialise the MGASO, several parameters are necessary that were discussed in the previous subsection. The blue boxes are functions that perform a core step of the optimisation. The parameters are passed to the 'Create archipelago' function, which creates the set of transfers using the given parameters and the RPS, and creates a pg.archipelago object. Once the archipelago has been created, the pg.archipelago object is passed to the 'Run evolution' function that performs the evolution and passes the relevant data structures with the champion fitnesses (ΔV) and design variable vectors to the 'Determine best PTB' function. This function converts the ΔV values of all islands per sequence into the fitness (f_s) of that sequence using Equation (7.2). Subsequently, the fitnesses of the sequences are sorted according to their PTB and then compressed into the fitness of that PTB (f_X) using Equation (7.3). Subsequently, the RTBA checks whether any branches are left that require another recursion. If so the optimal PTB is appended to the RPS and the same process is repeated. If not, all the results obtained thus far are saved in a file directory consisting of the RPS with all algorithm meta-data as well as a database of the sequences evaluated.

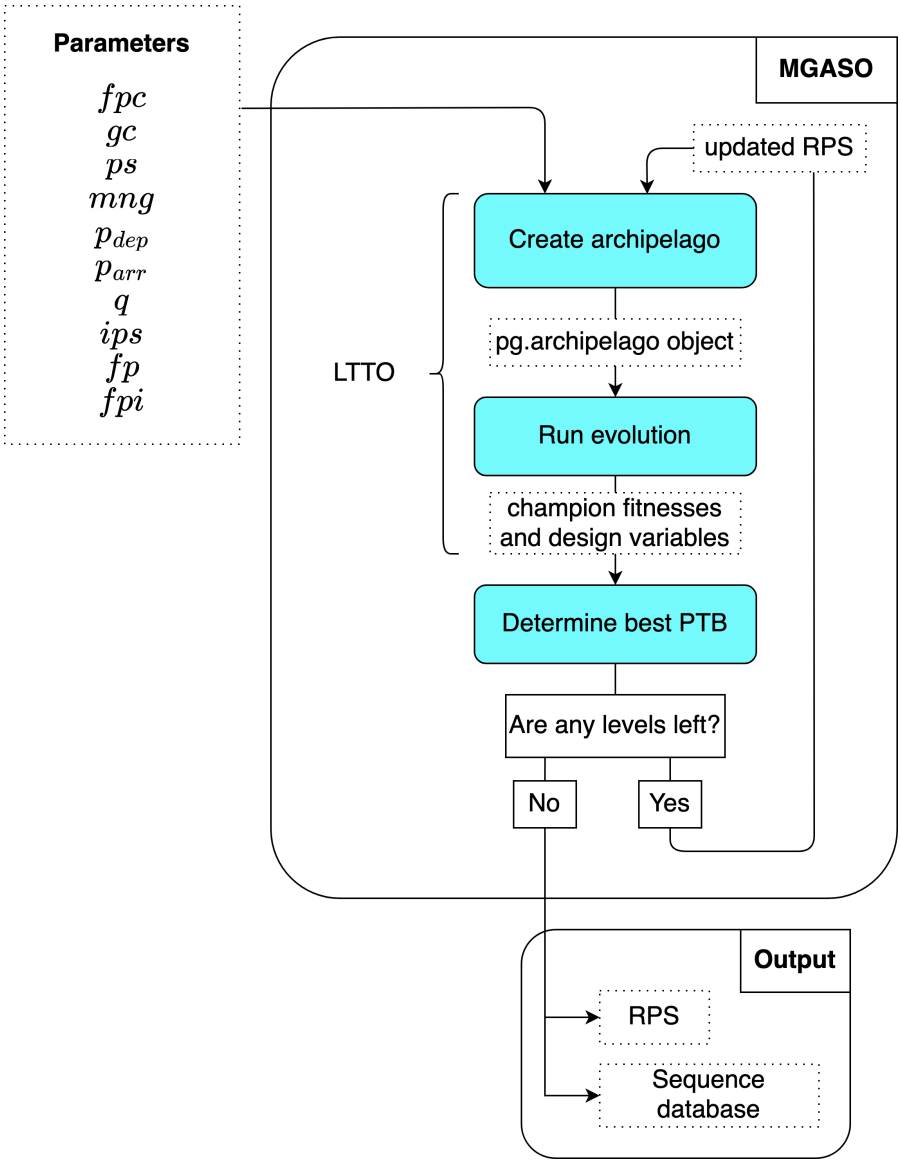


Figure 7.3: Block diagram of the MGASO process from a top-level perspective. The dotted lines denote the objects that are passed. The blue blocks denote sub-systems.

The creation of the archipelago is crucial to the approach and warrants further explanation. A block diagram is shown in Figure 7.4. The parameters and – if a recursion has already happened – the RPS are passed to the 'Determine archipelago configuration' function. All parameters are necessary except for 'gc', 'ps', and 'fpi': these are needed in the 'Run evolution' and 'Determine best PTB' functions. The 'Determine archipelago configuration' function executes a process similar to the MCTS 'expansion' phase, where randomly – yet evenly spread over the possible PTBs – sequences are created. All islands of a specific sequence are defined equivalently and all the parameters and bounds needed for the LTTO of this sequence are passed to these islands. Based on the parameters, a pg.topology object is created and added to the archipelago. However, because multiple islands consider the same sequence and in a single archipelago there are multiple sequences, the topology has to be specifically designed. A custom topology is defined that only links the islands of each sequence to all other islands of the same sequence. This topology now can recreate the behaviour of the topology used for the LTTO, where a 'fully_connected' topology could be used as all islands used the same sequence. This topology was only developed after the MGASO tuning process in Section 8.2.

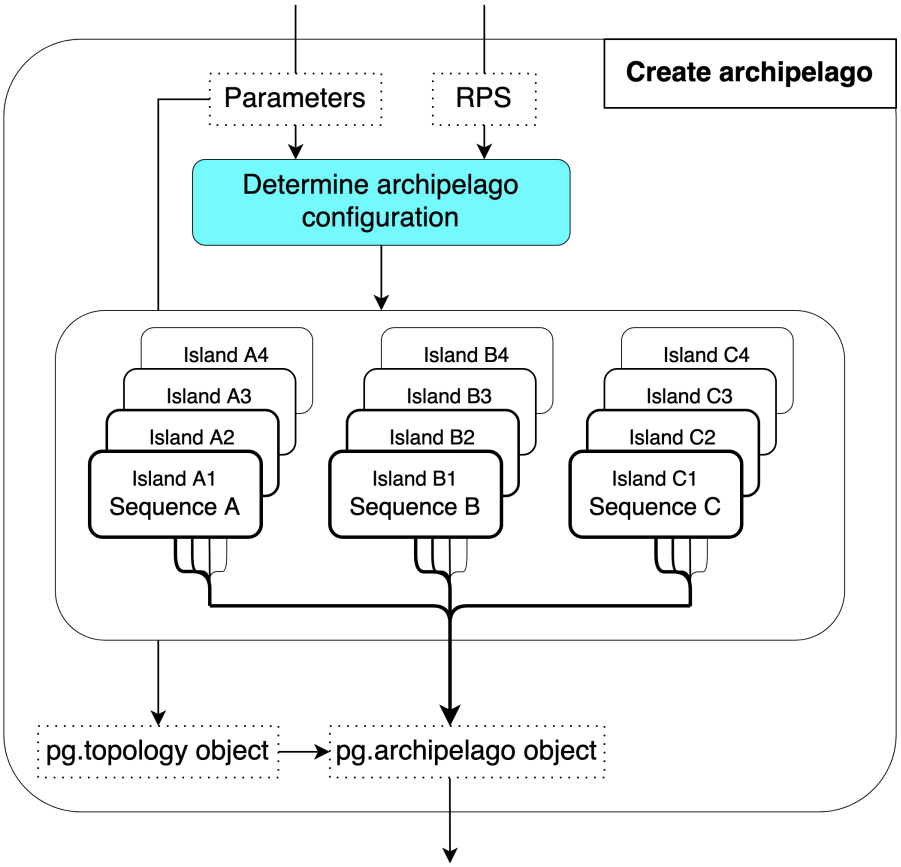


Figure 7.4: Block diagram visualising the creation of the archipelago. The dotted lines denote the objects that are passed. The blue blocks denote sub-systems.

The evolution is subsequently run and the resulting data structures are passed to the 'Determine best PTB' function. As was defined in the previous subsection, the minimum and mean values of the islands of each sequence are combined with the fp parameter to form fitness values for each sequence. The fitness values of each sequence are sorted according to the PTB of that layer. Finally, the fpi value is used to formulate the fitness values of each PTB, where the smallest one defines the best PTB and with that the updated RPS.

The outputs of the optimisation process are two-fold. On the one hand, the final RPS is output, being a sequence of planets that the optimisation deems the best sequence of GAs. This sequence does not by definition mean that the globally optimal sequence is equal to the final RPS: the RPS is always

equal in length to the number of recursions that are done. Consequently, the globally optimal sequence may only have one GA, whereas the RPS will be longer – and specifically equal in length to the number of recursions. On the other hand, the evaluated sequences database is output. This database is the collection of all the evaluated sequences, which are then sorted according to their f_s values. In short, the formal objective is the RPS, and the champion variables are the relevant sequences. This concludes the structure of the optimisation. In the next chapter, this approach is expanded with some features to increase the performance as well as a tuning process.

8

MGASO development

In this chapter, the MGASO is developed further by presenting specific adaptations that make the approach more efficient. Moreover, some tuning is discussed that assesses the relevant MGASO parameters.

8.1. Features

In this section, four aspects are considered. These can be seen as features that are an addition to the RTBA.

8.1.1. Possible GA candidates

The first feature determines the possible GA candidates, which can help with tuning any given MGASO problem definition. This step defines to a large extent the combinatorial complexity of the problem at hand, as the complexity grows exponentially with additional GA candidates. The relevant parameters are the departure and arrival planet. For this application, which only includes the planets in the Solar System, the outermost GA candidate is limited to the departure or arrival planet with the largest semi-major axis (the outermost of the two). An Earth-Jupiter transfer will not perform a GA with Neptune, for example. The Earth-Jupiter case is visualised in Figure 8.1. This is a safe step to make because the ΔV requirements for a transfer to an outer planet are reasonably high. Moreover, it is not expected to be worthwhile as one always has to decelerate in the final transfer leg, even if one flies by a planet with a higher semi-major axis compared to the target that decreases the heliocentric angular momentum perfectly. If the RTBA would be expanded to include moons or asteroids, this feature would have to be defined differently.

A final consideration is whether to allow Jupiter GAs. This is theoretically possible as some problem definitions require a rendezvous, rather than an intercept. A rendezvous may require that a trajectory performs a GA around Jupiter, to then rendezvous some time after. Because this transfer is a very small niche within the design space and the chances that a Jupiter GA is part of the optimal sequence are slim, Jupiter is left out. Its omission significantly decreases the combinatorial complexity, which is particularly useful for testing the RTBA approach.

As a quick indication of the combinatorial complexity that is pruned: in Table 8.1 the complexity can be seen with the number of GA candidates that are considered. The values are evaluated using Equation (7.1). The current problem definition without Jupiter has four possible GA candidates resulting in 85 possibilities in the first recursion. If Jupiter had been included, the combinatorial complexity would have increased by 80%. Consequently, the level of robustness that is desired may be infeasible due to the run-time constraint of DelftBlue.

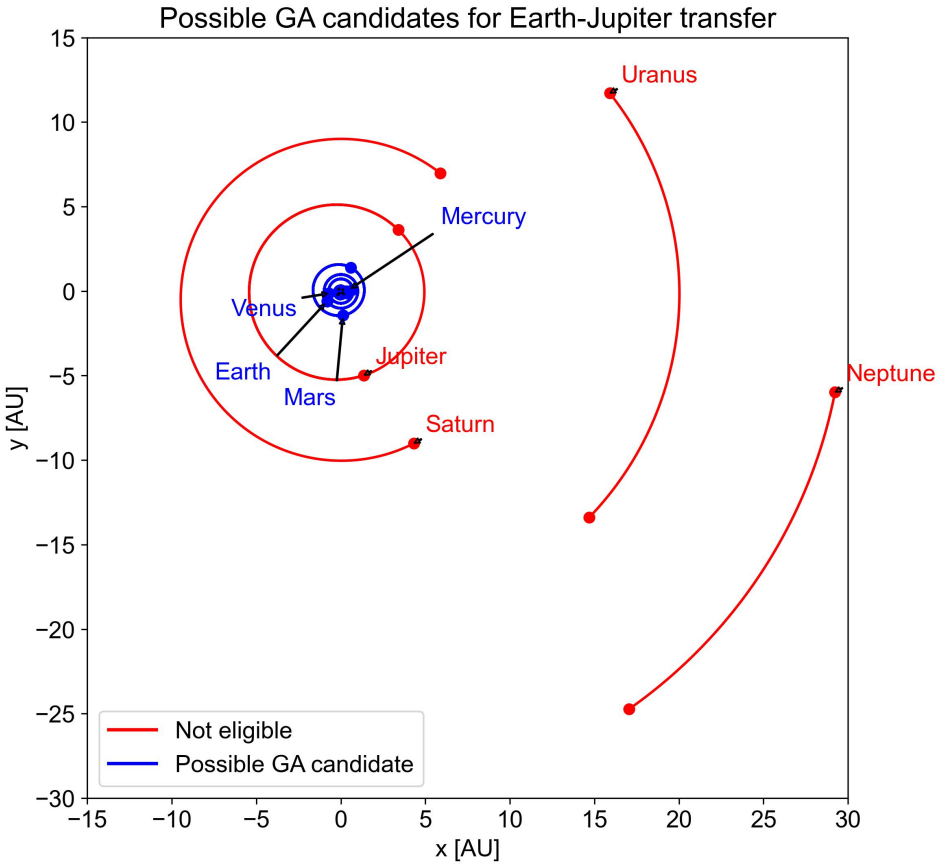


Figure 8.1: Permitted GA candidates for Earth-Jupiter transfer.

Table 8.1: Combinatorial complexity per number of possible GA candidates.

Possible GA candidates [-]	3	4	5	6	7	8
Combinatorial complexity (C) [-]	40	85	156	259	400	585

8.1.2. Unique sequences

The second feature of tuning is relatively straightforward and is used in MCTS as well. With the aim of making the RTBA as efficient as possible, it is beneficial to ensure that every sequence is only evaluated once. The random selection in the MCTS expansion step evenly distributes random sequences over every PTB. However, the random selection per PTB is also not entirely random: the selection is made without replacement, meaning that the sequences chosen have to be unique in that specific recursion. This ensures that no sequences are calculated twice within one recursion. If the next recursion randomly chooses a sequence that has already been evaluated, then the data from that sequence is passed on without the need for an optimisation – discussed in the next subsection. Each subsequent recursion has an increased chance of choosing sequences that have already been evaluated, which can accelerate the RTBA approach by preventing further LTTO steps.

8.1.3. Sequence inheritance

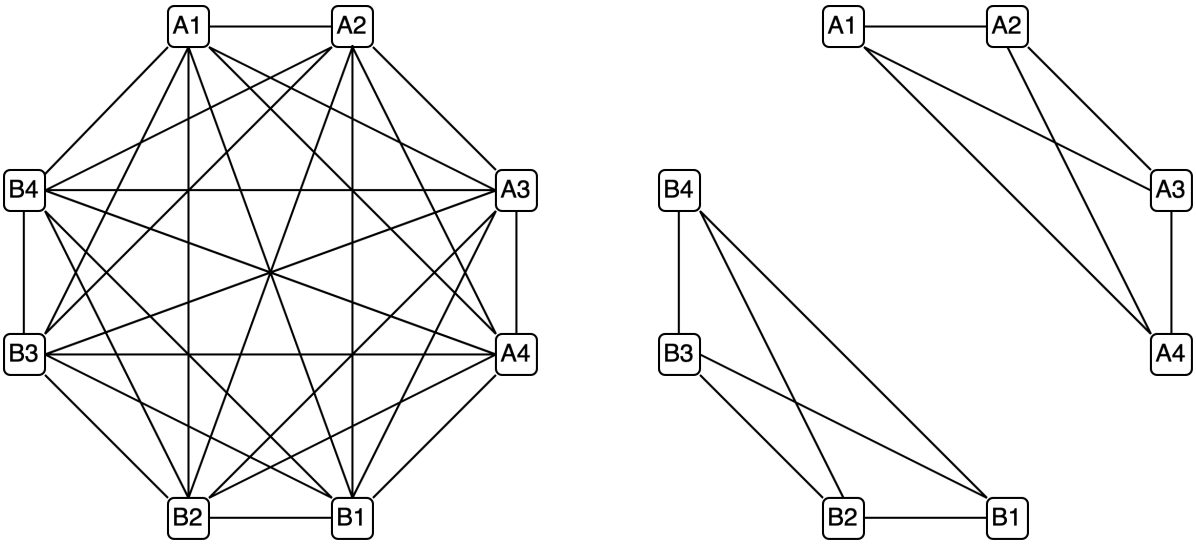
This subsection shortly describes another feature that is implemented in the MGASO. For context, the goal is to have the largest confidence in each PTB. The sequences of each PTB are gathered, however, the sequences do not have to be sourced only from the current recursion and Monte Carlo expansion. In addition, if a sequence has been evaluated in previous recursions, then that sequence is inherited by the current recursion, which increases the confidence in the PTB. This feature is not detrimental to the consistency of the RTBA: even if $fpi = 1.0$ – meaning that only the minimum f_s sequence determines

the best PTB – and a sequence is inherited that has the minimum f_s, then that sequence will determine the RPS. Moreover, each recursion can only append PTBs from that recursion level to the RPS. For example, if EEMMJ is inherited and it is the best sequence in the third recursion, then it can only append the third GA target, being the final 'M', which is different to the effect that EEMMJ would have had if it had been the best in the first recursion (where 'E' would have been appended to RPS). This feature can however mean that there is a small discrepancy in the confidence of each PTB choice, but this discrepancy is positive, as it increases the confidence relative to a minimum level of confidence.

8.1.4. Custom topology

Parallelisation has been mentioned on numerous occasions. Specifically, a topology is added to the archipelago. Though, the default topologies that PyGMO provides are found not adequate for the RTBA. Therefore, a custom topology is designed, which is discussed in this subsection. It should be noted that the custom topology was developed only after tuning the MGASO, and is therefore only present in the results of Chapter 10. The relevance of the tuning process does not decrease, as will be discussed later.

The custom topology shall only connect islands of single sequences because that is the only predictable and consistently effective use of the topology functionality for the RTBA. A topology that connects other sequences is not beneficial because the sequences have different lengths and different optimal design variables – the ToF can change drastically between different legs, for example. The topology structure is visualised in Figure 8.2 with an example using eight islands spread across two sequences.



(a) General 'fully_connected' topology. **(b)** Custom topology.

Figure 8.2: Graphical comparison of general vs custom topology.

The verification of this topology is given in Section 9.1.4. To compare the performance, two identical runs are done with a custom topology and the 'fully_connected' general topology from PyGMO. The problem formulation from Section 6.5 is used. Four islands per sequence are used. The results are tabulated in Table 8.2.

In Table 8.2, and in particular the 'Difference' row, it can be seen that there is a small increase in the ΔV values. This increase is due to the high ΔV sequences that have a higher uncertainty and the topology then does not make as much of a difference, because there are no optimal individuals to migrate. Moreover, the minimum ΔV is not persistently better with a general topology. The difference of 5.8 km/s can therefore be seen as negligible. The mean values, however, do consistently provide better results with the addition of the custom topology. This is the case for every sequence except the EMMMJ transfer, which also provided a high minimum ΔV. These results show that the custom topology systematically improves the results and specifically the mean, which is in line with what is found in Chapter 6.

The implementation of the custom topology in this thesis does not improve the run-time performance

Sequence [-]	Min ΔV [km/s]		Mean ΔV [km/s]	
	Custom topology	General topology	Custom topology	General topology
EMJ	19.227	19.240	19.297	20.598
EEJ	19.461	19.431	20.245	21.580
EJ	19.468	19.822	20.010	21.128
EMMJ	25.334	24.012	26.528	36.940
EEEEJ	36.370	35.429	46.327	51.169
EMEJ	38.711	46.865	39.974	72.030
EMMMJ	46.292	36.655	57.297	41.976
EEVJ	47.688	47.563	52.671	53.314
EVJ	48.765	50.795	49.275	54.565
EVMJ	52.118	47.816	53.894	63.601
Sum [km/s]	353.434	347.628	385.518	436.901
Difference [km/s]	-5.806		+51.383	

The '-' in the 'Difference' row indicates that the sum is smaller for the general topology, which is undesirable. The '+' indicates the opposite, namely a smaller sum for the custom topology.

Table 8.2: ΔV results of a selection of sequences evaluated with the custom and 'fully_connected' topology.

of the RTBA, because the number of generations has been fixed. As a future recommendation, a termination condition can be implemented to further increase the performance of the RTBA. The tuning process is aimed mainly at maximising the robustness of the results while remaining within the constraints of DelftBlue. This goal is not inhibited by the choice of topology, as the other parameters that are more indicative of the robustness and run-time complexity – such as ps, gc, fpc – are considered. The tuning process does consider the various MGASO-specific parameters that may be useful to compare in the results – mainly fp and fpi. However the reason for pruning values there is not caused by the optimisation results, but by theory. This constitutes the MGASO tuning process, the next part discusses the results obtained using the RTBA. These results form the core of this thesis, and the observations, interpretations, and conclusions made will enable the subsequent answering of the research question in Part V.

8.2. Tuning

In this section, a concise tuning process is conducted to test various parameter inputs of the MGASO. Ultimately, the goal of this section is to determine which combinations of these parameters lead to robust results. This tuning process is carried out with the problem definition from [Fan et al. 2021] as given in Section 6.5. This test case helps maintain consistency in the type of transfers looked at in this thesis.

8.2.1. Untuned RTBA

This subsection starts by presenting untuned results, which is the first of two iterations in the MGASO tuning process. The quantification of the tuning results is a challenge, because the output is not a ΔV value, but a sorted set of sequences and an RPS. Therefore, the focus will lie on the computational complexity and how to maximise robustness within the constraint.

The first iteration uses parameters as defined in Table 8.3. The parameters are chosen based on the tuning process from Part II. Specifically, the number of CPUs is maximised based on the maximum allowed CPUs of DelftBlue. ips is chosen based on the value used in Part II, but subtracted by one so that two sequences – with 23 islands each – can be evaluated in parallel using the 46 CPUs. Two free parameters are chosen initially as this gives the most freedom to the shaping functions. For the untuned MGASO process, a grid search is conducted on the fp and q parameters. The fpi parameter is equal to one as its development happened later during the thesis project.

As a short intermezzo, during this tuning process, a different method was used to calculate the remaining combinatorial complexity which was not exact and conservative in that it finds a lower combinatorial

Table 8.3: Fixed MGASO parameters for untuned results.

Parameter type	Value
gc	300
ps	1200
ips	23
CPUs	46
fpc	2
fpi	1.0
Recursion count	3

complexity than the correct method. The results in Chapter 10 do use the corrected formulation of the combinatorial complexity using Equation (7.1). Consequently, the original values chosen for the grid search are converted into correct q values, which results in non-round numbers. The difference between the converted q values found by the original method compared to the corrected q values used in the results is small: the differences are in the order of a few percentage points and the conclusions drawn in the tuning process are more general. Therefore, this discrepancy has no consequences for the quality of the conclusions drawn in this chapter.

The fp is tested on values of 1.0, 0.75, 0.5, 0.25, and 0.0 and q is tested at values of 0.08, 0.23, 0.38, and 0.53, resulting in 20 MGASO processes. The latter two q values however exceeded the DelftBlue maximum run-time constraint, and therefore terminated prematurely, tabulated in Table 8.4. The coarse resolution is chosen because each point on the grid requires an MGASO, which is time intensive. An MGASO is subject to the various DelftBlue constraints discussed in Section 2.2.1, and this can lead to the premature termination issue. Furthermore, the fp value is chosen such that it covers the entire possible range, and q is chosen for a wide array of possible values, though the objective, ultimately, is to minimise the q required to obtain robust results. A smaller q means that fewer sequences are evaluated. As mentioned before, the fpi parameter is not included in this initial iteration and was developed for the next iteration based on new insights on the behaviour of the MGASO. Specifically, the RPS was based on the sequence with the minimum f_s. As discussed in Section 7.1, the mean of the f_s values of every sequence using a given PTB also needs to be taken into account. In the Earth-Jupiter case, the EMJ transfer has the lowest f_s values. If the RTBA evaluates the EMJ in the first recursion, it will then most likely choose Mars as the most optimal PTB, assuming that there are no better results. This hypothesis is confirmed by the RPS strings per grid point shown in Table 8.5.

Table 8.4: Average run time per q value.

Fraction [-]	Total sequences evaluated [-]	Average run time [hh:mm]
0.08	16	12:00
0.23	30	22:30
0.38	-	>24:00
0.53	-	>24:00

The hh:mm unit stands for hours and minutes. This level of accuracy suffices for long run times.

In Table 8.5, every fp value finds a different RPS for $q = 0.08$, whereas this is not the case for $q = 0.23$. In the $q = 0.23$ run, the 'MMM' RPS is found three out of five times. It was previously mentioned that this is probably caused by the presence of the EMJ transfer in the set of evaluated sequences, which is confirmed by the database of sequences. In fact, in every grid point where the EMJ transfer is evaluated, the first element of the RPS is Mars. The RPS is further discussed in the next tuning iteration when all the q values have been evaluated. The sequences that are evaluated can be found in Appendix D.1. The most crucial aspect of this iteration, is that no seed was defined for the random sequencing. Therefore,

the difference in RPS can not be independently attributed to the fp value, rather it is caused by variations in the evaluated sequences at each iteration. Furthermore, the run times are too long for higher values of q. Therefore, a similar grid search is done with different parameters that reduce the computational complexity, discussed in the next subsection.

Table 8.5: Table with RPS strings for the grid search.

Fitprop value	Fraction	
fp	$q = 0.08$	$q = 0.23$
1.0	EEM	EMM
0.75	EMM	MMM
0.50	MEE	MMM
0.25	MMM	MMM
0.0	MEM	EEM

8.2.2. Reduced-time results

For this iteration, various parameters are changed to reduce the computational complexity without decreasing the robustness significantly. Moreover, a seed is added such that the set of sequences is consistent over multiple grid points – assuming an identical RPS. The fpi parameter is added as an additional parameter to the grid search. This tuning step is also conducted without the custom topology, as it was not yet developed.

Difference in setup

First, 200 generations and a population size of 800 individuals are chosen. Throughout the tuning process, the values proved to have converged relatively well after 200 generations. Moreover, a higher population size does not always guarantee better results and the grid search in Section 6.3.1 was too coarse to determine the optimal population size. The population size of 800 was used in some initial testing activities and performed well. Second, rather than an ips of 23, 14 is chosen. This number is based on the fact that now three sequences can be evaluated simultaneously while using the maximum number of 46 CPUs allowed by DelftBlue. The simulations are run with 42 CPUs, however, as this has the same computational efficiency but allows for quicker queuing in DelftBlue. Last but not least, the fpc is reduced from two to one. In Part II, it was found that the one free parameter count showed a similar robustness level for most configurations, with a significant drop in computational complexity. Depending on the run times observed and the level of robustness, a new and informed decision can be made on the configuration of the runs for the final results in Chapter 10.

Table 8.6: Parameters that are fixed for reduced-time iteration.

Parameter type	Value
gc	200
ps	800
ips	14
CPUs	42
fpc	1
Recursion count	3
Seeds per recursion	[266, 267, 268]

An additional change, as mentioned before, is that a seed is added also for the random selection of sequences, and not only for the initialisation of the archipelago. The 'random' package for Python is used and specifically the Mersenne Twister algorithm, which produces pseudo-random numbers. The Mersenne-Twister algorithm produces 53-bit precision floats and has a period of $2^{19937} - 1$. Because the number of experiments (moments at which one calls a random number) is relatively limited, the bias that

exists in the pseudo-random number generator is not considered critical. For Monte-Carlo simulations where one has orders of magnitude more runs and needs a high resolution, the pseudo-number generator from the 'random' package may not suffice.

A similar grid search is conducted as was done in the previous subsection. To further reduce the total run time of a single grid search, the resolution is decreased further to three values per parameter. These values still cover the entire possible space and are evenly distributed. q is evaluated at values of 0.08, 0.30, and 0.53, fp is evaluated at values of 1.0, 0.5, and 0.0, and fpi is also evaluated at values of 1.0, 0.5, and 0.0. Based on the results discussed next, a prediction can be made about the fp and fpi values that should be used for various fraction values in Part IV. Furthermore, an improved estimate can be made of the parameters from Table 8.6.

Analysis of sorted sequences

The sequences of the grid search are shown in Figure 8.3. To reiterate, this tuning process is focused on the robustness of the results in terms of the parameters from Table 8.6 but also the fp and fpi parameters. The q values are only assessed in the results because the most optimal parameters are used. In the final results, a q value of 0.5 is tested. Therefore only the $q = 0.53$ tuning case is shown for conciseness: results for the other two q values can be found in Appendix D. In Figure 8.3, the f_s is plotted against the evaluated sequences, sorted by increasing f_s values. The bottom left is more optimal and the top right is less optimal. It can be seen that a lower fp value – where the mean is prioritised more in the fitness of the sequence – results in a higher f_s value. This is plausible because the mean is equal to or higher than the minimum by definition. This trend is only observed for a single seed: a run with one seed could have a minimum f_s that is higher than the mean of a run with another seed.

The f_s values differ for various grid points evaluating the same sequence. As a general trend, the higher the f_s values of a sequence, the higher the spread across all grid points for that sequence – note the logarithmic scale of the y-axis. The spread here is defined as the difference between the maximum and minimum f_s values. However, this increase in spread does not apply to all sequences. For example, the EYJ sequence is very sub-optimal at around 100 km/s with a spread of around 15 km/s. The EMVYJ sequence, directly next to EYJ, has a similar minimum f_s value but a spread of 40 km/s. This inconsistency shows that the ranking of the sequences depends on the fp and fpi values, which is undesirable in terms of robustness. Of course, the $fp = 1.0$ runs are the most accurate in terms of actual optimality, the question remains as to whether it contributes to a more optimal RPS – which was the reason for implementing these parameters. This reasoning is elaborated upon shortly.

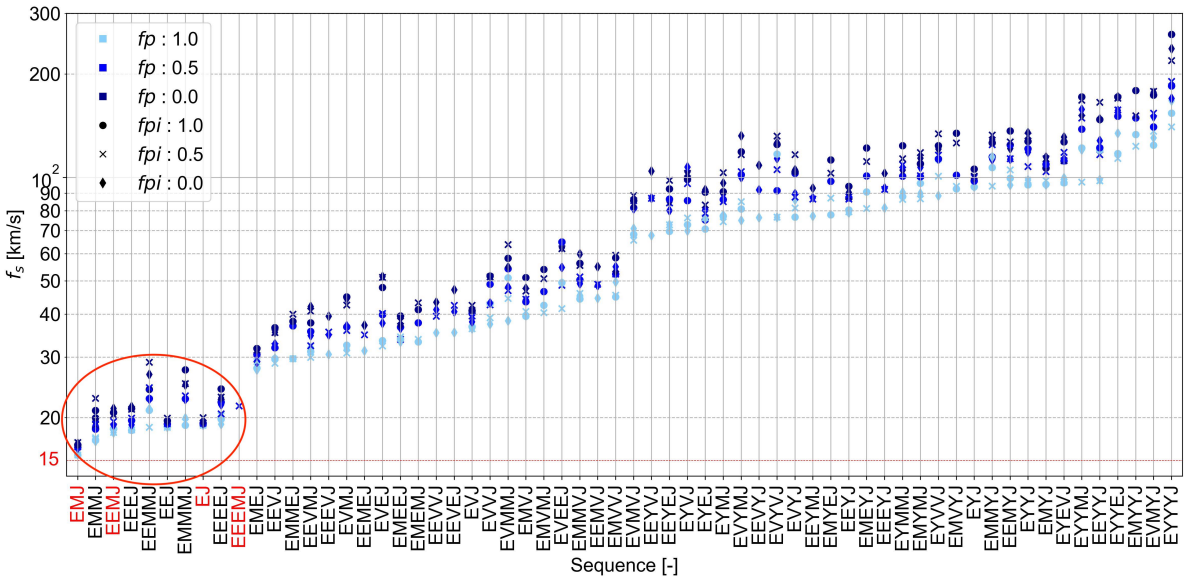


Figure 8.3: f_s of all evaluated sequences for $q = 0.53$.

The sequences at lower f_s are more relevant for the performance assessment and less spread is

observed there on average. This inconsistent spread observed at higher f_s values, therefore, does not contribute significantly to the optimality of fp, fpi, or q. The difference in the spread of low-f_s sequences is discussed in more detail later. As for the different fpi values, there is no trend similar to fp that can be observed. This is plausible because Figure 8.3 shows all sequences and the fpi variable only affects the RPS choice. fpi does influence what sequences are evaluated via the RPS; different RPS strings will result in different sequence sets, which can be seen by the fact that not all sequences have the same number of grid points. Therefore, if different fpi values – shown by different markers – evaluate the same sequence, its function reduces to an evaluation with the same seed which indicates the uncertainty. This uncertainty ranges from roughly 4 km/s for some low-f_s sequences to about 50 km/s for high-f_s sequences. These uncertainties are large, but it should be kept in mind that this tuning step does not include the custom topology, which was designed to reduce these uncertainties for the mean. In addition, the reduction in ps, gc, fpc, and ips may have contributed significantly.

As was determined in Chapter 6, with the current implementation, nuanced differences in the order of 1 km/s or less cannot be differentiated robustly. In higher-fidelity methods, the ranking of these sequences may change as a result. To remedy this, the spread should be minimised, which would result in reverting the decision to decrease the robustness of the results by decreasing the gc, ps, fpc, and ips to adhere to the run-time constraint of DelftBlue. Solutions for this are discussed later on. However, because the input to higher-fidelity methods would not be a single optimal sequence but rather a selection of sequences that includes the optimal sequence, the distinction does not have to be made in the RTBA. As long as there is a clear gradient in quality through the sequences, and certain sequences are better than others no matter what seed or fpi is defined, the output is physically relevant. Note that most of the RTBA is verified and partially validated, which is discussed in Chapter 9, making the relevance more reliable. This is the case in Figure 8.3: the low-f_s sequences are separated from all the other sequences by f_s values that are significantly lower than the best f_s of other sequences across all grid points. Therefore, some selection has to be made of the lower f_s sequences that have the potential to be the globally optimal MGA sequence within the definition of the problem. To enable the consistent determination of such a group, a metric is defined:

The low-f_s group is defined by the sequences that have a minimum f_s that is within 30% or 6 km/s of the overall lowest f_s.

This definition is given based on the uncertainties that are observed for the sequences, together with the differences in verification ΔV that have been observed. Two conditions are given in the definition, one relative and one absolute, to ensure the quality for smaller combinatorial spaces but also make the definition applicable to larger combinatorial spaces. In the case of Figure 8.3, the EMJ transfer has the lowest f_s and the left-most 10 sequences are part of the low-f_s group.

In Figure 8.3, this group is encircled in red. This group comprises the sequences up until and including the EEEMJ transfer. The spread in f_s of these sequences does overlap a small amount with the minimum of some sequences that are not in the low-f_s group. This result means that the selection of parameters for this tuning step does not provide a completely consistent result across all grid points for the low-f_s group. Therefore an increase in robustness, as alluded to before, is necessary to reduce that uncertainty and create an exclusive group of sequences that perform better than the other sequences. Though, because the previous iteration terminated prematurely at high q values, the robustness of the results with the higher-fidelity parameters is unknown. Nevertheless, it is still recommended to increase the robustness where possible while staying within the run-time constraint. This aspect is discussed later in this section. Specifically for the low-f_s group, more discussion is necessary; this group is plotted in Figure 8.4.

In Figure 8.4, it can be seen that the low-f_s group shows a relatively large difference in spread across the various fp and fpi values. As was said before, the fpi data points do not directly affect the fitness and therefore only affect the spread through the uncertainty in every LTTO process. The spread is correlated with the length of the sequence; this correlation is less pronounced with high-f_s sequences. Longer sequences have a higher spread, which is caused by the increased design complexity. This is an additional reason for increasing the robustness, as the length will then have less effect, in addition to the general improvement to the robustness. However, the main cause of the difference in f_s is expected to be the absence of a topology to filter out the high ΔV outliers. This leads to the development of the custom topology, discussed in Section 8.1.4.

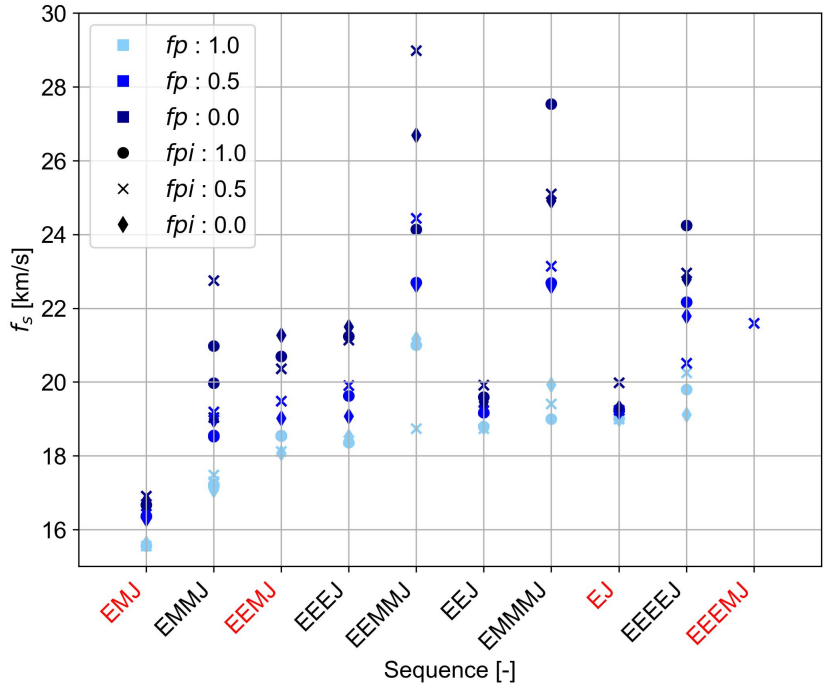


Figure 8.4: f_s of top 10 evaluated sequences for $q = 0.53$.

When analysing the spread per fp value, it can be seen that the spread is the smallest for $fp = 1.0$ and that this spread increases with a lower fp value. This can be explained by the fact that the mean value – with a non-zero for $fp < 1$ – is based on a distribution of ΔV values without the custom topology. This distribution of ΔV values is not symmetric because there is a physical limit to the lower ΔV values; the asymmetry can be observed in Chapter 6. Therefore, the minimum is expected to be relatively constant, whereas the mean varies per run. Even with the topology – which would have a weight of at least 0.01 based on the findings from Chapter 6 – there would be a non-zero spread for a single sequence and fp value. It can therefore be concluded that fp should be close to one, to minimise the spread, and increase the consistency of the results. In addition, fp values of 1.0 also more accurately represent the actual optimum found, and therefore provide a more reliable ranking of sequences. The other fp values return different sequence rankings, which is not ideal, but the collection of sequences themselves are the same.

A separate observation is that, although this is not a verification process, the sequences found by [Fan et al. 2021] are among the low-f_s group. Even across all fpi values, these same sequences are evaluated, albeit with one seed. Their f_s value – for the $fp = 1.0$ case – is similar to the results found in Chapter 6. This would indicate that the robustness of the simulations is accurate enough in terms of minimum ΔV, but not in terms of robustness across all simulations for a single sequence.

To shortly recap, the main takeaway is that the addition of the custom topology is key, but additional robustness can be achieved by changing the input parameters from Table 8.6. Besides the sorted sequences, the RPS is crucial to the performance of the algorithm, which is discussed next.

Analysis of RPS

This sub-subsection discusses the RPS. In particular, first, the best sequences from Figure 8.3 are mentioned and compared to the theoretical expectation. After this, these sequences are compared to the found RPS strings to compare. Finally, the RPS strings are analysed in terms of consistency and recommendations are made on the parameters that should be used and investigated further in the results.

Because of the addition of consistent seeds in this iteration, the assumption can now readily be made that any difference in RPS can be attributed to either the uncertainty in the results or the various grid search parameters. It should be noted that there is no optimal RPS, as this problem definition has no

verifiable optimal sequence. However, the RPS can be compared to the sequences with the lowest f_s shown in Figure 8.4: if the lowest f_s sequences overlap with the RPS, then the RPS has converged well, as it increases the probability of finding the highest fitness sequences. If there is no overlap, then the RPS converges incorrectly, and the subsequent recursions will evaluate lower fitness sequences. This difference is especially crucial with lower q values because the chance of evaluating the low-f_s sequences is smaller. Whether the RPS converges well or not can be verified by what one would expect theoretically. Table 8.7 shows all the RPS strings for all grid points.

From the low-f_s group in Figure 8.4, the EMJ and EEJ transfer are the only single-GA sequences, which start with 'M', and 'E'. From the two-GA sequences in Figure 8.4, the RPS strings 'EM', 'MM', or 'EE' specifically are expected to be optimal. The 'ME' RPS string – represented by the 11th sequence 'EMEJ' – is less optimal. This is plausible as an optimal sequence with the 'ME' sequence string would have to use significant thrust to decrease the heliocentric orbital energy to that of Earth after having increased it to reach the Mars orbit. Even if one does not circularise and use a highly eccentric orbit to save thrust, it is unlikely that this is better than consistently increasing the angular momentum (with an 'EMJ' transfer for example). Also, the MGA strings 'MMM', 'EMM', 'EEM', and 'EEE' are observed. These three-GA sequences follow the same pattern as the one- and two-GA sequences, where only Earth and Mars are used as GA targets, and specifically only outbound sequences are most optimal. To sum up what one would want in terms of RPS, based on the sequences from Figure 8.3 and the theoretical expectation, an RPS that includes Earth and/or Mars GAs with any number of GAs within the problem definitions. Moreover, Earth should never succeed Mars. Next, the actual RPS strings that can be seen in Table 8.7 are compared with the expectation.

q [-]	0.08			0.30			0.53		
fpi [-]	1.0	0.5	0.0	1.0	0.5	0.0	1.0	0.5	0.0
$fp = 1.0$	EMM	EMM	EMM	MMM	EMM	EMM	MMM	MMM	EVM
$fp = 0.5$	EMM	EMM	EMM	MMM	EMM	EMM	MMM	EEE	EVM
$fp = 0.0$	EMM	EMM	EMM	MMM	EMM	MMM	MMM	MMM	EVM

As a reminder, the values 0.08, 0.30, and 0.53 stem from the different methods of calculating q, as discussed in Section 8.2.1.

Table 8.7: RPS strings for all grid points of second tuning iteration.

First, the RPS strings are shortly compared to the previous iteration. In the previous iteration, there were a few occurrences of 'ME', and the 'MMM' occurrences increased for $q = 0.23$. These results were not reproducible due to the lack of a seed. Nevertheless, an improvement can be observed: there are no occurrences of 'ME', and there is more consistency across the various fp values. However, this comparison is also limited due to the small q values in question, which makes the RPS strings statistically less robust.

Second, the RPS strings are compared to the expectations discussed above: the sequences are almost exclusively in agreement with the expectation of the results in Figure 8.3. With one exception, only Earth and Mars are found in the RPS strings. Additionally, as said before, Earth never succeeds Mars. However, when diving deeper into why the RPS converged to the planets that they have, it is seen that for later recursions and low q values, there is a low statistical confidence in each PTB. This results in one strange RPS string in Figure 8.4: 'EVM'. To explain this, one should investigate the underlying sequences that are being evaluated. When looking at the sequences that caused the RTBA to converge to Venus in the second recursion, there are two 'EEV' sub-sequences and three 'EEM' sub-sequences. For the sake of simplicity, other evaluated sequences are not considered for this example. Specifically, EEVMJ and EEVJ are evaluated for Venus, and EEMYJ, EEMJ, and EEMMJ are evaluated for Mars. The deciding factor in finding the best PTB is that – independently of the fp value – the Mars sequences include an 'EEMY' sub-sequence, which is highly sub-optimal. The Venus sequences do not include such a highly sub-optimal transfer and therefore Venus is found to be a more optimal PTB. As a result, while one would expect Mars to be a better PTB, the RTBA converges to Venus instead. This sensitivity is considered further in the results in Chapter 10.

For almost all fpi values and q values, there are consistent RPS strings across all fp values. This means that despite the outliers and spread of the results discussed previously, the RPS is still the same. Two exceptions are observed at different fpi and q values. These exceptions are caused by a nuanced difference in the optimality of the PTBs as a consequence of the fp parameter. This behaviour is discussed more elaborately in the results. To increase the confidence in the consistency of the RPS determination, multiple seeds should be evaluated, which is done in Chapter 10.

Because it was previously determined that the statistical confidence per PTB decreases substantially with each subsequent recursion, the third and final RPS GA is therefore not as relevant – within the problem definition that limits the maximum number of GAs to three. Consequently, the number of recursions evaluated in the RTBA is changed to two, rather than three. This reduction saves a substantial amount of computational time, which can be used to obtain more robust results by using more generations with higher population sizes, for example. The choice for these changes in parameters is based on the LTTO findings from Part II, for which the quality and robustness of the results have been verified. The computational time for all q values is shown in Table 8.8. The run times are approximately evenly distributed and the deviation is caused by the initialisation during each recursion. Even for high q values, this iteration only takes roughly 10 hours to run, which is less than 50% of the maximum run time that is permitted.

Table 8.8: Run time per q value.

Fraction [-]	Total sequences evaluated [-]	Average run time [hh:mm]
0.08	13	03:12
0.30	33	06:08
0.53	45	10:08

The hh:mm unit stands for hours and minutes. This level of accuracy suffices for long run times.

As all the RPS strings are part of the pool of potentially optimal sequences, no conclusion can be drawn on the optimal fp and fpi values directly. However, from a theoretical perspective, if $fp = 0.0$, the PTB assessment is solely focused on the mean result of each sequence. If one island finds a highly-optimal ΔV, then it will be filtered out and the methodology will not take this island into account for the RPS determination. So while RPS strings are as good as those with other fp values, it is expected to be less optimal. This expectation is more valid for larger problems, and so the $fp = 0.0$ grid point is henceforth not considered. As for the $fpi = 0.0$, an analogous logic can be applied: only considering the mean of the sequences of one PTB results in an optimisation that converges to the highest average f_s fitness, whereas the goal is to find the best PTB that results in the minimum ΔV. As an additional reason, fewer grid points result in lower run-times for a grid search. In terms of q values, there is no clear difference in the RPS-string quality and/or in the overlap with the pool of optimal sequences determined earlier. Therefore, all q values are still considered in the results.

In conclusion, in this tuning step, it was found that higher robustness is needed. This is achieved by evaluating multiple seeds, adding the custom topology mechanism explained in Section 8.1.4, and increasing the robustness by enlarging the population size and generation count. To allow for this increase in terms of computational time without violating the run-time constraints, the final recursion is removed from the algorithm. Before presenting these results, it is crucial to determine how reliable this approach is: the next chapter discusses the verification and validation of the approach, after which the results are presented.

Part IV

Performance Assessment

9

Verification and validation

As is often the case with research in the space sector, there is little to no physical data that fits the problem at hand, which constrains the validation to component or sub-system level validation, rather than a fully integrated validation. The verification process is similarly challenging: few papers have optimised the same problem, which prevents the full verification of the approach. In this chapter, the verification and validation steps are presented that are performed.

9.1. Verification

This section verifies several elements of the approach to ensure their correct functioning. The verification includes the hodographic-shaping method implementation, the optimisation algorithm, the integrated LTTO verification as well as the custom topology.

9.1.1. Hodographic-shaping method

To begin with the hodographic-shaping method, this subsection discusses the verification of the implementation of the hodographic-shaping method in this thesis.

[Gondelach and Noomen 2015] developed the hodographic-shaping method, and computed a number of reference solutions for basic transfer problems. As this thesis only considers missions to planets, the Earth-Mars transfer is chosen as a suitable case – rather than a transfer to a NEO such as 1989ML. This test case is used with zero free coefficients and two free coefficients to show the correct implementation of the base functions, but also of the free coefficient velocity functions. The velocity functions are the recommended functions as presented in Table 4.3. The zero free coefficient case has no degrees of freedom, and is thus a single trajectory, whereas the free coefficients of the two free coefficient cases are design variables. An optimisation method is therefore needed here: in this case, the Nelder-Mead Simplex gradient-based optimisation method is used, as it was shown to be effective for this application in [Gondelach and Noomen 2015] and [Stubbig and Cowan 2021]. The results of this comparison are shown in Table 9.1. The Nelder-Mead Simplex optimisation method is verified in the next subsection.

The ΔV and f_{max} values found with the implementation in this thesis are almost identical to those of [Gondelach and Noomen 2015], with a difference of 10^{-3} km/s and 10^{-6} km/s respectively for dV and maximum thrust. These orders of magnitude are plausible as the method is semi-analytical, so there are few uncertainties that can lead to a different result: one cause may be the numerical integration with a different integrator or a different number of integration steps. It can now be said that the Tudat implementation of the hodographic-shaping method reflects the method as was designed by [Gondelach and Noomen 2015] and is considered verified.

Table 9.1: EM transfer characteristics for [Gondelach and Noomen 2015] and this thesis.

0 free coefficients				
V_r and V_θ CPowPow2		**V_z CosR5 P3CosR5 P3SinR5**		
Source	Departure date [MJD2000]	ToF [days]	ΔV [km/s]	f_{max} [10^{-4} m/s^2]
Gondelach	10025	1050	6.342	1.51
This thesis	10025	1050	6.341	1.516
2 free coefficients				
V_r and V_θ CPowPow2 PSin05 PCos05		**V_z CosR5 P3CosR5 P3SinR5 P4CosR5 P4SinR5**		
Source	Departure date [MJD2000]	ToF [days]	ΔV [km/s]	f_{max} [10^{-4} m/s^2]
Gondelach	9985	1100	5.771	1.50
This thesis	9985	1100	5.771	1.495

The departure and arrival velocity are set to 0. The number of revolutions is 2 for both runs.

9.1.2. Optimisation method

This subsection discusses the verification of the optimisation methods used from PyGMO. Specifically, the SGA and Nelder-Mead Simplex methods are verified against two reference problems.

In [Cowan 2022], both the Rastrigin and the Binh-and-Korn functions were found to be suitable for verification. As previously discussed, this thesis only considers single-objective optimisation. Consequently, only the Rastrigin optimisation problem has to be solved for a correct verification of the SGA. Specifically, PyGMO offers a built-in MINLP Rastrigin problem, which is ideal because of its mixed-integer nature and the ability to customise the complexity – the continuous and integer dimensions of the function are parameters of the function. In addition, the Ackley problem is used as well for comparison – this problem is also built into PyGMO. It is a continuous problem with an optimal function value of 0 and optimal variable values of 0 for any number of variables.

The MINLP Rastrigin function is shown in Equation (9.1). The continuous and original variant of this function is defined in [Rastrigin 1974].

$$F(x_1, ..., x_n) = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi \cdot x_i)) \tag{9.1}$$

Here, continuous variables are within $x_i \in [-5.12, 5.12]$, integer variables are within $x_i \in [-10, -5]$, F is the function value, and n is the total dimension. While the optimum is not found in references, it can be derived. An optimum can be deducted from Equation (9.1). First of all, $10 \cdot n$ is constant with respect to the variables, which means that its derivative is zero. Second, the sums can be split into integer and continuous elements for a better overview. The function can be rewritten as Equation (9.2).

$$F(x_1, ..., x_n) = \sum_{i=1}^{n-m} (x_i^2 - 10 \cdot \cos(2\pi \cdot x_i)) + \sum_{i=n-m}^{n} (x_i^2 - 10 \cdot \cos(2\pi \cdot x_i)) \tag{9.2}$$

Here, m is the integer dimension of the problem. A short proof follows:

Assumption: $f(x) \geq \min\{f(x)\}$ and $g(x) \geq \min\{g(x)\}$

Therefore: $f(x) + g(x) \geq \min\{f(x)\} + \min\{g(x)\}$

And therefore: $\min\{f(x) + g(x)\} \geq \min\{f(x)\} + \min\{g(x)\}$

Here, f and g are generic functions, but they represent any sum of terms in Equation (9.2). Consequently, only the minimum of each term in both sums has to be found to find the global optimum. This leaves Equation (9.3).

$$F(x_i) = x_i^2 - 10 \cdot \cos(2\pi \cdot x_i) \tag{9.3}$$

The second term in Equation (9.3) has a negative cosine, which can be minimised to -1 for multiples of 2π – where the variable values are integers within their respective bounds. The second term can therefore be minimised to -10. All that remains is the minimum of a square in the first term. For the continuous variables between $x_i \in [-5.12, 5.12]$, $x_i = 0$ is the minimum value. The integer variables lie between $x_i \in [-10, -5]$, where $x_i = -5$ is the minimum value. These minimum values also are equal to the minimum value of the cosine in the second term. Now that the optimum is known, the verification can be performed.

The MINLP Rastrigin problem is configured as follows: the dimension of both the continuous and integer part of the problem is set to five – $n = 10$ and $m = 5$. This value is chosen based on two factors. On the one hand, if the value is too small, then the algorithm might 'randomly' reach an optimal value, even if one would not expect it to converge. On the other hand, the dimensions only need to be proportional to the dimension of the problem in this thesis to prevent any unnecessary run times. The theoretical minimum for the MINLP Rastrigin problem with the aforementioned dimensions is 125. As for the second test problem, the optima are already known: the Ackley problem is defined by Equation (9.4).

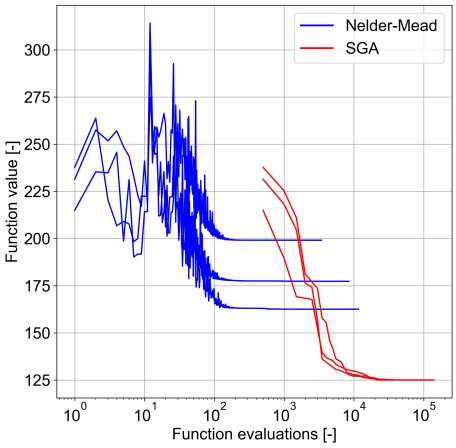
$$F(x_1, \ldots, x_n) = 20 + e - 20e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^{n} \cos(2\pi x_i)} \tag{9.4}$$

Here, $x_i \in [-15, 30]$. The Ackley problem is configured with six parameters, this is a relatively low number. The dimension is not important because the Ackley problem verification is an additional test to compare the performance and cross-check the convergence on more than one formulation.

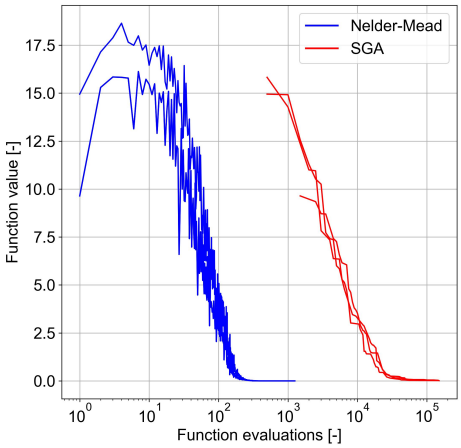
In Figure 9.1, the function value is plotted against the function evaluations of the MINLP Rastrigin problem. In addition to the SGA applied to an MINLP verification, this method is shortly compared to the Nelder-Mead Simplex method applied to the same problem, and both are also applied to a second reference problem as a second check. To add reliability to the verification, three seeds are used per reference problem: 4217, 4218, and 4219 were used for the MINLP Rastrigin problem, and 4212, 4213, and 4214 for the Ackley problem. The SGA is configured to use 300 generations and a population size of 500. The Nelder-Mead Simplex method is limited to a maximum number of evaluations of 50000, but this is not reached.

It can be seen that the SGA performs well on the MINLP Rastrigin problem seen in Figure 9.1a. The function value converges to the expected value of 125 within a reasonable level of computational effort. This behaviour is observed for all three seeds, making the convergence reliable. The SGA method is hereby considered verified for MINLP problems. This verification is extended to Non-Linear Program (NLP) problems when looking at the successful convergence of the SGA applied to the Ackley problem in Figure 9.1b. The Nelder-Mead Simplex method, contrary to the SGA, seems to converge to sub-optimal values for the MINLP problem. As mentioned at various points during this thesis, this is plausible as there are many local optima in the MINLP Rastrigin problem and Nelder-Mead Simplex is a gradient-based method that cannot escape a local optimum. One advantage, however, is the limited number of function evaluations necessary to converge. The different seeds converge to different optima, making the algorithm unreliable. As for the Ackley problem, the Nelder-Mead Simplex method does converge to 1e-14 with substantially fewer function evaluations than the SGA.

The main takeaway from this subsection is that the SGA method is verified in terms of optimising MINLP problems. This can be said for more than one reference problem, and with multiple seeds. The next subsection discusses the integrated verification of the LTTO, which shortly recaps the results found in Chapter 6.



(a) The MINLP Rastrigin problem with 5 continuous and discrete variables.



(b) The Ackley problem with 6 variables.

Figure 9.1: Comparison of optimisation using SGA and Nelder-Mead Simplex for two reference problems.

9.1.3. Integrated verification of LTTO

In this subsection, the integrated verification of the LTTO is presented. The goal is to recreate specific MGA sequences and compare them to the results found by [Fan et al. 2021]. These results use results from Chapter 6, so no new simulations need to be run.

To recap on the LTTO problem, in [Fan et al. 2021] several sequences are investigated: an EJ, EMJ, EEMJ, and EEEMJ transfer. The bounds were defined in Section 6.5 and are reiterated below in Table 9.2. The verification results are summarised in Tables 9.3, 9.4, 9.5 and 9.6. These include the optima found by the Bezier shape method, FFS method, and GPM method in [Fan et al. 2021] as well as the optima found in this thesis. The GPM method functions more as extra information, as this is a higher-fidelity step. The verification is more challenging as the reference quantities use a different method, however, some comparison can be made about the optimal values as well as their locations.

Table 9.2: Problem definition for transfers using parameters from [Fan et al. 2021].

Variable Name	Lower bound	Upper bound	Unit
Launch window	61420	63382	[MJD]
ToF	100	4500	days
Incoming velocity	0	5000	m/s
GA altitude	$2 \cdot 10^5$	$1 \cdot 10^9$	m
β	0	2π	rad
θ_g	0	2π	rad
ϕ_g	$-\frac{\pi}{2}$	$\frac{\pi}{2}$	rad
Free coefficients	$-1 \cdot 10^4$	$1 \cdot 10^4$	-
Number of revolutions	0	2	-

The tinted value originates from [Fan et al. 2021]. The ToF bounds were fixed manually according to Table 6.9.

Earth-Jupiter Transfer

To determine the utility of any gravity assist manoeuvre, the direct transfer must also be optimised, as seen below in Table 9.3.

In Table 9.3, it can be seen that the ΔV value is competitive, where the date is also within the same synodic period optimality window. Note that the departure date for all shape-based methods and all

sequences from [Fan et al. 2021] are identical; it is expected that the FFS and GPM methods are fixed to the optimal departure date that the Bezier method finds. For the EJ sequence, as was discussed in Section 6.3.3, the synodic period is about 399 days. Here a difference of 74 days is observed, which is not ideal but could be attributed to a difference in the ephemeris model or the different shape-based method. The difference in ΔV observed between Bezier, FFS, and hodographic shaping in this thesis is limited to 0.42 km/s. This is a 2% deviation from the ΔV values in [Fan et al. 2021], which is relatively accurate for low-fidelity shaping methods. When put into perspective with the difference in ΔV observed during the MGASO tuning process in Section 8.2 between different sequences, this 2% deviation is insignificant. During the MGASO tuning, the difference between the optimal sequences and the sub-optimal sequences started at a 43% increase in f_s – and by extension ΔV. This result shows that the implementation of the LTTO is verified for single-leg transfers. Next, MGA transfers are considered.

Table 9.3: ΔV values for optimised trajectories for an EJ transfer.

Source	Launch date [MJD]	ΔV [km/s]	Relative performance [%]
[Fan et al. 2021] Bezier	62654	17.81	-
[Fan et al. 2021] FFS	62654	17.90	-0.5
[Fan et al. 2021] GPM	62654	17.49	1.8
This thesis	62580	17.48	1.9

The relative performance is measured relative to the Bezier shape method developed by [Fan et al. 2021].

Earth-Mars-Jupiter Transfer

An analogously structured table is shown in Table 9.4 with the results for the EMJ transfer.

The ΔV values are once again competitive. A deviation can be observed of 0.8% for the Bezier shapes, which is not significant. The FFS method performs up to 10.6% worse than the LTTO in this thesis. The LTTO is therefore verified for a trajectory that includes a GA. Because the design complexity increases with every GA and the convergence will be more challenging with larger sequences, more verification is necessary. Therefore, the EEMJ and EEEMJ transfer performance is also compared. The optimal departure date only differs by 20 days, which is well within the 780-day Earth-Mars synodic period, and therefore is not expected to have a large contribution to the difference between models.

Table 9.4: ΔV values for optimised trajectories for an EMJ transfer.

Source	Launch date [MJD]	ΔV [km/s]	Relative performance [%]
[Fan et al. 2021] Bezier	61872	15.18	-
[Fan et al. 2021] FFS	61872	16.98	-10.6
[Fan et al. 2021] GPM	61872	14.83	2.4
This thesis	61892	15.30	-0.8

Earth-Earth-Mars-Jupiter Transfer

The EEMJ transfer results are shown in Table 9.5. In terms of ΔV the LTTO performs slightly worse than the reference value. Again the small deviation is negligible. The slight increase compared to the EMJ transfer could be that the optimum was found at a different departure date. As was discussed in Section 6.3.3, the EEMJ transfer in [Fan et al. 2021] has a different period of ΔV optima over the departure date window. The optimal departure dates are 517 days apart, which is substantially different from the 780-day deviation that one would expect. Nevertheless, this small difference in ΔV still shows that the LTTO implementation can produce competitive results. The LTTO performs 5% better than FFS, which is also a verified method for low-fidelity low-thrust trajectory design.

Table 9.5: ΔV values for optimised trajectories for an EEMJ transfer.

Source	Launch date [MJD]	ΔV [km/s]	Relative performance [%]
[Fan et al. 2021] Bezier	61452	14.95	-
[Fan et al. 2021] FFS	61452	16.30	-8.3
[Fan et al. 2021] GPM	61452	14.85	0.7
This thesis	61969	15.45	-3.2

Earth-Earth-Earth-Mars-Jupiter Transfer

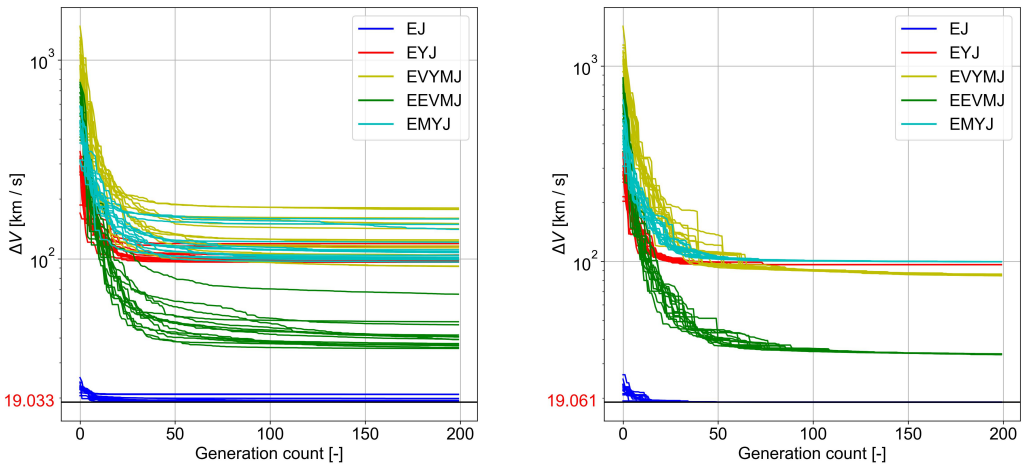
EEEMJ is the last transfer that is considered in [Fan et al. 2021]. Although the ΔV performance, seen in Table 9.6, has slightly decreased for these sequences, it may be attributed to the specific transfer. As was seen in Section 6.4, more suitable transfers can be found that have three GAs. This result is dependent on the departure date, which is to be expected. Specifically, a 9 % decrease is still well within the margin of 43% mentioned previously which is more important for the determination of the optimal sequences. The LTTO performs better than the FFS method, which is verified as mentioned before. With this, the LTTO is considered to be verified.

Table 9.6: ΔV values for optimised trajectories for an EEEMJ transfer.

Source	Launch date [MJD]	ΔV [km/s]	Relative performance [%]
[Fan et al. 2021] Bezier	61872	14.39	-
[Fan et al. 2021] FFS	61872	15.92	-9.6
[Fan et al. 2021] GPM	61872	14.32	0.5
This thesis	61854	15.83	-9.1

9.1.4. Custom topology

The custom topology is a UDT that only connects the islands of specific sequences together, which was introduced in Section 8.1.4. Here the correct functioning of the custom topology is verified against an optimisation without a topology.



(a) ΔV per generation for 1 recursion without topology.

(b) ΔV per generation for 1 recursion with 10% custom topology.

Figure 9.2: Comparison of custom topology vs no topology.

To verify the correct functioning of the custom topology, a selection of sequences is plotted with a custom topology and without a topology. A topology weight (tw) of 0.1 is assigned to exaggerate the differences. In Figure 9.2, it can be seen that the jumps are present only when the custom topology is added, resulting in a small difference between the mean and minimum ΔV value in Figure 9.2b as compared to Figure 9.2a. In addition, after observing the specific migrations that occurred, it was confirmed that individuals only exchanged between islands of the same sequence. The custom topology is therefore considered verified.

9.1.5. Verification sum-up

In the past subsections, several elements of the system have been verified. This subsection quickly revises what has been verified and why.

The verification is performed where possible. For several aspects, the verification was deemed too rudimentary, and for a different group of aspects, there was too little data to perform a reliable verification. To sum up, the hodographic-shaping implementation has been verified, as well as the optimisation methods used in this work, an integrated verification of the LTTO was performed, and the custom topology was verified. Besides these aspects, several other aspects can be considered verified without an explicit test. The flyby model is considered verified because the model implemented in Tudat is unit tested using the verified and validated GTOP Cassini 1 problem. The more fundamental elements of the methodology – using Tudat – such as the frame conversions are also considered verified as they are also unit tested. It is not possible to verify the MGASO implementation due to a lack of a perfectly suited reference problem. [Fan et al. 2021] was used as a reference problem, and [Fan et al. 2022] performed an MGASO on this problem, however, it remains unclear – even after inquiry with the authors – exactly how the optimisation is defined. Moreover, the outputs of the optimisation are not given, preventing any quantitative verification. Verification is of course only part of the task, the work and methods also have to comply with physical data, which is discussed in the next section on validation.

9.2. Validation

This section discusses the validation steps to prove the physical realism of the results to as high a degree as possible. While the entire methodology cannot be validated, due to a lack of real mission data, it can be ensured that the individually validated elements discussed here provide high confidence in terms of physical realism.

The hodographic-shaping method, for one, could be validated against space missions that have flown, for example, BepiColombo or Dawn. Real-world data is available via the SPICE[1] database on the state, orientation, and telemetry for the duration of the respective missions. To reliably validate the hodographic-shaping method, a transfer leg with continuous low-thrust propulsion is required where the state – full position, velocity, thrust magnitude and direction – and time are known. No thrust direction was found, which prevents the full validation using this dataset. Furthermore, the space missions that have flown with low thrust so far have been somewhat experimental, meaning that the ΔV optimality of the trajectories can not be guaranteed, and therefore the optimality of the hodographic-shaping legs would not be proven.

What can be said is that the hodographic-shaping method has been verified to produce similar results to other shaping methods, but also compared to software packages such as DITAN. These other packages have been validated. It is possible that the specific optimisation in this thesis does not apply to all cases, meaning that the comparison with DITAN is not a full validation. However, in absence of a direct way to validate, this provides the highest confidence.

The sequence optimisation process cannot be validated properly because there is no physical proof that any mission has flown the optimal sequence. The validation of the individual optimisation elements that are used in this thesis can be considered. The PyGMO library, which was developed by the Advanced Concepts Team (ACT) at the European Space Agency (ESA), has been validated using the Global Trajectory Optimisation Problem (GTOP) database. In addition, this software package has been used in countless other research activities. Therefore the methods that are used from PyGMO are considered validated. All the elements that are discussed in Sections 2.3.3 and 5.3 are included.

[1] " Spice Data (SPICE kernels) *Ron Baalke* Jun. 2023 " `https://naif.jpl.nasa.gov/naif/data.html`

This concludes the validation that can readily be performed on this approach. Having completed the verification and validation, which provides a relatively high confidence in the reliability and physical realism of the approach, the results can be presented.

10

Results

In this chapter, the results of the RTBA approach are presented. The approach is applied to the MGA sequence problem as defined in Section 6.1.1. The context within which the results are analysed is divided into three parts and presented explicitly. Subsequently, the parameters used for the RTBA are presented and the results are analysed. The results are split into the sorted sequences and RPS analysis, analogously to the analysis of the tuning results in Section 8.2.2.

10.1. Context for performance assessment

In the analysis of the results, multiple scopes are considered to determine their quality, robustness, and scientific significance. The scopes consist of three parts, and are discussed in the next three subsections: the results are discussed within the context of this thesis, within the context of verification concerning other MGASO methods, and when compared to other fidelity levels. These different scopes should be kept in mind when reading the analysis of the results in this chapter and also during the conclusions in Chapter 11.

10.1.1. Reliability within thesis bounds

The aspect discussed in this subsection is the main point of discussion; the methods by which it will be shown that the RTBA is robust and reliable in the context of this problem are presented. Moreover, with the problem definition and model choices that have been made for this thesis, the question arises as to whether the results hold any engineering value, and if so what value.

The robustness of two outputs needs to be examined: the MGA ranking and the RPS. For the former output, it is crucial that the general ranking of MGA sequences is the same if the same MGA sequences are evaluated. Of course, due to the nature of the approach, it is not always guaranteed that all the same sequences are evaluated for different seeds. However, it should be guaranteed that the highly optimal sequences are consistent across multiple seeds. For the latter output, ideally, there is a consistent convergence to one of the RPS strings that are expected to perform the best. What RPS is considered best was already introduced and shortly discussed during the tuning in Section 8.2, and will be elaborated upon in the results. In terms of computational performance, various parameter options are considered and compared, but most of the computational optimisation remains as a recommendation. This thesis is aimed at demonstrating the quality of the methodology and the concept, rather than optimising its performance.

If both outputs return results that are robust, consistent, and require limited computational resources, then this approach has significant engineering value as it can effectively and swiftly determine the MGA sequences that should be considered for a low-thrust interplanetary mission of up to three GAs. These results would also set a precedent for further development of MGASO applied to other low-thrust missions.

10.1.2. MGASO methods with similar fidelity

Another aspect that will be discussed in the results, and is useful to put the RTBA into perspective, is whether it is verifiable and comparable with other MGASO methods within the low-fidelity level shown in Figure 2.1.

During Chapter 6, a thorough tuning process was conducted to approach ΔV values that were found by [Fan et al. 2021], but also to obtain the most robust results with the implementation in this thesis. It was found that the ΔV values can be reproduced to a large extent, but because [Fan et al. 2021] provides no information on the optimality of the sequences that are evaluated, these results can not be used to verify the optimality of the MGASO, and thereby can not verify the RTBA. Even with the follow-up paper by the same author [Fan et al. 2022] that specifically analyses MGA sequences with the same LTTO method as in [Fan et al. 2021], a lack of results presented in the paper combined with a lack of transparency when contacting the author has made the results unusable. As a result, no independent verification of the MGASO can be done. However, the run times of the Bezier shape MGA optimisation with low thrust from [Fan et al. 2022] are given, and a limited comparison can be made despite the model differences (the inclusion of large asteroids in addition to the planets).

10.1.3. Comparison to other fidelity levels

This subsection describes how the results will be put into context with methods at other fidelity levels. The relevance and usability of the output for higher-fidelity methods are considered.

In this thesis, hodographic shaping is used. Recall that this is a shape-based method, which is a semi-analytical method. Semi-analytical methods are widely used throughout astrodynamics for trajectory design but have limited accuracy due to the many assumptions that are made: hodographic shaping only considers the two-body problem, for example. These low-fidelity methods are used in a preliminary phase of trajectory optimisation, especially when the MGA sequence has not been fixed yet, and the mission characteristics are flexible. Besides the LTTO problem, there are also uncertainties in the MGASO: the RTBA is a greedy approach that also includes chance, both of which carry uncertainties. One aspect that is therefore discussed in the next section is whether the output of the MGASO is robust enough such that it could be used as an input for high-fidelity methods, where only a single or very few sequences have to be evaluated. The next section presents the parameters used for the results.

10.2. Parameters for results

The results of this thesis are presented using the example that has been used throughout the thesis, which is based on [Fan et al. 2021]. The problem definition – defined in Section 6.1.1 – is used again for consistency but more importantly because of its completeness and relative simplicity.

From Section 8.2, it is known that the computational resources are limited by the constraints of using DelftBlue. In Section 8.2.2, it was concluded that the last recursion is not particularly useful. It was also previously found that the configuration used in Section 8.2.2 results in a large difference between the minimum and mean values using the same seed. This may be caused by gc, ps, fpc, ips, a lack of a topology, or a combination of these elements. Consequently, a selection of variables is reversed to the configuration in Section 8.2.1, which is computationally possible due to the removal of a recursion level. In particular, gc and ps are increased which increases the total complexity by a factor of four. A custom topology is added, which improves the robustness of the results as discussed in Section 8.1.4. An additional advantage of extra generations and individuals is more potential for the topology to migrate individuals. Furthermore, three seeds are evaluated rather than one. The seeds mean that the sequences are generated using different reproducible sets of pseudo-random numbers which translate into a different selection of MGA sequences to evaluate. This allows for a more accurate assessment of the robustness of the methodology. Ideally, more seeds would be evaluated to increase the statistical confidence, however, this is left as a future recommendation due to time constraints. These changes are summarised in Table 10.1.

From the tuning process in Section 8.2, the optimal choice for the fp and fpi parameters has not been made yet. Some recommendations are made based on the tuning results, which allows for a small comparison of various parameter combinations. Specifically, the fp and fpi parameters are evaluated at 0.5 and 1, because it was shown that a value of 0.0 is unlikely to provide the most consistent RPS.

Table 10.1: Parameter settings for the final results.

Parameter type	Value
gc	300
ps	1200
ips	14
CPUs	42
fpc	1
tw	0.01
Seed 1 per recursion	[123978, 123979]
Seed 2 per recursion	[26604, 26605]
Seed 3 per recursion	[12, 13]

This was due to the inconsistency of the results if only the mean is used as fitness. Finally, q is evaluated at 0.1, 0.3, and 0.5. Note that these values are comparable to the q values from the tuning process, with a maximum difference of 3%, which results in a maximum of one sequence extra per PTB. This concludes the introduction to the results, the next section presents and analyses the sorted sequences.

10.3. Sorted sequences

This section comprises the analysis of the sorted sequences that are output of the RTBA.

10.3.1. Figure contents

Similarly to the MGASO tuning results, the results are sorted by fitness: in each figure with sequence rankings, the lower left is considered a more optimal sequence, and the upper right is less optimal. The two blue colours represent an fpi of 0.5. To shortly reiterate, this implies that the minimum and mean f_s sequence for each PTB are both given a weight of 0.5 – according to Equation (7.3). The grey colours represent an fpi of 1.0, where only the minimum f_s sequence is used. The darker tints of each colour represent an fp of 0.5 which – according to Equation (7.2) – means that the mean and minimum ΔV of all islands are given a respective weight of 0.5. Analogously, the lighter tints of each colour correspond to a minimum-only ΔV of each sequence. Three markers are used to depict the results for each individual seed. The y-axis is shown logarithmically, to show the differences in f_s more clearly.

The analysis of the sorted sequences is split into the general trends and results that can be seen across all sequences on the one hand, and the specific differences of the low-f_s group sequences on the other hand. The former is discussed first, followed by the latter later on.

10.3.2. General trends

The general trends that are observed in Figure 10.1 are split into various aspects, discussed below.

Higher number of sequences

One key observation that is valid for all sub-figures is that the number of sequences plotted is higher than one would expect. One would expect 9 sequences with a combinatorial complexity of 85 and a q of 0.1, for example. One reason for this is that the figures include the sequences evaluated for all grid points, and because the fp and fpi values can influence the RPS, a different set of sequences can be evaluated despite having the same seed. The other reason is that q is defined for one recursion, but there are two recursions, which results in the evaluation of more sequences. Specifically, the remaining complexity is calculated and the same fraction of that space is evaluated. Note as well that every sequence is unique, as was seen in Section 8.1.2, so no overlap is possible resulting in lower fractions of evaluated sequences. The total number of evaluated sequences in each sub-figure is 29, 60, and 75 for a q of 0.1, 0.3, and 0.5, respectively. Per run, however, 14, 30, and 50 sequences are evaluated. This difference is to be expected as a wide variety of RPS strings are found in Section 10.4 which all result in a higher probability of finding unique sequences. For context, the total combinatorial complexity

using Equation (7.1) is 85 and 21 for the first and second recursion, respectively. The total fraction (Q) then becomes 0.16, 0.35, and 0.58 for all q values, respectively. To recap, a complexity of 85 means that 85 possible MGA sequences could be evaluated and Q was the cumulative fraction of sequences evaluated after all recursions.

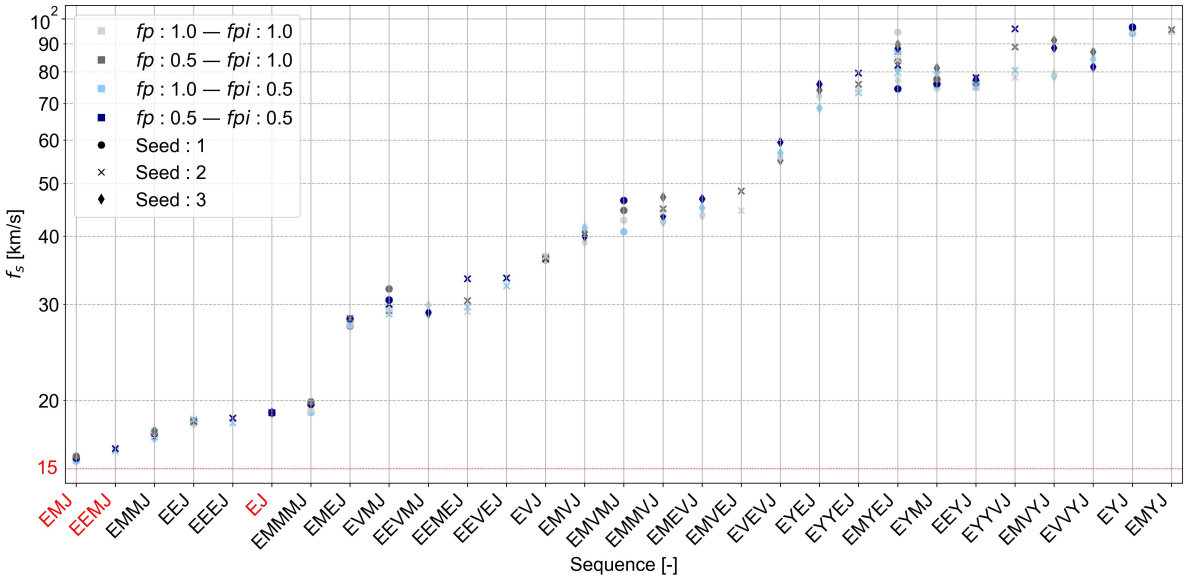
Minimum f_s
The minimum f_s observed is approximately 15.4 km/s for the EMJ transfer for all q values but not for all seeds. This value is in agreement with the tuned ΔV values found in Chapter 6, which is to be expected as the optimisation is identical. Moreover, the values are effectively equal across all fp, fpi, and q values. This means that the minimum and mean f_s are almost identical, and thus all islands have found similar results. In the MGASO tuning from Section 8.2.2, where EMJ was only evaluated for $q = 0.30$ and $q = 0.53$, the f_s values across all grid points have a substantially higher spread than in Figure 10.1. Despite the higher spread, EMJ consistently produces the lowest f_s values in the tuning results as well as the final results, which shows that EMJ is a highly optimal sequence.

Maximum f_s
Regarding the maximum f_s values observed at the right end of Figure 10.1, the f_s differs from roughly 94 km/s for $q = 0.1$ to 166 km/s for $q = 0.5$. The main cause is that the highest f_s sequence is different for $q = 0.1$ than for the other fractions. When comparing the EMYJ transfer, which is the highest f_s transfer for $q = 0.1$, the f_s values are close together with a spread of around 5 km/s. The spread is discussed more in the next paragraph. It is expected that a higher q value results in the evaluation of more sub-optimal trajectories. This 'wasted' computational time is mostly spent in the first recursion, assuming that the RPS converges to an optimal sequence. A remedy for this 'waste' could be to define q differently such that the number of sequences evaluated in one recursion is above some minimum number, which would reduce the number of sequences in the first recursion relatively compared to the subsequent recursions.

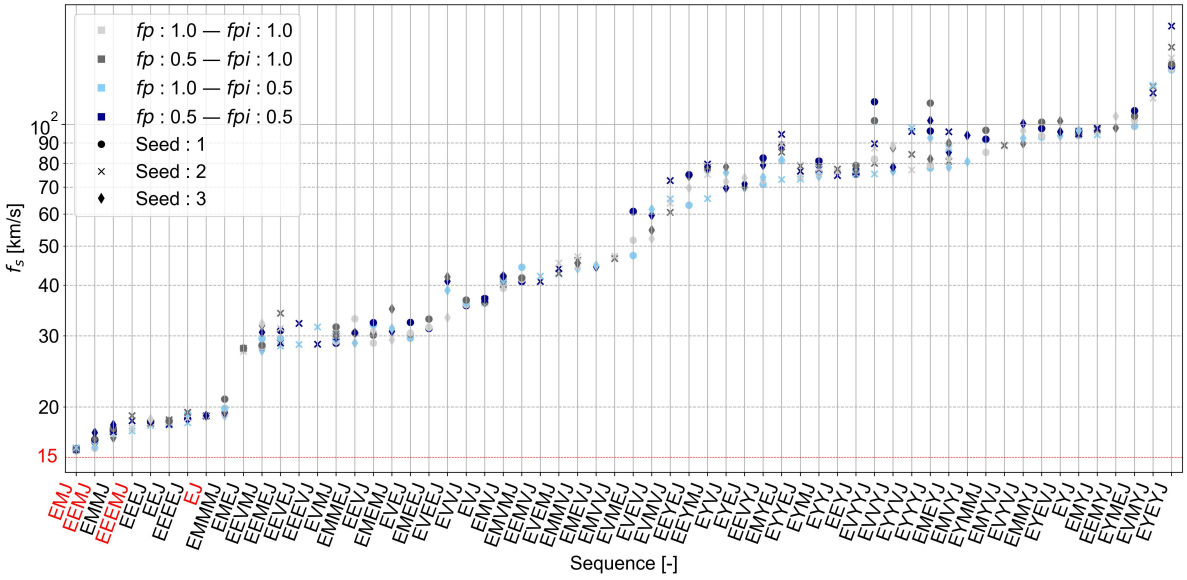
In Section 8.2.2, the highest f_s sequences were EVMYJ and EYYYJ with a maximum of 170 and 250 km/s, respectively. In Figure 10.1c, where the fraction is comparable to the tuning results, EVMYJ is also the second highest f_s at 127 km/s, which is a 25 % decrease in f_s. EYYYJ performed substantially better compared to the tuning result with a f_s of 94 km/s, which is an improvement of 63%. Note that these values are for different grid points – fp and fpi combinations. Furthermore, the spread is smaller than the difference in f_s between tuning results and final results, so the addition of the custom topology and improvement of robustness in terms of gc and ps is significant. Unfortunately, there is no easy way to distinguish if the improvement of any result is caused by specifically the topology or the change in parameters.

Spread of results
The spread of f_s values across all the runs for a single sequence is considered next. To reiterate, the spread here is the difference between the maximum and minimum f_s values. The spread is relevant, as it visualises if different parameters or different seeds affect the overall fitness f_s, which in turn influences the robustness of the optimisation. As can be seen in Figure 10.1c, the spread of some sequences is more heavily affected than others, as was already touched on in Section 8.2.2. However, a key difference in these results is the addition of the custom topology, which is noticeable by the reduced spread of most sequences in Figure 10.1 compared to Figure 8.4. Consequently, there is a more clear ranking of MGA sequences: the sequences overlap significantly less in terms of f_s value, which is particularly true for the low-f_s group of sequences, discussed shortly.

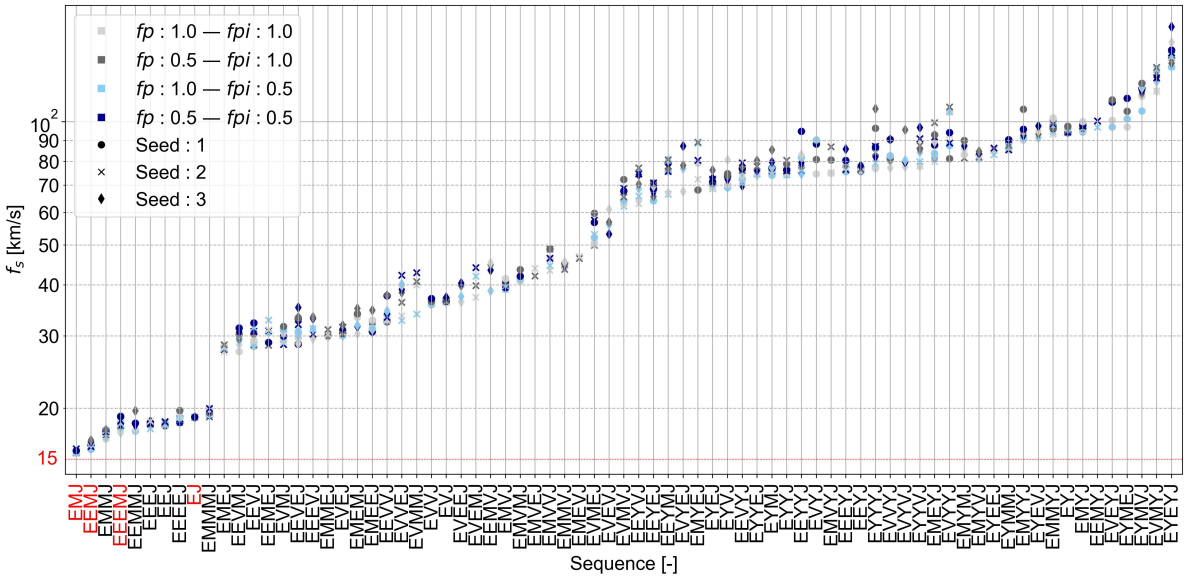
The spread that is still present for all sequences is caused by the difference between mean and minimum ΔV values, which is visualised by the various fp values. The $fp = 1.0$ runs still generally find lower f_s values than the $fp = 0.5$ variants, which was observed for all sequences in Section 8.2.2, but this is less pronounced in Figure 10.1. Due to the addition of the custom topology, the mean and minimum are substantially closer together. If there is an uncertainty of a few km/s in the LTTO for the same parameters with the same seed, the mean ΔV may be lower than the minimum ΔV in two subsequent runs. For the fpi, there is no additional difference, other than that some sequences were evaluated only with one fpi value, resulting in a low spread across fpi values.



(a) $q = 0.1$, $Q = 0.16$



(b) $q = 0.3$, $Q = 0.35$



(c) $q = 0.5$, $Q = 0.58$

Figure 10.1: MGA sequences sorted by f_s, for three different q values and for three different seeds.

An important trend is that the spread increases with f_s value. It is not immediately evident due to the logarithmic scale. In absolute values, the spread grows from about 1 km/s at low-f_s sequences to the order of 10 to 30 km/s at high-f_s sequences across all q values. This trend can be explained by the fact that the configuration of the LTTO optimisation was tuned for specific transfers. As a result, the sequences that differ greatly from the type of transfer observed in Chapter 6 are less well-tuned and may have a harder time converging. For example, the magnitude of the incoming velocity of each GA may be severely different for transfers to inner planets compared to outer planets. The same holds for the revolution count and ToF among others, resulting in the worse convergence. With more extensive tuning and further development of dynamic bounds, for example, this increase in spread could have been decreased. The dependency on the type of tuning is discussed more later in this section. In a large design space such as the one considered for this test case with many non-linearities and few local optima, a single component of the design variable vector can cause substantial degradation of optimised fitness values. The design space is unforgiving and consequently, the islands converge to different ΔV values, resulting in means and minima that are spread out despite the custom topology and increased ps and gc.

While one could expect a strong correlation between the uncertainty and the length of the sequence because the sequence length has a severe impact on the design complexity and therefore the ability to converge, the correlation in Figure 10.1 is less pronounced. This is in contrast to the results from Section 8.2.2, where there was a clear and strong correlation. This result shows that the combination of custom topology with various increases in the parameters – namely an increase in ps and gc – seems to contribute to a more independent optimisation across all possible sequence lengths. This result is useful as it shows the robustness of the methodology against differences in sequence length. However, it is still expected that for even longer sequences the design complexity may still become problematic for the convergence.

Jumps in f_s

A phenomenon that is observable across all q values both here and in Section 8.2.2 is the severe jump in f_s around 20 km/s. This jump is caused by the sequences. In particular, when looking at the occurrence of certain GA targets, there are obvious and distinguishable trends. For one, the sequences below this jump in terms of f_s all only have specific sub-sequences that were recommended from Section 8.2.2. To recap, these sub-sequences were 'MM', 'EM', and 'EE'. All other sequences (above the jump) do not exclusively use these sub-sequences. The transfers above the jump either include sub-optimal sub-sequences consisting of Earth and Mars – the 'ME' sub-sequence mainly – or the transfers include Venus or Mercury. Moreover, Venus is observed in the middle of the sorted sequences and Mercury is observed at the right of the sorted sequences in Figure 10.1. This trend is clear in all three sub-figures and was also, to a lesser extent, already observable in Figure 8.3. The optimality of the recommended sub-sequences was already argued previously. Furthermore, the peculiarity of Venus and Mercury was shortly touched on, however, more analysis is required.

This trend in GA candidate occurrence can be explained from a theoretical point of view: Venus and Mercury are the only two planets on the 'inner' side of Earth in radial distance from the Sun. From astrodynamics, it is known that the heliocentric angular momentum has to be decreased to reach the inner planets. This means that the spacecraft is thrusting to reduce its angular momentum as an investment to perform a GA on an inner planet and subsequently gain enough angular momentum to reach Jupiter with minimal ΔV expenditure. This combination is therefore expected to be a less attractive region within the design space compared to the consistent increase in angular momentum to outer planet GA targets. In addition, a GA is more effective with bodies that have a higher mass. Mercury has only 5.5% times the mass of Earth and Venus has 81%. This makes Mercury especially unattractive as a candidate. It should be noted that Mars also only has 10% times the mass of Earth, which also decreases its suitability. These reasons would indicate that transfers including the sub-sequences that were recommended are indeed the more optimal ones; Mars is found regularly in the lowest f_s sequences, indicating that its position in the Solar System has more impact than its relatively low mass. Some evidence, showcased by JUICE, for example, has shown that a Venus GA is a viable way of reaching Jupiter.

As was previously mentioned, the approach in this thesis was tuned for the sequences EJ, EMJ, EEMJ, EEEMJ, based on [Fan et al. 2021]. This resulted in the assumption of identical shaping functions that

[Gondelach and Noomen 2015] found to be optimal for an EM transfer, for example. Moreover, shaping coefficients for EY transfers were found of up to 10^{11}, whereas the coefficients in the LTTO are bound by a magnitude of 10^4. This tuning choice affects the potential for the shapes to recreate optimal low-thrust trajectories when headed toward inner planets. In addition, a key difference between transfers to inner and outer planets is generally the ToF and revolution count. For inner planet transfers from Earth, the ToF is generally significantly lower, and the revolution count is substantially higher. The time of flight bound was quite broad – [20, 4000] days – which should be less of an issue, however, the maximum of two revolutions may cause issues for low-thrust transfers to inner planets. The angular rate also increases which affects the number of revolutions per ToF, because the orbital periods decrease as one approaches the Sun. Because low thrust provides the same thrust independent of the nature of the transfer, this generally requires more time and thus more revolutions are needed for inner transfers. During the tuning process, the maximum observed revolution count for highly fit individuals was two, however, this is not a particularly robust choice for a general-purpose MGA sequencing optimisation. The further investigation of revolution count bounds remains as a recommendation, discussed in Chapter 12. Due to time constraints, the approach could not be tested on the same test case without Mercury. If such a profound discrepancy between the ranking including and excluding Mercury would still be observed, then as a further recommendation one could add a pruning feature that removes Mercury – or whatever body it would be in a more general case – from the design space in subsequent recursions. This is discussed more in Chapter 12. Ideally, these trends and results in general should be fully verified and validated. However, due to a lack of physical data, there is no method to prove whether these trends in GA targets are correct or merely a product of the specific tuning. The theory as well as the verification and validation steps discussed in Chapter 9 do support the observations made. As an additional note, these optimal sequences would likely change with a different departure date bound. This variable has an enormous effect on the optimality of certain bodies due to phasing. The departure date window is 60 days, so it is possible that the Venus GA, for example, is more optimal for a different bound. Due to time constraints, this possible explanation is not further investigated.

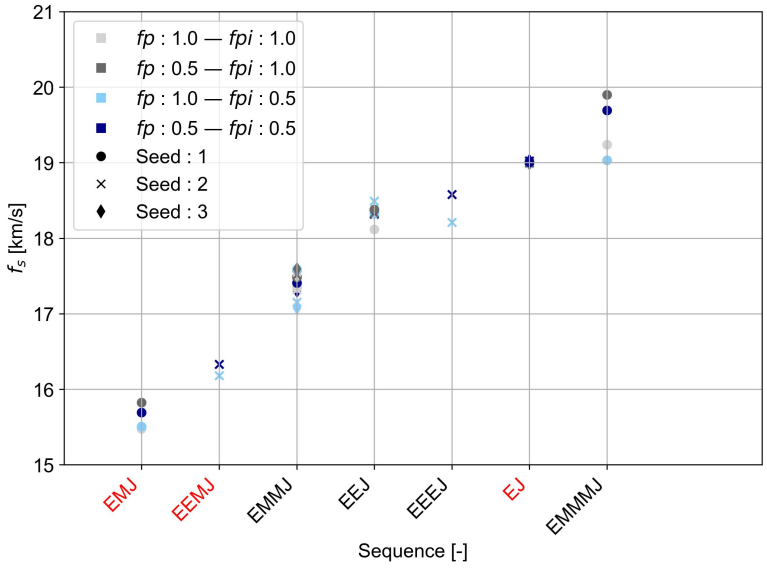
Focus on best sequences

When analysing sequences with higher fidelity, the computational cost increases because more dynamics are considered which results in the numerical integration including many terms. As input to these higher-fidelity methods, only a small selection of the most optimal sequences is considered. For the RTBA, it is therefore necessary to determine whether a selection can be made of those sequences and whether this can be distinguished from all the other sequences. A group can be observed in all figures, which was also observed in Section 8.2.2 and defined explicitly. To reiterate, this group is defined by the sequences that have a minimum f_s that is within 30% or 5 km/s of the overall lowest f_s. This group shows significantly lower f_s values at all three q values. Moreover, this group also remains consistent across all the grid points: the change across the various seeds, fp values, and fpi values is small enough that the sequences in this low-f_s group all are lower than the lowest f_s value of the sequences that are not included in this group. The group varies slightly in size depending on the q value, which is plausible as more sequences are evaluated. As a side note, the red sequences are those evaluated by [Fan et al. 2021]. Besides the omission of EEEMJ in the $q = 0.1$ run, all the sequences from [Fan et al. 2021] are evaluated and are found to be part of this low-f_s group. This group is further considered, and the results are shown explicitly in Figure 10.2 below.

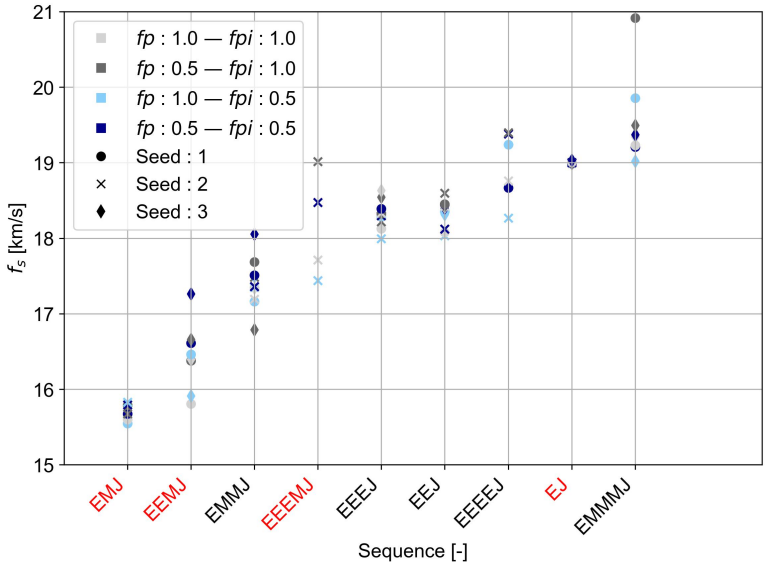
10.3.3. Low-f_s group analysis

This subsection looks specifically at the low-f_s group of sequences. These sequences are found to be most useful from an engineering perspective and are therefore subjected to additional analysis.

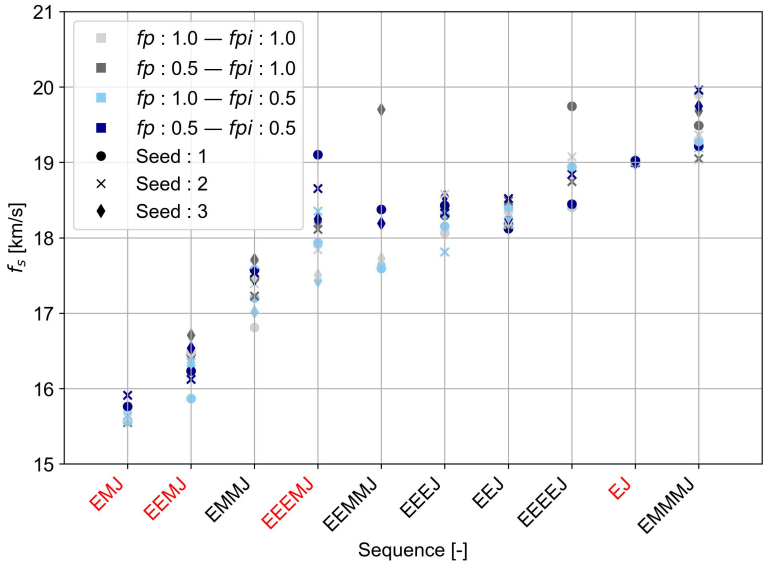
The low-f_s group is first compared to the tuning, after which more specific observations are discussed. In comparison to the tuning from Section 8.2.2, more sequences have entered the low-f_s group using the definition defined previously. For the high q value in the tuning process, nine sequences were found across all grid points compared to the 10 in Figure 10.2c, but this is not the full picture. All grid points in the results presented here adhere to the requirements for the low-f_s group. By contrast, not all grid points adhered to the low-f_s group in the tuning steps, only the lowest-f_s grid points did. This is explained following the same logic as before: the addition of the custom topology and increase of gc and ps.



(a) $q = 0.1$



(b) $q = 0.3$



(c) $q = 0.5$

Figure 10.2: Sorted top 10 sequences of three different q values for three different seeds.

Ultimately, the fpi, fp, and q parameters are assessed within the definition of the problem in this thesis. To achieve this, the robustness and reliability per parameter should be analysed. Therefore, in Table 10.2, the number of seeds are shown that evaluate a certain sequence per grid point. The number of seeds that evaluate certain sequences gives a good indication of the consistency with which a certain parameter combination will find the sequences that have been shown to perform best.

Table 10.2: Sequence evaluation statistics for various q values using three seeds. At each grid point, the number of seeds that evaluate each sequence in the low-f_s group are given. The sequences are sorted by f_s value of the $q = 0.5$ run from left to right.

(a) $q = 0.1$.

fpi	fp	EMJ	EEMJ	EMMJ	EEEMJ	EEMMJ	EEEJ	EEJ	EEEEJ	EJ	EMMMJ
1.0	1.0	1	0	0	0	3	0	2	0	3	1
	0.5	1	0	0	0	3	0	2	0	3	1
0.5	1.0	1	1	0	0	3	1	2	0	3	1
	0.5	1	1	0	0	3	1	2	0	3	1

(b) $q = 0.3$.

fpi	fp	EMJ	EEMJ	EMMJ	EEEMJ	EEMMJ	EEEJ	EEJ	EEEEJ	EJ	EMMMJ
1.0	1.0	2	2	3	0	1	3	3	1	3	2
	0.5	2	2	3	0	1	3	3	1	3	2
0.5	1.0	2	2	3	0	1	2	3	2	3	2
	0.5	2	2	3	0	1	2	3	2	3	2

(c) $q = 0.5$.

fpi	fp	EMJ	EEMJ	EMMJ	EEEMJ	EEMMJ	EEEJ	EEJ	EEEEJ	EJ	EMMMJ
1.0	1.0	2	2	3	2	1	3	3	2	3	3
	0.5	2	2	3	2	1	3	3	2	3	3
0.5	1.0	2	3	3	3	2	3	3	2	3	3
	0.5	2	3	3	3	2	3	3	2	3	3

Across the various q values, as one would expect, the number of occurrences increases with increasing q. The average number of seeds that are evaluated is 1.1, 2.0, 2.55 for a q of 0.1, 0.3, and 0.5 respectively. Ideally, one wants an average of 3.0, however, this is statistically challenging to achieve with a Monte-Carlo based approach. The choice for an optimal q value is a trade-off of computational complexity and consistency of results. If the top five sequences are prioritised – as this may be the number of sequences that are passed on to a higher-fidelity method – and a q is selected based on this, it seems worthwhile to fix q to 0.5. However, an interesting result is that the $q = 0.1$ runs already find seven out of 10 sequences across all seeds, with only a fraction of the computation time. This is probably caused by a similar choice of RPS despite fewer sequences, which is discussed more in Section 10.4. Less computational power is spent on evaluating sequences in the first recursion that are going to find high ΔV sequences. Despite a higher relative performance across all grid points, only two sequences were found in all seeds. As a result, this q value is currently not reliable for discovering the low-f_s group of sequences. If only two out of three seeds evaluate certain sequences, then this methodology would have to be run more than once to be confident in finding the optimal MGA sequences, which is ineffective. The core of the problem lies in how many sequences are evaluated in a certain recursion. As it is implemented currently, the q is applied to all recursions equally, resulting in later recursions with smaller combinatorial complexities that still only evaluate a fraction of the space. This results in a lower confidence for further exploration and RPS determination. For example, with $q = 0.5$, there is by definition a 50% chance of not finding the minimum f_s PTB in the second recursion – assuming the first RPS GA is optimal. This contributes significantly to part of the f_s, resulting in an incorrect convergence of the RPS. One solution would be to redefine q in a way that balances out the computational effort across the recursions such that the

later recursions have a higher chance of finding the optimal sequences. As an additional note, this problem occurs in the later recursions, which scales with the complexity of the problem. If more bodies would be considered with more GAs, then multiple recursions will have a higher confidence, before deteriorating. Another solution is to fix fpi to 0.5, so that the minimum f_s does not have to be found for the RPS determination to converge properly. This preference is discussed more shortly.

Looking at various fp in Table 10.2 for all q values, there is no difference in the number of seeds evaluating each sequence. This indicates that the mean and minimum ΔV of each island are similar, resulting in the same choices for the RPS independently of the fpi value. Moreover, the equivalency can be attributed to the custom topology and added generations and individuals decreasing the difference between the minimum and mean of a single island. This result shows that the fp, using the configuration as it is given in Section 10.2, can be chosen such that it provides the most optimal ΔV, which in this case is $fp = 1.0$ by definition.

The differences in the fpi values are present yet nuanced. The $fpi = 0.5$ runs show a slight increase in the number of seeds observed. Specifically, there are five cases spread across all q values where an $fpi = 0.5$ run found one low f_s sequence more than the $fpi = 1.0$ run. This result shows that an fpi value of 0.5 is better than 1.0. However, this is a coarse grid with only two points. One could experiment with other values, which is left as a further recommendation due to time constraints.

Another observation is that the lowest f_s sequence, EMJ, according to this LTTO procedure across all grid points and q values is only observed in two of the three seeds. It is found that, due to the design of the RPS, only the first recursion allows for a single GA transfer to be evaluated. The sequences are built up starting from the departure planet and subsequently adding the RPS, the PTB, and a random sequence of a random length. Because the PTB is always added, evaluating EMJ after the first recursion becomes an impossibility, even if the best PTB that gets appended to the RPS is Mars. This artefact should be removed in future work.

This concludes the analysis of the results for the sorted sequences. In the next section, the RPS strings of these simulations are discussed together with the computational load.

10.4. RPS analysis

This section discusses the analysis of the quality of the RTBA by assessing the convergence of the RPS. Similar to the previous analysis, the trends and general recommendations for parameters are given first followed by some specific observations.

10.4.1. General trends

The recommended RPS sub-strings were 'EE', 'EM', and 'MM'. Looking at Table 10.3, only these RPS strings are found, which is a promising outcome. This shows the correct functioning of the RPS determination in terms of convergence for all q values: Venus and Mercury are not seen in any RPS string. These results are a slight improvement compared to the tuning results from Section 8.2.2, where an 'EVM' RPS was observed. In terms of consistency across all parameters – q, fp, and fpi – there is no apparent improvement. This is explained by the fact that the addition of the custom topology and increased gc and ps contribute mostly to the difference between the minimum and mean ΔV for each sequence and not to how the RPS is determined. Therefore, the RPS is mostly determined by the sequences that are found in the Monte-Carlo search. As was seen in the previous section, the q values did not find all low-f_s MGA sequences across all seeds, which explains in part the remaining inconsistency of the RPS strings.

Similar to the previous section, all parameters are analysed in turn. For the q values, there is a decrease in the number of 'MM' RPS strings, from five to three to two for q values of 0.1, 0.3, and 0.5, respectively. Observing fewer 'MM' sub-sequences could indicate a better result, as it can be seen in Figure 10.2c that only two of the 10 transfers have MM as the first two GAs. However, EM is also present twice, and EE is present three times. This distribution shows that there is no single best RPS, and that therefore a greedy approach is potentially detrimental to finding all low-f_s sequences. With further development, the RPS can also be parallelised, where a small selection of seemingly optimal sequences are split into various RPS processes. This would allow for the various RPS values that are considered optimal here to be developed further.

Regarding fp, similar to the tuning process, there is no difference in RPS, which is to be expected for the same reasons mentioned previously on the small difference between the minimum and the mean of each island. The fpi does show some difference: for $fpi = 0.5$, three seeds across all q values find 'MM', whereas $fpi = 1.0$ finds seven. The $fpi = 0.5$ was previously shown to be the better candidate with slightly more low-f_s sequences that were found. Here the decrease in 'MM' sub-sequences can be seen as a slight improvement. The occurrence of 'EE' and 'EM' is seemingly random. To restate the tentative recommendations from the sequence ranking analysis: an fp of 1.0 seems best together with an fpi of 0.5. Moreover, there was no decisive outcome regarding the q value, although $q = 0.1$ is shown not to be reliable enough. This recommendation is in agreement with the findings when observing the RPS strings, though the differences are nuanced and no configuration was significantly better. Next, some more detailed examples are presented as to why the RPS converges to certain values.

		$q = 0.1$		$q = 0.3$		$q = 0.5$	
		$fpi = 1.0$	$fpi = 0.5$	$fpi = 1.0$	$fpi = 0.5$	$fpi = 1.0$	$fpi = 0.5$
Seed 1	$fp = 1.0$	MM	MM	MM	EE	MM	EM
	$fp = 0.5$	MM	MM	MM	EE	MM	EM
Seed 2	$fp = 1.0$	MM	EE	MM	EE	MM	EE
	$fp = 0.5$	MM	EE	MM	EE	MM	EE
Seed 3	$fp = 1.0$	MM	MM	EM	MM	EM	EM
	$fp = 0.5$	MM	MM	EM	MM	EM	EM

Table 10.3: RPS strings for three seeds, three q values, and two fp and fpi values.

10.4.2. Specific observations

One observation from Table 10.2 is that EEMMJ is exclusively evaluated in the $q = 0.5$ runs. This is peculiar, as it is seen in Table 10.3 that there are two runs with $q = 0.3$ that converge to EM, and still EEMMJ is not found. One consequence of the reduction to two recursions is that when two RPS GAs have been fixed, no further evaluation is done by definition. As a result, the occurrence of optimal trajectories is decreased significantly. In other words, even though all RPS strings seem to converge to sequences that are in the low-f_s group, the trajectories that use the RPS string are not evaluated. Only the first RPS GA is used for the selection of sequences in the second recursion. In future work, three recursions should be investigated further, without the constraints of DelftBlue. To reiterate, the choice for two recursions was made due to the maximum run-time constraints set by DelftBlue.

Take the $fp = 1.0$, $fpi = 1.0$, and $q = 0.1$ case, which only considers the minimum. Seed 1 finds EMJ in the first recursion, and EMMJ in the second recursion as minimum ΔV sequences, which results in the 'MM' RPS. For seed 2, EMMJ is the minimum ΔV sequence that is passed to the second iteration. No better sequence is found and thus EMMJ determines the entire RPS. This shows that for parameter values of 1.0, the choice is not balanced. Although, within the trajectories that the RTBA does find, EMMJ is the best. In terms of finding low-f_s group sequences the RTBA works, but this is left to chance for this configuration. Recall that only 14 sequences are evaluated in total, 9 of which are in the first recursion. For comparison, take the $fp = 1.0$, $fpi = 1.0$, and $q = 0.5$ case in Table 10.3. Here, for seed 3, the RPS is 'EM', rather than the 'MM' sub-sequence observed for all other q values and seeds. EMJ was not evaluated for this run despite 42 sequences being evaluated. The choice for Earth in the EM sequence is therefore still coincidental, showing again that $fpi = 1.0$ and $q = 0.1$ are not reliable settings for determining RPS strings.

Now a case is analysed using $fp = 0.5$, $fpi = 0.5$, and $q = 0.1$ during the first recursion. The trajectories that contributed to the RPS in this case are tabulated in Table 10.4. Across all seeds, the decision-making process for the best PTB is based on only one or two sequences, which is statistically insignificant compared to the total complexity of 85. It can be seen that even in the first recursion, only two sequences are evaluated per PTB. The RPS addition is then determined by the PTB with the fewest occurrences of 'Y' or 'V'. This dependency on the number of sub-optimal PTB candidates decreases with increasing q. In Table 10.4, for seed 1 for example, one can see that the Earth PTB finds a sequence with 'Y', resulting in a significantly higher mean ΔV, leading to 'M' being the optimal PTB in this run. By contrast,

Table 10.4: Transfers and f_s of Earth and Mars transfers for the first recursion of the case with $q = 0.1$, $fp = 0.5$, and $fpi = 0.5$.

PTB [-]	Seed 1		Seed 2		Seed 3	
	Transfer evaluated [-]	f_s [km/s]	Transfer evaluated [-]	f_s [km/s]	Transfer evaluated [-]	f_s [km/s]
M	EMJ	15.7	EMMJ	17.5	EMMJ	17.3
	EMMMJ	19.7	EMYEJ	82.3	EMVJ	40.0
E	EEJ	18.4	EEJ	18.3	EEYJ	77.5
	EEYJ	76.9	EEMEJ	33.4	EEVMJ	29.0
RPS	MM		EE		MM	

in seed 2 the Mars PTB finds a sequence with 'Y', resulting in the convergence toward 'E'. Similar to seed 1, in seed 3 Mars is found as the optimal PTB. Here, both PTB candidates also find a sequence with 'V', but this is insignificant in the presence of a 'Y' transfer. This q level is clearly not robust, reliable, or consistent. To compare with other q values, for the same test case, an analogous analysis is done. 10 sequences are evaluated per PTB, which is already significantly better statistically, however, the Earth PTB only evaluates one 'Y' transfer, compared to the four 'Y' transfers evaluated by Mars, resulting in the 'E' addition to the RPS in Table 10.3. It is not attractive to evaluate more sequences with q values higher than 0.5 because a full enumeration is then approached, which is not an efficient MGASO. Consequently, a crucial improvement needs to be the real-time pruning of GA candidates that are shown to be sub-optimal on multiple occasions. This improvement can be applied to expansions of this approach to the optimisation of asteroid flyby missions, for example. For the most robust results, in the current implementation, a q of 0.5 is required. However, one can also take a substantially lower q and then run the optimisation several times.

To summarise, the RPS strings are quite optimal, however, there is still room for improvement: the RPS strings are not consistent when using the current implementation of the RTBA, but the direction of convergence is correct for all RPS strings. Next, the computational complexity is presented for these results. Due to time constraints, no further iterations of grid searches were tested. There are a plethora of possible improvements, which are discussed in Chapter 12.

10.5. Computational characteristics

An essential part of this approach is also the computational performance on top of the robustness within the bounds of the problem. Several useful quantities are shown per q value in Table 10.5 to show what q value might be best from a computational point of view. The run-time difference between the fp and fpi grid points is negligible. First, the run time is given, which indicates the time it takes for the complete MGASO to be executed. This quantity is useful as it indicates the time it takes to do one iteration of this problem and how it may affect a mission analysis. However, the run-time relevance is limited because one can add more and more cores, which requires more resources. These resources are often limited, therefore the efficiency should be maximised as well, which is depicted by the average CPU efficiency as the second quantity. The efficiency is defined by Equation (10.1).

$$\eta = \frac{ct}{rt \cdot nc} \tag{10.1}$$

Here, η is the efficiency, ct is the CPU time, rt is the run time, and nc is the number of CPUs. Last, the memory utilisation is tabulated to show how intensive this method is for the RAM. The memory usage is given as the total amount of RAM used, and not per CPU. These quantities are not given in literature, preventing a comparison of the performance, but they are useful nevertheless as a stand-alone quantity.

The run times increase significantly, which is to be expected. Recall the 14, 30, and 50 sequences that are evaluated per grid point per q value. The run time is not exactly proportional, but comes reasonably close: from $q = 0.1$ to $q = 0.3$, there is a 214% increase in sequences evaluated, compared to a 260% increase in run time. This is different to the step from $q = 0.3$ to $q = 0.5$, where there is a 167% increase

Table 10.5: Simulation characteristics per q value.

q [-]	Average run time [hh:mm]	Average η [%]	Average Memory utilised [GB]
0.1	05:00	70	6.80
0.3	13:00	80	7.50
0.5	21:30	85	8.15

The hh:mm unit stands for hours and minutes. The average run time is rounded to the nearest half hour for clarity. The memory used is specifically the RAM usage across all cores, not the disk space.

in sequences evaluated, compared to a 165% increase in run time. This difference can be explained by the fact that there is a relatively longer initialisation period. Moreover, the higher the q, the more time the parallel mechanisms are working, relatively. This is further confirmed by the increased CPU efficiency, which increases noticeably for higher q values. The final aspect is memory usage. Using DelftBlue, there was some strange behaviour that sometimes resulted in memory usage of 24 kB. It is assumed that this value does not represent the actual memory usage. Therefore these runs are not taken into account for the average. On the one hand, the marginal increase in memory usage is peculiar, because the number of islands increases by a factor of more than two for increased q values. The memory usage increase is only 20%. On the other hand, the approach is quite complicated and not optimised for memory usage. This means that the initialisation of the RTBA is memory intensive, irrespective of the q value.

When considering the computational complexity, the run time is the most indicative, as this quantity is used in literature: the CPU efficiency can be improved through the specific implementation of the parallelisation and the memory usage can be improved by optimising the code. The run time, however, is mostly dependent on the RTBA configuration. The computational complexity of the most complex case – $q = 0.5$ – is still well below the 67 hours found in [Englander and Conway 2017]. From [Fan et al. 2022], run times of two to three hours are found. As [Fan et al. 2022] has not properly defined the problem, no verification or comparison is possible and there is no other literature that has optimised the problem used in this thesis.

All results are shortly considered and a trade-off is made between the computational complexity and the quality of the results. With the differences in what low-f_s sequences are evaluated combined with the lack of consistency of the RPS strings, there was no conclusively optimal q value. $q = 0.5$ is the most robust from what is tested in this thesis, however an alternative with multiple seeds of a lower q value is promising. The fp value had a negligible effect on the results with the test problem considered and was therefore fixed to 1.0 as it provides the minimum ΔV which directly relates to the propellant mass and by extension the payload mass. $fpi = 0.5$ is found to be optimal both when looking at the sequences and the RPS. A more statistically spread consideration of the various PTB values was better. However, various improvements are possible here too. The run times can not directly be compared to literature, but it can be concluded that the run times are reasonable from a mission design perspective. The next chapter draws conclusions on all aspects of this thesis.

Part V

Conclusions and Recommendations

11

Conclusions

In this chapter, conclusions are drawn from the work that has been done in this thesis. The chapter reiterates the research question, after which the conclusions are separated into LTTO- and MGASO-related conclusions, and finally the sub-questions and research question are answered.

11.1. Research question reiteration

This section shortly reiterates the research question and accompanying sub-questions. The main research question is:

How can the MGA sequencing problem using low-thrust trajectories be tackled with an automated parallel-computing optimisation approach for preliminary trajectory design?

Several accompanying sub-questions were defined to break the overarching question into more processable blocks:

- How should low-thrust trajectories be represented to guarantee the accurate assessment of the performance of an MGA sequence?
- How can parallelisation be used to assist the optimisation process?
- What metric should be used for the quality of a sequence to increase the robustness of the sequence optimisation?
- How effective is the developed methodology in optimising a low-thrust MGA sequencing problem?

The answers to these questions are divided into conclusions about the LTTO process – the inner loop – and the MGASO process – the outer loop, spread over the next two sections.

11.2. LTTO

This section focuses on the conclusions related to the LTTO process.

LTTO model choices

A hodographic-shaping representation was used together with a simplified patched-conics approach for modelling GAs. The hodographic-shaping implementation in Tudat, which is used for this thesis, has been verified. Moreover, the combination of hodographic shaping with the simplified patched-conics approach has shown to be an accurate method to represent MGA trajectories within the definition of the problem. For the optimisation, the SGA is used, which has shown to be a suitable optimisation method for the mixed-integer and non-linear nature of the problem. The low-thrust trajectory optimisation process is not accurate for all possible trajectories without several tuning aspects, discussed next.

Tuning process and results
A rigorous tuning process has been performed on multiple test cases that improved the ΔV values found for MGA trajectories by more than 50%, resulting in a verified set of MGA trajectories with up to three GAs. The convergence becomes more difficult with more GAs, as this increases the design complexity substantially. Most tuning has been conducted on an Earth-Jupiter transfer based on [Fan et al. 2021]. A population size of 1200 and a generation count of 300 is found to produce the best results, further expansion showed negligible improvement. Multiple LTTO configurations have been tested with various selections of variables: 'Configuration IV' performs best. It is essential to include θ, ϕ as well as β to properly optimise the GAs and make a GA worthwhile in terms of minimising ΔV. Trajectories using one and two free parameters can both produce accurate results. A local optimisation is tested and found to systematically reduce the ΔV further with little added computational time, however, this is not added to the methodology for the results.

Design-space exploration
The bounds that were tuned are problem specific and would require generalisation to apply to other test cases. For the Earth-Jupiter test case, a departure date bound of 60 days showed robust results and provided enough accuracy to consistently find a verifiable optimum. The ToF values change significantly depending on the transfer leg. In particular, the direction of the transfer – inward or outward assuming Earth is the departure planet – is an important factor. A number of revolutions of up to two were found: this bound is tuned for outward Earth-Jupiter transfers, whereas inner transfers may require more revolutions due to the decreased orbital period and constrained thrust magnitude. In general, the bounds have a smaller effect on the quality of the result compared to the other tuning aspects, therefore a conservatively wide estimate is used.

Parallelisation for LTTO
A core part of the LTTO is parallelisation. In the LTTO, multiple identical optimisation processes are run in parallel using the GIM. This allows for the migration of highly fit individuals to accelerate the optimisation process and improve robustness. The migration of individuals between islands defined by a topology improved the standard deviation of the results significantly, leading to consistent and robust results. In particular, the outliers that converged to significantly sub-optimal ΔV values are filtered out. In addition, the custom topology increases the convergence speed.

In conclusion, the LTTO has been shown to provide accurate ΔV values compared to results from literature. Moreover, these accurate results are found consistently when rerunning the optimisation. The combination of these results makes the LTTO implementation a suitable inner optimisation step, which can very well be used in conjunction with an outer loop optimisation.

11.3. MGASO

This section focuses on the conclusions related to the MGASO process.

RTBA in general
The MGASO is performed using the RTBA. This approach applies the GIM in a novel way, by splitting the combinatorial space into parallel tasks defined by the PTBs. Every PTB is evaluated through multiple sequences with multiple islands. The approach is recursive because an identical process is performed for each subsequent GA. Moreover, the RTBA is developed by combining two tree-search methods that performed well in literature: the MCTS and BSS method. This combination allows for efficient pruning of the combinatorial space while maintaining a high degree of exploration through each recursion.

Pruning strategies
To improve the performance of the RTBA, multiple features were developed in terms of computational efficiency. First, the initial candidates for possible GAs were pruned away using theory from astrodynamics. This reduced the combinatorial complexity from 21845 to 85 possible GA sequences within the bounds of the problem considered. Second, the Monte-Carlo process is slightly tweaked such that it cannot evaluate the same sequence twice. This feature reduces wasted computational resources. In addition, the current recursion can use the evaluated sequences from each prior recursion for the RPS determination. This addition helps increase the statistical confidence in the optimality of each

target body, which leads to a more informed decision on further RPS additions. Last but not least, a custom topology is applied. A topology has not been applied to an optimisation problem like the one considered. The custom topology only connects the islands of single sequences to prevent migrating between incompatible islands. The migration of individuals between different sequences is not beneficial, as each sequence has unique optimal design variables as well as a potentially different length.

Fitness proportion metrics
In the quantification of the optimality of MGA sequences found by the RTBA, several new quantities have been defined to improve the RPS determination process as well as to quantify the degree of exploration. Three quantities are crucial here: q, fp, and fpi. q is defined as the fraction of the combinatorial space that is to be explored in each recursion. In this thesis, q is kept constant over all recursions. The choice for this definition is to maintain a similar relative level of statistical confidence in the various PTBs. The fp parameter is defined to take into account all the optima of islands for one sequence. In particular, a linear combination of the minimum and mean ΔV values is used, where fp defines the ratio between these two quantities. fpi is defined analogously, but on a different level of the approach: while determining the RPS addition, the ΔV of the best sequence with a certain PTB can be combined with the mean of the ΔV of all sequences of that same PTB to determine the optimality of that PTB. Once this step is executed on all PTBs, the best PTB of that recursion is appended to the RPS. These parameters allow for a distributed and well-informed decision on the addition to the RPS.

Tuning process
For the MGASO, a tuning process of the previously mentioned parameters is found to be necessary. Specifically, the fp and fpi parameters are considered for this using a grid search, as the q value is the quantity that is to be minimised while maintaining robust performance. After one iteration, the optimisation complexity was reduced slightly from the values that were found in the LTTO tuning process. This iteration showed an improvement in the run time by 75%. Moreover, the ranking of sequences found is in strong agreement with the expected optimal MGA sequences. For this assessment, it is assumed that a small selection of sequences forms the output that is subsequently input into higher-fidelity methods. This group of sequences is defined to include all sequences that have a minimum f_s within 30 % or 5 km/s of the lowest-f_s sequence. The definition leads to a group with up to 10 sequences, depending on the q value.

For the Earth-Jupiter case considered, Earth and Mars GAs were expected to be more optimal. In particular, the 'EM', 'EE', and 'MM' sub-sequences are considered best: these transfers were found exclusively in the low-f_s group. Furthermore, the ΔV values found in this group are comparable to the LTTO verification ΔV values. While the optimality of the MGASO is not explicitly verified, the verified ΔV values of the LTTO show that optimal MGA sequences can be found within a short departure date window. However, these values were found when considering all grid points: per grid point only a portion of the low-f_s sequences were found, showing a lack of consistency. As for the RPS, the strings only converge to the RPS strings of the theoretically expected transfers. This convergence behaviour is in agreement with the sequence ranking found. However, no consistency is observed across the various parameter combinations, similar to the sequence ranking. To remedy this, an increase in robustness is required for the final results. The final recursion – three recursions are permitted – is removed as it is less relevant in proving whether the RTBA converges in the correct direction while decreasing the computational complexity significantly. At this point, the custom topology is added, ps is set to 1200 and gc is set to 300. Furthermore, multiple seeds are evaluated to more thoroughly test the robustness. The grid search could also be simplified because $fp = 0.0$ and $fpi = 0.0$ were not effective in finding the MGA sequences with the minimum ΔV.

Results
The results are split into sequence ranking and RPS strings. With the changes mentioned before, a significant increase in robustness was observed. The spread of sequences at all optimality levels improved. The general f_s values of sequences improved, although this was more so the case for sub-optimal sequences. The low-f_s group became more clearly identifiable and the discrepancy between the sequences with the recommended transfers – 'EE', 'EM', and 'MM' – compared to all other sequences increased. It is shown that for the Earth-Jupiter test case, the Venus and Mercury GA sequences systematically produce higher ΔV values; Mercury performs significantly worse than Venus. This

trend exposed a significant waste of computational resources on sub-optimal transfers. Regarding the consistency across three seeds, the low-f_s sequences are analysed. It was concluded that the $q = 0.5$ runs performed best because of the increased exploration: the average number of seeds that evaluated each sequence in the low-f_s group increased from 1.1 to 2.55, with an increase in run time of a little over 400%. Smaller q values are found to be too sensitive to small differences in the sequences that are evaluated. As no other literature presents the results with q or Q, a comparison between the total combinatorial complexity explored is not possible. Regarding fp and fpi, $fp = 1.0$ and $fpi = 0.5$ are shown to be the best combination to find the actual optimum by finding more low-f_s sequences on average, though the differences are small. The inclusion of the mean value for the RPS determination helps to filter out the sensitivity for Mercury and Venus transfers mentioned before. These conclusions are supported by the RPS strings found. For the RPS strings, however, no consistency was found across seeds. The RTBA consistently converges in the correct direction but converges differently for different seeds, q values, and fpi values. The statistical confidence in the optimality of each PTB is limited and therefore more sensitive to occurrences of Mercury and Venus. This result increases the importance of pruning away sub-optimal GA candidates during the optimisation, rather than a priori. Despite the lack of consistency in the RPS, the seeds find mostly the same sequences. It can be concluded that the RTBA can optimise low-thrust MGA sequences and find comparably low ΔV values for sequences verified by literature. Most aspects of the RTBA are verified and some elements are validated, which confirms to a large extent the correct and physically real functioning of the RTBA. Besides the quality of each sequence, the consistency of the RTBA in general needs further development. The cause of this inconsistency is mostly due to the variety of sequences present in the low-f_s region. The RPS can only converge to one sequence, however, the relevant output of this approach is a small selection of multiple sequences. Therefore, due to the nature of the optimal sequences, a single RPS – that can only converge to one sub-sequence in the low-f_s group – is not ideal.

Context and implications
The run times are difficult to compare with literature; run times for somewhat comparable problems range from 2 to 67 hours. The run time for $q = 0.5$ is 21 hours on average for the Earth-Jupiter transfer that has a combinatorial complexity of 21845. The combinatorial complexity decreases to 85 after pruning. The run time is spread over 42 CPUs. A Mercury rendezvous mission from [Englander and Conway 2017] with a combinatorial complexity of 9841 finds a run time of 67 hours over 60 cores – CPU is equivalent to core. The individual cores perform similarly. By comparing the run time and combinatorial complexity of both methods, the RTBA is shown to be competitive for low-thrust MGA sequencing optimisation. This comparison is not perfect because different models and different problem definitions are considered. While other methods [Fan et al. 2022; Englander and Conway 2017; Morante et al. 2019] do not extensively discuss the confidence in the sequence ranking given, it is clear that the uncertainty in the ΔV values compared to higher-fidelity methods and other shape-based methods shows that no unique, perfect sequence ranking can be given. With the RTBA, a high level of confidence is present for the superiority of the low-f_s sequences compared to the high-f_s sequences, which is a good result for run times of 21 hours. While 21 hours is still long, there are many facets where the computational performance can be increased. The main takeaway from this thesis is that the method can reliably optimise low-thrust MGA sequences without full enumeration within a limited amount of run time. With further optimisation of the approach, the RTBA is expected to be a highly-performing candidate for preliminary design.

11.4. Answer to research questions

The previous sections have drawn conclusions on the work in this thesis. This section explicitly answers the research questions. The sub-questions are answered first after which the main research question is answered:

- **How should low-thrust trajectories be represented to guarantee the accurate assessment of the performance of an MGA sequence?**

  ▷ There are numerous methods for representing a low-thrust trajectory that can accurately assess the ΔV of an MGA sequence. For single transfers, hodographic shaping has been shown to provide competitive ΔV values that have been verified using validated software such as DITAN. For the GAs, the simplified patched-conics

approach has been proven to improve ΔV values of MGA sequences when defined properly. The combination of these methods forms the LTTO, which has consistently been shown to converge to realistic ΔV values for MGA sequences up to three GAs. In particular, ΔV values are within 10% of two other shaping methods from literature. It is expected that the LTTO with more GAs and other sequences would also converge to accurate results, but this remains to be tested.

- **How can parallelisation be used to assist the optimisation process?**

  ▷ For the RTBA, the GIM has been used to parallelise the optimisation of MGA sequences by allocating multiple islands to various sequences. Using the custom topology, the convergence of the LTTO is improved significantly. As a result, the highly-optimal MGA sequences can consistently be distinguished from the sub-optimal MGA sequences. The run-time improvement as a result of the custom topology has not been investigated.

- **What metric should be used for the quality of a sequence to increase the robustness of the sequence optimisation?**

  ▷ The quality of a sequence is defined by the objective ΔV, however, to determine its quality overall, a combination is taken of the minimum and mean ΔV values across multiple islands. Specifically, a linear combination with a weight of 0.5 is used. Furthermore, the RTBA is developed to assess the quality of the PTBs, for which a combination is used of the minimum and mean fitness of sequences that evaluate each PTB. It is shown that using the mean of the sequences for the quality of a PTB improves the robustness of the sequence optimisation.

- **How effective is the developed methodology in optimising a low-thrust MGA sequencing problem?**

  ▷ The RTBA can optimise a low-thrust MGA sequencing problem within run times of less than a day. Three seeds are used, and with a q value of 0.5, an average of 2.55 seeds find each highly-optimal sequence. Furthermore, the RPS consistently converges in the direction that is expected to be optimal, although the specific RPS strings are not identical across multiple seeds.

With the answers to all sub-questions, the main research question can be answered:

- **How can the MGA sequencing problem using low-thrust trajectories be tackled with an automated parallel-computing optimisation approach for preliminary trajectory design?**

  ▷ The MGA sequencing problem for low-thrust trajectories can be tackled with an automated parallel-computing optimisation approach by using the RTBA. This approach consists of a nested-loop optimisation approach that exploits multi-processing capabilities and combines the existing MCTS and BSS tree-search method with hodographic shaping. A high-fitness group of sequences was consistently found for an Earth-Jupiter transfer with up to three GAs. The combination of a greedy approach with a Monte-Carlo-based search is effective, however, some difficulty arises due to the limits of using a single RPS. The novel quantities introduced for the RTBA show a marginal increase in convergence.

There are a plethora of further recommendations, which are discussed next and constitute the final chapter of this thesis.

12

Recommendations

This chapter discusses several useful insights that can be further explored. These insights range from small improvements that can contribute to slightly better results, but also more profound changes that will expand the scope of this approach to make it more attractive.

12.1. Expansion of applicability

This section includes recommendations that could expand the applicability of the methodology to more general cases.

Expansion of GA candidates

This thesis only considers the planets as potential GA candidates which is beneficial for the combinatorial complexity of the problem, but has several downsides. This assumption limits the approach to missions that depart from and arrive at a planet. Targets including asteroids and comets may be of more interest because one can learn about the formation of the Solar System, for example. Moreover, only rendezvous cases were used even though intercepts and flybys are increasingly used for extensive science missions. This addition would immediately increase the number of potential GA candidates by multiple orders of magnitude. In this case, the tuned parameters must be revised in addition to the general idea of having a greedy approach. Adding bodies other than the planets is not only interesting scientifically, but they can also be relevant for the optimal sequence even in interplanetary design cases. [Fan et al. 2022] finds a Juno-Jupiter-Jupiter optimal sequence for an Earth-Jupiter transfer case, for example.

Multi-objective optimisation

Future work could expand the RTBA to use multi-objective optimisation. The objectives tested here are limited to ΔV. ΔV is one way of indicating whether a mission is feasible in terms of propulsion system design and propellant mass constraints. While ΔV remains a core objective for most trajectory optimisation problems, limiting the objective to this quantity has a number of downsides. The main reason is that there are other stakes in the design of any trajectory. Time is of the essence. The most pressing risk of longer ToF values is that the spacecraft is designed for a certain lifetime. Lower ToF values, therefore, ease the design process, because the scientific instruments can be designed for a shorter lifetime. Current low-thrust propulsion technologies can provide many km/s but are constrained by the mechanical design: constant stress and fatigue make the design of a propulsion system that must function for a decade an enormous challenge. Another essential constraint is that the mass that can be launched into space is limited by cost, geo-political considerations, and volume – a launcher only has so much space in its fairing. Another way of quantifying the mass requirement is by using a mass-related objective instead. Therefore, a multi-objective optimisation would expand the optimisation to more realistic scenarios for mission design.

Coasting arcs

Another core addition that can be contributed is that of coasting arcs. With low-thrust propulsion, as mentioned sporadically throughout this thesis, the thrusters do not thrust continuously for years. They

need to be commissioned in addition to perhaps periodic cooling or further maintenance checks. The navigation of a spacecraft also requires that spacecraft are not constantly thrusting, as the navigation is mainly done using Doppler data. The quality of observations using Doppler data when a spacecraft is under thrust is severely impaired. Also, the objectives related to the ΔV can be improved upon with the addition of coasting arcs, as the thrust can be zero for a considerable portion of the trajectories. It should be noted that this may also come at the cost of ToF. Nevertheless, it gives another degree of freedom to consider. Another reason to include coasting arcs is the wider availability of reference papers that have already included coasting arcs with their approaches. A better comparison can be made of the final results, and a complete validation step also becomes possible. Coasting is thus an important part of low-thrust mission design and should be added in future work.

Dynamic bounds
To make the approach more robust and widely applicable than it already is, methods can be developed that automatically generate the bounds necessary for the LTTO based on the input parameters. As an example, the ToF can be automatically adjusted based on the GA candidates at each leg. An EJ transfer has fundamentally different ToF values than an EV transfer, for example. Not only can the ToF bounds be dynamically determined, but the GA angles necessary for the conditions at each GA can also be dynamically determined. The same holds for the shaping functions, and effectively any bound that is part of the LTTO. The methodology for the dynamic bounds has been considered, but the exact tuning of them is unknown. This addition can contribute to more robust results.

Grid search as extra level
It is found that the departure date is very indicative of the performance of certain sequences. In this thesis, only a 60-day window was considered for the results. Mission designers never only consider 60 days when designing an interplanetary mission due to many uncertainties in the project development. To increase the completeness of the methodology for mission design, the departure date can be added as an extra abstraction layer in the form of a grid search. This may have severe implications for the RTBA, because more optimisations need to be performed, while only one parameter is removed from the design variable vector in the LTTO.

This concludes the main recommendations that can make the applicability of the RTBA greater. Next, recommendations are given for the potential improvements to the current implementation.

12.2. Improvement of current implementation

This section discusses recommendations that can be added to the approach developed in this thesis to improve its performance.

Parallel RPS strings
The current implementation of the RTBA only allows for one RPS string. This fact combined with the greedy nature of the optimisation means that the low-f_s group can never fully be explored, as it is found that this group includes multiple different RPS sub-strings. It is recommended to allow for multiple RPS strings to be converging in parallel, based on the relative performance of various sequences. It is expected that, for the Earth-Jupiter test case, both Earth and Mars will be chosen as initial RPS characters, as they both perform equally well in terms of fitness. This does make the RTBA not strictly greedy, however, the RPS strings individually do act as greedy optimisations.

Run-time pruning
Another limitation of the RTBA is that significant computational time is spent evaluating sequences that find high ΔV values. It would be better to prune the sub-optimal candidates or sequences out of the optimisation. This can be done in two ways. One option would be to prune certain candidates after recursions. This has disadvantages, especially if expanded to larger problems with more candidates, where the sequences are less predictable for the test case in this thesis. The second option would be to implement a certain critical f_s value below which the transfer would contribute to f_x. This critical value can filter out sub-optimal sequences that substantially increase the fitness value and nullify the contribution of more optimal sequences.

Exchange between sequences
A leg database that the entire archipelago can access can be added. Specific legs with specific conditions can have similar optima, and if those conditions are saved, they can be inserted into design variable vectors. This proposal goes further than inserting certain variable values because that is what the genetic algorithm is already supposed to solve. However, because the design variables are dependent on one another, the leg database can save a specific combination of variables that is only useful for that leg, whereas the genetic algorithm does not have any physical interpretation or intuition about the dependencies. This can improve the fitness of single islands across the entire archipelago. If two different sequences (that now cannot communicate with each other by design) have a leg in common, and they happen to find a leg that has a high fitness that would apply to the other sequence, then it should be able to migrate. This concept can be investigated more thoroughly, however, it does make the methodology more memory intensive, which should be considered as well. Note that the current implementation of the RTBA is not particularly memory intensive, with total RAM usage of up to 8 GB.

Improvement of custom topology
Currently, the islands allow for migrating individuals, but the optimisation problems themselves are identical. Inspiration can be found in the original intent of the GIM, where various optimisation algorithms are used to converge more quickly. As an addition to the custom topology used in this work, the islands can also use other optimisation algorithms to improve convergence. This would allow for fewer generations and individuals, which would decrease computation time.

Expand fitness proportion metrics
The fitnesses of sequences, and also of PTBs are determined by the minimum and mean of the respective collection of values. However, the minimum and the mean do not give a complete perspective on the results of all the islands. In the LTTO, a spread of islands is observed. This distribution is not exactly Gaussian, and can be defined with a mean and standard deviation. The latter of the two indicates how wide the spread is of the quantities. It could be beneficial to investigate whether or not the addition of this quantity may help in the assessment of the fp and fpi at both the sequence level and the PTB level.

Elaborate tuning process
The tuning process is extensive, however, further tuning may give more insight into the type of problem. From design-space exploration methodologies, a Central Composite Design (CCD) or Fractional Factorial Design (FFD) can be performed in combination with an Analysis Of Variance (ANOVA) to quantify the dependencies, rather than theoretically establishing them. This may have a limited effect on the end result and may change per test case, but it would also contribute to the dynamic bound recommendations discussed before.

References

Abdelkhalik, Ossama and Ahmed Gad (2012). “Dynamic-size multiple populations genetic algorithm for multigravity-assist trajectory optimization”. In: *Journal of Guidance, Control, and Dynamics* 35.2, pp. 520–529.

Bellome, Andrea et al. (June 2021). “A multi-fidelity optimization process for complex multiple gravity assist trajectory design”. In: 8th International Conference on Astrodynamics Tools and Techniques.

Bellome, Andrea et al. (2020). “Modified tisserand map exploration for preliminary multiple gravity assist trajectory design”. In: 71st International Astronautical Congress - the Cyberspace Edition.

Biscani, Francesco and Dario Izzo (2020). “A parallel global multiobjective framework for optimization: pagmo”. In: *Journal of Open Source Software* 5.53, p. 2338.

Campagnola, Stefano et al. (2015). “Low-thrust trajectory design and operations of PROCYON, the first deep-space micro-spacecraft”. In: *25th International Symposium on Space Flight Dynamics*. Vol. 7. German Aerospace Center (DLR) Munich, Germany.

Carnelli, Ian et al. (2009). “Evolutionary neurocontrol: A novel method for low-thrust gravity-assist trajectory optimization”. In: *Journal of Guidance, Control, and Dynamics* 32.2, pp. 616–625.

Casteren, Jan van and M Novara (2011). “BepiColombo mission”. In: *Memorie della Societa Astronomica Italiana* 82, p. 394.

Ceriotti, Matteo and Massimiliano Vasile (2010). “MGA trajectory planning with an ACO-inspired algorithm”. In: *Acta Astronautica* 67.9, pp. 1202–1217. ISSN: 0094-5765.

Chilan, Christian M and Bruce A Conway (2013). “Automated design of multiphase space missions using hybrid optimal control”. In: *Journal of Guidance, Control, and Dynamics* 36.5, pp. 1410–1424.

Conway, Bruce A (2010). *Spacecraft trajectory optimization*. Vol. 29. Cambridge University Press.

Cowan, Sean (July 2022). *Automated Low-thrust Trajectory Optimization*. Delft University of Technology. Literature Study.

Crain, Timothy et al. (2000). “Interplanetary flyby mission optimization using a hybrid global-local search method”. In: *Journal of Spacecraft and Rockets* 37.4, pp. 468–474.

Dachwald, Bernd (2004). “Low-thrust trajectory optimization and interplanetary mission analysis using evolutionary neurocontrol”. PhD thesis. Universität der Bundeswehr, München.

Dachwald, Bernd and Andreas Ohndorf (2019). “Global optimization of continuous-thrust trajectories using evolutionary neurocontrol”. In: *Modeling and Optimization in Space Engineering*. Springer, pp. 33–57.

De Pascale, Paolo and Massimiliano Vasile (2006). “Preliminary design of low-thrust multiple gravity-assist trajectories”. In: *Journal of Spacecraft and Rockets* 43.5, pp. 1065–1076.

Debban, Theresa et al. (2002). “Design and optimization of low-thrust gravity-assist trajectories to selected planets”. In: *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, p. 4729.

Dirkx, Dominic et al. (2022). “The open-source astrodynamics Tudatpy software–overview for planetary mission design and science analysis”. In: *EPSC2022* EPSC2022-253.

Ellison, Donald Hamilton (2018). “Robust preliminary design for multiple gravity assist spacecraft trajectories”. PhD thesis. University of Illinois at Urbana-Champaign.

Englander, Jacob A and Bruce A Conway (2017). “Automated solution of the low-thrust interplanetary trajectory problem”. In: *Journal of Guidance, Control, and Dynamics* 40.1, pp. 15–27.

Englander, Jacob A et al. (2012). “Automated mission planning via evolutionary algorithms”. In: *Journal of Guidance, Control, and Dynamics* 35.6, pp. 1878–1887.

Englander, Jacob A et al. (2015). “Multi-objective hybrid optimal control for multiple-flyby low-thrust mission design”. In: *AAS/AIAA Space Flight Mechanics Meeting*. GSFC-E-DAA-TN19664.

Fan, Zichen et al. (2021). “Fast initial design of low-thrust multiple gravity-assist three-dimensional trajectories based on the Bezier shape-based method”. In: *Acta Astronautica* 178, pp. 233–240.

Fan, Zichen et al. (2022). “Improved Monte Carlo Tree Search-based approach to low-thrust multiple gravity-assist trajectory design”. In: *Aerospace Science and Technology* 130, p. 107946.

Gad, Ahmed and Ossama Abdelkhalik (2011). “Hidden genes genetic algorithm for multi-gravity-assist trajectories optimization”. In: *Journal of Spacecraft and Rockets* 48.4, pp. 629–641.

Galletti, Elena (2017). *Fast computation of SEP transfers to Mars using analytic curve-fit functions*. Delft University Of Technology. MSc Thesis.

Gondelach, David J and Ron Noomen (2015). “Hodographic-shaping method for low-thrust interplanetary trajectory design”. In: *Journal of Spacecraft and Rockets* 52.3, pp. 728–738.

Hennes, Daniel and Dario Izzo (2015). “Interplanetary trajectory planning with Monte Carlo tree search”. In: *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

Izzo, Dario et al. (2012). “The generalized island model”. In: *Parallel Architectures and Bioinspired Algorithms*. Springer, pp. 151–169.

James, Steven et al. (Feb. 2017). “An Analysis of Monte Carlo Tree Search”. In: *Thirty-First AAAI Conference on Artificial Intelligence*. Vol. 31. 1.

Johnson, Greg et al. (2003). “Copernicus: A generalized trajectory design and optimization system”. In: *UT Austin Engineering Communication ASE 333T, available online*.

Kranen, Tommy (2019). *Low-Thrust Gravity Assist Trajectory Optimisation using Evolutionary Neurocontrol*. Delft University Of Technology. MSc Thesis.

Maiwald, Volker (2017). “A new method for optimization of low-thrust gravity-assist sequences”. In: *CEAS Space Journal* 9.3, pp. 243–256.

Massari, Mauro and Alexander Wittig (2015). “Optimization of multiple-rendezvous low-thrust missions on general-purpose graphics processing units”. In: *Journal of Aerospace Information Systems* 13.2, pp. 1–13.

McConaghy, T Troy et al. (2003). “Design and optimization of low-thrust trajectories with gravity assists”. In: *Journal of spacecraft and rockets* 40.3, pp. 380–387.

Morante, David et al. (2021). “A survey on low-thrust trajectory optimization approaches”. In: *Aerospace* 8.3, p. 88.

Morante, David et al. (2019). “Multi-objective low-thrust interplanetary trajectory optimization based on generalized logarithmic spirals”. In: *Journal of Guidance, Control, and Dynamics* 42.3, pp. 476–490.

Moreno Gonzalez, Andres (2020). *Characterisation of Shape-Based Methods and Combination with Coasting Arcs*. Delft University of Technology. MSc Thesis.

Musegaas, Paul (2013). *Optimization of space trajectories including multiple gravity assists and deep space maneuvers*. Delft University of Technology. MSc Thesis.

Novak, Daniel M and Massimiliano Vasile (2011). “Improved shaping approach to the preliminary design of low-thrust trajectories”. In: *Journal of Guidance, Control and Dynamics* 34.1, pp. 128–147.

Nyew, Hui Meen et al. (2015). “Structured-chromosome evolutionary algorithms for variable-size autonomous interplanetary trajectory planning optimization”. In: *Journal of Aerospace Information Systems* 12.3, pp. 314–328.

Ozimek, Martin et al. (2019). “The low-thrust interplanetary explorer: A medium-fidelity algorithm for multi-gravity assist low-thrust trajectory optimization”. In: *Proceedings of the AAS/AIAA Space Flight Mechanics Meeting, Maui, HI, USA*, pp. 13–17.

Petropoulos, Anastassios E and James M Longuski (2004). “Shape-based algorithm for the automated design of low-thrust, gravity assist trajectories”. In: *Journal of Spacecraft and Rockets* 41.5, pp. 787–796.

Rastrigin, Leonard Andreevič (1974). “Systems of extremal control”. In: *Nauka*.

Ricciardi, Lorenzo Angelo and Massimiliano Vasile (2018). “Modhoc-Multi Objective Direct Hybrid Optimal Control”. In: *7th International Conference on Astrodynamics Tools and Techniques*.

Stegmaier, Philipp et al. (2022). “Cooperative Trajectory Planning in Uncertain Environments With Monte Carlo Tree Search and Risk Metrics”. In: *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 4109–4116.

Strange, Nathan J and James M Longuski (2002). “Graphical method for gravity-assist trajectory design”. In: *Journal of Spacecraft and Rockets* 39.1, pp. 9–16.

Stubbig, Leon and Kevin Cowan (2021). “Improving the Evolutionary Optimization of Interplanetary Low-Thrust Trajectories Using a Neural Network Surrogate Model”. In: *2020 Astrodynamics Specialist Conference*, AAS–20.

Tsuda, Yuichi et al. (2013). “System design of the Hayabusa 2—Asteroid sample return mission to 1999 JU3”. In: *Acta Astronautica* 91, pp. 356–362. ISSN: 0094-5765.

Turner, Martin JL (2008). *Rocket and spacecraft propulsion: principles, practice and new developments*. Springer Science & Business Media.

Ueda, Satoshi and Hideaki Ogawa (2021). “Multi-fidelity approach for global trajectory optimization using GPU-based highly parallel architecture”. In: *Aerospace Science and Technology* 116, p. 106829.

Vasile, M et al. (2002). “Design of interplanetary and lunar missions combining low thrust and gravity assists”. In: *Final report of ESA/ESOC study contract* 14126.00.

Vasile, Massimiliano et al. (2015). “Incremental planning of multi-gravity assist trajectories”. In: *Acta Astronautica* 115, pp. 407–421.

Wakker, Karel F. (2015). *Fundamentals of Astrodynamics*. Delft University of Technology Library. URL: https://repository.tudelft.nl/islandora/object/uuid%3A3fc91471-8e47-4215-af43-718740e6694e.

Wall, Bradley J and Bruce A Conway (2009). “Shape-based approach to low-thrust rendezvous trajectory design”. In: *Journal of Guidance, Control, and Dynamics* 32.1, pp. 95–101.

Zhang, Jin et al. (2015). “Analysis of multiple asteroids rendezvous optimization using genetic algorithms”. In: *2015 IEEE Congress on Evolutionary Computation (CEC)*, pp. 596–602.

Part VI

Appendix

A

Hardware performance

The DelftBlue supercomputer is used to perform the MGASO. This chapter compares the performance of DelftBlue to that of a personal computer. The two hardware systems are introduced in Section 2.2.1.

An LTTO and MGASO test is performed using [Morante et al. 2019]. In particular, the problem definition from Section 6.2 is taken defining an Earth-Jupiter transfer. For the LTTO, the EVEMJ sequence is optimised. For both optimisation problems, it is expected that DelftBlue leads to an improvement in run time.

Table A.1: Run time comparison between Macbook Pro 2017 and DelftBlue for LTTO and MGASO.

Simulation type	Run time [hh:mm]	
	MacBook Pro	DelftBlue
LTTO		
$gc = 150$, $ps = 300$	00:17	00:14 (4 CPUs) 00:13:23 (22 CPUs)
MGASO		
$mng = 3$, $sr = 2$ $spp = 10$, $gc = 150$, $ps = 300$	01:56	01:26 (4 CPUs) 00:31 (22 CPUs)

mng is the maximum number of GAs allowed, sr is the number of sequence recursions, spp is the number of sequences per permitted GA candidate. These quantities are discussed elaborately in Part III.

It can be seen that the run times are shorter when running on DelftBlue. The LTTO test only results in a marginal increase in performance per CPU, which is plausible because there is no elevated level of parallelisation, only an increase in the power of a given CPU. The MGASO case does result in significant run time improvements, again limited when a similar number of CPUs are used, but when a high CPU count is used, the performance is almost four times as fast. This can be the difference between 100 hours and 25 hours. The extra time required to move files between DelftBlue and the personal computer is negligible relative to this increase in time saved. These values were found in an early stage of the thesis without the fully developed approach. To further verify these results, the test cases were run three times, and similar results came out.

B

Hodographic shaping

This appendix provides the coefficient derivation and full hodographic-shaping methodology steps. Both sections are taken from [Gondelach and Noomen 2015].

B.1. Coefficient determination

In this section, the derivation is given of the coefficients that follow from the three boundary conditions per axis. For each of the three velocity functions, three boundary conditions have to be met, resulting in nine total boundary conditions.

$$V_r(0) = V_{r,0}; \qquad V_r(t_f) = V_{r,f}; \qquad \int_0^{t_f} V_r dt = r_f - r_0 \tag{B.1}$$

$$V_\theta(0) = V_{\theta,0}; \qquad V_\theta(t_f) = V_{\theta,f}; \qquad \int_0^{t_f} \frac{V_\theta}{r} dt = \theta_f = \psi + 2\pi N \tag{B.2}$$

$$V_z(0) = V_{z,0}; \qquad V_z(t_f) = V_{z,f}; \qquad \int_0^{t_f} V_z dt = z_f - z_0 \tag{B.3}$$

Here, ψ is the transfer angle and N is the number of revolutions. Three boundary conditions also mean that three coefficients (for each base function) are the minimum number of coefficients to define a feasible trajectory. A linear system follows from this for every velocity direction, in Equation (B.4):

$$\begin{bmatrix} v_1(0) & v_2(0) & v_3(0) \\ v_1(t_f) & v_2(t_f) & v_3(t_f) \\ \tilde{v}_1(t_f) - \tilde{v}_1(0) & \tilde{v}_2(t_f) - \tilde{v}_2(0) & \tilde{v}_3(t_f) - \tilde{v}_3(0) \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix} = \begin{bmatrix} V_0 - \sum_{i=4}^{n} c_i v_i(0) \\ V_f - \sum_{i=4}^{n} c_i v_i(t_f) \\ P_f - P_0 - \sum_{i=4}^{n} c_i \left[\tilde{v}_i(t_f) - \tilde{v}_i(0)\right] \end{bmatrix} \tag{B.4}$$

Here, V and P are the boundary conditions on velocity and position, respectively, and the tilde represents the integral of the velocity function. Of all directions, the radial and axial directions can be solved analytically. Calculation of the third coefficient requires an extra step, because the polar angle needs to be integrated numerically due to its dependence on the radial component – seen in Equation (B.2).

$$c_3 = \frac{\theta_f - \int_0^{t_f} \frac{K_3 v_1 + K_4 v_2 + \sum_{i=4}^{n} c_i v_i}{r} dt}{\int_0^{t_f} \frac{K_1 v_1 + K_2 v_2 + v_3}{r} dt} \tag{B.5}$$

In Equation (B.5), K_i are coefficients from [Gondelach and Noomen 2015]. Once c_3 is calculated, the other two coefficients follow from the other equations. The thrust acceleration can then be determined

when the individual terms are substituted into the EOM. The ΔV is calculated by the numerical integration of the thrust acceleration, as seen in Equation (B.6).

$$\Delta V = \int_0^{ToF} f dt \tag{B.6}$$

B.2. Full methodology

This section presents the full methodology in a number of steps [Gondelach and Noomen 2015]:

1. Pick the departure and target body
2. Pick the departure date and ToF (design variables)
3. Compute the corresponding initial and final position and velocity
4. Define the radial, normal, and axial velocity functions
5. Pick values for the coefficients in the velocity functions which can be chosen freely (design variables)
6. Compute the values of some of the coefficients in the velocity functions to satisfy the boundary conditions on the initial and final velocities
7. Analytically integrate the radial and axial velocity over time. Use the results to adjust the velocity functions to satisfy the boundary conditions on the final radial and axial position
8. Integrate the angular velocity ($\dot{\theta} = \frac{V_\theta}{r}$) to obtain the change in polar angle θ. The polar angle at t_{final} can be used to adjust the normal velocity function in order to meet the polar angle boundary condition
9. Compute the thrust acceleration profile using the equations of motion
10. Compute the ΔV by integrating the thrust acceleration over time

C

LTTO tuning results

In this chapter, some extra results are shown for the interested individual. The extra results consist of extra plots that do not contribute to a significant conclusion, but are still useful to present the entire process.

C.1. Departure date grid search EJ, EMJ, EEMJ

This section shows extra results for the LTTO tuning of the departure date. The EJ, EMJ, and EEMJ transfers are presented, the EEEMJ transfer results can be seen in Section 6.3.3.

C.1.1. 60-day interval grid search

The local optimisations are included for the 60-day departure date window, although the local optimisation was discussed after the departure date grid search in Section 6.3.7. The EJ transfer does not include the local optimisation, because the local optimisation does not improve on the already optimal results. The maximum improvement found was 0.04%, compared to 10% or more for MGA transfers.

Below are the 60-day interval grid-search results for EJ, EMJ, and EEMJ. To shortly recap, the departure date bound used throughout the results of this thesis is based on the consistency of the local optima around this departure date – as discussed in Section 6.3.3. For the EJ transfer in Figure C.1, the departure date interval of [61842,61902] is not explicitly plotted, however, it has been verified that there is also a local optimum around this departure date bound.

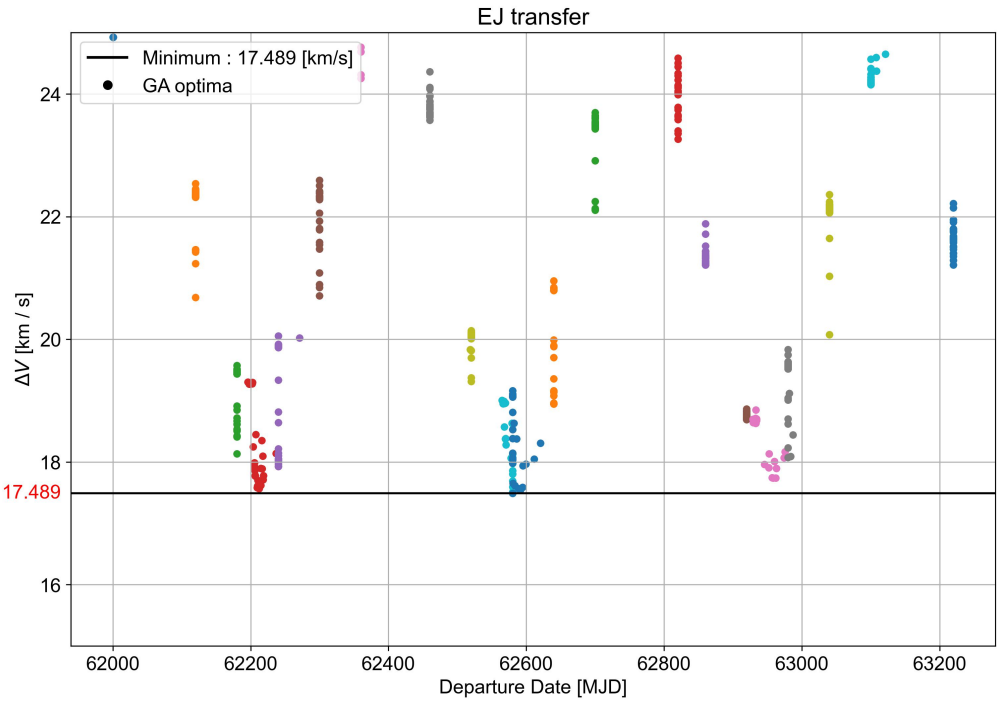


Figure C.1: Departure date grid search with 60-day interval for EJ sequence.

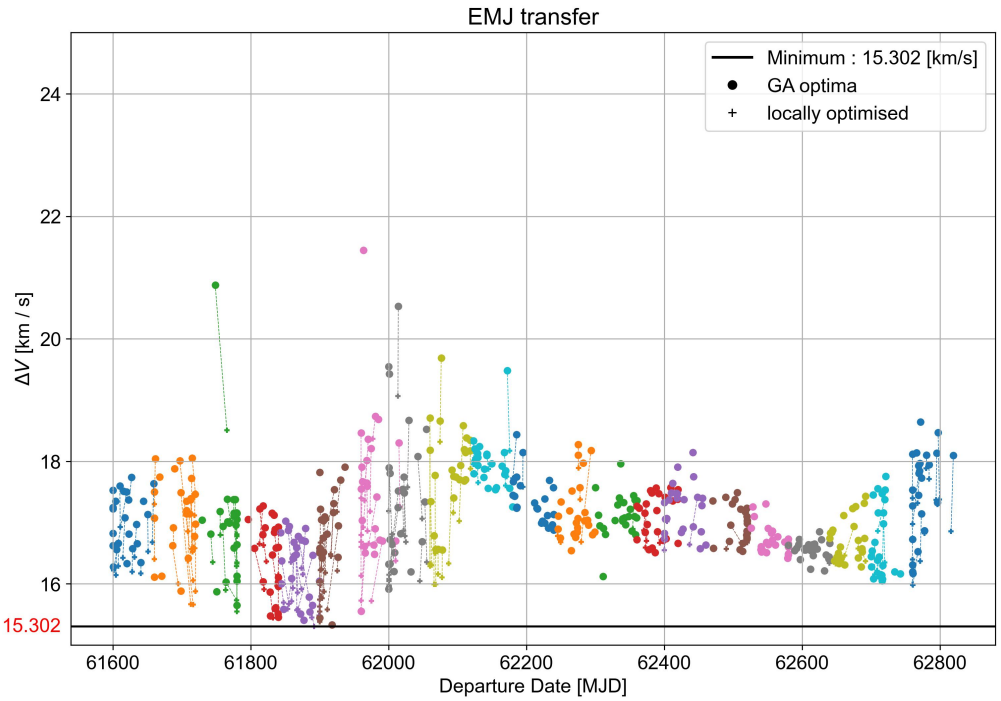


Figure C.2: Departure date grid search with 60-day interval for EMJ sequence.

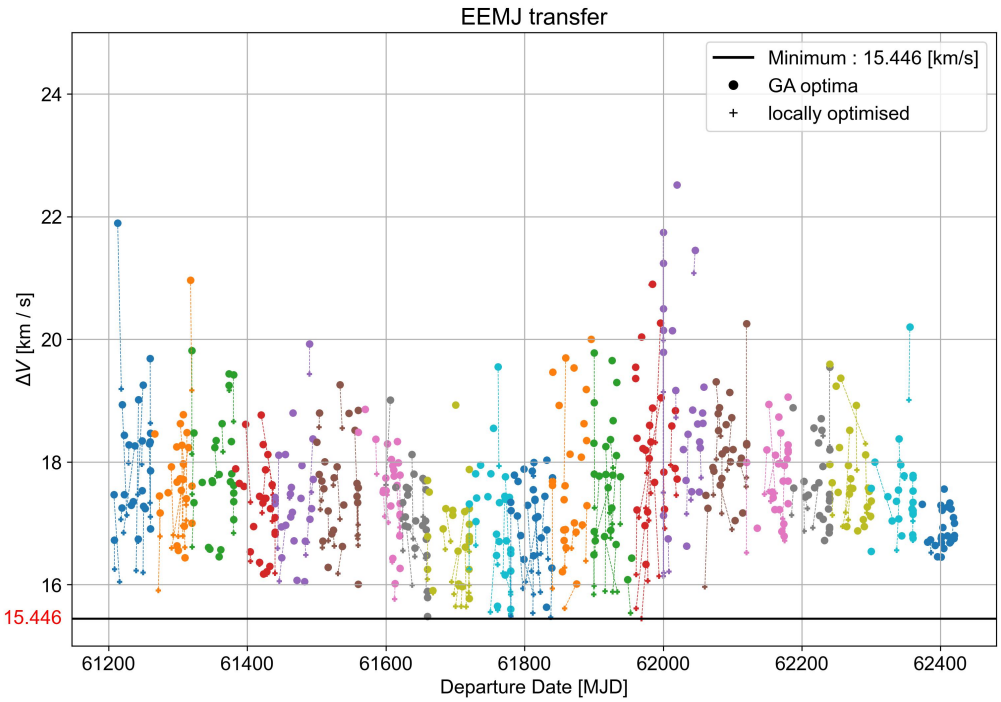


Figure C.3: Departure date grid search with 60-day interval for EEMJ sequence.

C.1.2. 400-day interval grid search

Below are the 400-day interval grid search results for EJ, EMJ, and EEMJ. For the same reason as for the 60-day interval in Section 6.3.3, the 400-day intervals are also not locally optimised. Interestingly, the minimum ΔV is not much higher than for the 60-day interval case. However, this is not tested across multiple seeds, nor does it guarantee an accurate optimisation with different sequences. It does decrease the computational time by up to 85%.

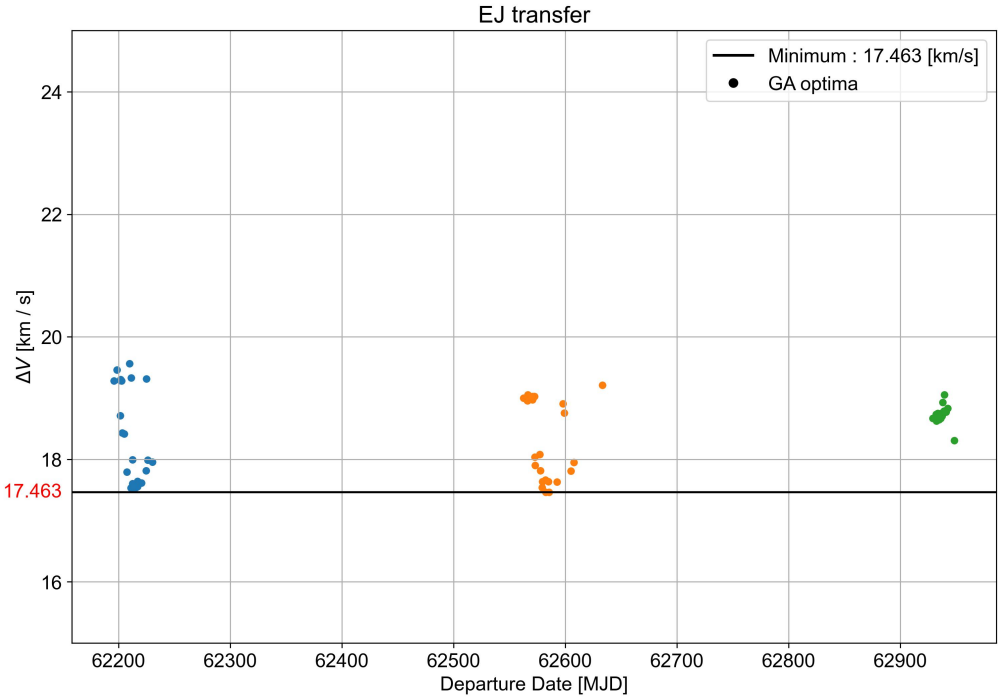


Figure C.4: Departure date grid search with 400-day interval for EJ sequence.

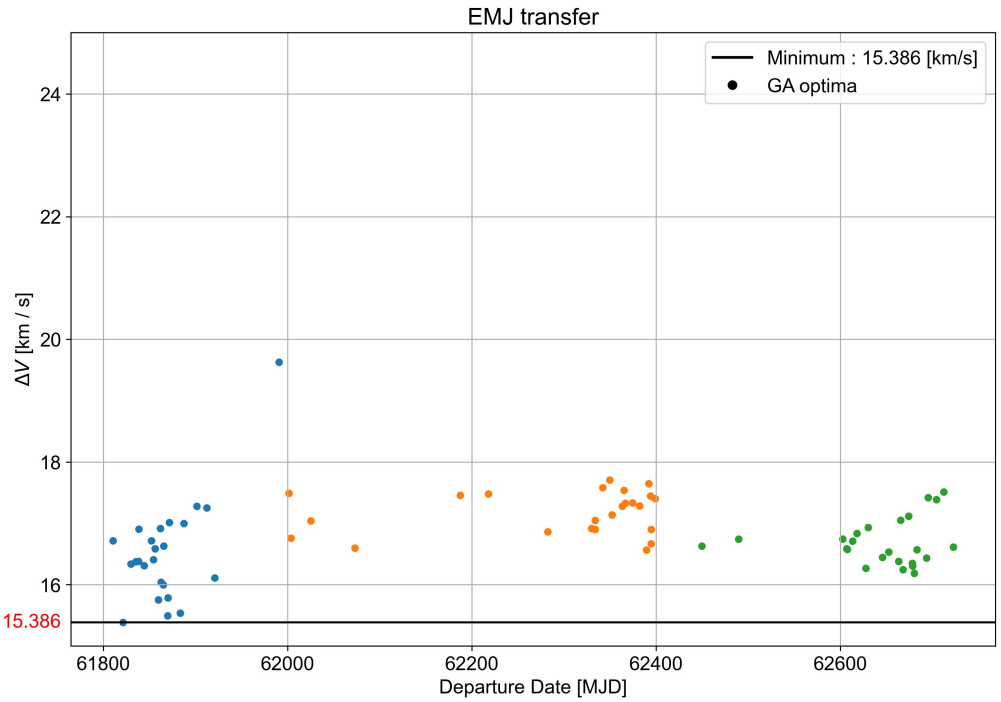


Figure C.5: Departure date grid search with 400-day interval for EMJ sequence.

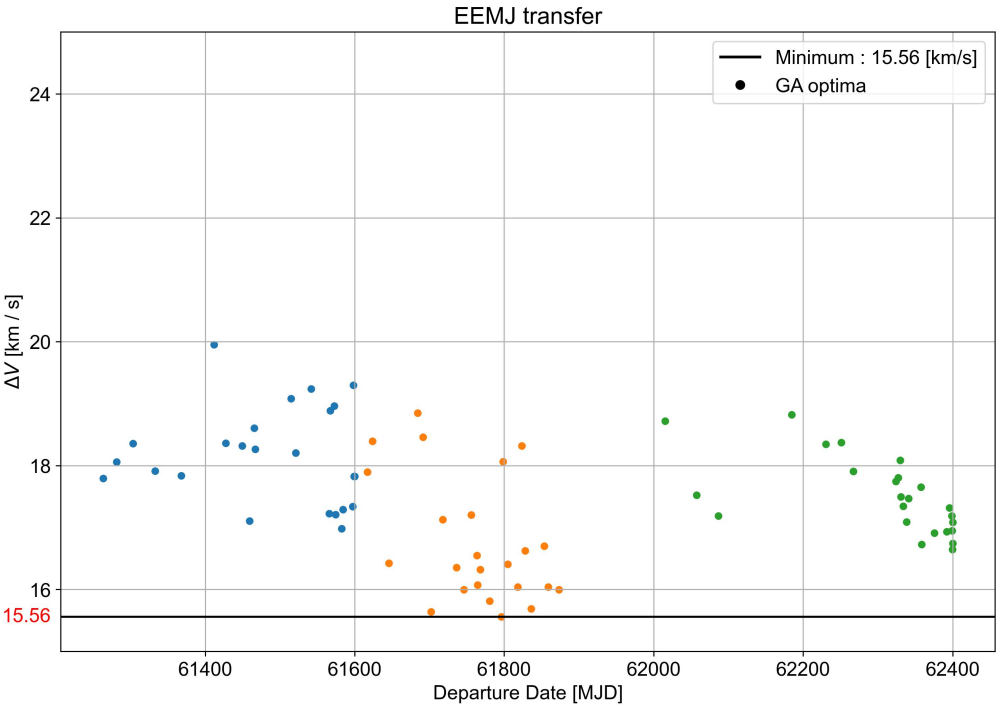


Figure C.6: Departure date grid search with 400-day interval for EEMJ sequence.

C.2. Optimisation parameters

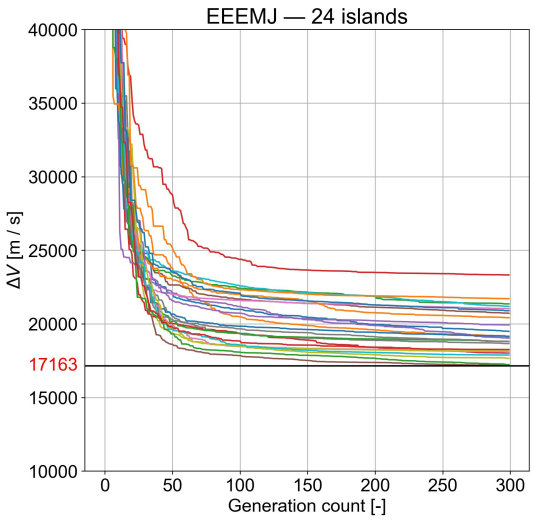
As an addition to Sections 6.3.1 and 6.3.6, extra tuning results are plotted. Table C.1 shows the results for the grid search of the ps and gc parameters with four islands. These results supported the conclusion that more islands would be necessary to determine the correct configuration, resulting in the choice for 24 islands during Part II and 14 islands during Part III.

Table C.1: Grid search results for ps and gc parameters with four islands.

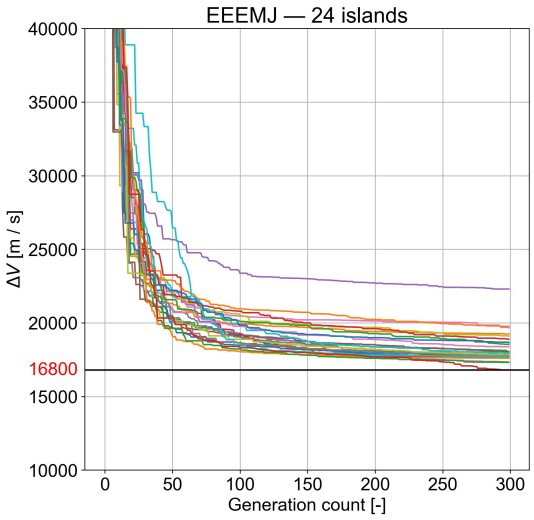
Minimum ΔV [km/s]			Generation count		
			30	100	300
Population size	100	Min	53.257	25.422	28.121
		Mean	73.145	46.754	52.634
	600	Min	28.339	23.142	24.086
		Mean	36.114	25.613	25.802
	1200	Min	28.948	23.541	20.903
		Mean	26.160	20.640	18.011

In Figures C.7, C.8, C.9 and C.10, a number of additional experiments are shown for various grid search results from phase one and two of Section 6.3.6. In particular, the experiments from Table 6.12 are plotted, followed by the phase two experiments from Table 6.13 that used the 'uniform' mutation type. In Figure C.7, it can be seen that there is an improvement in terms of standard deviation of the final ΔV, however, no significant drop in the minimum ΔV is observed. In Figure C.8, different mutation types are used, which results in a minimal change, with only a few outlier islands. A slight increase in convergence can be observed from Figure C.9a to Figure C.9b, whereas there is a formidable decrease in convergence from Figure C.9b to Figure C.9c.

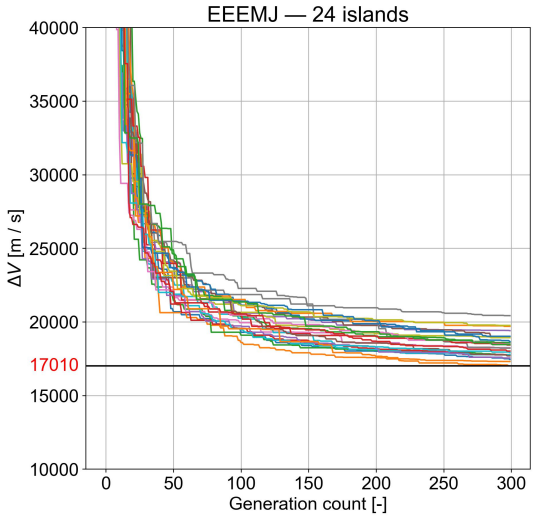
In Figure C.10, different mutation probabilities are plotted for the 'uniform' mutation type. It can be seen that any mutation probability above 8% results in a significant degradation of the minimum ΔV. The higher mutation probability can be recognised by the increase in large ΔV drops from one generation to another, implying the mutation of at least one design variable.



(a) The 'Benchmark' experiment with a mutation probability of 0.02.

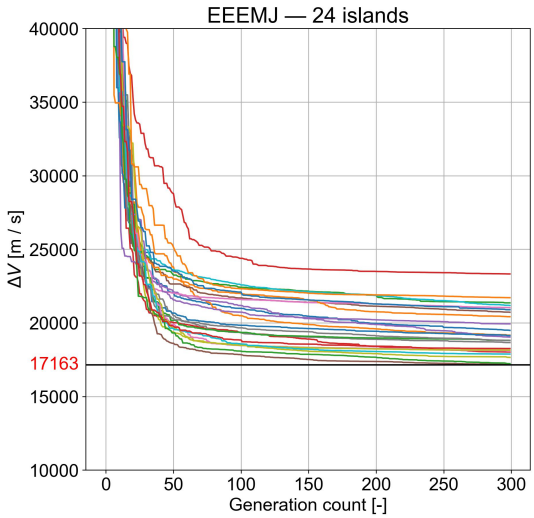


(b) The 'm0.08_mut-poly_cr0.9' experiment with a mutation probability of 0.08.

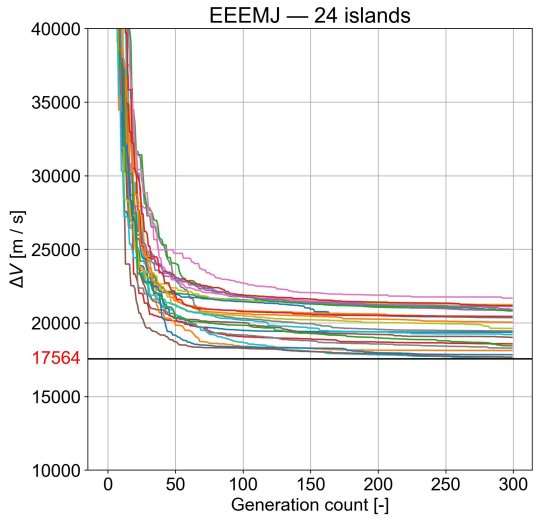


(c) The 'm0.16_mut-poly_cr0.9' experiment with a mutation probability of 0.16.

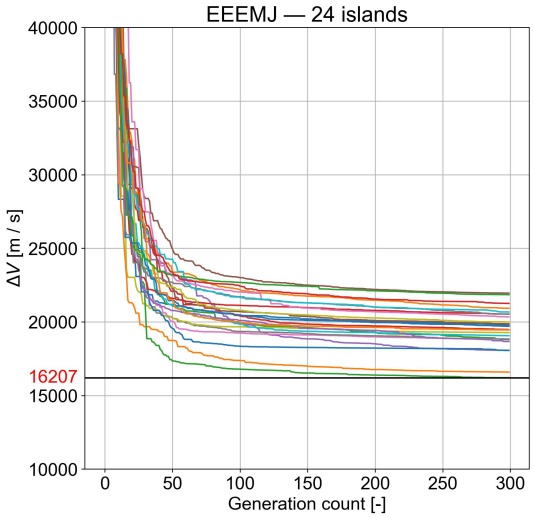
Figure C.7: ΔV per generation for various mutation probabilities.



(a) The 'Benchmark' experiment with a mutation type of 'polynomial'.

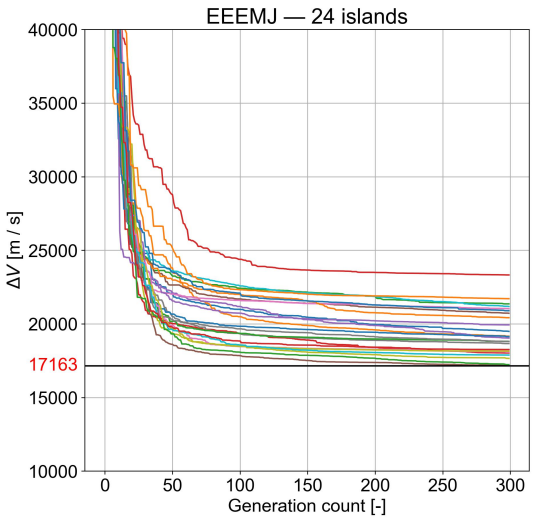


(b) The 'm0.02_mut-gaus_cr0.9' experiment with a mutation type of 'Gaussian'.

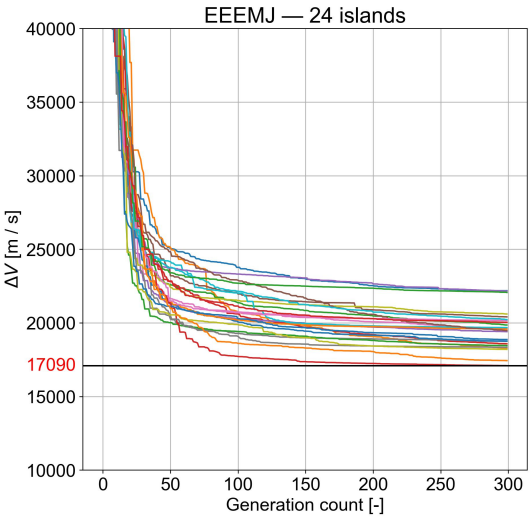


(c) The 'm0.02_mut-uni_cr0.9' experiment with a mutation type of 'uniform'.

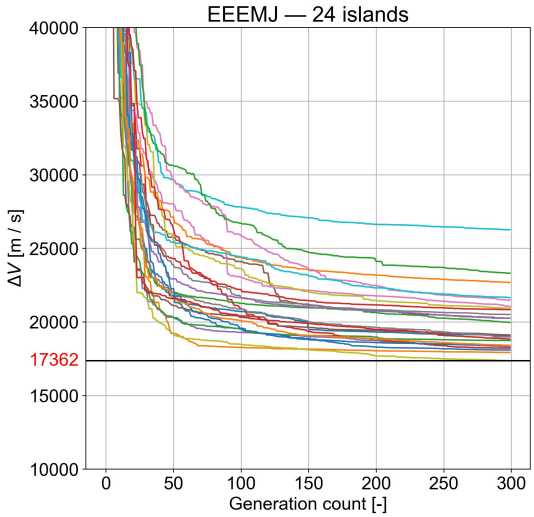
Figure C.8: ΔV per generation for various mutation types.



(a) The 'Benchmark' experiment with a crossover probability of 0.9.

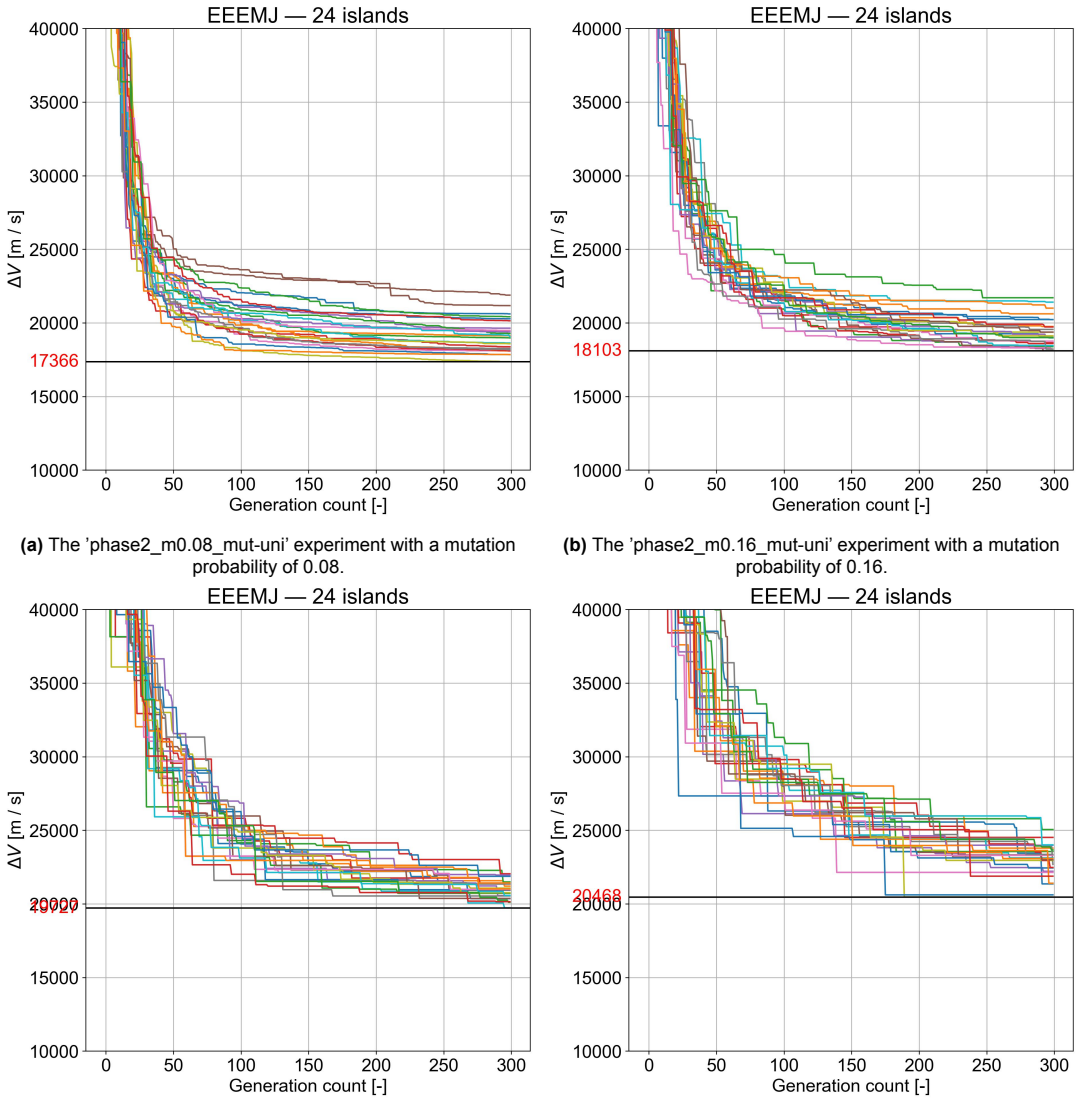


(b) The 'm0.02_mut-poly_cr0.5' experiment with a crossover probability of 0.5.



(c) The 'm0.02_mut-poly_cr0.2' experiment with a crossover probability of 0.2.

Figure C.9: ΔV per generation for various crossover probabilities.



(a) The 'phase2_m0.08_mut-uni' experiment with a mutation probability of 0.08.

(b) The 'phase2_m0.16_mut-uni' experiment with a mutation probability of 0.16.

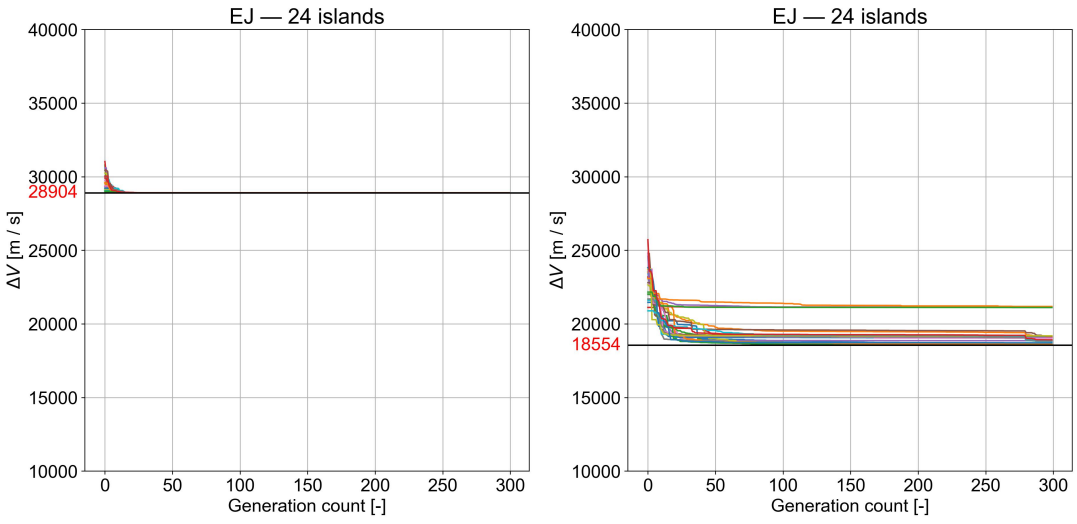
(c) The 'phase2_m0.24_mut-uni' experiment with a mutation probability of 0.24.

(d) The 'phase2_m0.32_mut-uni' experiment with a mutation probability of 0.32.

Figure C.10: ΔV per generation for uniform mutation with more wider mutation ranges.

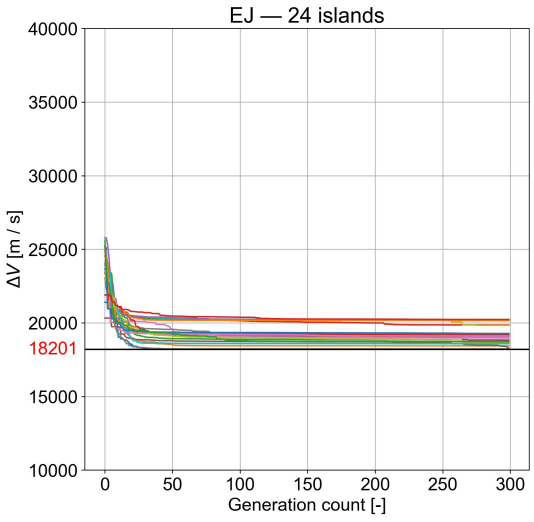
C.3. Free coefficient count

This section includes figures that show analogous results to Section 6.3.4 using zero, one, and two free parameters, but for the other sequences from [Fan et al. 2021]. Specifically, Figures C.11, C.12 and C.13 show the ΔV per generation resulting from the LTTO for the EJ, EMJ, and EEMJ sequence, respectively.



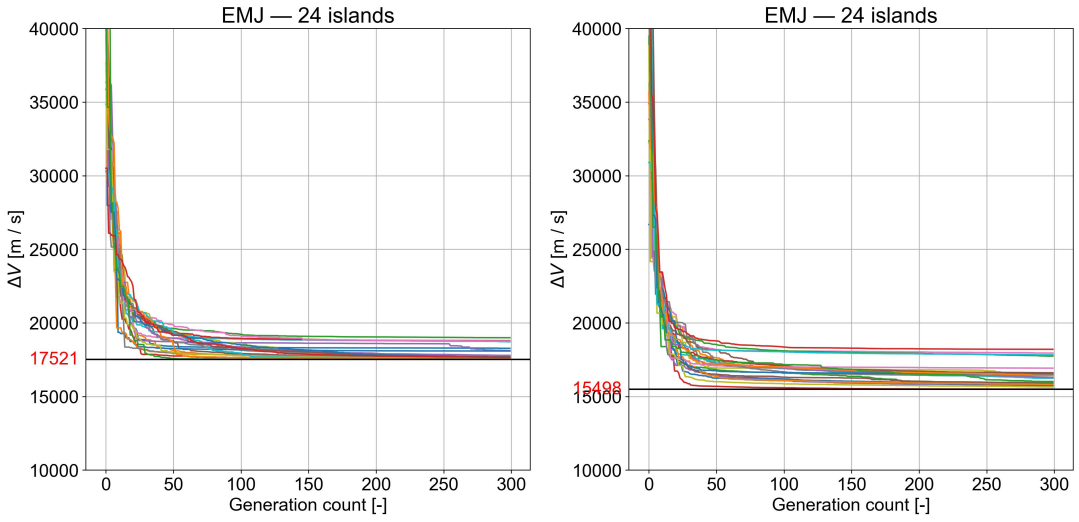
(a) Zero free parameters.

(b) One free parameter.



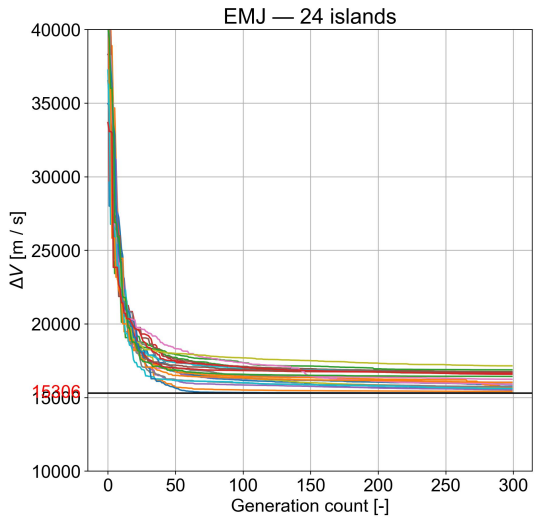
(c) Two free parameters.

Figure C.11: ΔV per generation comparison between zero, one, and two free parameters for the EJ transfer.



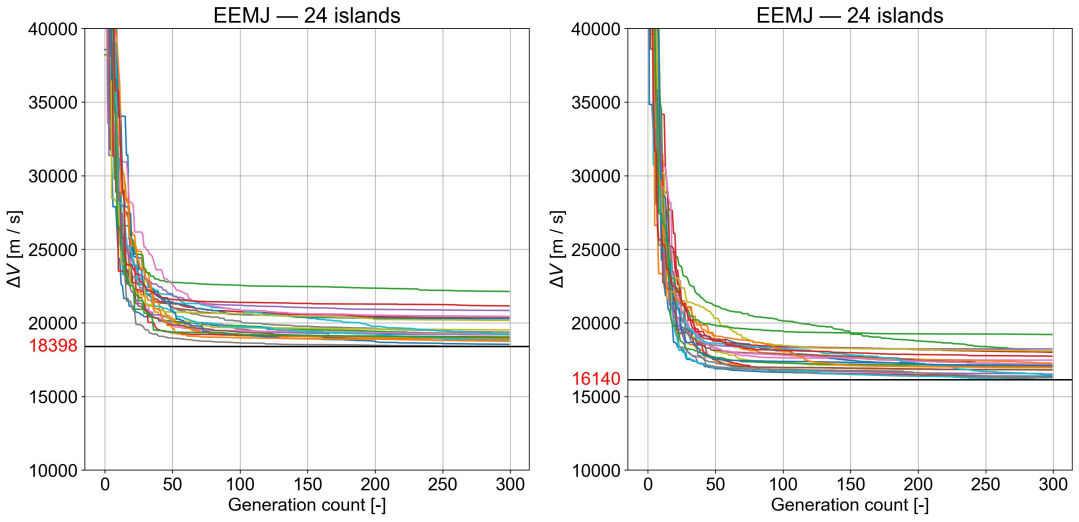
(a) Zero free parameters.

(b) One free parameter.



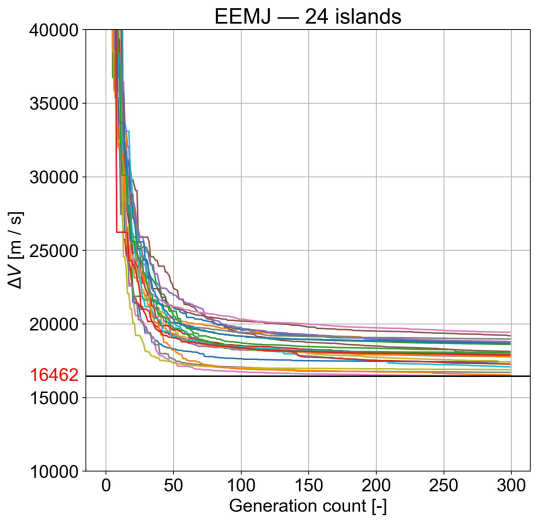
(c) Two free parameters.

Figure C.12: ΔV per generation comparison between zero, one, and two free parameters for the EMJ transfer.



(a) Zero free parameters.

(b) One free parameter.

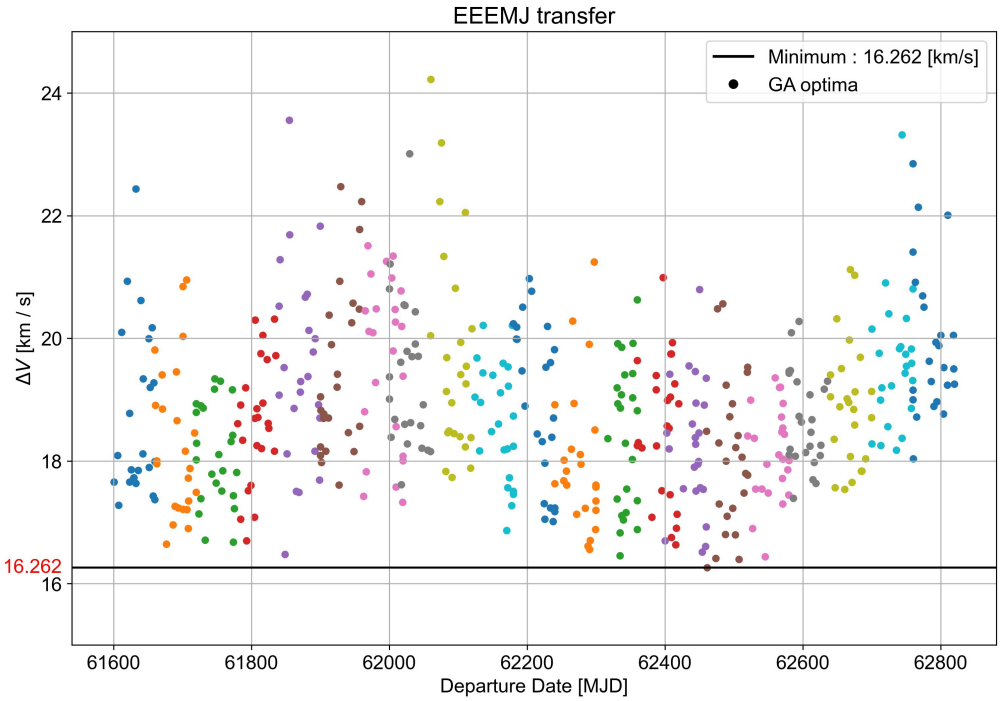


(c) Two free parameters.

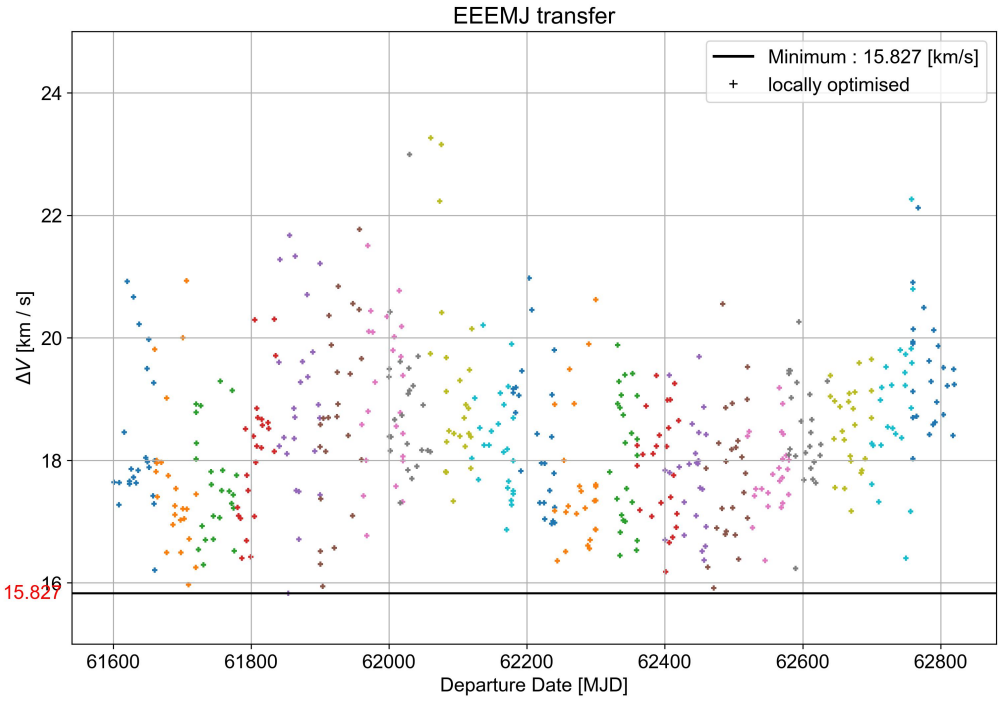
Figure C.13: ΔV per generation comparison between zero, one, and two free parameters for the EEMJ transfer.

C.4. Local optimisation

In Figure C.14, the normal optimisation grid search can be compared to the locally optimised grid search. It can be seen that the behaviour that has been explained in Section 6.3.7 is consistent when locally optimised. Specifically, the trend in optimality per departure date and the difference in ΔV between the best and worst island at a given departure date are almost identical.



(a) Grid search from 61400-62800 MJD.



(b) Locally optimised grid search from 61400-62800 MJD.

Figure C.14: Comparison of grid search with and without local optimisation step for EEEMJ transfer.

C.5. EN testing grid search results

This section provides additional results for the Earth-Neptune transfer case, defined in Section 6.4. The locally optimised grid searches are shown for the EN, EJN, EMJN, EEMJN, and EEEMJN transfer. Both 1200-day interval windows are plotted.

For the EN transfer in Figure C.15, the ΔV values converged well, similar to the EJ transfer from [Fan et al. 2021]. The average ΔV found was $\approx$ 73.7 km/s. In Figure C.16, the periodicity is found to be similar to those found in Figure 6.3a. However, the relevant synodic periods are Earth-Jupiter at $\approx$ 399 days and Jupiter-Neptune at $\approx$ 12.8 years – or 4672 days. The EJ periodicity can be observed between the various groups of optimal points. A trend throughout the groups can be observed which may hint at the JN periodicity, however, this is not confirmed. The minimum found is 25.04 km/s, which is a 66% decrease in ΔV as compared to the direct EN transfer. The local optimisation does not show significant improvement, indicating that it still converges relatively well, albeit to slightly different local optima. Interestingly, the optimal trajectories were only found for short portions of the departure date interval, which indicates that all islands converge to the same departure date window. Similar behaviour can be observed for the longer sequences shown in Figures C.17, C.18 and C.19, with an analogous analysis of the relevant synodic periods and location of the optima.

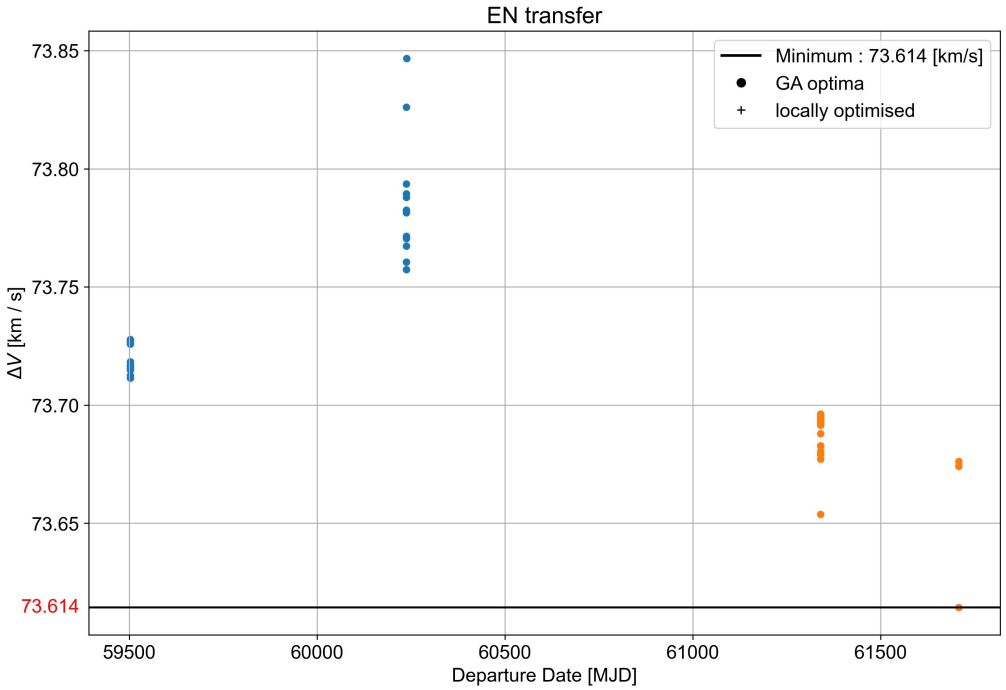


Figure C.15: Grid search for EN transfer.

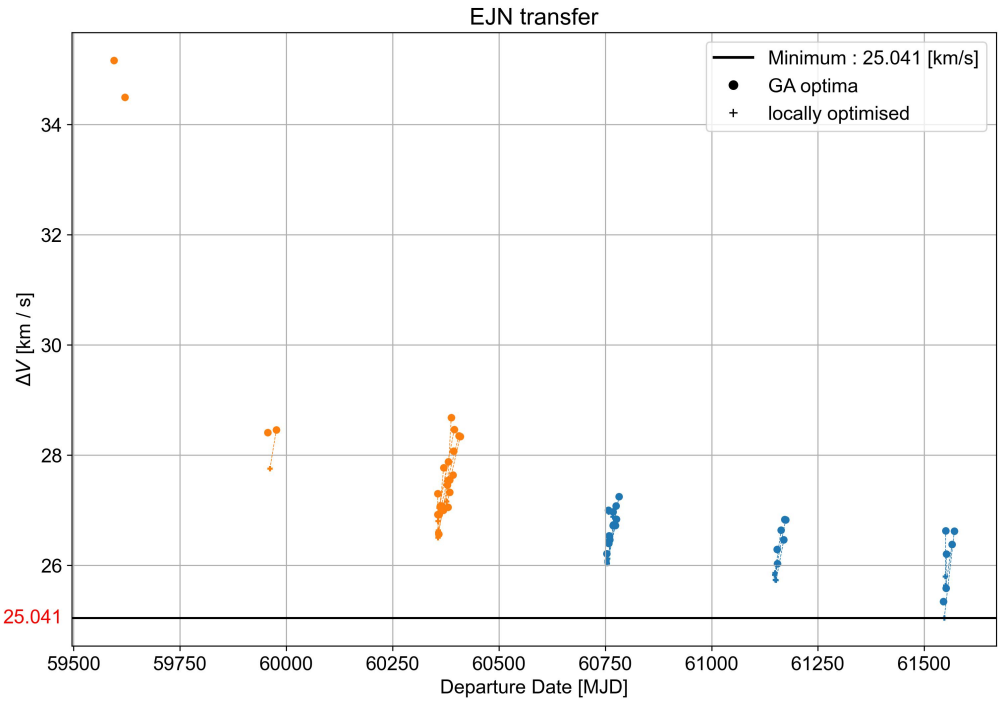


Figure C.16: Grid search for EJN transfer.

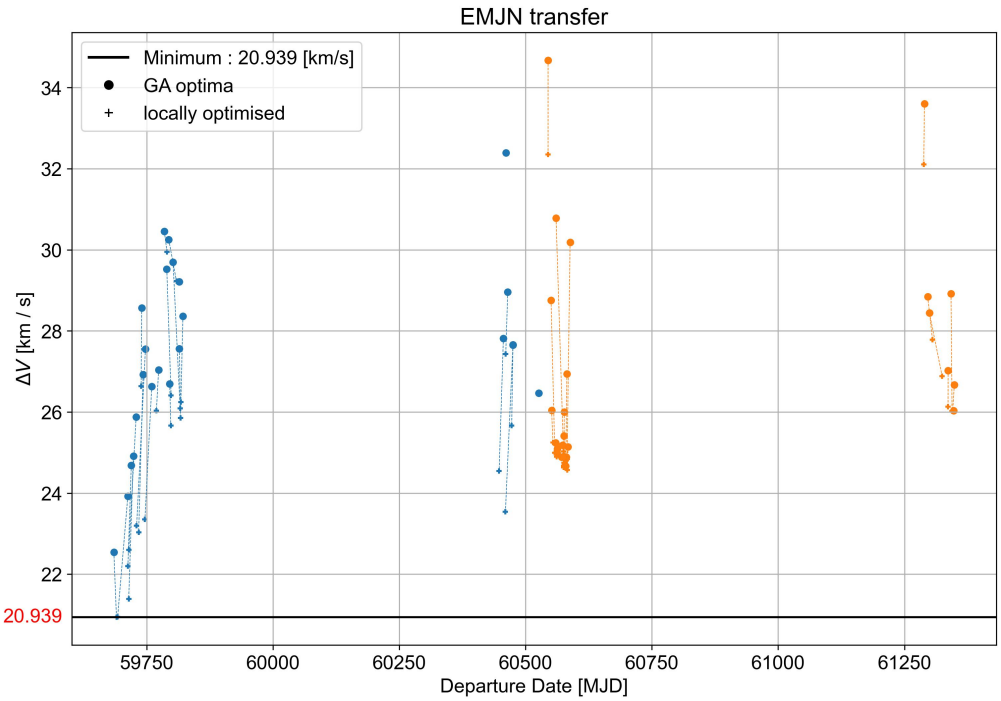


Figure C.17: Grid search for EMJN transfer.

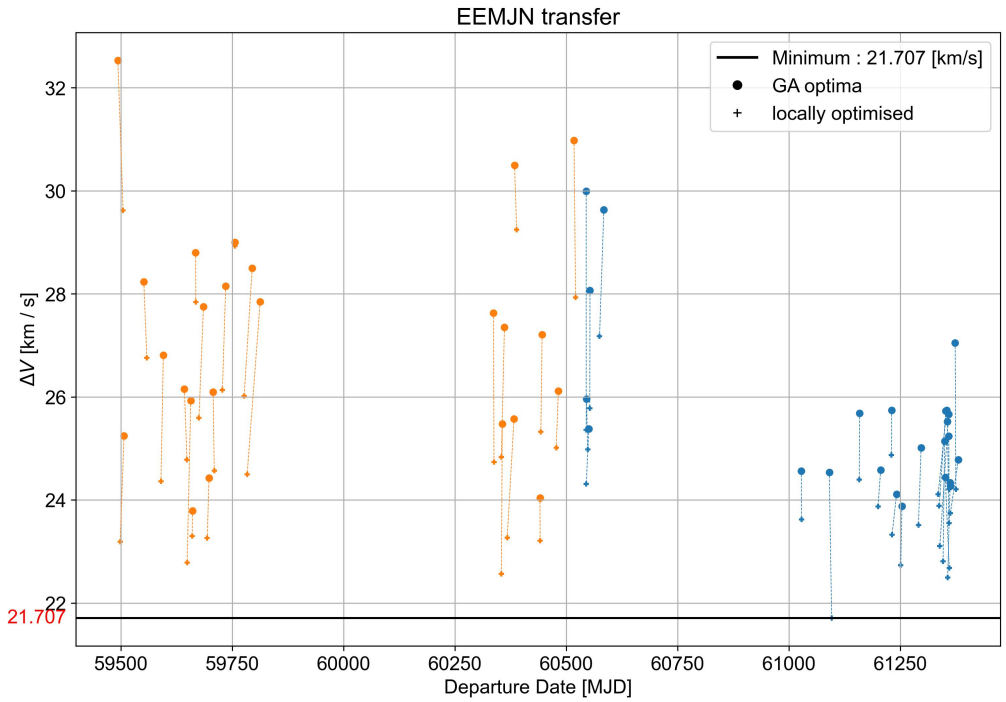


Figure C.18: Grid search for EEMJN transfer.

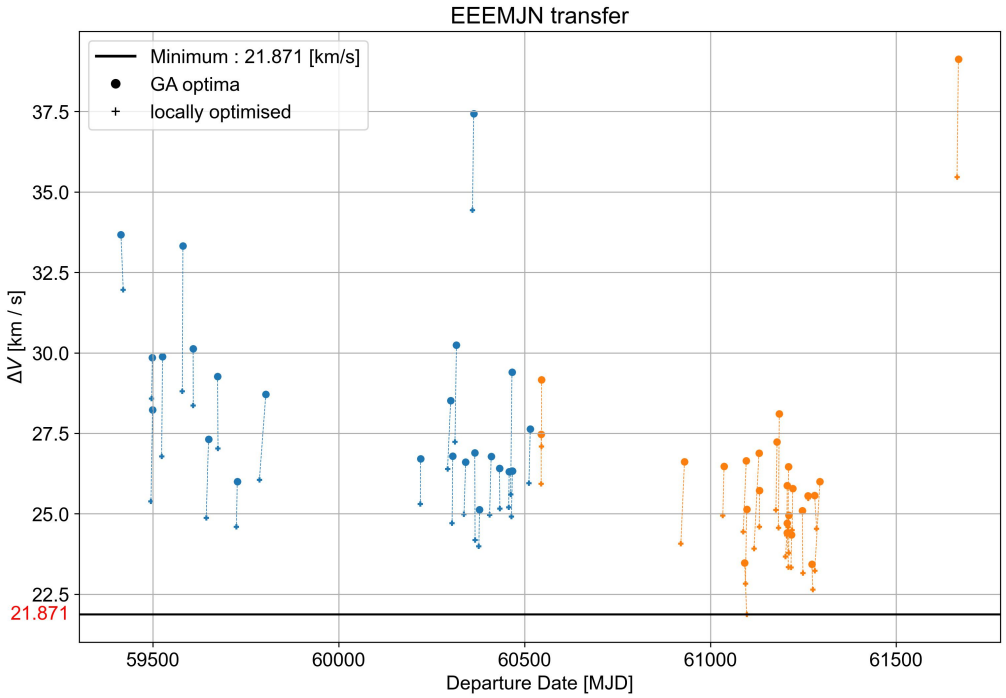


Figure C.19: Grid search for EEEMJN transfer.

D

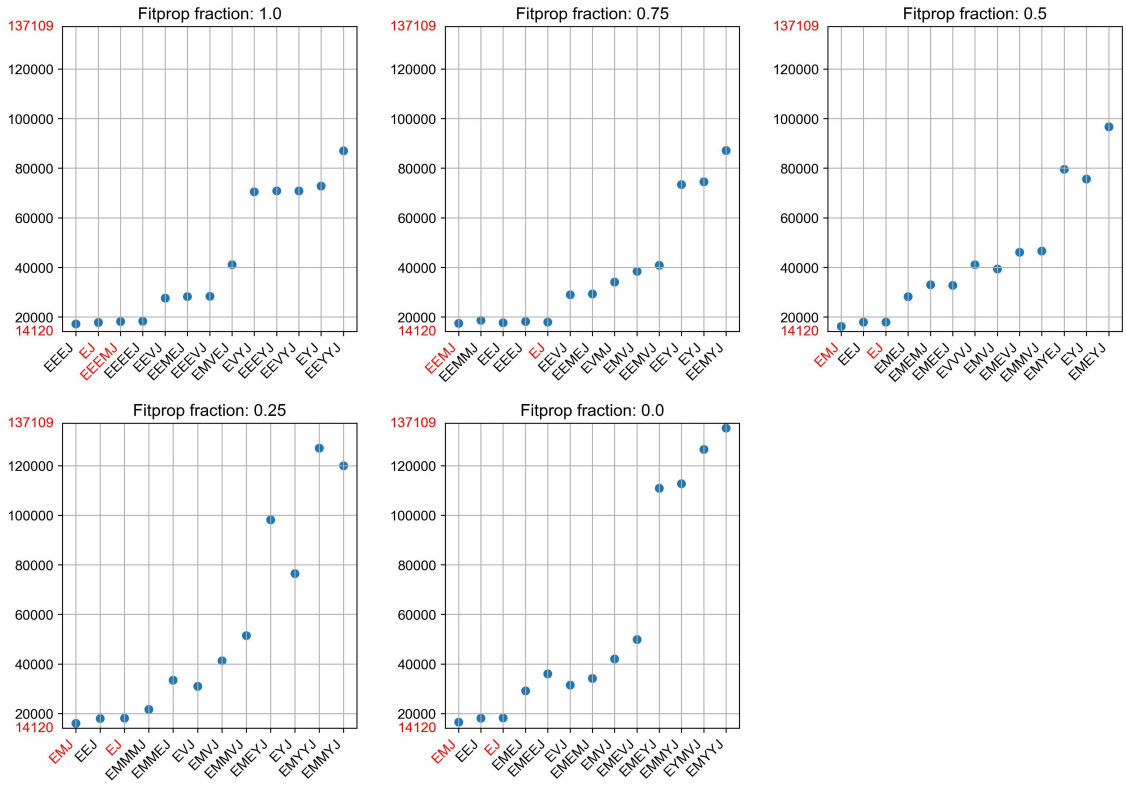
MGASO tuning results

D.1. Iteration one

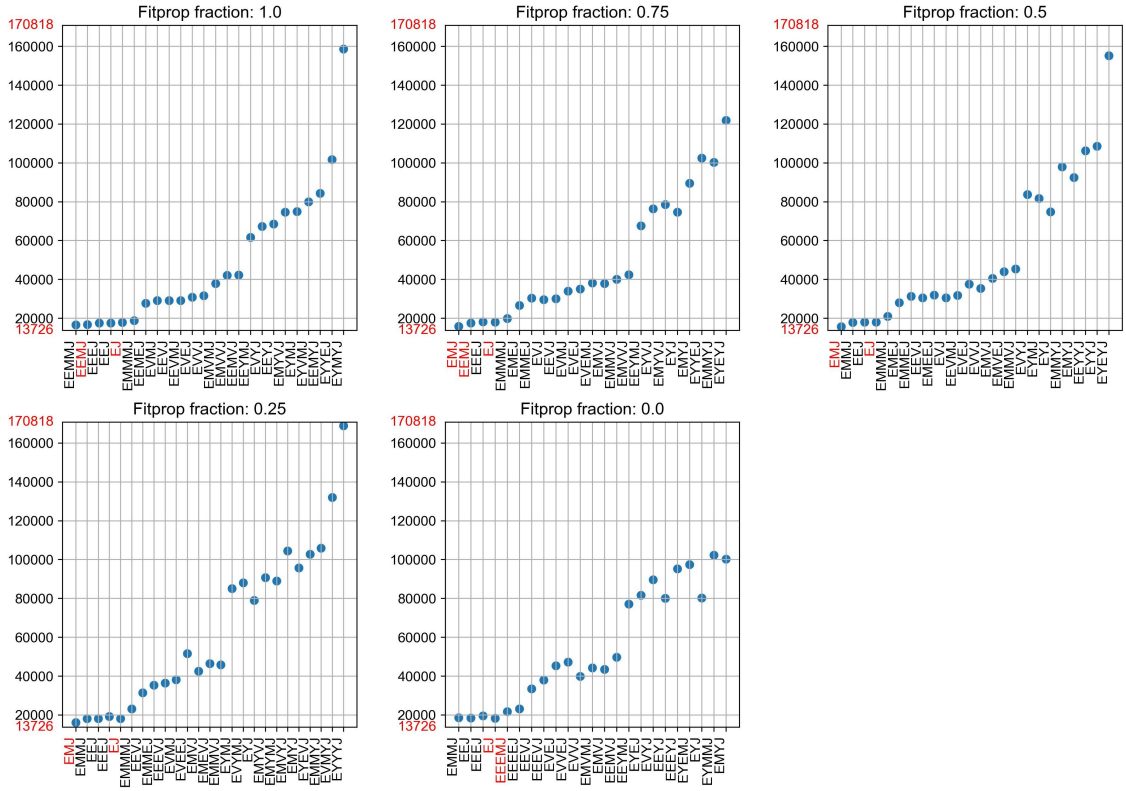
A number of runs were performed with no seed, so the fraction of the sequences that is evaluated was different every time, and nothing can therefore be said on the effect of fp on the results. Note that the results in this section were obtained in an earlier phase of the thesis work with the insight at the time.

The grid search was completed with q values of 0.08 and 0.30, shown in Figures D.1a and D.1b. The fp values were 1.0, 0.75, 0.5, 0.25, and 0.0. These parameters are chosen based on the tuning process from Part II. Specifically, the number of CPUs is maximised based on the maximum allowed CPUs of DelftBlue. The ips is chosen based on the value used in the Part II, but subtracted by one so that two sequences can be evaluated in parallel using the 46 CPUs. Two free parameters are chosen because that gives the highest robustness. As far as the RPS is concerned, there is no real trend. The only pattern is that $q = 0.3$ leads to MMM three out of five times. However, due to the randomness of the chosen sequences for each run, no conclusion can be made over the differences between various fp values. It also is not immediately clear whether – independently of the fp value – there are consistent results or not.

In Figures D.1a and D.1b, an increasing trend can be seen from highly fit sequences to low-fitness sequences. For each fp value, there are between three and five sequences that all have a very low fitness. Ideally, these sequences can be distinguished in optimality, because then the best sequence can be determined. On the x-axis, the sequences that are also tested by [Fan et al. 2021] are shown. It is a good sign that in each test, the sequences that are evaluated by [Fan et al. 2021] have a high fitness. The direct transfer in this case does not really count as it is evaluated each time. It is also a good sign that there are sequences with a lower ΔV than the direct transfer, meaning that the tuning in Part II have resulted in improvements and are resembled in the mgaso algorithm. The last observation that shows some consistency is that in the runs where there were overlapping sequences, they perform equally well in each. For example, the EMJ transfer that is evaluated in the third, fourth, and fifth figure. Do to the non-consistent nature of these optimisations there is no way to tell the effect of fp. These runs therefore contain only very limited information. With these results, there are no consistent results. As a side note, the run times for the various q values were 12:45:00 and 20:31:00 for 0.08 and 0.3 respectively.



(a) $q = 0.08$.

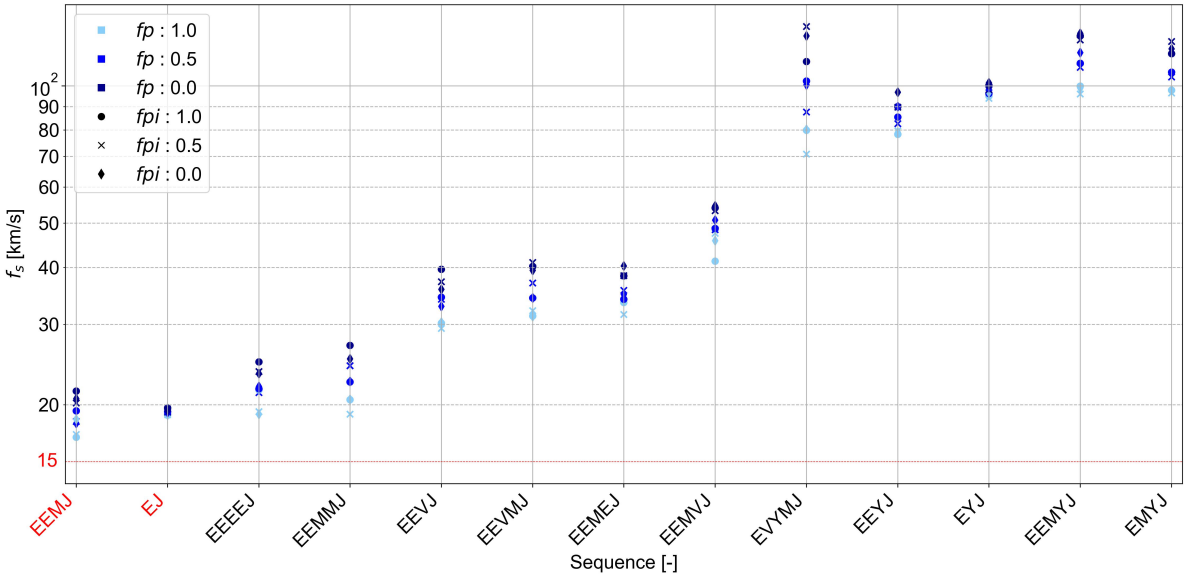


(b) $q = 0.30$.

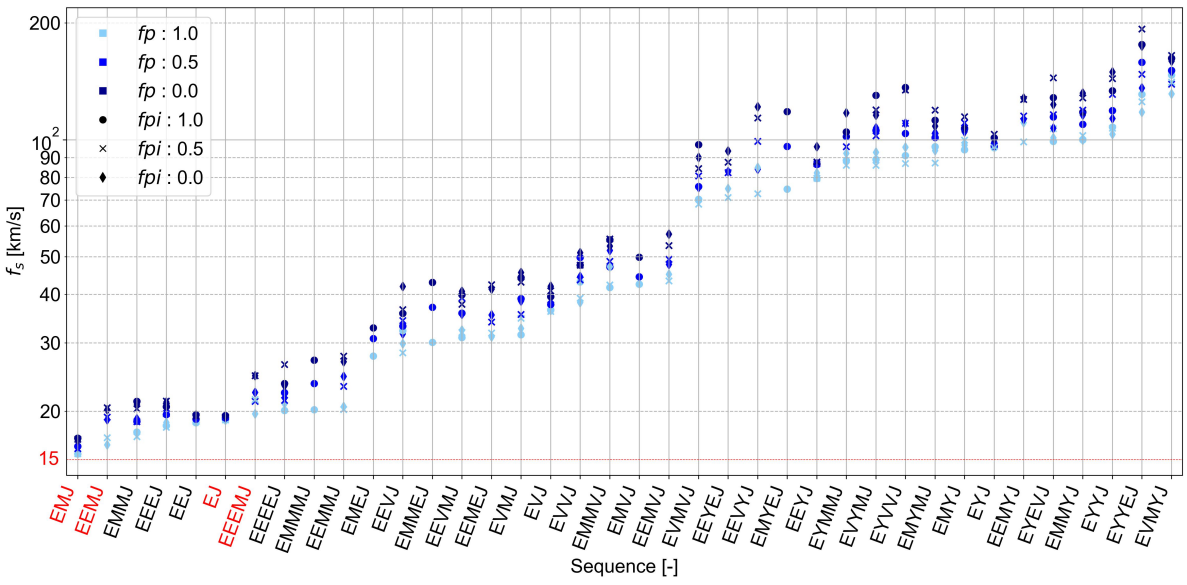
Figure D.1: f_s values for all sequences for various fp (fitprop) values and two q values.

D.2. Iteration two

This subsection presents extra results found in Section 8.2.2. In particular, Figure D.2 includes the sorted sequences for the grid search with q values of 0.08 and 0.30 for all sequences, and Figure D.3 shows the low-f_s group of Figure D.2.

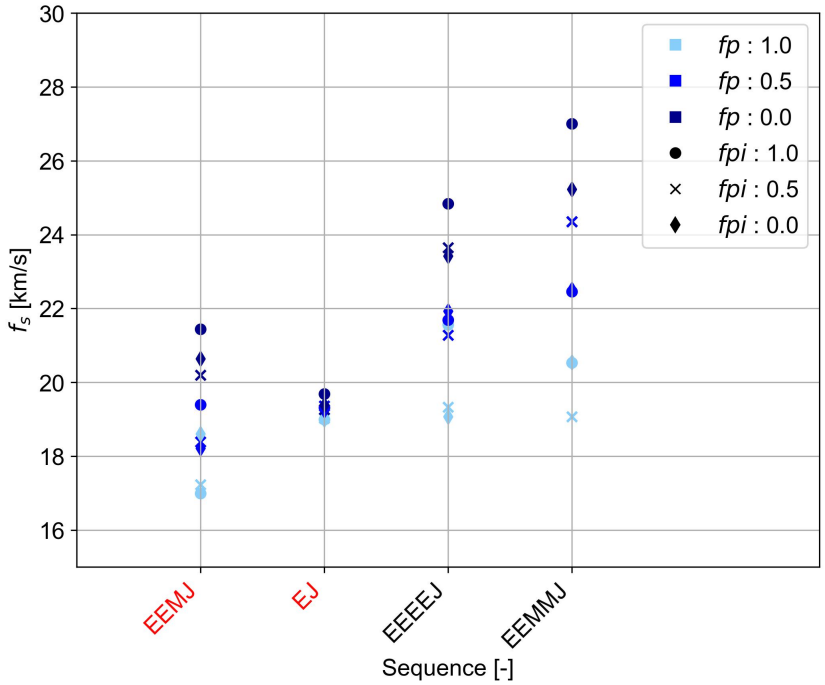


(a) $q = 0.08$.

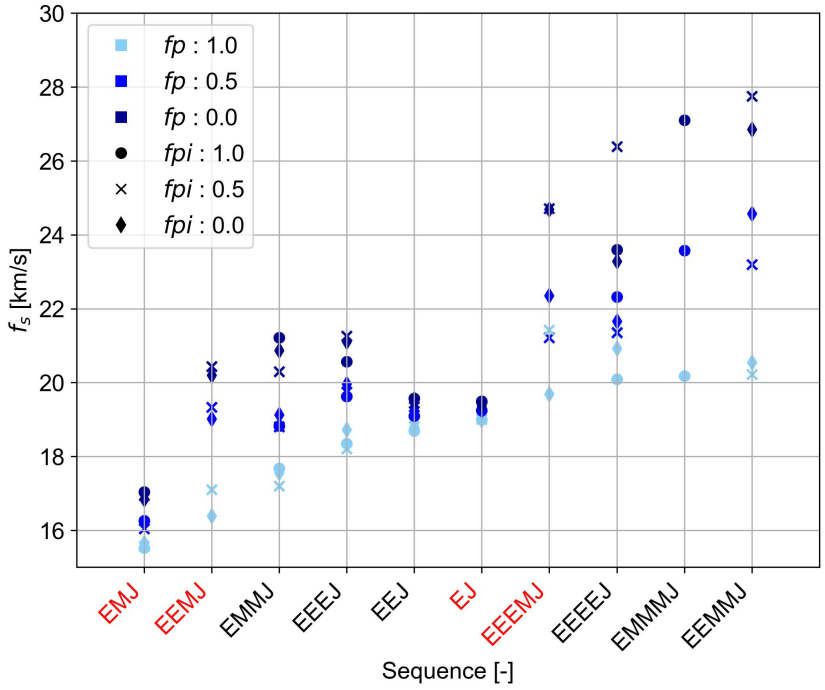


(b) $q = 0.30$.

Figure D.2: f_s of all evaluated sequences for two q values.



(a) q=0.08.



(b) q=0.30.

Figure D.3: f_s of low-f_s group for two q values.

E

Dynamic bounds

This chapter introduces some additional theory on the implementation of the dynamic bounds, which was a recommendation given in Chapter 12. The goal of this particular recommendation is to make the bounds insensitive to the specific transfer case considered. A number of quantities can be considered for dynamic bounds: ToF, β, θ_g, ϕ_g, and shaping function.

E.1. Time of flight

A dynamic ToF bound is desired as the ToF bound differs greatly per transfer leg. The dynamic ToF bounds are based on [Englander and Conway 2017], in which a pseudo period is defined – shown in Equation (E.1). The pseudoperiod is meant predominantly to take account of highly eccentric orbiting bodies, which is not the scope of this thesis. Nevertheless, an automated flight time bound is defined as seen in Table E.1 which could be used in future work.

$$\tilde{P} = 2\pi\sqrt{\frac{r_a^3}{\mu}} \tag{E.1}$$

where $\tilde{P}$ is the pseudo period and r_a is the apocenter of the body.

Table E.1: Automated choice of phase flight-time bounds [Englander and Conway 2017].

Case [-]	Lower bound [s]	Upper bound [s]
Repeated flyby of same planet	$\tilde{P}$/2	20$\tilde{P}$
Outermost body has a $<$ 2 LU	0.1 min($\tilde{P}_1$, $\tilde{P}_2$)	2.0 max($\tilde{P}_1$, $\tilde{P}_2$)
Outermost body has a $\geq$ 2 LU	0.1 min($\tilde{P}_1$, $\tilde{P}_2$)	max($\tilde{P}_1$, $\tilde{P}_2$)

LU is equal to Astronomical Unit (AU) in [Englander and Conway 2017].

E.2. GA angles

Besides the ToF bound, β could be automated as well by allowing a certain range depending on the inclination of the departure and arrival bodies. However, as was seen in Section 6.3.3, β takes on values in the whole design space, so this is left as a future endeavour. θ can be dynamically constrained properly. Specifically, depending on whether the arrival body of any given leg is an inner or outer transfer, the incoming velocity, relative to the planet velocity can be specified. For instance, if an MJ transfer is used, the spacecraft will not approach the body from outside Jupiter.

E.3. Shaping functions

The shaping functions can also be changed depending on the transfer. [Gondelach and Noomen 2015] tuned shaping functions for single leg low-thrust arcs to various arrival bodies. This could be implemented as follows:

Table E.2: Automated choice of phase flight-time bounds [Englander and Conway 2017].

Case [-]	Shaping function configuration [-]
$a_{arr} = a_{dep}$	Earth-Mars shaping functions
$a_{arr} > a_{dep}$	Earth-Mars shaping functions
$a_{arr} < a_{dep}$	Earth-Mercury shaping functions

The Earth-Mars shaping functions are given in Table 4.3. The Earth-Mercury shaping functions are shown below in Table E.3. The dynamic shaping functions from Table E.2 can be further improved by investigating the performance of other shaping functions.

Table E.3: Recommended base functions for an EY transfer [Gondelach and Noomen 2015].

Type of function	Axis	Name and equation		
Base	R + N	**C**	**Pow**	**Pow2**
		c_1	$+c_2 t$	$+c_3 t^2$
	A	**CosR5**	**P3CosR5**	**P3SinR5**
		$c_1 \cos(2\pi t(N+0.5))$	$+c_2 t^5 \cos(2\pi t(N+0.5))$	$+c_3 t^5 \sin(2\pi t(N+0.5))$
Additional	R + N	**PSin05**	**PCos05**	
		$c_4 t \sin(0.5t\pi)$	$+c_5 t \cos(0.5t\pi)$	
	A	**P4CosR5**	**P4SinR5**	
		$c_4 t^6 \cos(2t\pi(N+0.5))$	$+c_5 t^6 \sin(2t\pi(N+0.5))$	

R is radial, N is normal, and A is axial. C is Constant. Pow or P is Power. Px or Powx where x is the exponent. CosR5 where Cos is a cosine. R refers to the factor N. R5 refers to (N+0.5); 05 refers to the factor 0.5