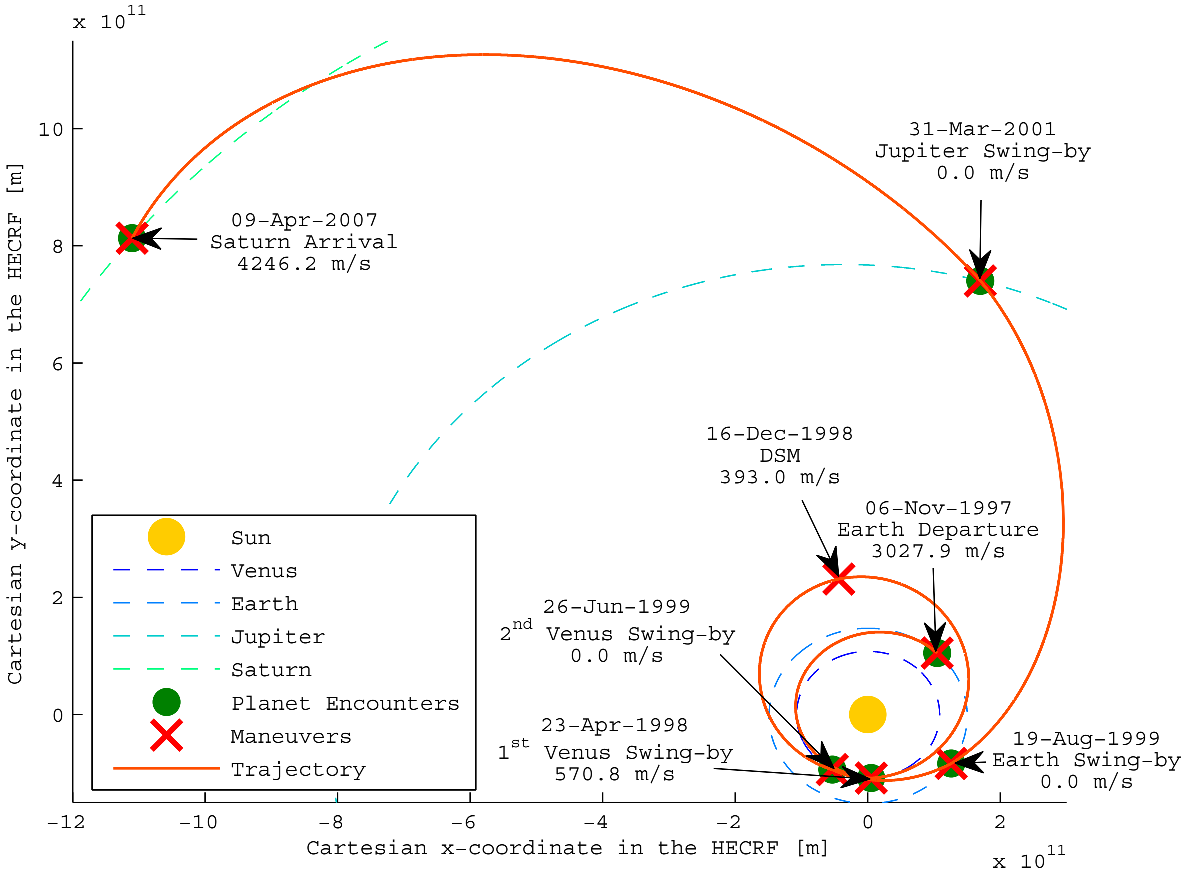
Optimization of Space Trajectories Including Multiple Gravity Assists and Deep Space Maneuvers



Thesis Report

Author: Paul Musegaas

Supervisor: Ron Noomen

Institute: Delft University of Technology

Date: December 20, 2012

TU Delft Delft University of Technology

Challenge the future

Preface

The final part of the study to become an aerospace engineer at the TU Delft consists of a Master thesis project. The project is assigned 42 ECTS and is the graduation project to become a Master of Science (MSc).

This report contains very interesting results for anyone interested in optimization of high-thrust interplanetary trajectories. Although this thesis looked especially into trajectories to Saturn, many conclusions and strategies are likely valid for other trajectories as well. Those interested in more background information on the optimization of high-thrust trajectories are encouraged to read Part I of this thesis, as well as the literature study that preceded this thesis [Musegaas, 2012]. Those interested in the software development required for the optimization of these trajectories are referred to Part II. Part III discusses the tuning process used in this thesis, which proved to be very effective for obtaining good settings of the optimization algorithms. Subsequently Part IV discusses the results obtained in this thesis, both related to the modeling as well as to the optimization of the trajectories.

I would very much like to thank my supervisor, Ron Noomen. We have had weekly meetings throughout a large part of my graduation project. These have guided me to investigate the right aspects during my research. Also they have helped me very much in keeping up my motivation. It was a pity that we could not hold these meetings in the later stages of my thesis project where I obtained my results.
Also I would like to thank Dario Izzo for all his efforts both on setting up GTOP as well as on setting up PaGMO. Without these toolboxes this thesis would definitely not have achieved the same results. Especially GTOP is acknowledged for its usefulness in comparing different strategies of optimizing difficult high-thrust trajectories.
Also I would like to thank the students at the ninth floor for making the 8 months spent on this thesis very worthwile. The Space Bars and massive music quizzes were very entertaining. Finally I would like to thank my girlfriend for supporting me during the hard work.

Table of Contents

List of Symbols

The following table gives an overview of the symbols that are used throughout the report. Vectors are printed **bold** in the report. So **r** refers to the vector 'r', whereas r refers to the magnitude of 'r'. Common symbols are used for the summations ($\sum$), derivatives ($\frac{\delta}{\delta t}$, $\dot{x}$ or $f'()$), functions ($f(x)$), difference operators Δ, etcetera.

Sometimes a hat symbol is used to differentiate between a special value of a variable, instead of the variable in general. For instance $\hat{x}$ may be a certain solution of x, whose meaning is clear from the context.
Also sometimes a tilde symbol is used to distinguish between velocity definitions. $\tilde{\boldsymbol{V}}$ may refer to the velocity with respect to the target planet, whereas $\boldsymbol{V}$ is used for the accompanying heliocentric velocity. Again this is explained also in the surrounding text.

First the Roman symbols are shown, then Greek symbols and finally the indices are shown. Note that for some variables the subscript is always present, in which case they are included in the symbol list as one.

Roman

Symbol	Meaning	Unit
A	Algorithm	[-]
a	Semi-major axis	$[m]$
B	Impact parameter	$[m]$
b_{incl}	Rotation angle of a gravity assist	$[rad]$
CR	Crossover constant	[-]
E	Elliptic eccentric anomaly	$[rad]$
e	Eccentricity	[-]
F	Hyperbolic eccentric anomaly	$[rad]$
F	Scale factor (DE)	[-]
i	Unit vector	[-]
iun	Maximum # of iterations without improvement before restart (in IDEA)	[-]
M	Elliptic mean anomaly	$[rad]$
$\overline{M}$	Hyperbolic mean anomaly	$[rad]$
m	Counter	[-]
m	Mutation rate	[-]
m_{noImp}	Maximum number of runs without improvement (TIA)	[-]
m_{runs}	Number of runs in determining algorithm performance	[-]
m_{type}	Mutation type	[-]
m_{width}	Mutation width	[-]
n	Number of evaluations	[-]
$n_{95\%}$	Number of evaluations to optimize with 95% probability	[-]
n_{eli}	Number of elite individuals	[-]
n_p	Population size	[-]
n_{eval}	Number of evaluations to optimize with 95% probability	[-]
$neigh_{type}$	Neighbor type (PSO)	[-]
$neigh_{param}$	Neighbor parameter (PSO)	[-]

Symbol	Meaning	Unit
$p_{success}$	Success rate	[-]
r	Random number	[-]
r	Radius	$[m]$
r	Dimensionless DSM radius (PF model)	[-]
r	Position in space (from origin)	$[m]$
r_i	Random individual in DE, i is an index (1,2..)	[-]
sel_{type}	Selection type	[-]
T	Time of flight of an interplanetary leg	[s]
T	Temperature (in SA/ASA)	[-]
t	Time	$[MJD]$ or $[s]$
tol_{conv}	Tolerance for convergence (in IDEA)	[-]
V	Velocity	$[m \cdot s^{-1}]$
V_∞	Magnitude of velocity relative to departure planet	$[m \cdot s^{-1}]$
v	Value defining direction of V_∞ in 1st DSM leg	[-]
v_{coeff}	Velocity coefficient (PSO)	[-]
var	Variant of PSO	[-]
x	x-coordinate	$[m]$
x	Decision vector for trajectory	various
y	y-coordinate	$[m]$
z	z-coordinate	$[m]$

Greek

Symbol	Meaning	Unit
α	Bending angle (gravity assist)	$[rad]$
β	Angle between r_p and V'_{pl} (gravity assist)	$[rad]$
Δ	Bubble size (in IDEA)	[-]
Δ_{fit}	Fitness threshold before resetting m_{noImp} (TIA)	[-]
ΔV	Magnitude of the velocity change of a maneuver	$[rad]$
δ_c	Margin distance for restart (in IDEA)	[-]
η	Fraction of time of flight at which DSM occurs	[-]
η_1	Cognitive component coefficient (PSO)	[-]
η_2	Social component coefficient (PSO)	[-]
θ	True anomaly	$[rad]$
θ	Angle defining direction of V_∞ in 1st DSM leg	$[rad]$
θ	In-plane angle defining the DSM location (PF model)	$[rad]$
μ	Gravitational parameter	$[m^3 \cdot s^{-2}]$
π	Constant (3.14159265)	[-]
ρ	Average distance between individuals in a population	[-]
ϕ	Angle defining direction of V_∞ in 1st DSM leg	$[rad]$
ω	Particle's inertia or constriction coefficient (PSO)	[-]

In the next table, the most frequently used indices are given. Note that of course numbers are also frequently used. The meaning of the number varies and should be clear from the context.

Indices

Index	Meaning
av	Average
dsm	Of the DSM
f	Final
g	Global
i	Initial
i	Index in iteration
i	Individual (number) in metaheuristics
in	Incoming (before gravity assist)

Index	Meaning
j	Index in iteration
k	Index in iteration
l	Local
max	Maximum
out	Outgoing (after gravity assist)
p	At pericenter
pl	Of the planet
rem	Remaining
S	Of the Sun
SoI	Of the sphere of influence
s	Of the spacecraft
tot	Total
x	In the x-direction
y	In the y-direction
z	In the z-direction
∞	Hyperbolic excess velocity (velocity at infinity)
$-$	Before a maneuver
$+$	After a maneuver
$\tilde{}$	Of a different reference frame (planetocentric instead of heliocentric)

List of Abbreviations

The following table gives an overview of the abbreviations used in this report. It is noted that apart from these abbreviations also the planetary sequence is often abbreviated. For this the first letter is used of the planetary sequence. Mercury is an exception to this, where Me is used instead. EVVMe for example stands for an Earth-Venus-Venus-Mercury trajectory.

Abbreviation	Meaning
ACT	Advanced Concepts Team
A&S	Astrodynamics & Space missions (chair at TU Delft)
ASA	Adaptive Simulated Annealing
BFGS	Broyden-Fletcher-Goldfarb-Shanno (algorithm)
BOBYQA	Bound Optimization BY Quadratic Approximation
BSD	Berkeley Software Distribution
CMAES	Covariance Matrix Adaptation Evolutionary Strategy
COBYLA	Constrained Optimization BY Linear Approximation
CPU	Central Processing Unit
DE	Differential Evolution
DIRECT	DIvided RECTangles
DSM	Deep Space Maneuver
ESA	European Space Agency
FR	Fletcher-Reeves (algorithm)
GA	Genetic Algorithm
GALOMUSIT	Genetic ALgorithm Optimization of a MUltiple Swingby Interplanetary Trajectory
GNU	GNU's Not Unix!
GSL	GNU Scientific Library
GTOP	Global Trajectory Optimisation Problem
HECRF	non-rotating Heliocentric ECliptic Reference Frame
IDEA	Inflationary Differential Evolution Algorithm
IF	Implicit Filtering
IPOPT	Interior Point OPTimizer
JD	Julian Date
LHS	Latin Hypercube Sampling
MBH	Monotonic Basin Hopping
MC	Monte Carlo
MCS	Multilevel Coordinate Search
MESSENGER	MErcury Surface, Space ENvironment, GEochemistry and Ranging
MGA	Multiple Gravity Assist
MGADSM	Multiple Gravity Assist with Deep Space Maneuvers
MGA-1DSM	Multiple Gravity Assist with 1 Deep Space Maneuver per leg
MGA-1DSM-PF	Multiple Gravity Assist with 1 Deep Space Maneuver per leg adopting a Position Formulation (trajectory model)
MGA-1DSM-VF	Multiple Gravity Assist with 1 Deep Space Maneuver per leg adopting a Velocity Formulation (trajectory model)
MGA-1DSM-VF-UNP	Multiple Gravity Assist with 1 Deep Space Maneuver per leg adopting a Velocity Formulation and assuming UNPowered swing-bys (trajectory model)
MIDACO	Mixed Integer Distributed Ant Colony Optimization

Abbreviation	Meaning
MJD	Modified Julian Date
MPSO	Multiple Particle Swarm Optimization
MS	Multi-Start (algorithm)
MSc	Master of Science
NLOPT	NonLinear OPTimization (software library)
NM	Nelder-Mead (algorithm)
OOP	Object-Oriented Programming
OPTIDUS	OPTimization of Interplanetary trajectories by Delft University Students
PaGMO	Parallel Global Multi-objective Optimizer
PF	Position Formulation
PR	Polak-Ribiere (algorithm)
PSO	Particle Swarm Optimization
SA	Simulated Annealing
SAGAS	Search for Anomalous Gravitation using Atomic Sensors
SAGES	Self-Adaptive Gaussian Evolutionary Strategy
SNOPT	Sparse Nonlinear OPTimizer
SoI	Sphere of Influence
SQP	Sequential Quadratic Programming
TandEM	Titan and Enceladus Mission
TIA	Time Inflationary Algorithm
Tudat	Technical University of Delft Astrodynamics Toolbox
TU Delft	Technische Universiteit Delft; Dutch acronym for Delft University of Technology
VF	Velocity Formulation

Abstract

The optimization of high-thrust trajectories continues to draw attention. A large number of options are namely available to end up at the target planet. The optimization of these trajectories is very difficult, even when making various assumptions to allow the trajectory to be modelled semi-analytically. The search space may be characterized by a large number of minima and is furthermore highly sensitive to small deviations in the decision vector. This report investigates the effects of both the trajectory model as well as the optimization algorithms that are employed to find optimal trajectories. This report focuses on trajectories to Saturn in doing so.

Various options are available to model these trajectories. Maneuvers may be limited to planetary encounters, but the trajectory will likely be suboptimal as a consequence. Deep Space Maneuvers (DSMs) may namely significantly improve the efficiency of the trajectory. Furthermore the option to perform powered swing-bys may be included or excluded by the trajectory model. Trajectories employing DSMs may furthermore be modelled in various ways, among which a Position Formulation (PF) and a Velocity Formulation (VF).

This report shows that the inclusion of DSMs improves the solutions that can be found drastically. The average final mass for 24 trajectories to Saturn was found to be twice as high using models including the option to perform DSMs. Also the advantages of powered swing-bys are found to be significant, resulting in a final mass that was 7.8% higher on average. Furthermore this report shows that a VF model is a better choice than a PF model. The VF model is less sensitive in general and can be optimized easier. Disadvantages of the VF model have led to the proposition of a new trajectory model. If hybridized with legs of a VF model, the proposed model may allow a trajectory to be optimized more easily.

Many options are available for the optimization of these trajectories as well. Metaheuristics have been employed frequently, but many publications did not properly tune their algorithms before applying and comparing them. This report applies a proper tuning scheme to Differential Evolution (DE), Genetic Algorithm (GA) and Particle Swarm Optimization (PSO). Especially for DE this tuning process was found to be very effective, resulting in an algorithm that is more efficient than all other DE algorithms encountered in literature. The performance of GA and PSO was found to be significantly worse than that of DE. The performance of the tuned versions of DE is very competitive with the performance of other efficient algorithms encountered in literature, such as Inflationary Differential Evolution (IDEA) and Monotonic Basin Hopping (MBH).

Also a new algorithm is developed to optimize a large set of standardized trajectory problems available online. The algorithm, Time Inflationary Algorithm (TIA), combines global and local DE searches in a clever way to find the optimal solution. TIA managed to improve 16 of the 24 test cases it was subjected to, thereby proving that it is an effective algorithm in the optimization of interplanetary trajectories.

Chapter 1

Introduction

Background

The optimization of high-thrust interplanetary trajectories continues to draw attention. Because of the complex dynamical system and the large number of options, high-thrust optimization problems are very difficult to solve. Typically perturbations are neglected in first-order analyses and the patched-conics assumption is adopted. This simplifies the problem, but still leaves a large number of options in terms of swing-bys and powered maneuvers. Although other strategies are also used, the problem may be further simplified by fixing the swing-by sequence. These different swing-by sequences may be optimized separately because the number of options is very limited due to the small number of feasible swing-by planets.

A further assumption that is sometimes made, is that maneuvers are only performed near planetary encounters. Because the planetary locations are simply dependent on time, this allows for a relatively easy problem definition in which only the planetary visitation times need to be determined. The trajectories that are obtained in this way can be considered a nice initial estimate, but are in general not representative for the actual trajectories.

A more representative approach is to include the option to perform Deep Space Maneuvers (DSMs). The addition of DSMs however also significantly complicates the optimization process: the number of minima increases significantly and the search space becomes very sensitive.

Various strategies have been developed recently to cope with the increased difficulty of the optimization of trajectories including DSMs. Optimal control theory has been applied to solve the problem, but good results are only obtained if a good initial guess of the solution is available. Other strategies employ semi-analytic trajectory models. Two different trajectory models are employed frequently. The first one adopts a velocity formulation to define the trajectory and assumes unpowered swing-bys. The second model uses a position formulation and assumes powered swing-bys for patching the interplanetary legs. No quantitative studies have been encountered that compare both models and most authors simply use the first model.

Various optimization strategies have been employed to solve the trajectory problems. Many nature-inspired metaheuristics have been employed but their performance differs significantly across publications. Differential Evolution is often mentioned as a good algorithm. Olds et al. [2007] have shown that metaheuristics may be very sensitive to their tuning parameters. Given that most comparative studies did not employ rigorous tuning schemes, it is difficult to make a fair comparison.

Also repeated local searches have been applied with success, especially if these searches were repeated in a smart way. Both local optimization algorithms [Addis et al., 2008], as well as global optimization algorithms [Vasile et al., 2008] can be used for these local searches.

Hybrids between various methods have also been proposed frequently. A fair performance comparison could not be made for these methods based on the various publications. The performance of these hybrids is however likely to be strongly dependent on the performance and tuning of the individual algorithms.

A last approach type is that of repeatedly pruning the search space and building the trajectory incrementally. The performance of this approach can be very good. Disadvantages are that proper knowledge of the problem and the corresponding feasible trajectories is required to define good pruning criteria.

Purpose

This thesis project focuses on six aspects related to the optimization of high-thrust interplanetary trajectories. First of all software will be developed to model these trajectories using different models with a general set-up. Secondly a search space analysis is performed in which the sensitivity of various parameters belonging to different trajectory models will be analyzed. Thirdly a rigorous tuning scheme will be used to identify the best settings of various metaheuristics. The importance of proper tuning will be analyzed. Subsequently, this thesis compares the performances of these metaheuristics after they were subjected to a this tuning scheme. The fifth aspect is a comparison between the three most commonly used trajectory models and the quantification of the gain of both DSMs as well as powered gravity assists. Finally a general strategy is developed which is verified by optimizing a large number of trajectories belonging to an online database of difficult trajectory optimization problems (GTOP).

Structure

This report is divided into four parts. The first part, Heritage, gives a further introduction into the optimization of high-thrust interplanetary trajectories in Chapter 2. Subsequently the goals for this thesis are discussed in more detail in Chapter 3.

The second part, Methods and Software, deals with the software needed for this thesis. Chapter 4 discusses the astrodynamic tools that were developed and tested. Chapter 5 subsequently deals with the development and verification of the various trajectory models. Finally Chapter 6 discusses the optimization software that will be used in this thesis.

The third part, Algorithm Tuning, deals with the tuning process used for the different metaheuristics. The tuning procedure is discussed in Chapter 7. Subsequently Chapter 8 discusses the problems that will be used to tune the algorithms. Chapter 9 then discusses the tuning of Differential Evolution, Chapter 10 that of a Genetic Algorithm, Chapter 11 deals with Particle Swarm Optimization and finally repeated local searches are discussed in Chapter 12.

The fourth part, Results, discusses the results of the various analyses. Chapter 13 deals with the search space analysis. Chapter 14 subsequently compares different optimization algorithms and discusses the effects of the tuning process. Also the performance of the best algorithms is compared with performances encountered in literature. Chapter 15 discusses the optimization of a large number of difficult trajectory problems available in GTOP. Finally a comparison between different trajectory models is made in Chapter 16. Also the effects of the inclusion of both DSMs and powered swing-bys are quantified. Finally an overview of the conclusions is given in Chapter 17, along with recommendations for future research.

Part I

Heritage

Chapter 2

Optimization of High-Thrust Interplanetary Trajectories

This chapter gives an overview of the various methods that have been used to optimize high-thrust interplanetary trajectories. First of all an introduction into this topic is given in Section 2.1. Subsequently different approaches to modeling the trajectory are discussed in Section 2.2. Section 2.3 then discusses the optimization aspects of finding good high-thrust interplanetary trajectories. Finally Section 2.4 discusses the heritage of previous students at the TU Delft in this field.

It is noted that a large part of this chapter is based on the main findings of an earlier study done by the author. Hence for a more in-depth discussion, the reader is referred to [Musegaas, 2012].

2.1 Introduction

Optimization of interplanetary trajectories is a complex business. The dynamical system is nonlinear due to the gravitational attraction of the Sun and the planets. The trajectories typically include discontinuities in the state variables that are caused by rocket engine maneuvers and gravity assists along the trajectory. The environment is constantly changing as a result of planetary motion around the Sun. This causes the location of possible gravity assists to change as a function of time. Also the initial conditions and/or terminal conditions change as a result of this motion.

A large number of variables is typically required to model trajectories that include a sufficient amount of gravity assists and rocket burns. Less variables typically means that only suboptimal trajectories can be found. The problem is further complicated by the fact that the basic structure of gravity assists and maneuvers is typically unknown and the number of gravity assists and maneuvers varies from trajectory to trajectory.

Examples of typical high-thrust trajectories can be seen in Figures 2.1 and 2.2. Both trajectories represent missions that have actually been flown. Note that both trajectories utilize a combination of multiple planetary swing-bys and DSMs.

To effectively search through all possible trajectories, some assumptions are typically used in the first phase of optimizing high-thrust interplanetary trajectories.

Perturbations can typically be neglected in first-order analyses. The absolute magnitude of perturbing forces is typically much smaller than that of the main forces. The Sun is the main force during the largest part of the trajectory. The gravitation of planets only takes over close to planetary encounters. Because the planetary encounters have a very small duration, typically the patched-conics assumption is used. This means that the trajectory is modeled in the heliocentric reference frame and that gravity assists are modeled as instantaneous maneuvers in the relevant planetocentric reference frames. Similarly, due to the fact that high-thrust impulses have a very short duration, they are also modeled as instantaneous velocity changes.

These first-order analyses are subsequently used as initial guesses in more detailed studies of the tra-

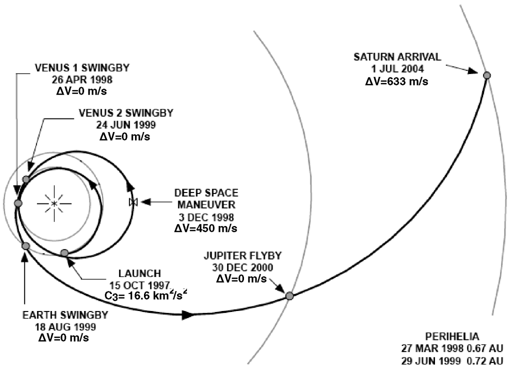


Figure 2.1: Schematic representation of the Cassini trajectory. Taken from [Olds et al., 2007], verified against a lower resolution version in [Peralta and Flanagan, 1995].

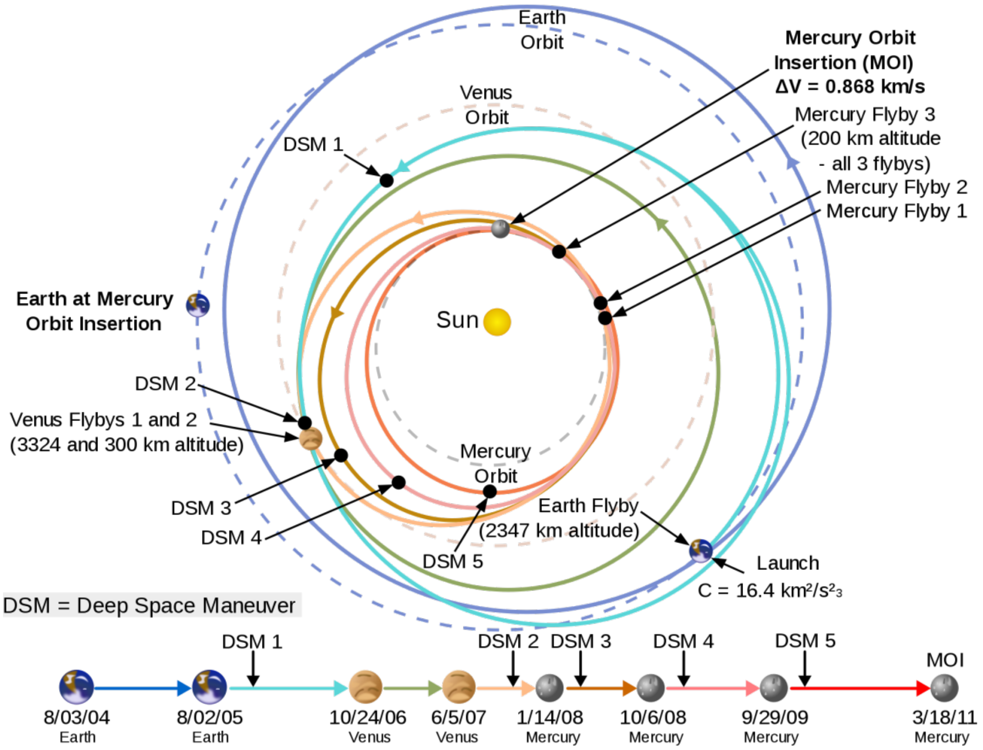


Figure 2.2: Schematic representation of the Messenger trajectory. Taken from [Wikipedia, 2012], verified against a lower resolution version in [McAdams et al., 2006].

jectory. Perturbations such as third-body effects, irregularities in gravity fields, solar radiation pressure and aerodynamic forces have to be taken into account in these more detailed analyses. Also other aspects may play an important role in deciding for the final trajectory, such as options to visit additional asteroids or timing requirements of the critical maneuvers.

Despite of all these simplifications that are used in the first-order analyses, high-thrust interplanetary trajectories are still very difficult to optimize. The selection of the swing-by planets can be simplified by assuming a certain swing-by sequence and subsequently optimizing all feasible sequences. Given that the number of planets is limited and the number of planets useful for gravity assists are typically even more limited, this results in a solvable number of different trajectories. Interesting applications of this methods can be seen in [Petropoulos et al., 2000] and also in [Melman, 2007]. It is noted that also methods are developed for solving the planetary sequence and actual trajectory in a single optimization run. An example can be seen in [Gad and Abdelkhalik, 2011]. It is noted though that the only way to perform such an analysis is to simplify the problem on other aspects, in this case by narrowing bounds on departure and arrival times.

If the planetary sequence is fixed, trajectories that further assume that maneuvers to be performed during the planetary encounters only are comparatively simple problems to solve. The planetary encounter dates namely also fix the locations of these maneuvers. These trajectories are however typically suboptimal and the overall mission cost may be reduced significantly by the addition of maneuvers between planetary encounters. These maneuvers are commonly referred to as Deep Space Maneuvers (DSMs).
The addition of these maneuvers comes at a high price, because the number of variables required to determine such trajectories is significantly higher. Furthermore it was already noted by Myatt et al. [2004] that the search space is very complex and high-oscillatory. The optimization becomes very difficult because many suboptimal minima are present and because the final objective is typically extremely sensitive to the optimization variables. A quick sensitivity test in [Musegaas, 2012] revealed that a deviation of 0.01 % of the search space of some variables could result in a penalty of up to 3 km/s in the objective value. Given that deviations of about 100 m/s can typically be considered very significant already, these problems can be deemed highly sensitive.
Molenaar [2009] also showed that many unrelated minima are found in the search space belonging to complex high-thrust trajectories, as can be seen in Figure 2.3.

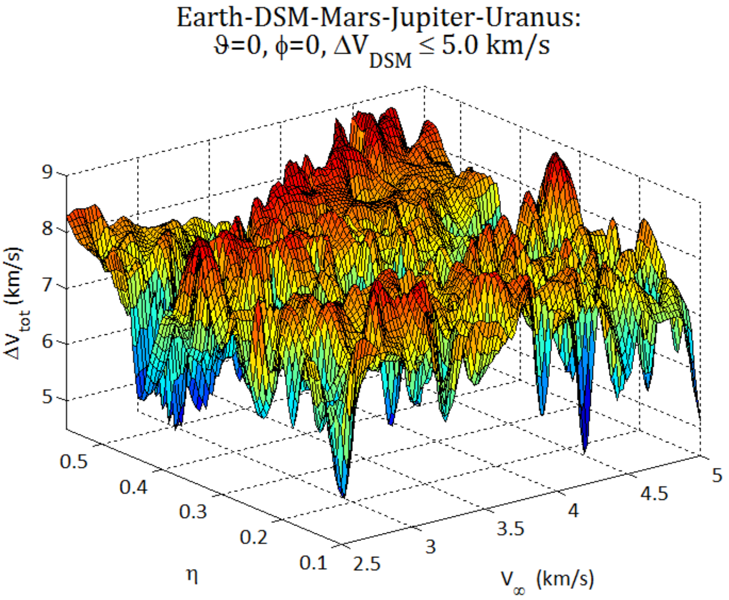


Figure 2.3: A contour plot of the ΔV cost in terms of two variables, for a mission to Uranus [Molenaar, 2009]. Note that the problem is simpler than many other optimization problems encountered in practice. Also note that this is the *optimized* contour for two variables. In practice most other variables, which may be well over 20, will also have a similar structure of minima. Also note that two variables specifying the DSM location were fixed to create this plot.

2.2 Trajectory Models

This section gives an overview of the trajectory models that can be employed in the optimization of high-thrust interplanetary trajectories. First an introduction is given in Section 2.2.1. Subsequently the methods derived from optimal control theory are discussed in Section 2.2.2, which calculates the trajectory using numeric methods. After that different options for semi-analytic trajectory modeling are presented in Section 2.2.3.

2.2.1 Introduction

The design of high-thrust interplanetary trajectories used to be done mainly by proper manual analysis of all transfer options, aided by computer-driven local optimization of apparent good trajectories. Purely computer-driven optimization started gaining popularity in the 1990's. Researchers and mission planners developed a semi-analytic trajectory model that assumed maneuvers were only possible during gravity assists. The trajectory in-between the planetary encounters could then simply be solved using a Lambert targeter. This trajectory model is often referred to as the Multiple Gravity Assisted (MGA) trajectory model. The MGA trajectory model allowed for a small decision vector and hence comparatively simple optimization. Although by far not the first, a comprehensive application of this method was done by Petropoulos et al. [2000].

This trajectory model typically does not include the best trajectory options though. Because no maneuvers can be performed between planets, many potentially good trajectories are pruned from the set of possible solutions. [Lawden, 1963] already showed that many trajectories would benefit from intermediate impulses. The introduction of these intermediate impulses, or Deep Space Maneuvers (DSMs), is however not straightforward.

Two different approaches to solve this problem are typically encountered in literature. One method is to treat the problem as an optimal control problem. The implications of this approach are further discussed in Section 2.2.2. Another approach is to adopt a semi-analytic model in which additional parameters are used to define the DSM location.

2.2.2 Optimal Control Theory Methods

Optimal control theory is an extension of the calculus of variations. It is used to mathematically optimize control inputs. The theory behind optimal control theory was largely developed by Pontryagin and Bellman [Lawden, 1963]. A first space-related application was proposed by Lawden [1963]. His primer vector theory is an indirect method of trajectory optimization. An initial guess of the trajectory is simulated numerically. A set of necessary conditions is found for a Lagrange multiplier that indicate whether a trajectory is optimal or not. This information can subsequently be used in the search for the optimal number of impulses and their locations.

Many authors have subsequently contributed to and improved the original theory. Very notable contributions have been made by Jezewski [1975]. Furthermore a good application of the method can be found in [Navagh, 1993]. A good comprehensive overview of the primer vector method has been written by Prussing in [Conway, 2010].

The exact method has already been discussed in [Musegaas, 2012] and the reader is referred to that report for a further discussion. This section will focus on the application of the primer vector in optimization of high-thrust interplanetary trajectories.

Primer vector theory has recently caught ESA's attention. The subsequent ARIADNA study resulted in a series of three very interesting publications on the application of primer vector theory: [Olympio et al., 2007], [Olympio and Marmorat, 2007] and [Olympio, 2009]. Two substantially different strategies to the optimization of high-thrust interplanetary trajectories are discussed therein.

The first strategy is to model the entire trajectory as a whole using the primer vector theory. The main difficulty is that each gravity assist is a discontinuity. To cope with that, a transition matrix is derived along with optimality conditions. An augmented Langrangian is set up that contains the complete cost function as well as all constraint and coordination variables required by the trajectory. An iterative algorithm is then presented that can be used to solve the problem.

The strategy is very promising, as it allows to optimize the trajectory with great precision and for very general cases. A practical problem is that the problem may become very sensitive as more gravity assists are added. In practice the method only works if an initial guess is given of the optimal solution. Olympio [2009] showed that this method may then be used to obtain better solutions. The addition of a second DSM in the first leg of a trajectory inspired by Messenger resulted in a ΔV reduction of 140 m/s.

The second strategy is to prune the search space using the primer vector. To do so the problem is decomposed into multiple simple body-to-body transfers. These subproblems are subsequently optimized

to obtain a large set of optimized subproblem trajectories. Various pruning strategies are subsequently used to obtain a reduced set of subproblem trajectories. These can subsequently be patched together by matching specified linking conditions. In this way the trajectory is built up gradually, reducing the dependence on previous legs. The method is however prone to wrong decisions that could lead to pruning the optimal trajectory: the best overall trajectory may very well be suboptimal in a certain leg.
This strategy has not been applied to difficult problems yet and its actual performance can not be compared because of that.

Because the primer vector theory has no proven performance on difficult problems, it will not be the subject of this thesis. It is however noted that the primer vector theory may be used to improve solutions found using semi-analytic methods that will be described in the next section. Hence in future work the primer vector theory may be a very interesting method to examine more thoroughly.

2.2.3 Semi-Analytic Trajectory Models

A completely different approach is to use a semi-analytic trajectory model. These models can be regarded as an extension of the MGA-trajectory models. The idea is that a DSM may be added to the trajectory by employing additional parameters that define its location and moment of application. For a general case this would require at least four new variables for each DSM.
Two different trajectory models are often encountered in literature, both of which model the trajectory in 3D. The first one is commonly referred to as the Velocity Formulation (VF), the second one as the Position Formulation (PF). The main difference between both methods is the way in which the DSM position is specified. The PF specifies the position of this DSM directly, whereas the VF variant specifies the velocities after the last planet encounter and subsequently propagates the trajectory for a certain time, thereby obtaining the DSM location. This difference is represented schematically in Figure 2.4.
Most publications assume a maximum of 1 DSM per leg, because the first DSM typically yields the largest gain. These trajectory models will be referred to as MGA-1DSM-VF (Multiple Gravity Assist with 1 Deep Space Maneuver per leg using a Velocity Formulation) and MGA-1DSM-PF (Multiple Gravity Assist with 1 Deep Space Maneuver per leg using a Position Formulation) trajectory models.

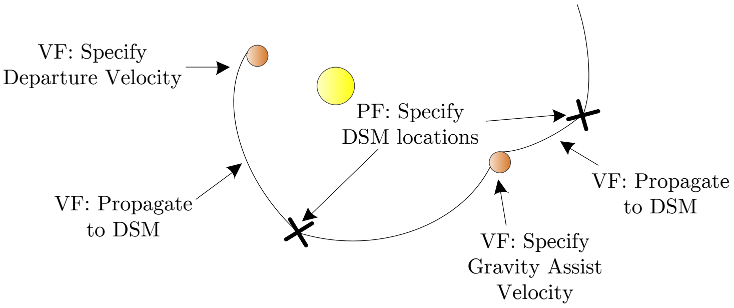


Figure 2.4: Schematic representation of the difference between the MGA-1DSM-VF and MGA-1DSM-PF trajectory models.

The majority of publications use the VF trajectory model. They further assume the gravity assists to be unpowered and this trajectory model is hence further referred to as the MGA-1DSM-VF-UNP (MGA-1DSM-VF-Unpowered) trajectory model. This assumption reduces the number of variables needed to define the DSM to three, thereby simplifying the problem. Two variables are used to define the properties of the gravity assist leading up to the DSM. Subsequently the trajectory is propagated for a certain time after which the DSM is applied. A Lambert targeter is subsequently employed for the second part of the leg. [Vasile and de Pascale, 2006]
The PF variant splits the leg into two parts that are solved independently using a Lambert targeter. Subsequently maneuvers at the gravity assist and at the DSM location are used to patch the trajectory together. In contrast to the VF variant encountered in literature, the PF variant does include the option to perform powered gravity assists. [Becarra et al., 2007]
Both trajectory methods are further discussed in Chapter 5 and are hence not discussed in detail here.

It suffices to say that both methods can be used to obtain good trajectories in practice.

An especially interesting development was the publication of GTOP, Global Trajectory Optimization Problems [Vinko et al., 2007a] [Izzo and Vinko, 2012]. GTOP is a database that consists of various standardized trajectory problems that resemble actual missions. This allows researchers to compare optimization results with each other. Implementation details in the trajectory models made such comparison difficult before. Furthermore GTOP made a clear split between the modeling and the optimization of high-thrust interplanetary trajectories. This allowed researchers from outside the space world to help in optimizing these complex trajectories. This happened various times and it is very interesting to note that at this point the best putative solution to the most difficult problem in GTOP is held by Stracquadanio et al. [2011], who is a biomedical engineer.

Almost all trajectory models assume that only one DSM is present in each leg. Again this is not generally the case, as was also pointed out by Olympio [2009]. Semi-analytic trajectory models with more options for maneuvers, and a swing-by sequence that is not fixed, have been investigated in many subsequent papers by Abdelkhalik. Among the publications, [Abdelkhalik and Mortari, 2011], [Gad and Abdelkhalik, 2011] and [Abdelkhalik and Gad, 2012] were available to the author of this thesis. Although a very nice and general approach was used in all papers, they were only able to optimize the trajectories with much smaller bounds on departure and arrival times than GTOP. Furthermore they were unable to find any improvement with respect to the GTOP variants of the problems. One of the included trajectories was the one for which Olympio [2009] did find a better solution.
It is hence concluded that this approach may be interesting once better optimization techniques or computational performance is available. This approach is considered too general to be useful in practice using the current methods.
Another approach to solving the problem with a more general set-up can be found in [Ceriotti and Vasile, 2009] and [Ceriotti and Vasile, 2010]. A discrete 2D trajectory model is discussed therein. Again the solution found for this trajectory model appears to have been obtained using some a-priori information of the optimal trajectory. Hence also this trajectory model is unproven so far and will not be considered in this thesis.

2.3 Optimization Methods

This section discusses the optimization aspects of finding a good trajectory. First an introduction is given in Section 2.3.1. Various nature-inspired metaheuristics are discussed in Section 2.3.2. Subsequently various approaches to repeated local searches are presented in Section 2.3.3. Section 2.3.4 discusses branching options and hybrids between various metaheuristics. Section 2.3.5 discusses pruning and incremental approaches to solving the problem. Finally some recent interesting developments from outside the space world are discussed in Section 2.3.6.

2.3.1 Introduction

Given the relatively low complexity, the MGA trajectory models were much easier to optimize than trajectories including DSMs. Petropoulos et al. [2000] and various other researchers found good trajectories within their trajectory models with relative ease. Also at the TU Delft Schlijper [2003], Melman [2007] and others found good trajectories with relative ease.
This changed drastically as two research groups started to investigate options to add DSMs to these MGA trajectory models. This investigation followed an ESA ARIADNA study in 2003. Both Myatt et al. [2004] and Di Lizia et al. [2004] did not manage to find a reliable method to optimize these trajectories. Myatt et al. [2004] noted that the performance of deterministic methods, such as DIvided RECTangles (DIRECT) and Multilevel Coordinate Search (MCS), deteriorated rapidly as the problems became more complex. Metaheuristics, such as Differential Evolution (DE) and Particle Swarm Optimization (PSO), were selected as the best techniques to optimize these difficult trajectory optimization problems.

Since these publications, many different strategies have been developed to solve these difficult problems.

The following sections intend to give an overview of these strategies and their performances. These strategies have been grouped where possible and will be discussed per category. Because the majority of this analysis was also present in [Musegaas, 2012], the descriptions of all methods have been kept short. The reader is hence referred to [Musegaas, 2012], or to the original publications, for more information about the optimization methods.

2.3.2 Nature-Inspired Metaheuristics

Four nature-inspired metaheuristics are encountered very often in publications on solving interplanetary trajectories. These are a Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) and Adaptive Simulated Annealing (ASA). Each of these methods is described in short first, after which their performance will be discussed.

Algorithm Description

Genetic Algorithms are stochastic search methods taking their inspiration from the Darwinian natural selection and 'survival of the fittest' principle. The variables corresponding to the optimization problem are encoded in 'chromosomes' of an individual. After initializing a population of these individuals, various genetic operations are performed on these individuals. These operations typically include cross-over, or 'breeding', and mutation. More operations may be included depending on the GA version. Subsequently selection criteria determine which individuals 'survive'. These genetic operations are used to generate new solutions in the vicinity of good solutions. This may significantly help in finding better solutions.
GAs have a long history. Evolutionary algorithms very similar to the current definition of a GA have been used since the 1960's [Bosworth et al., 1972]. Since then many different versions have been developed. For good sources on GAs, the reader is referred to [Goldberg, 1989] or [Michalewicz, 1996].

Differential Evolution is a heuristic that was first developed in 1996 by Storn and Price [1997]. Similar to GA it is population based. The basic version of DE generates a mutant vector by taking the weighted difference between two randomly chosen individuals from the population. This mutant vector is then added to a third individual. If the resulting fitness is better than that of the corresponding individual in the population, it is replaced. Many variations exist for DE, but the basic idea is similar. Note that not all 'chromosomes' or variables are typically included in the mutant vector. Instead the inclusion is determined randomly with a certain cross-over probability CR.
For more information on DE, the reader is referred to [Price et al., 2005] or [Feoktistov, 2006].

Particle Swarm Optimization is an optimization technique developed by Eberhart and Kennedy [1995] and subsequently improved in [Kennedy and Eberhart, 1995]. It is another population-based heuristic that was inspired by the flocking behavior of birds. In PSO each particle keeps track of its own position and velocity in the search space. Its new position is dependent on the direction it had been moving in in the previous generation, the position of the best fitness it encountered so far, and the best location found by the particles in its neighborhood. These three components are referred to as the inertia, cognitive and social components. Contrary to DE and GA, there is no selection in PSO.
The reader is referred to [Eberhart and Yuhui, 2001] and [Conway, 2010] for more information on PSO.

Adaptive Simulated Annealing is a specific implementation of Simulated Annealing (SA). SA was developed by Kirkpatrick et al. [1983]. The underlying principle is based on the solidification process of liquids into crystalline. SA picks a neighbor of a certain point and subsequently moves to this new point depending on a random number which depends on the distance of the corresponding fitnesses and the global temperature T. This temperature decreases gradually during the optimization. In ASA the parameters controlling the temperature schedule and random step selection are updated automatically

and are dependent on the optimization results.
Many variants exist on (A)SA. More information on SA can be found in [Kirkpatrick et al., 1983]. ASA is discussed in [Ingber, 1993] and [Ingber, 1996]. Related to space trajectory optimization, ASA is encountered in [Vinko et al., 2007a].

Algorithm Performance

Many publications have been made that compare various metaheuristics on complex trajectory optimization problems that include DSMs. It is however noted that the results of most of these papers are difficult to compare. Performance criteria have not been defined properly and algorithms have not been tuned properly, as will be further discussed in Chapter 7. Furthermore publications before 2008 used different problem definitions. The introduction of GTOP allows for a much better comparison of most papers after 2008.
That said, a first comparison is found in [Myatt et al., 2004]. DE was ranked first on the problems that included a DSM. PSO was ranked second. GA was found to perform badly in this study. Di Lizia et al. [2004] did not report an equally good performance of DE, although it is argued by Olds et al. [2007] that this is purely caused by the bad settings used for DE in [Di Lizia et al., 2004].
More comparisons can be found in [Vinko et al., 2007a] and [Vinko and Izzo, 2008]. Among other things, GA, DE, PSO and ASA were tested on various MGA and MGA-1DSM problems. GA and PSO were typically ranked worst and DE typically managed to find the best solutions. ASA had a good average performance but found less good solutions than DE. The number of evaluations used in the comparison was not enough for it to be deemed a good comparison either.

A very interesting paper was written by Olds et al. [2007]. The performance criterion used therein is very robust. Also a proper tuning process was used to find good settings of DE. Olds et al. [2007] also showed the extremely high sensitivity of DE to its tuning parameters, with performance dropping by a factor of 100 or more with slightly different tuning parameter settings. Olds et al. [2007] managed to reliably solve problems with a large number of swing-bys and one DSM in roughly one minute.

A very interesting comparison between different metaheuristics can be found in [Vasile et al., 2008]. DE, PSO, GA and various other methods were tested on problems of increasing difficulty. The results were very interesting as DE was found to be the only nature-inspired metaheuristic with a non-zero chance of optimizing difficult MGA-1DSM problems. Similar results are presented in [Vasile et al., 2009], [Vasile et al., 2010] and [Vasile et al., 2011]. The algorithms in these papers were tuned to some extent, although it is noted that a more rigorous tuning process may still yield large improvements.

2.3.3 Repeated Local Search

A completely different strategy to solving the problems is to repeatedly start local optimizations. Because a single local optimization run is typically much faster than a global optimization run, many runs can be performed at equal computational cost as a global optimization run. Typically local optimization techniques are used for these local runs, although also global optimization algorithms may be employed for these local searches.
Many different local optimization algorithms exist. Two common types are discussed first, which are Nelder-Mead (NM) and Sequential Quadratic Programming (SQP). Subsequently three different strategies for the repeated local search are discussed. Finally the algorithm performance is discussed.

Local Optimization Algorithm Description

Nelder-Mead is a comparatively simple local algorithm that was developed by Nelder and Mead [1965]. The algorithm initializes a simplex (a polytope with n+1 vertices) around the initial guess of the solution. The simplex then ‘runs’ over the search space in search of the best solution. Depending on the fitness of all points in the simplex, the simplex can make different moves. These moves are illustrated in

Figure 2.5 for a 2-dimensional problem. These steps are taken until the vertex converges within a certain tolerance.
More information on the Nelder-Mead algorithm can be found in [Press et al., 1992] and [Press et al., 2007].

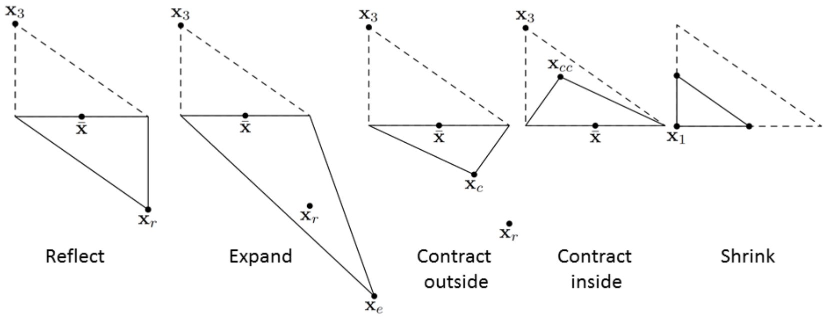


Figure 2.5: The various steps that can be taken in Nelder-Mead. Adapted from [Lagarias et al., 1998].

Sequential Quadratic Programming is a local optimization algorithm that was first applied in 1963 in the PhD thesis of Wilson [Boggs and Tolle, 1995]. SQP essentially approximates the objective function in a small region around a certain point by a quadratic function. This local quadratic model is then solved analytically. The results are then used to construct a better approximation for the next iteration. This process is repeated to obtain a sequence of approximations that hopefully converges to a solution. Many variants of SQP have appeared, each with their own advantages and disadvantages. An overview of some options is given in [Sun and Yuan, 2006]. Especially the Sparse Nonlinear OPTimizer, SNOPT, as developed by Gill et al. [2002] is often mentioned as a very efficient algorithm.

Search Strategies

The local searches can be combined in various ways. This section discusses three of those ways. The first two strategies are more general algorithms, the third one was built specifically for space trajectory optimization.

Multi-Start (MS) is the simplest way to perform repeated local samples. A number of points is selected to start a local search. These points may be selected at random or following a certain distribution such as a prescribed grid or Latin Hypercube Sampling (LHS).

Monotonic Basin Hopping (MBH) is a special case of the Basin Hopping (BH) algorithm developed by Wales and Doye [1997]. The algorithm was adapted by Leary [2000] and further generalized by Locatelli [2005] and Addis et al. [2005]. These authors also first applied the algorithm to space trajectories in Addis et al. [2008].
In its basic version MBH is very similar to MS. Instead of sampling in the entire area, samples are generated in the neighborhood of the current best local minimum. In this way it is able to 'hop' from a certain minimum to a next minimum that is nearby. Thereby it exploits the 'funneling' structure consisting of a large number of minima that are close together. These minima often inhibit the best solution to be found, but are used by MBH to 'guide' the way to the best minimum. After a certain amount of samples have been taken in which no improvement is found, a global restart is performed.

Inflationary Differential Evolution Algorithm (IDEA) is an algorithm developed by Vasile et al. [2008]. It was inspired by the success of MBH. Basically the algorithm is very similar to MBH that uses DE as a local optimization technique. Vasile et al. [2008] namely noted that DE was the best metaheuristic to optimize difficult MGA-1DSM trajectory problems. DE had difficulties to converge on difficult problems and MBH (with local optimizers) suffered from the fact that the problems contained a large number of minima at different locations in the search space. Hence Vasile et al. [2008] hybridized

them. Minor changes are present in subsequent publications: [Vasile et al., 2009], [Vasile et al., 2010] and [Vasile et al., 2011].
The algorithm can best be explained using a picture. Figure 2.6 gives an overview of the algorithm *as interpreted by the author*. Some aspects are ambiguous in the publications and hence it is not entirely clear if this is a good representation of the algorithm.

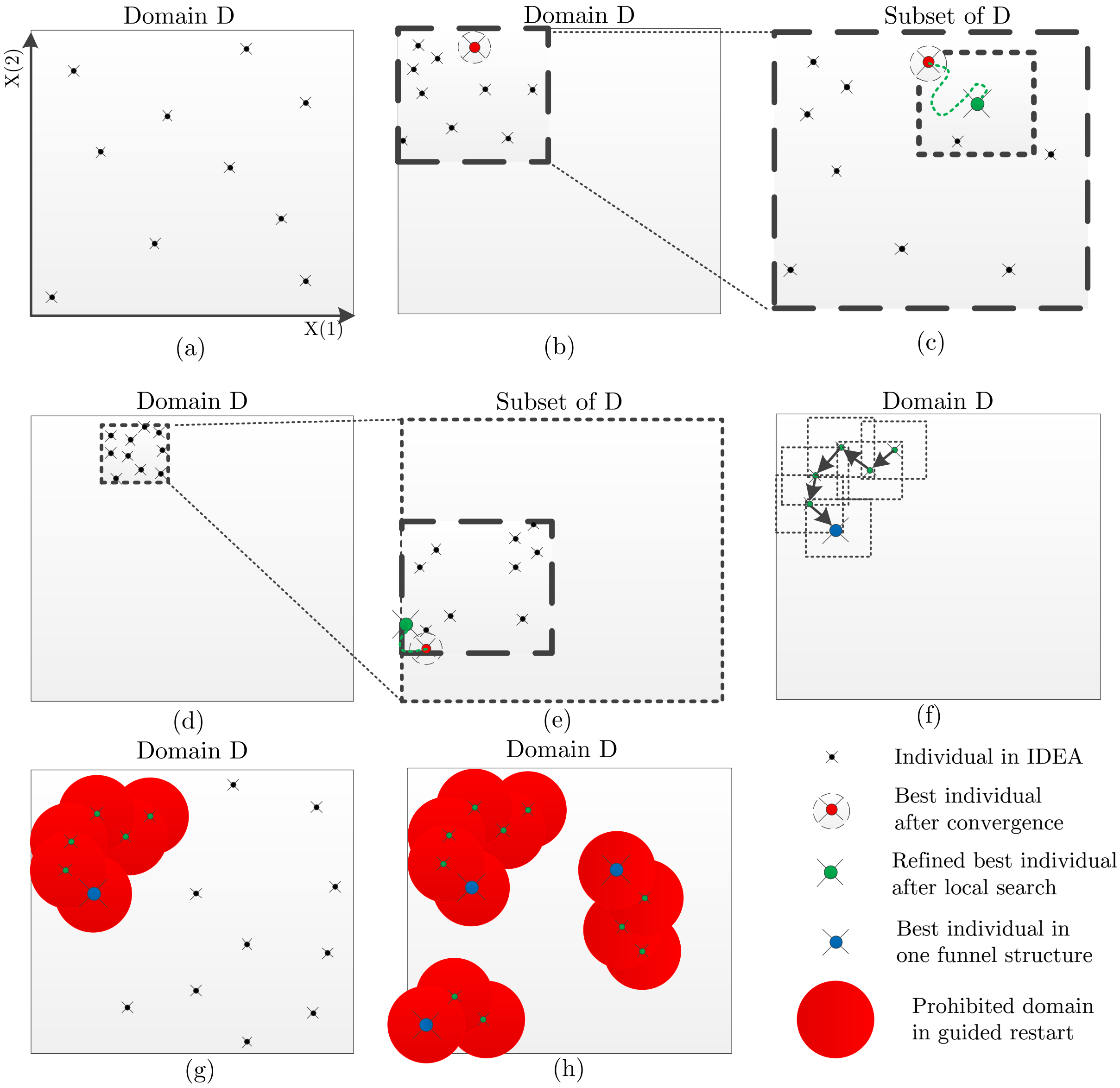

Domain D
X(2)
X(1)
(a)
Domain D
(b)
Subset of D
(c)
Domain D
(d)
Subset of D
(e)
Domain D
(f)
Domain D
(g)
Domain D
(h)
Individual in IDEA
Best individual after convergence
Refined best individual after local search
Best individual in one funnel structure
Prohibited domain in guided restart

Figure 2.6: A schematic representation of IDEA. (a): a DE population is started in the search space. (b): the population converges to within tolerance bounds. (c): the best individual is refined by a local search method. (d): a small bubble around this best solution is set up with a new DE population (e): The evolution within this bubble converges (after which it could be refined by local optimizer and then a new bubble is initiated). (f): This process is repeated, the bubble 'hops' the search space to lower minima, until it hits the funnel bottom. (g): a global restart is performed, in which areas leading to the bottom of (f) are prohibited for the *initial* new population. (h) this process is repeated, potentially revealing multiple funnels. Derived from the algorithm in [Vasile et al., 2011].

Algorithm Performance

Two different series of publications are available of MBH on high-thrust trajectory optimization. Addis et al. [2008], and a later published version of essentially the same paper [Addis et al., 2011], have applied MBH on many complex problems in GTOP. They managed to find the best solutions at the time. Although many solutions have been improved since, Addis et al. [2008] still hold the best solutions to many instances of the TandEM missions in GTOP. It is noted that some of these solutions used a learning algorithm to improve the chance of finding good solutions [Cassioli et al., 2009].

Vasile et al. [2008] tested both MS and MBH with the same test suite they used to test DE, GA and PSO. The results were very interesting as it appeared that both MS and MBH outperformed DE, GA, and PSO. MBH performed significantly better than MS. It was already noted in Section 2.3.2 that, although various settings were used for DE, GA and PSO, their performance may be improved by employing a more rigorous tuning process.

Similar results are presented in Vasile et al. [2009], Vasile et al. [2010] and Vasile et al. [2011]. These papers however also compare these algorithms with IDEA. Especially at more difficult problems it was noted that IDEA outperforms MBH, MS and DE. An example of the results is given in Figure 2.7.

Finally it is noted that Zhiqing et al. [2011] developed an inflationary PSO algorithm. It used a scheme very similar to IDEA, but used PSO instead of DE. The results of their algorithm did not come close to the performance of IDEA though.

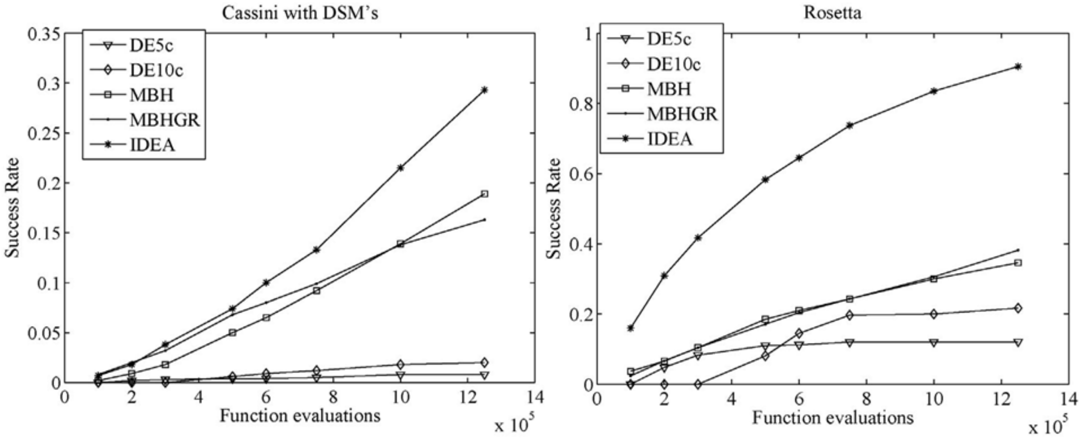


Figure 2.7: Performance of various algorithms applied to the Cassini2 (left) and Rosetta (right) problems in the GTOP database [Vasile et al., 2011]. Both are difficult MGA-1DSM problems from the GTOP database. Success is defined as finding a solution that is closer than 111 m/s (Cassini2) or 57 m/s (Rosetta) to the best putative solution to the problem. To represent performance, the success rate is plotted against the required number of function evaluations.

2.3.4 Branching and Hybrids Metaheuristics

Algorithms relying on the separate evolution of different populations have initially arisen from the need for parallelization of big computational effort. The basic idea is to spread the total computational effort over multiple processors and to start (semi-)independent searches on each processor. Typically these different populations are referred to as islands. Although not the first, Martin et al. [1997] noted especially that an island migration model based on several GAs consistently outperformed a model in which migration was not applied systematically.

An often encountered example of such a model is Multiple Particle Swarm Optimization (MPSO), in which multiple swarms of PSO are used to search for the minimum. Also parallelization of DE is easily achieved as shown in [Tassoulis et al., 2004].

Attempts of combining different metaheuristics, and local search methods, have also been published frequently, as will be discussed in the following.

An early space-related example of a branching scheme was presented by Vasile and de Pascale [2006]. A GA was severely adapted to optimize difficult MGA-1DSM problems. The algorithm was applied to various difficult test cases, some of which were later adopted in the GTOP database. The final results obtained by Vasile and de Pascale [2006] were good, but significantly better solutions have been found nevertheless. Hence their algorithm is not considered powerful enough.

A space-related version of a parallel island model was described by Izzo et al. [2009]. The performance criteria used therein are not good enough to allow for good comparison. It appeared though that the parallel island versions outperformed their sequential counterparts.

Attempts of combining different metaheuristics have also been published frequently. Examples can be found in [Vinko et al., 2007a], [Vinko and Izzo, 2008], [Vinko et al., 2007b] and [Biscani et al., 2010]. A similar implementation was written by Oldenhuis [2010] at the TU Delft, although that algorithm was not applied to problems including DSMs.
Although Vinko et al. [2007a] and Vinko and Izzo [2008] conclude that these cooperative methods perform better than their individual counterparts, the author of this thesis deems this conclusion premature. These publications namely do not use a good performance criterion and the solutions of all algorithms are far away from the best putative solutions.
Biscani et al. [2010] did find good solutions. The computational effort to do so is however considered very significant. The optimization of the Cassini2 problem lasted for more than 8 hours on 7 CPUs, as an example. Compared to the results presented by Vasile et al. [2011], this is a very large computational effort. A success rate of 30 % is reported in [Vasile et al., 2011] after only $1.2 \cdot 10^6$ function evaluations. This number of function evaluations can be performed in roughly 2-3 minutes. Note that success does not mean that the best solution is found exactly, but refining the best result will not last 56 hours.

2.3.5 Pruning and Incremental Methods

Only a very small part of the search space typically contains good solutions as a result of the complex problem structure. Hence it is very appealing to try to define this very small part and prune out the rest of the search space. This would enable a much more focused search for the best solution. This technique is called search space pruning.
Various approaches have been attempted in literature. A first automated approach was presented by Myatt et al. [2004], which was subsequently improved by Izzo et al. [2007]. The heuristic developed therein was mainly intended for MGA trajectories and is hence not further discussed here.
An approach for MGA-1DSM trajectories was developed by Ceriotti et al. [2007] and is presented with more detail in [Conway, 2010]. An incremental approach is followed in which first the first leg is optimized. All solutions below a certain threshold are stored and grouped in 'boxes'. The next leg is subsequently optimized making use of the much smaller search space in the variables of the previous leg. In this way the entire trajectory is built up incrementally before finally optimizing the entire trajectory.
A similar approach was developed by Becarra et al. [2007]. Becarra et al. [2007] used a position formulation instead of the velocity formulation, as the last legs are less dependent on the previous legs in the position formulation variant of the trajectory model. Both approaches are the result of two ARIADNA studies, [Vasile et al., 2007] and [Zazzera et al., 2007], which are very useful and explanatory sources for someone who intends to work on search space pruning in the future.

Becarra et al. [2007] only applied their algorithm to relatively easy problems, hence it is not possible to measure the algorithm's performance. Ceriotti et al. [2007] did apply the algorithm to a complex problem. Their problem is however different from other problems and does hence not allow for a good comparison. Ceriotti et al. [2007] compared their algorithm to DE, PSO and MS techniques, but given that the tuning parameters of those algorithms are not given, a fair comparison can not be made.
A more rigorous description of the algorithm developed by Ceriotti is given in Conway [2010]. Still a good comparison was difficult to make, although results seem to indicate that the method is very com-

petitive with other solutions. It is noted that, in general, disadvantages of this incremental approach are that good knowledge of the problem is required to properly define the pruning criteria. Furthermore the book-keeping may become very difficult if the trajectory becomes even longer than the tested trajectory consisting of 4 legs.

Del Rey Zapatero presented a very different approach to pruning in [Conway, 2010]. He proposed to use a stochastic pruning process. The problem is optimized very often using a certain optimization technique. Subsequently the search space bounds are narrowed around a certain percentage of the best solutions that have been found. This process is repeated several times until the bounds are narrow enough to reliably converge to a solution.

A main advantage of this method is that it is comparatively simple with respect to other pruning methods. A disadvantage is of course that it does not guarantee the inclusion of the actual optimum. Although the test case presented in [Conway, 2010] was the best solution at the time, it has been improved very significantly since [Izzo and Vinko, 2012].

2.3.6 Recent Developments in Optimization Techniques

The fact that many difficult interplanetary trajectory problems have been posted online has led various developers of optimization techniques to use these problems for benchmarking their algorithms. This is considered a very interesting development and good for comparison. Two of those algorithms are discussed here as they have demonstrated their ability to optimize difficult interplanetary trajectory optimization problems.

MIDACO: Mixed Integer Distributed Ant Colony Optimization was developed by Schlüter et al. [2011]. MIDACO is an adaptation of Ant Colony Optimization (ACO) that uses a special penalty method. It has been applied to many difficult optimization problems in different fields, among which some instances in the GTOP database. It managed slight improvements to two problems in GTOP, and furthermore managed to make a large improvement in the most difficult problem in the database. This problem was subsequently improved by various authors afterwards though. It is noted though that the computational effort used to optimize these problems is likely significantly more than the effort required by some other authors. It could be deduced that MIDACO required $3{\cdot}10^6$ evaluations to come within 50 m/s of the best solution to the Cassini1 trajectory and a total of $7.5{\cdot}10^6$ to actually find the minimum. Vasile et al. [2010] reported a success rate of 80 % after only $4{\cdot}10^5$ evaluations, where success was defined as coming within 63 m/s of the minimum.

SAGES: Self-Adaptive Gaussian Evolutionary Strategy (SAGES) was developed by Stracquadanio et al. [2011]. Working on optimization problems in biomedical engineering and general optimization, the authors have developed a method with the aim to robustly solve the trajectory optimization problems in GTOP. For all problems it was applied to, SAGES managed to either match the best putative solutions or improve them. Especially the performance on the most difficult problem in the database (Messenger-Full) is impressive. The best putative solution of 4.254 km/s was improved to 2.970 km/s and subsequently to 2.113 km/s, which is currently still the record [Izzo and Vinko, 2012]. Again the computational time required for this is likely to be very high.

2.4 Heritage at TU Delft

Also at TU Delft many students have optimized high-thrust trajectories as part of their graduation projects. Many of these students have used the MGA trajectory model and neglected the option to perform DSMs. Interesting results in this field have been published by Schlijper [2003], Melman [2007] and many others. Most students working on the optimization of high-thrust interplanetary trajectories made use of GALOMUSIT (Genetic ALgorithm Optimization of a MUltiple Swingby Interplanetary Trajectory). GALOMUSIT was a program developed by various students that can be used to optimize interplanetary trajectories using a GA.

A different approach was taken by Oldenhuis [2010], who used a combination of metaheuristics to optimize his trajectories. The optimization algorithm developed therein has great resemblance to the algorithms presented in [Vinko and Izzo, 2008]. Given that also the subject of that thesis was to optimize comparatively easy MGA trajectory problems, the results are deemed less applicable to this study.

Molenaar [2009] incorporated DSMs in GALOMUSIT as part of his graduation project. It turned out that the addition of a DSM in one leg of a trajectory with multiple swing-bys caused the search space to become too complex to be optimized reliably. To obtain good results some of the variables were fixed to certain reasonable values. This allowed Molenaar [2009] to find solutions, although it is noted that they are likely to be suboptimal.
Given the comparatively bad performance of GA reported in [Myatt et al., 2004] and [Vasile et al., 2008], a possible cause for this difficult optimization is that GA is not the best algorithm to use for optimizing trajectories including DSMs. This hypothesis will be investigated more thoroughly in this thesis.

Also Iorfida [2011] tried to solve high-thrust trajectories including DSMs. The primer vector theory is used to tackle the problem, but Iorfida [2011] made a critical mistake in the calculations. The *initial estimate* of the DSM location was mistaken for the *optimal* DSM location in Equation 4.68 of [Iorfida, 2011]. This equation namely only provides an initial guess of the DSM location and should be iterated upon to find the optimal location. More information on this can be found in [Musegaas, 2012] or Chapter 2 of [Conway, 2010].

In terms of trajectory modelling, it is interesting to note that TU Delft is developing a new astrodynamics toolbox called Tudat. Because not many students have been working on high-thrust trajectories, many methods still have to be developed and tested for this toolbox before they can be used. This is another goal for this thesis, as will also be discussed in the next chapter.

Chapter 3

Thesis Goals

This chapter gives an overview of the goals that were defined for this thesis. Most goals come from the literature study performed earlier [Musegaas, 2012], but some goals were adapted as the thesis progressed. The following six objectives are defined:

- Develop astrodynamic tools for modelling high-thrust space trajectories.
- Investigate the search space belonging to MGA-1DSM problems.
- Subject various metaheuristics to a rigorous tuning scheme and investigate the importance of tuning.
- Compare these metaheuristics after they have been properly tuned.
- Compare different trajectory models.
- Develop a strategy that allows for reliable optimization of complicated DSM problems.

Each of these goals will be discussed in more detail below. Subsequently the last paragraph will discuss the focus on the type of trajectories that will be analyzed in this thesis.

Astrodynamic Tool Development

This goal consists of two main objectives. First of all the generic astrodynamic functions that are still missing in the Tudat library should be developed within the Tudat context, such that future students also benefit from the code as much as possible. This code will be developed with stricter guidelines than other parts of the software, as will be further discussed in Chapter 4.
Secondly, various properly working models will need to be made to calculate high-thrust trajectories including DSMs. These models should be written in a very understandable manner, such that future students may use and improve them. Also these models should be written in a very general manner, as other thesis goals include the comparison between different trajectory models and hybrids between them. The development of the trajectory models is further discussed in Chapter 5.

Search Space Investigation

Little investigation has been done on the characteristics of the search space belonging to the MGA-1DSM trajectories. Apart from some initial analysis in [Myatt et al., 2004] and a sensitivity analysis in [Stracquadanio et al., 2011] nothing has been published on this topic. An investigation of the search space characteristics may help in understanding optimization algorithm performance, differences between trajectory models and the development of better algorithms.
The search space investigation will be presented in Chapter 13. The analysis focuses on the sensitivity of individual parameters and the sensitivity of the overall trajectory model. Also a new method is applied to obtain a first-order estimation of the size of the basins of attraction of various problems.

Importance of Tuning

Olds et al. [2007] pointed out the importance of proper algorithm tuning, which is also often overlooked in publications. This thesis will investigate the performance gain in case a proper tuning suite is used. The commonly used metaheuristics DE, GA and PSO are subjected to a thorough tuning process in which they will be applied to increasingly more difficult problems. Also various local optimization algorithms are compared. The tuning process and performance criteria used therein will be discussed in Chapter 7. Subsequently the trajectory problems used for the tuning process will be discussed in Chapter 8. Chapters 9, 10, 11 and 12 will subsequently discuss the tuning process for DE, GA, PSO and repeated local searches respectively.

Comparison Between Different Optimization Algorithms

As noted in Section 2.3, very few publications compare different optimization algorithms using a proper test suite. Typically either too simple and unrealistic problems are used, not enough function evaluations are used or bad performance criteria are used for the comparison. This thesis will investigate the performance of DE, GA, PSO and repeated local searches after each one has been carefully tuned. This comparison is discussed in Chapter 14.

Strategy to Optimize Complex MGA-1DSM Trajectories

An important goal of this thesis is to develop a heuristic that is able to reliably and automatically optimize complex MGA-1DSM trajectories. The heuristic will be developed for and applied to a set of difficult trajectory problems in GTOP. The strategy will be draft depending on the results of both the search space analysis and the optimization algorithm comparison. Chapter 15 will discuss this strategy and discuss the results obtained with it.

Comparison Between Different Trajectory Models

This thesis will focus on three aspects related to the trajectory models. First of all the gain of using DSMs will be quantified, by comparing the results of an MGA trajectory model with those of the MGA-1DSM trajectory models.
Secondly the potential gain of using powered swing-bys will be quantified. A powered swing-by poses constraints on navigation systems and increases overall mission risk. On the other hand it may significantly improve the efficiency of the trajectory model. A good comparison of the ΔV-cost for both methods was not encountered in literature. This will hence be further investigated in this thesis. To that end the MGA-1DSM-VF-UNP model will be rewritten to cope also with powered swing-bys.
Thirdly a comparison will be made between the MGA-1DSM-PF and MGA-1DSM-VF trajectory models. Because the MGA-1DSM-PF typically includes powered swing-bys, the MGA-1DSM-VF trajectory model with powered swing-bys will be used for fair comparison.
These three analyses on the influence of the trajectory model will be discussed in Chapter 16.

Focus of Trajectories

It is likely that the search space, the performance of various trajectory models and the performance of optimization strategies is dependent on the trajectories that are to be optimized. Hence a certain type of trajectory is selected for this thesis. The main focus will be on trajectories to Saturn. This choice was made for various reasons:

- Many benchmark problems in GTOP go to Saturn. The Cassini1 MGA problem offers a good test bed for the MGA trajectory model and the Cassini2 MGA-1DSM-VF-UNP trajectory offers a nice test bed for MGA-1DSM trajectory models. Furthermore the TandEM case consists of a total of 48 trajectory problems, and thereby provides a large number of problems to test the effectiveness of the optimization strategy. Because TandEM is the only such test case in GTOP, this is deemed a large advantage of selecting Saturn as a destination.

- Multiple-revolution transfers are less common in missions to the outer planets. This thesis will only use single-revolution Lambert targeters and this therefore provides the fairest comparison between different trajectory models. The VF model namely does allow for multiple-revolution transfers because of the fact that a Kepler propagator is used in the first part of the leg.
- Saturn is a popular destination because of various reasons, including the possibility of extra-terrestrial life on its moons.

Part II

Methods and Software

Chapter 4

Astrodynamic Functions

This chapter discusses the basic astrodynamic functions that form the building blocks in drafting an interplanetary trajectory. An introduction into this chapter and the development of these functions for Tudat is given in Section 4.1. Subsequently the Kepler propagator is discussed in Section 4.2. The Lambert targeter is disccussed in Section 4.3. Section 4.4 discusses gravity-assist propagators and Section 4.5 discusses the gravity-assist calculator. Finally Section 4.6 discusses the escape and capture maneuver calculations.

The verification tests belonging to these functions are also discussed in this chapter. It is noted here that a relative accuracy representation is used throughout this thesis because it is independent of units and absolute variable sizes. It is also the accuracy that is typically used in unit testing. Absolute accuracy tests are actually not present in the unit test suite implemented in Tudat.

4.1 Introduction

The astrodynamic functions discussed in this chapter form the basic blocks for many different applications, not just for the application in this thesis. Hence one of the goals for this graduation project was to write these functions for the Tudat library. Tudat is the astrodynamic toolbox under development by all students of the A&S chair at the Faculty of Aerospace Engineering of TU Delft. Section 4.1.1 gives a further introduction into the Tudat project. Subsequently Section 4.1.2 gives an overview of the status of the methods relating to this thesis that were available at the start of this project.

4.1.1 Tudat

Different astrodynamic toolboxes, such as GALUMUSIT and OPTIDUS, have been developed by students of the Astrodynamics & Space Missions (A&S) chair of the Aerospace Engineering faculty of TU Delft. These toolboxes lacked a clear structure, causing the wheel to be reinvented frequently. Also these toolboxes had a very steep learning curve and were written for very specific applications. The Tudat project was founded to overcome these issues. [Kumar et al., 2011]
Tudat aims to be a multi-purpose modular software library with robust and well-tested functions. Also it aims to take advantage of the synergy between students that often graduate in related topics. C++ was chosen as the programming language for Tudat. This decision was taken because it is a relatively modern language that has a very fast compuational speed, which is often desired for astrodynamic purposes. The project is open-source and is licensed under the BSD 3-Clause License.

The Tudat project is being run by students and staff at A&S. To manage such a large project, a good organization was set up, consisting of a management team and a working group team. The latter consists of all active developers and all programming issues are discussed in the working group. A good documentation system was set up on the website, consisting both of a Doxygen code documentation system and a wiki. The wiki explains topics such as project set-up, coding guidelines and tips & trics.
A good code development and checking process was set up for Tudat to ensure the quality. This includes

setting and upholding coding standards, employing strict code checking processes and unit testing for all functions.
The project is split up in two different libraries: Tudat Core and Tudat. Tudat Core consists of well-tested and frequently-used functions and undergoes slow evolution. Tudat Core was set up to ensure stable interfaces for code blocks that are frequently used, such that users can rely on the fact that this code will remain available throughout their graduation project. Backwards compatibility is offered for one release (about 4-6 months). On the other hand the Tudat library is more subject to updates and also includes functions that are well tested, but not yet widely used. This means that interfaces or functionality may change if it is required by other applications of the code. The Tudat library undergoes fast evolution and contains 'state-of-the-art' functions.
Recently the list of publications that used Tudat has grown significantly, as the library is being used increasingly. An overview of some publications is given in [Kumar et al., 2012].
More on the Tudat Project can be found in [Musegaas, 2012] or on the Tudat website: `tudat.tudelft.nl`.

Finally it is noted here that throughout the entire graduation project the author of this thesis was both an active developer and code-checker of code not directly related to this astrodynamics part. Also the author was a member of the management team as a documentation manager. The author has also developed the assignment for a course on Tudat, which will be given for the first time in the academic year 2012-2013. The assignment is to optimize the dates of departure and times of flight for a sample-return mission to Mars using a grid search.
This activity did not directly contribute to the goals for this thesis, but nevertheless took up a considerable amount of time throughout the graduation project.

4.1.2 Status Mission Segments of Tudat

This section gives an overview of the status of the mission segments part of Tudat. The astrodynamic functions that were to be developed in this thesis are namely categorized under mission segments in Tudat.
Unlike some other parts of the library, such as orbit integration, not many students had worked on the mission segments part of the library. Various functions were present, but a thorough testing procedure revealed that many functions did not work properly yet. An overview of this is given in Table 4.1.

Table 4.1: Overview of the mission segments in Tudat at the start of the thesis project.

Function	Status
Kepler propagator	Working only for eccentric orbits with $e < 0.8$
Lambert targeter	Gives incorrect values for various cases
Gravity-assist propagator	Not present
Gravity-assist calculator	Present, but multiple mistakes in implementation
Escape/Capture	Present, but devious

Hence many new functions had to be written. Luckily a different student was working on the Lambert targeter already and implemented Izzo's method as part of his graduation [Leito Pinto Secretin, 2012]. The other functions had to be developed and tested by the author though.

4.2 Kepler Propagator

This section discusses the development and testing of a Kepler propagation function. A description of the method and its basic calculation scheme is given in Section 4.2.1. Subsequently Section 4.2.2 discusses the conversion from elliptic mean anomaly to elliptic eccentric anomaly. Section 4.2.3 then discusses the conversion from yperbolic mean anomaly to hyperbolic eccentric anomaly. The verification of these methods is included in the relevant sections. Finally the verification of the Kepler propagator is discussed in Section 4.2.4.

4.2.1 Description

Kepler propagation is a fast and analytic solution to solve a basic orbit propagation problem, which can be stated as follows: given $\boldsymbol{r}_i$, $\boldsymbol{V}_i$ and time t_i, calculate $\boldsymbol{r}_f$, $\boldsymbol{V}_f$ at time t_f.
Orbit perturbations are neglected in Kepler propagation. Five of the six Keplerian elements are integrals of motion in case perturbations are neglected. The true anomaly θ, is dependent on time. Kepler propagation makes use of these notions to solve the orbit propagation problem: first the initial position and velocity are converted into Keplerian elements, subsequently the true anomaly is updated and finally the Keplerian elements are converted back to Cartesian elements to yield the final position and velocity.
The following paragraphs provide the calculation scheme for Kepler propagation of both an elliptic as well as a hyperbolic orbit. Note that the basic idea is the same, but exact formulas differ between these two cases. Also it is noted that parabolic orbits exist as well, which would require a different calculation scheme. Given that an eccentricity of exactly 1 is typically not encountered in practice, no special transformation was written for this case. Eccentricities as close to 1 as 10^{-15} have been tested both for the hyperbolic as well as for the elliptic case, meaning that any near-parabolic case can be handled in practice.

First the Cartesian elements are converted to Keplerian elements. This well-defined process is not given in this thesis, but may be looked up in any astrodynamics textbook, or in [Musegaas, 2012] or [Wakker, 2007]. The next step is to transform the initial true anomaly, θ_i to an initial elliptic eccentric anomaly, E_i, or hyperbolic eccentric anomaly, F_i. Equations 4.1 and 4.2 can be used for that.

$$E_i = 2\ \arctan\left(\sqrt{\frac{1-e}{1+e}}\cdot\tan\frac{\theta_i}{2}\right) \tag{4.1}$$

$$F_i = 2\ \text{arctanh}\left(\sqrt{\frac{e-1}{e+1}}\cdot\tan\frac{\theta_i}{2}\right) \tag{4.2}$$

Note that the 'arctan' function in these equations, and in Equations 4.7 and 4.8, depends on the quadrant of the angle. Many computer languages employ an 'atan2' function to automatically deal with this ambiguity.
After this transformation the initial elliptic mean anomaly, M_i, and hyperbolic mean anomaly, $\overline{M}_i$, are calculated using Equations 4.3 and 4.4.

$$M_i = E_i - e\sin E_i \tag{4.3}$$

$$\overline{M}_i = e\sinh F_i - F_i \tag{4.4}$$

From these initial mean anomalies, the final mean anomalies can be calculated as follows:

$$M_f = M_i + \sqrt{\frac{\mu}{a^3}}(t_f - t_i) \tag{4.5}$$

$$\overline{M}_f = \overline{M}_i + \sqrt{\frac{\mu}{-a^3}}(t_f - t_i) \tag{4.6}$$

These final mean anomalies now need to be transformed into the final elliptic eccentric anomaly, E_f, and the final hyperbolic eccentric anomaly, F_f. The calculation scheme to do so is further discussed in Sections 4.2.2 and 4.2.3. After this transformation, the final true anomalies can be calculated by rewriting Equations 4.1 and 4.2. The rewritten equations to do so are given below in Equations 4.7 and 4.8.

$$\theta_f = 2\ \arctan\left(\sqrt{\frac{1+e}{1-e}}\cdot\tan\frac{E_f}{2}\right) \tag{4.7}$$

$$\theta_f = 2\ \arctan\left(\sqrt{\frac{e+1}{e-1}}\cdot\tanh\frac{F_f}{2}\right) \tag{4.8}$$

Finally the resulting Keplerian elements can be converted back to Cartesian elements. Again this well-defined calculation scheme is not given here, but the reader is referred to other publications such as

[Musegaas, 2012] or [Wakker, 2007].

As a final note it is important to mention that different definitions of the hyperbolic eccentric anomalies are used. Hence care has to be taken that the same definition is used when comparing different results. As an example, the GTOP database uses a different definition [Izzo and Vinko, 2012].

4.2.2 Conversion from Elliptic Mean Anomaly to Elliptic Eccentric Anomaly

The elliptic mean anomaly can not be transformed into an elliptic eccentric anomaly directly. Instead an iterative procedure is required. A Newton-Raphson root-finding scheme can be employed for that. The corresponding scheme is given in Equation 4.9.

$$E_{k+1} = E_k - \frac{f(E_k, M)}{\frac{d}{dE}(f(E, M))|_{E=E_k}} = E_k - \frac{E_k - e\sin E_k - M}{1 - e\cos E_k} \tag{4.9}$$

It is noted by various authors that this iterations scheme converges badly or not at all for eccentricities close to 1. Among many other authors, [Bate et al., 1971] and [Molenaar, 2009] have suggested a universal variable formulation approach to solve this problem. Also many authors have reported very chaotic behavior of this root-finding scheme as e approaches 1.
This issue was investigated thoroughly by Charles and Tatum [1997]. A remarkable conclusion was drawn, namely that this chaotic behavior disappeared if π was used as a starting value.

To further analyze this behavior, various starter values were tested that were encountered in [Taff and Brennan, 1989], [Charles and Tatum, 1997] and [Wakker, 2007]. The eccentricity spectrum was divided into 100 parts. Each part was subjected to 10^6 orbital element conversions with a random mean anomaly and a random eccentricity within the corresponding range. The average number of iterations is calculated from this. The results can be seen in Figure 4.1. Note that an accuracy of $5.0 \cdot 10^{-14}$ was required in M and 100 was used as the maximum number of iterations. Experimentally it had been determined that the root-finder would converge within 40 iterations, unless it showed the chaotic behavior mentioned earlier.
Also the failure rate of the various starter values was recorded. These results can be seen in Figure 4.2.

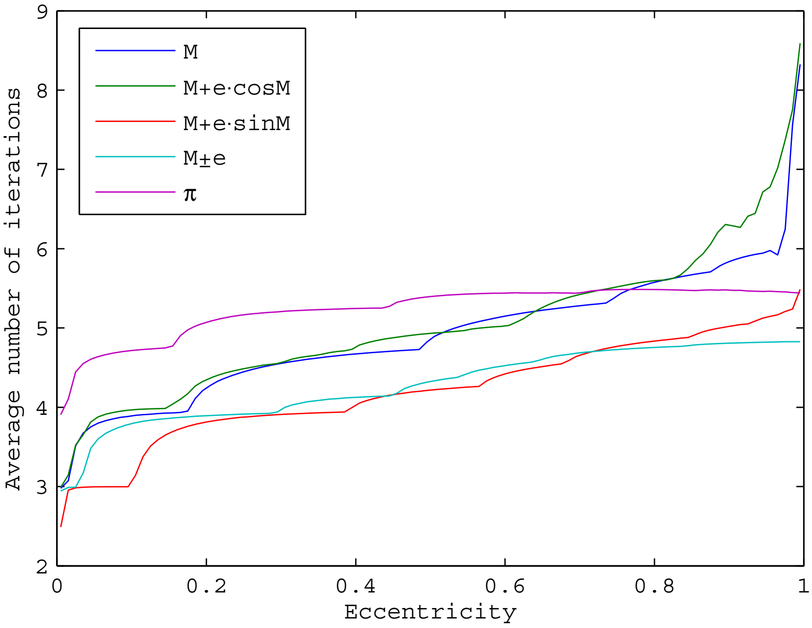


Figure 4.1: The average required number of iterations as a function of the eccentricity for various start values of the Newton-Raphson iterations scheme.

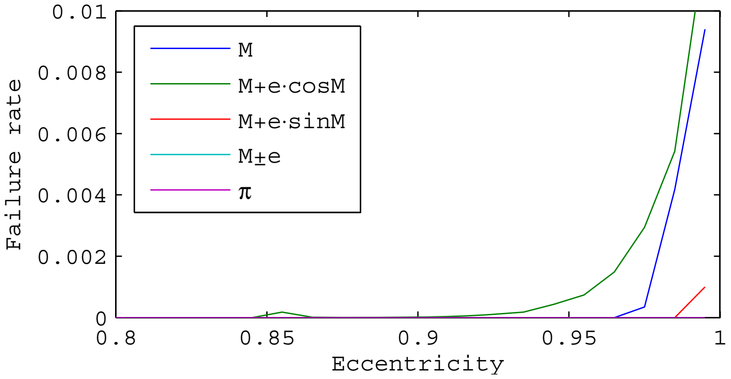


Figure 4.2: The failure rate as a function of the eccentricity for various start values of the Newton-Raphson iterations scheme.

It can be seen in Figure 4.2 that only π and $M \pm e$ are stable throughout the entire spectrum. Furthermore it can be seen in Figure 4.1 that $M \pm e$ converges faster than π throughout the entire spectrum. Also it converges practically equally fast as or faster than $M + e \sin M$ throughout the largest part of the spectrum. Hence $M \pm e$ is chosen as initial guess. Equation 4.10 gives the test which determines which starter value is used.

$$\text{if} \quad M_{\text{mod}(2\pi)} \leq \pi, \qquad E_0 = M + e \tag{4.10a}$$

$$\text{if} \quad M_{\text{mod}(2\pi)} > \pi, \qquad E_0 = M - e \tag{4.10b}$$

The old conversion function was subsequently rewritten to a free function and a large unit test was set-up to verify the implementation. Four pseudo-random conversions were compared against the same conversions done using the GTOP toolbox [Izzo and Vinko, 2012] and the ODTBX toolbox [NASA, 2012]. Additionally 17 conversions with near-parabolic eccentricities of 1.0-10^{-15} and a combination of possibly singular values and pseudo-random values for the mean anomaly were compared to GTOP and ODTBX. Furthermore a random test was performed of 10^8 random conversions, which were subsequently converted back and compared with the initial value. Another test was set-up with the eccentricity fixed at 1.0-1.0·10^{-15} and 10^8 random values of M.
All tests gave correct results with a relative accuracy of at least 1.0·10^{-13} for the 'normal' cases and 1.0·10^{-9} for the near-parabolic cases.
As an additional check the back-and-forth random tests were added to the unit test with a smaller number of random samples. In this way the test will eventually be run on many different computers, to help build up statistical evidence of the correct convergence of the starter value.

4.2.3 Conversion from Hyperbolic Mean Anomaly to Hyperbolic Eccentric Anomaly

Similar to the elliptic case, an iterative procedure is required for the hyperbolic case as well. Again Newton-Raphson may be employed to do so. The accompanying recursive formula is given below in Equation 4.11.

$$F_{k+1} = F_k - \frac{f(F_k, \overline{M})}{\frac{d}{dF}(f(F, \overline{M}))|_{F=F_k}} = F_k - \frac{F_k - e \sinh F_k - \overline{M}}{1 - e \cosh F_k} \tag{4.11}$$

Again it is very important to select the correct initial guess. As it turns out, this iteration scheme tends to be chaotic for wrong initial guesses as well. There was an old version available in Tudat, which was included without unit test. The initial guess present therein is given in Equation 4.12 below.

$$F_0 = \ln\left(\frac{2\overline{M}}{e} + 1.8\right) \tag{4.12}$$

The guess was based on an internet page (`http://www.cdeagle.com/omnum/pdf/demokep1.pdf`), whose source could be traced back to three subsequent publications: [Danby and Burkardt, 1983], [Burkardt and Danby, 1987] and [Danby, 1987]. It turned out that this formula was derived empirically for a specific domain of e and M. Simulations revealed that this initial guess was not suited for all values, as it resulted into chaotic behavior for a large set of combinations of e and M.

A different solution was found in [Wakker, 2007], in which a starter is selected dependent on the values of $\overline{M}_f$ and e as in Equation 4.13.. It is noted here that a mistake was present in Equation 8.28 [Wakker, 2007], which is corrected in Equation 4.13c.

$$\text{if } |\overline{M}_f| < 6e: \quad F_0 = \sqrt{\frac{8(e-1)}{e}} \cdot \sinh\left(\frac{1}{3}\text{arcsinh}\left(\frac{3\overline{M}_f}{\sqrt{\frac{8(e-1)}{e}}(e-1)}\right)\right) \tag{4.13a}$$

$$\text{if } \overline{M}_f > 6e: \quad F_0 = \ln\left(\frac{2\overline{M}_f}{e}\right) \tag{4.13b}$$

$$\text{if } \overline{M}_f < -6e: \quad F_0 = -\ln\left(\frac{-2\overline{M}_f}{e}\right) \tag{4.13c}$$

It turned out that these starter values had a stable performance. Hence these starter values were adopted.

The next step was to verify the correctness of the results. In this case this was considerably more difficult, as the conversion is offered by less toolboxes. The best comparison appeared to be the GTOP toolbox [Izzo and Vinko, 2012], although a different definition of the hyperbolic eccentric anomaly is used therein. Also no explicit transformation to a true anomaly is made. To allow for a good comparison, the eccentric anomalies were further transformed back to Cartesian elements and subsequently compared.
Six pseudo-random values were compared in this way and another four pseudo-random values were compared with an eccentricity close to 1.0. All results had a relative accuracy of at least 10^{-14}.
Furthermore three different random suites were performed, all with 10^8 back-and-forth conversions. The first suite extensively tests the behavior in the most common regime, with $1 < e < 10$ and $-20 < M < 20$. The second suite tests a very wide regime, with $2 < e < 10^{15}$ and $-10^{-13} < M < 10^{13}$. Finally the third suite tests the near-parabolic behavior with $e = 1.0 + 10^{-15}$ and $-20 < M < 20$. All tests were positive with a relative accuracy of 10^{-14}.

4.2.4 Verification of Kepler Propagator

Although the various functions that the Kepler propagator consists of are well tested, another test bed was set up to make sure that no mistakes had been made in the implementation of the Kepler propagator itself. The elliptic orbit propagation was already verified against benchmark data from ODTBX [NASA, 2012] and benchmark data from Melman [Tudat, 2012].
The hyperbolic test case consists of propagating a hyperbolic orbit around the Earth for four consecutive periods of 100 days. It was compared against a similar simulation in GTOP [Izzo and Vinko, 2012]. The simulation had a relative accuracy of 10^{-15}.

4.3 Lambert Targeter

Lambert targeting solves the problem of finding an unperturbed orbit around a central body connecting two points in a specified flight time. The typical geometry of this problem is given in Figure 4.3. For our problem, the central body is the Sun. P_1 represents the position of the departure planet at time t_1. P_2 represents the position of the arrival planet at t_2. The flight time then simply is the difference between t_1 and t_2. [Gooding, 1990]

Still many solutions exist to this Lambert problem. One solution is the trajectory over angle θ, another solution is given by the trajectory over angle $2\pi - \theta$ in the opposite direction. Since all planets in the

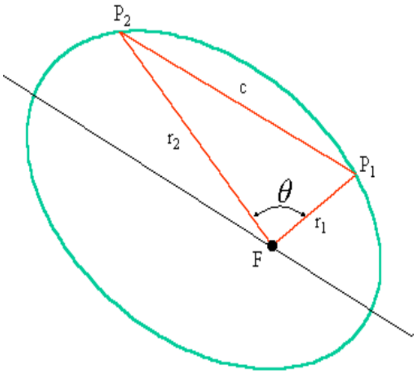


Figure 4.3: Geometry of the Lambert targeting problem [Battin and Vaughan, 1983].

solar system orbit in a counterclockwise direction, the trajectories with a clockwise direction can be omitted.
Also the solution space is not necessarily bound to the domain from 0 to 2π. Multiple revolutions are sometimes also possible solutions to the problem. As mentioned in Chapter 3, these will however not be investigated in this thesis. The prime objective is to implement the DSM into interplanetary trajectory optimization. To achieve this, first an optimization technique must be developed capable of handling the basic scenarios before moving on to even more general, and hence more difficult, scenarios.

The key to solve Lambert's problem is to solve the time of flight equation, given in Equation 4.14.

$$\Delta t = t_2 - t_1 = \sqrt{\frac{a^3}{\mu}}(E_2 - E_1 - e(\sin E_2 - \sin E_1)) \tag{4.14}$$

Many different approaches have been developed to solve this problem [Gooding, 1990]. Typically the semi-major axis a is rewritten to a different form to allow the equation to be solved more easily. In [Musegaas, 2012] one of the calculation schemes to solve the problem was presented, based on the approach by Gooding [1990]. It is not repeated here for the sake of preserving space.

In the scope of the Tudat project, Leito Pinto Secretin [2012] has implemented a newer solution to the Lambert problem worked out by Izzo as part of the GTOP project [Izzo and Vinko, 2012]. Izzo substitutes a variable into the time-of-flight equation which has a linear relation with the time-of-flight. This is done with the intent of reducing the number of iterations required by the root-finder to solve the problem. Furthermore the secant method is used, to avoid costly derivative evaluations. The exact calculation scheme for Izzo's method can be found in [Leito Pinto Secretin, 2012] and is not repeated here.

Both Gooding's method and Izzo's method are compared in [Leito Pinto Secretin, 2012]. For a test consisting of 360 samples over the entire spectrum of θ and requiring the same accuracy, the results of both methods are given in Table 4.2.

Table 4.2: Comparison between computation times for Gooding's method and Izzo's method [Leito Pinto Secretin, 2012].

Method	Maximum [μs]	RMS [μs]	σ [μs]
Gooding's Method	18	11.35	1.40
Izzo's Method	35	6.48	1.09

Because of the faster average computation speed of Izzo's method, this method will be used in this thesis. Leito Pinto Secretin [2012] already subjected the method to a rigorous verification test bed. The reader is hence referred to that thesis report for more information.

4.4 Gravity-Assist Propagator

This section discusses the gravity-assist propagators that were written for this thesis. First of all the basic principle of a gravity assist is discussed in Section 4.4.1. Subsequently the unpowered gravity-assist propagation method is dealt with in Section 4.4.2. Finally Section 4.4.3 discusses the powered gravity-assist propagation.

It is noted here that gravity assist, swing-by and fly-by are synonyms for the same thing.

4.4.1 Basic Principle

When a spacecraft flies close to a planet, their gravitational fields interact. This interaction can be used to change the magnitude and the direction of the velocity of the spacecraft. During the interaction between a planet and the spacecraft, the law of impulses states that the total momentum is preserved. Assuming constant masses of the spacecraft and planet, this interaction can be represented mathematically as follows:

$$m_s \cdot V_{s,i} + m_{pl} \cdot V_{pl,i} = m_s \cdot V_{s,f} + m_{pl} \cdot V_{pl,f} \tag{4.15}$$

In these equations the subscripts s and pl represent the spacecraft and the planet respectively. The subscripts i and f represent the initial situation and final situation. Equation 4.15 can be rewritten to an equation relating the velocity change of the spacecraft to that of the planet:

$$\Delta V_s = V_{s,f} - V_{s,i} = (V_{pl,i} - V_{pl,f})\frac{m_{pl}}{m_s} \tag{4.16}$$

A typical planet mass is between 10^{24} kg and 10^{27} kg, whereas a typical spacecraft mass is 10^3 kg. This means that the planet's velocity remains effectively unchanged (10^{-18} to 10^{-21} m/s) for a typical fly-by velocity in the order of 10^3 m/s.

Figure 4.4 gives an overview of the in-plane geometry of a swing-by. The spacecraft enters the SoI of the planet at hyperbolic excess velocity $V_{\infty,i}$. The distance from the asymptote of this hyperbolic trajectory to the center of mass of the planet is denoted by the impact parameter B. $V_{\infty,f}$ represents the outgoing hyperbolic excess velocity. V_p and r_p are the velocity and radius of the hyperbolic trajectory at pericenter. V'_{pl} is the velocity of the planet in the plane of the hyperbolic trajectory in the heliocentric reference frame. The definition of the bending angle α and the angle β becomes clear in Figure 4.4.

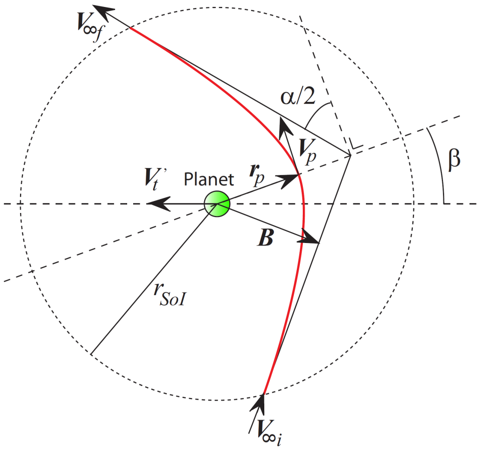


Figure 4.4: An in-plane geometry representation of a gravity assist [Melman, 2007]. Note that $\mathbf{V}'_t$ is equal to $\mathbf{V}'_{pl}$ in the text.

As a result of energy conservation in the planetocentric reference frame, the magnitude of $V_{\infty,i}$ and $V_{\infty,f}$ is equal. Their direction is different though. This has implications on the velocities of the spacecraft in the heliocentric reference frame. This principle is sketched in Figure 4.5. In this figure $V_{s,i}$ and $V_{s,f}$ are the initial and final spacecraft velocities in the heliocentric reference frame. It can thus be seen that the magnitude of the velocity, and thus the orbital energy, can be changed in the heliocentric reference frame.

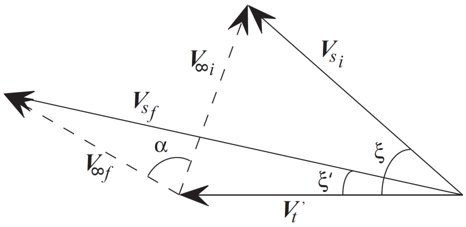


Figure 4.5: A 2D velocity vector diagram for a gravity assist [Melman, 2007].

By selecting the appropriate impact parameter B and the right approach, the amount and direction of this velocity change can be changed. Passing in front of the planet reduces the velocity, passing behind increases this velocity. By selecting the right incoming rotation angle, the swing-by can also be used to change the 3D orientation of the spacecraft orbit.

Calculations

By looking at Figure 4.4 and applying conservation of momentum, the following equation can be set up:

$$BV_\infty = r_p V_p \tag{4.17}$$

This equation can be rewritten to the following equation for the pericenter radius [Cornelisse et al., 1979]:

$$r_p = -\frac{\mu_{pl}}{V_\infty^2} + \sqrt{\frac{\mu_{pl}^2}{V_\infty^4} + B^2} \tag{4.18}$$

It can be derived that the deflection angle and eccentricity of the hyperbolic trajectory are related as in Equation 4.19 [Cornelisse et al., 1979].

$$\cos\left(\frac{\pi}{2} + \frac{\alpha}{2}\right) = -\sin\frac{\alpha}{2} = -\frac{1}{e} \tag{4.19}$$

From this and using the geometrical relations for the hyperbola, the following relation for α can be derived:

$$\sin\frac{\alpha}{2} = \frac{1}{1 + \frac{r_P V_\infty^2}{\mu_{pl}}} = \frac{1}{\sqrt{1 + \frac{B^2 V_\infty^4}{\mu_{pl}}}} \tag{4.20}$$

Equation 4.20 shows that the deflection angle depends on the planet's gravitational parameter, μ_{pl}; the impact parameter, B; and the excess velocity, V_∞.

Restrictions

Not all gravity assists will be possible in practice. Of course the pericenter radius needs to be larger than the planet radius to prevent the spacecraft from crashing. Also the planet may have a dense atmosphere at low altitudes, which could cause an unwanted degradation of the trajectory. This effect can also be used as an advantage, because it may help to increase the deflection angle. This is called an aerogravity assist. Although much research is done into this field, it has not yet been used in practice [Johnson and Longuski, 2002] [Melman, 2007]. Because of this, and the fact that it is not the main subject in this thesis, the aerogravity assist will not be treated in this thesis. To prevent atmospheric effects, a small additional margin will be put on the minimum pericenter radius.

Besides the atmosphere, other aspects may also disable a spacecraft to fly very close to the planets. Amongst others, these can be the navigation accuracy that is required, or the radiation that the spacecraft must endure near the planet. These radiation levels tends to be very high for some planets, for instance Jupiter.

4.4.2 Unpowered Gravity-Assist Propagation

The gravity-assist propagation problem can be stated as follows: given the heliocentric incoming velocity $\boldsymbol{V}_{in}$ at which the spacecraft enters the SoI and various parameters describing the gravity assist, calculate the heliocentric outgoing velocity $\boldsymbol{V}_{out}$.

The parameters that are used to define the gravity assist may differ. Two parameters are required to define an unpowered gravity assist. In here the definitions used in GTOP will be adopted [Izzo and Vinko, 2012]. It is noted explicitly that this is NOT the calculation scheme presented in both papers on GTOP: [Vinko et al., 2007a] and [Vinko and Izzo, 2008]. A different definition of the coordinate frame is used therein. Furthermore a mistake is present in calculating the eccentricity. The equation scheme presented here was taken directly from the GTOP code and is similar to the scheme presented by Izzo in Appendix 7A of [Conway, 2010].

To define the gravity assist bending angle, the pericenter radius, r_p, will be used. This is useful because it allows to constrain the possible range of pericenter radii directly in the decision vector belonging to a trajectory. A 3D rotation inclination angle, b_{incl} is then used to define the bending angle in 3D. Its definition can be seen in Figure 4.6. Note that the definitions of the unit vector directions i_x, i_y and i_z are discussed in the following calculation scheme.

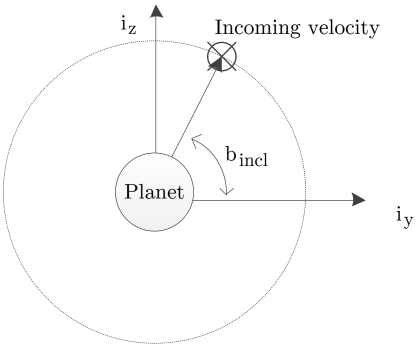


Figure 4.6: The definition of the 3D rotation inclination angle b_{incl}.

First the incoming planetocentric velocity vector $\tilde{\boldsymbol{V}}_{in}$ is calculated as in Equation 4.21. Notice that the tilde (˜) is used to distinguish between planetocentric and heliocentric reference frames.

$$\tilde{\boldsymbol{V}}_{in} = \boldsymbol{V}_{in} - \boldsymbol{V}_{pl} \tag{4.21}$$

From this the eccentricity and bending angle are calculated as in Equations 4.22 and 4.23.

$$e = 1 + \frac{r_p |\tilde{\boldsymbol{V}}_{in}^2|}{\mu_{pl}} \tag{4.22}$$

$$\alpha = 2 \arcsin \frac{1}{e} \tag{4.23}$$

Subsequently the outgoing planetocentric velocity vector $\tilde{\boldsymbol{V}}_{out}$ is calculated:

$$\tilde{\boldsymbol{V}}_{out} = |\tilde{\boldsymbol{V}}_{in}| \cdot (\cos\alpha \; \boldsymbol{i}_x + \cos b_{incl} \sin\alpha \; \boldsymbol{i}_y + \sin b_{incl} \sin\alpha \; \boldsymbol{i}_z) \tag{4.24}$$

in which the coordinate frame is determined as follows:

$$\boldsymbol{i}_x = \frac{\tilde{\boldsymbol{V}}_{in}}{|\tilde{\boldsymbol{V}}_{in}|} \tag{4.25a}$$

$$\boldsymbol{i}_y = \frac{\boldsymbol{i}_x \times \boldsymbol{V}_{pl}}{|\boldsymbol{i}_x \times \boldsymbol{V}_{pl}|} \tag{4.25b}$$

$$\boldsymbol{i}_z = \boldsymbol{i}_x \times \boldsymbol{i}_y \tag{4.25c}$$

Finally the outgoing heliocentric velocity $\boldsymbol{V}_{out}$ is calculated using Equation 4.26.

$$\boldsymbol{V}_{out} = \boldsymbol{V}_{pl} + \tilde{\boldsymbol{V}}_{out} \tag{4.26}$$

Verification

The unpowered gravity-assist propagator was verified using a test case based on GTOP. The first swing-by of the best putative Messenger-Easy problem was used as test case. The incoming heliocentric velocity and the gravitational parameters were taken directly from GTOP with 15-digit accuracy. The pericenter radius and rotation angle were taken from the GTOP website. Subsequently the gravity assist was propagated using both the GTOP code as well as using the newly developed Tudat function. The outgoing heliocentric velocities are shown in Table 4.3.

Table 4.3: Comparison of the unpowered gravity assist method with GTOP results.

Variable	GTOP result [m/s]	Tudat result [m/s]	Absolute difference [m/s]	Relative difference [-]
$V_{out,x}$	12868.524	12868.524	$3.8{\cdot}10^{-11}$	$3.0{\cdot}10^{-15}$
$V_{out,y}$	-22821.445	-22821.445	$5.1{\cdot}10^{-11}$	$2.2{\cdot}10^{-15}$
$V_{out,z}$	-775.698	-775.698	$1.3{\cdot}10^{-11}$	$1.7{\cdot}10^{-14}$

It is concluded from Table 4.3 that the differences between both methods are negligable.

4.4.3 Powered Gravity-Assist Propagation

The velocity change of a gravity assist may be increased by including a powered maneuver. The most efficient location to perform an impulse is at the pericenter of the hyperbolic trajectory. The geometry of the swing-by changes as a result of the impulse and the bending angle of the first and second part are different, as can be seen in Figure 4.7. [Melman, 2007]

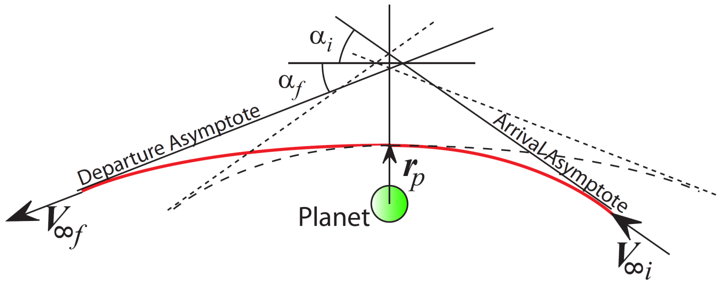


Figure 4.7: The geometry of a powered swing-by with $V_{\infty,out} > V_{\infty,in}$. Adaptation of picture in [Melman, 2007].

A powered gravity assist is often used for gravity-assist calculators, but has not been encountered for propagators in literature. Myatt et al. [2004], Di Lizia et al. [2004], Vasile and de Pascale [2006], Vinko et al. [2007a], Vasile et al. [2011] and Conway [2010] all used unpowered gravity-assist propagators. A powered gravity assist requires a better guidance and navigation system, but may help in reducing the ΔV significantly. Because this thesis aims to quantify this potential gain, the calculation scheme was adapted to allow for a powered gravity assist.

The following section describes the calculation scheme of a powered gravity-assist propagator. It is assumed that the ΔV will be applied at the pericenter. It is argued by Melman [2007] that a maneuver to increase the bending angle may be beneficial as well, which would be applied upon entering the SoI. In practice this maneuver is very expensive though, because of which it is not encountered in optimal trajectories. Hence it seems like a fair assumption that the ΔV will only be applied at pericenter.

Calculation Scheme

The basic calculation scheme of the unpowered case will be used for the powered case as well. The difference is that the incoming and outgoing bending angles are different and need to be calculated separately. Also the absolute outgoing velocity changes. The pericenter radius, r_p, and the inclination angle, b_{incl}, will be used here as well to define the swing-by. Additionally the ΔV applied at pericenter will be used as a third parameter.

After having calculated $\tilde{\boldsymbol{V}}_{in}$ and e_{in} using Equations 4.21 and 4.22, the bending angle of the incoming leg can be calculated using Equation 4.27 below.

$$\alpha_{in} = \arcsin \frac{1}{e_{in}} \tag{4.27}$$

Subsequently the incoming and outgoing pericenter velocities, $V_{p,in}$ and $V_{p,out}$, are calculated using Equations 4.28 and 4.29.

$$V_{p,in} = \sqrt{|\tilde{\boldsymbol{V}}_{in}|^2 \cdot \frac{e_{in}+1}{e_{in}-1}} \tag{4.28}$$

$$V_{p,out} = V_{p,in} + \Delta V \tag{4.29}$$

from which the outgoing bending angle and total bending angle can be calculated using Equations 4.30 and 4.31.

$$\alpha_{out} = \arcsin \frac{1}{1 + \frac{r_p \cdot |\tilde{\boldsymbol{V}}_{in}|^2}{\mu_{pl}}} \tag{4.30}$$

$$\alpha = \alpha_{in} + \alpha_{out} \tag{4.31}$$

Also the absolute planetocentric outgoing velocity, $\tilde{V}_{out}$, can be determined as follows:

$$\tilde{V}_{out} = \sqrt{V_{p,out}^2 - \frac{2 \cdot \mu_{pl}}{r_p}} \tag{4.32}$$

Now the planetocentric outgoing velocity, $\tilde{\boldsymbol{V}}_{out}$, can be calculated using Equation 4.33.

$$\tilde{\boldsymbol{V}}_{out} = \tilde{V}_{out} \cdot (\cos\alpha \ \boldsymbol{i}_x + \cos b_{incl} \sin\alpha \ \boldsymbol{i}_y + \sin b_{incl} \sin\alpha \ \boldsymbol{i}_z) \tag{4.33}$$

in which the unit vectors are determined as in Equation 4.25. Finally the heliocentric outgoing velocity $\boldsymbol{V}_{out}$ is calculated using Equation 4.26.

Verification

The powered gravity-assist propagator was first verified using the same test as the unpowered gravity-assist propagator. In case the ΔV is set to 0, the powered gravity-assist propagator should of course yield the same results as the unpowered gravity assist calculated. The resulting outgoing heliocentric velocities are shown in Table 4.4.

Table 4.4: Comparison of the powered gravity-assist propagator with GTOP results for an unpowered gravity assist.

Variable	GTOP result [m/s]	Tudat result [m/s]	Absolute difference [m/s]	Relative difference [-]
$V_{out,x}$	12868.524	12868.524	$4.0 \cdot 10^{-11}$	$3.1 \cdot 10^{-15}$
$V_{out,y}$	-22821.445	-22821.445	$5.1 \cdot 10^{-11}$	$2.2 \cdot 10^{-15}$
$V_{out,z}$	-775.698	-775.698	$1.3 \cdot 10^{-11}$	$1.7 \cdot 10^{-14}$

Again the differences are negligable. It is noted that the difference in $V_{out,x}$ is slightly larger than for the unpowered case. Given that more computations are required in calculating the powered gravity assist, this may be caused by rounding errors. The difference namely approaches machine precision.

Finding a good test case for a powered gravity assist was more difficult, as no similar software was found. Instead an instance of a gravity-assist calculator was reverse-engineered to obtain the accompanying r_p, b_{incl} and ΔV. Subsequently the incoming heliocentric velocity was propagated and compared to the outgoing velocity of the gravity-assist calculator. The test case was based on the first swing-by of the best putative Cassini-1 trajectory. The resulting outgoing velocity is given in Table 4.5.

Table 4.5: Comparison of the powered gravity-assist propagator with a powered GTOP test case.

Variable	GTOP result [m/s]	Tudat result [m/s]	Absolute difference [m/s]	Relative difference [-]
$V_{out,x}$	37954.243	37954.243	$8.7 \cdot 10^{-11}$	$2.3 \cdot 10^{-15}$
$V_{out,y}$	-14093.047	-14093.047	$6.9 \cdot 10^{-11}$	$4.9 \cdot 10^{-15}$
$V_{out,z}$	-5753.537	-5753.537	$4.7 \cdot 10^{-11}$	$8.2 \cdot 10^{-14}$

It is concluded from Table 4.5 that the difference is again negligible, validating the implementation of the powered gravity-assist propagation.

4.5 Gravity-Assist Calculator

This section discusses the gravity-assist calculator. First an introduction into the method is given in Section 4.5.1. Subsequently the calculation scheme is discussed in Section 4.5.2. Finally the verification of the function that was written is done in Section 4.5.3.

4.5.1 Introduction

The physics behind the gravity-assist calculator are of course the same as that behind the propagator, which was discussed in Section 4.4.1.
The gravity-assist calculator calculates the ΔV required to achieve a certain heliocentric outgoing velocity, $\boldsymbol{V}_{out}$, given a certain heliocentric incoming velocity $\boldsymbol{V}_{in}$. The ΔV further depends on the gravitational parameter, μ_{pl} and the minimum pericenter radius, $r_{p,min}$ of the planet in question.

In this thesis, the basics of the gravity-assist calculator described by Melman [2007] are worked out. Similar to the gravity-assist calculator in GTOP, Melman [2007] uses a maneuver at pericenter to patch the absolute incoming and outgoing velocities. A difference with respect to the gravity-assist calculator in GTOP is that an additional impulse is considered by Melman [2007]. In case the bending angle can not be met Melman [2007] applies additional burns upon entering or leaving the SoI to obtain the correct bending angle. Other calculators just give a penalty in case the bending angle cannot be achieved with the minimum pericenter constraint. In practice both methods yield the same result when optimizing trajectories. The bending angle maneuver namely requires a large amount of ΔV that is not present in an optimal trajectory in practice, at least as far as the author is aware of. On the other hand this ΔV may be a better way to distinguish between suboptimal trajectories and is also computationally faster than the method described in [Izzo and Vinko, 2012].

4.5.2 Calculation Scheme

First the incoming and outgoing planetocentric velocities, $\tilde{\boldsymbol{V}}_{in}$ and $\tilde{\boldsymbol{V}}_{out}$, are determined using Equations 4.21 and 4.26. The angle between $\tilde{\boldsymbol{V}}_{in}$ and $\tilde{\boldsymbol{V}}_{out}$ is equal to the desired bending angle α. By rewriting Equation 4.20 for the incoming and outgoing bending angle, the following relation between α and the pericenter radius r_p can be determined.

$$\alpha = \alpha_i + \alpha_f = \arcsin \frac{1}{1 + \frac{r_p |\tilde{\boldsymbol{V}}_{in}|^2}{\mu_t}} + \arcsin \frac{1}{1 + \frac{r_p |\tilde{\boldsymbol{V}}_{out}|^2}{\mu_t}} \tag{4.34}$$

The maximum bending angle can now be determined by setting r_p to the minimum pericenter radius $r_{p,min}$ in Equation 4.34. A different calculation scheme is followed depending on whether this required bending angle can be met without additional impulses. First the case where the bending angle can be met is discussed. Subsequently the case where it cannot be met will be dealt with.

Sufficient Bending Angle

The key in this case is to determine the incoming and outgoing pericenter velocities. To do so the pericenter needs to be determined. Equation 4.34 relates r_p to α. Although this equation cannot be solved directly, it can be found using a root-finder. A Newton-Raphson scheme, as given in Equation 4.35 [Montenbruck and Gill, 2000], suffices for this operation.

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \tag{4.35}$$

Two different options are available to find the root. First of all the pericenter radius can be used as iteration parameter, which is done by Izzo and Vinko [2012]. Secondly the equation can be rewritten to an equation for the incoming eccentricity e_i. Subsequently e_i can be used as iteration parameter, as proposed by Melman [2007].

Iteration Using the Pericenter Radius

First the incoming and outgoing semi-major axes are determined using Equation 4.36.

$$a_{in/out} = \frac{-\mu_{pl}}{|\tilde{\boldsymbol{V}}_{in/out}|^2} \tag{4.36}$$

From Equation 4.34 the root-finding function and its derivative can now be derived to be:

$$f(r_p) = \arcsin \frac{a_{in}}{a_{in} - r_p} + \arcsin \frac{a_{out}}{a_{out} - r_p} - \alpha \tag{4.37}$$

$$f'(r_p) = -\frac{\frac{a_{in}}{a_{in} - r_p}}{\sqrt{r_p \cdot (r_p - 2 \cdot a_{in})}} - \frac{\frac{a_{out}}{a_{out} - r_p}}{\sqrt{r_p \cdot (r_p - 2 \cdot a_{out})}} \tag{4.38}$$

in which $r_p = r_{p,min}$ can be used to start up the iteration process. From this pericenter radius the incoming and outgoing eccentricities and pericenter velocities can be determined using Equations 4.39 and 4.40.

$$e_{in/out} = 1 - \frac{r_p}{a_{in/out}} \tag{4.39}$$

$$V_{p,in/out} = |\tilde{\boldsymbol{V}}_{in/out}| \cdot \sqrt{\frac{e_{in/out} + 1}{e_{in/out}}} \tag{4.40}$$

Finally the difference between the pericenter velocities is the ΔV that is required for the gravity assist.

Iteration Using the Incoming Eccentricity

Also for this scenario the incoming and outgoing semi-major axes are determined first using Equation 4.36. Subsequently Equation 4.34 is rewritten using the Vis-Viva integral to an equation relating e_{in} to α. The result is given in Equation 4.41.

$$\alpha = \arcsin\frac{1}{e_{in}} + \arcsin\frac{1}{e_{out}} = \arcsin\frac{1}{e_{in}} + \arcsin\frac{1}{1-\frac{a_{in}}{a_{out}}(1-e_{in})} \tag{4.41}$$

From Equation 4.41 the root-finding function and its derivative can now be derived to be:

$$f(e_{in}) = \arcsin\frac{1}{e_{in}} + \arcsin\frac{1}{e_{out}} - \alpha = \arcsin\frac{1}{e_{in}} + \arcsin\frac{1}{1-\frac{a_{in}}{a_{out}}(1-e_{in})} - \alpha \tag{4.42}$$

$$f'(e_{in}) = -\frac{1}{e_{in}\sqrt{e_{in}^2-1}} - \frac{\frac{a_{in}}{a_{out}}}{\left(1-\frac{a_{in}}{a_{out}}(1-e_{in})\right)\cdot\sqrt{\left(1-\frac{a_{in}}{a_{out}}(1-e_{in})\right)^2-1}} \tag{4.43}$$

[Melman, 2007] used $e_{in,0} = 1.01$ to start up the iteration process. The author in this thesis found that the Newton Raphson algorithm would indicate $e_{in,1}$ to be lower than 1.0 for cases where $|\tilde{\boldsymbol{V}}_{out}| \gg |\tilde{\boldsymbol{V}}_{in}|$. This subsequently caused the trigonometric equations to crash in these limit cases. Experimentally it was determined that $e_{in,0}$ should be set to $1 + 10^{-10}$ in case $\frac{|\tilde{\boldsymbol{V}}_{out}|}{|\tilde{\boldsymbol{V}}_{in}|} > 100$.

Having determined the incoming eccentricity, the outgoing eccentricity can be determined using Equation 4.44. Subsequently the pericenter velocities and the ΔV are calculated similar to the other method.

$$e_{out} = 1 - \frac{a_{in}}{a_{out}}(1-e_{in}) \tag{4.44}$$

Insufficient Bending Angle

In case the bending angle is insufficient, the pericenter radius will be chosen equal to the minimum pericenter radius to ensure the largest possible bending angle is achieved. The remaining bending angle will need to be patched using an additional burn. The ΔV to achieve this extra bending of the trajectory can be calculated using Equation 4.45. [Wakker, 2007]

$$\Delta V_{bend} = 2V_{\infty}\sin\frac{\Delta\alpha}{2} \tag{4.45}$$

It is argued by [Wakker, 2007] that this change can be performed most efficiently at the point where the velocity is lowest, which is either upon entering or leaving the SoI. Note that the ΔV rises quickly as a result of this burn.

Apart from matching the bending angle, also the absolute incoming and outgoing velocities need to be matched. Because the pericenter radius was already determined, this can be done using Equations 4.36, 4.39 and 4.40.

4.5.3 Verification

One test was present in Tudat. It was however completely inadequate to verify the method. Only one very trivial case was tested in which only the bending angle effect ΔV was tested of the gravity-assist calculator.

The previous version included the bending angle calculation and the pericenter velocity matching function based on iteration on the incoming eccentricity. The author of this thesis found various mistakes in the logical structure of the old calculation scheme, leading to completely wrong values in some cases. It was hence completely rewritten and also an option to iterate on the pericenter radius was added.

To verify the new implementation a total of 10 additional test cases were written.

First of all four test cases were written that require the iteration scheme. One test was based on the first swing-by of the best putative solution to the Cassini-1 problem in GTOP [Izzo and Vinko, 2012]. Subsequently three limit cases were set up. One in which $|\tilde{\boldsymbol{V}}_{in}|$ was only 0.01 m/s, one in which $|\tilde{\boldsymbol{V}}_{out}|$ was only 0.01 m/s and one in which both were small, but not equal. The calculated ΔV of the last three were verified using an excel sheet. The excel sheet was also uploaded to the Tudat site [Tudat, 2012]. Both the r_p-iteration scheme and the e_{in}-iteration scheme were subjected to these test. The Cassini-1 trajectory tests had a relative accuracy of $6{\cdot}10^{-14}$ and the limit cases were accurate to at least $1{\cdot}10^{-9}$.

Furthermore the test in which only the bending angle effect ΔV is present still functioned. Another test case in which both the bending and the velocity effect ΔV was present was written. An excel sheet was written for this case as well, which was accurate to $1{\cdot}10^{-12}$. Finally a test case was written in which no maneuver should be necessary, which was indeed correctly identified by the gravity-assist calculator.

Finally a brief test was done by the author to determine which iteration method would be used throughout the thesis. The number of iterations required to solve all gravity assists in a short optimization run of the Cassini-1 MGA trajectory revealed that iteration on e_{in} was 20 % faster than iteration on r_p. Hence iteration on e_{in} will be used throughout the thesis.
It has to be noted though that it is possible that the iteration on r_p can be sped up by selecting the initial guess more carefully. Also it is possible that the performance changes from trajectory to trajectory.

4.6 Escape/Capture

Escape and capture maneuvers are easy to calculate. The ΔV that should be applied at pericenter can be calculated using Equation 4.46. [Wakker, 2007]

$$\Delta V = \sqrt{\frac{2\mu}{a(1-e)} + V_\infty^2} - \sqrt{\frac{2\mu}{a(1-e)} - \frac{\mu}{a}} \tag{4.46}$$

in which μ is the gravitational parameter of the departure/arrival planet, a and e are the semi-major axis and the eccentricity of the parking/capture orbit and V_∞ is the hyperbolic excess velocity after the escape maneuver, or before the capture maneuver.

The old method worked fine, but was rather devious. A class had to be made, after which all four input variables had to be set independently before a calculation could finally be made. Hence the code was rewritten as a free function. Also only circular parking/capture orbits were present in the unit test. Hence an elliptical test case was added, which was verified using a different formula in [Wakker, 2007]. All results had a relative accuracy of 10^{-15}.

Chapter 5

Trajectory Models

This chapter discusses the different trajectory models that were implemented. To this end Section 5.1 gives a general introduction into the trajectory models. Also the general set-up of the software is discussed here. Subsequently the trajectory models are discussed indvidually, starting with the MGA trajectory model in Section 5.2, followed by the MGA-1DSM-VF trajectory model in Section 5.3 and finally the MGA-1DSM-PF trajectory model in Section 5.4. Section 5.5 discusses the capture leg definition, which is applicable to all trajectory models. The verification process of the trajectory models is discussed in the relevant sections.

5.1 Introduction

This section gives an introduction into the software architecture for the trajectory models in Section 5.1.1. Subsequently the verification process that was used to verify the functionality and accuracy of the trajectory models is discussed in Section 5.1.2.

5.1.1 Trajectory Modeling Software Architecture

As was already discussed in Section 2.2, many options are available for the choice of the trajectory model. Three different trajectory models will be investigated in this thesis: the Multiple Gravity Assist (MGA) trajectory model, the Multiple Gravity Assist with 1 Deep Space Maneuver per leg using Velocity Formulation (MGA-1DSM-VF) trajectory model, and the Multiple Gravity Assist with 1 Deep Space Maneuver per leg using Position Formulation (MGA-1DSM-PF) trajectory model.

These trajectory models all share one common feature: they are all built up of multiple legs. A leg in this context means a transfer from a planet to another planet. The way in which this transfer is performed varies for the different models, but the initial and final conditions are the same. One of the goals of this thesis is to set up a very general tool that can calculate a wide variety of trajectories. Hence it is desirable to make sure that a trajectory can be built up by adding different legs, possibly belonging to different trajectory models. A distinction is made here into three different leg types. These three leg types are represented schematically in Figure 5.1 and described in short in Table 5.1.

The three different trajectory models under consideration all have different calculation schemes for the departure and swing-by legs. The capture leg definition is uniform for the different models. This leads to the inheritance architecture diagram that can be seen in Figure 5.2.

Each leg has a set of basic functions:

- CalculateLeg: fully calculates the leg returning the ΔV required.
- Maneuvers: returns location and size of all the maneuvers in the leg.
- IntermediatePoints: returns the position of the spacecraft at specified time-intervals during the leg.

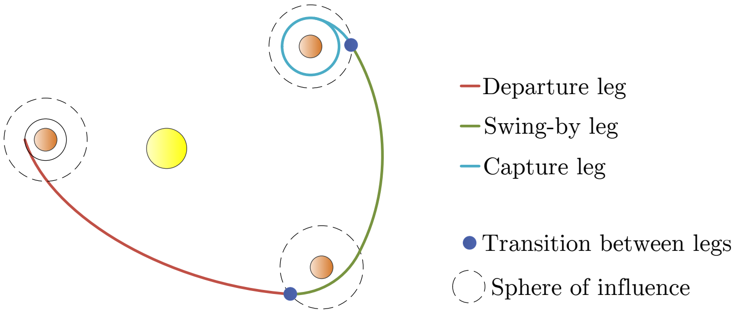


Figure 5.1: A schematic representation of the three different leg types. Note that a capture leg may also be followed by another departure leg for some missions. 'Leg' in this sense does not stand for a transfer from planet to planet, but for a capture maneuver at a certain planet that is possibly followed by a capture period at that planet.

Table 5.1: Three different leg type definitions that are applicable to all three trajectory models.

Leg Type	Definition	Example applications
Departure	From a parking orbit around a planet to a different planet.	An interplanetary mission departing Earth. Sample-return missions. Missions visiting multiple asteroids.
Swing-by	From just before performing a swing-by to a different planet	An interplanetary mission involving swing-bys.
Capture	From just before arriving at a planet to just before leaving the parking orbit.	An interplanetary mission involving capture. Sample-return missions. Missions visiting multiple asteroids.

And some functions to speed up computation by allowing different trajectories with a similar leg structure to be computated using the same trajectory object:

- UpdateDefiningVariables: updates the variables that define a specific instance of the leg.
- UpdateEphemeris: updates planet positions and velocities.

The implementations of the departure and swing-by legs for each trajectory model are discussed to greater detail in Sections 5.2 to 5.4. The implementation of the capture leg, which is the same for all three trajectory models, is discussed in Section 5.5.

The basic idea is now that a trajectory class concatenates various legs to obtain one trajectory. These legs can be of different trajectory models and differ depending on the mission requirements. Note that of course a departure leg is only added at the start of the mission or after a capture leg; a swing-by leg and a capture leg are only added after a departure leg or a swing-by leg.
The trajectory class also implements the ephemeris data, which is used to determine the planetary positions and velocities. Since this thesis is looking into preliminary trajectory design, the precision of the ephemeris data is comparatively less important than the speed of computation. Hence the approximate planet positions calculation scheme given by Standish [2012] is used by default. This is also the default ephemeris available in Tudat [Tudat, 2012]. In order to compare this trajectory class with the trajectories available in GTOP, the ephemeris that is used in GTOP is also imported into this trajectory class. This ephemeris uses a quadratic to quintic polynomial fit to determine planetary positions, depending on the planet in question [Izzo and Vinko, 2012].
Finally, the trajectory also contains functionality to update the defining variables and the ephemeris for all legs to speed up the computation process.

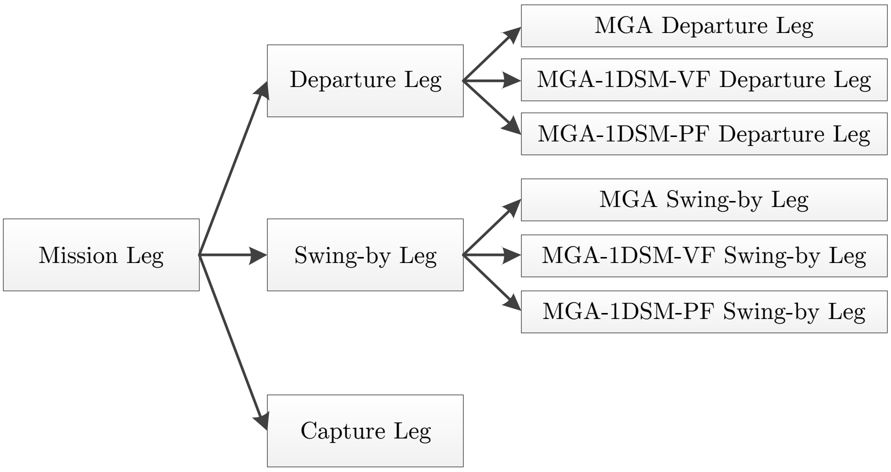


Figure 5.2: A diagram showing the inheritance of the different mission legs. Note that the capture leg definition is not dependent on the trajectory model that is used.

5.1.2 Verification Method

Verifying the trajectory modeling software can be a difficult process, since very often no good benchmark results are present. The results of the mission legs depend heavily on the ephemeris that is used and also the values for the astrodynamical constants that were assumed. Many publications do not give a complete list of all the values assumed, complicating the verification process.
However in this case a very good benchmark is available. Vinko and Izzo [2008] have developed the Global Trajectory Optimimization Problems (GTOP). The accompanying software is also publicly available, both at the ESA-ACT website [Izzo and Vinko, 2012], as well as incorporated in the PaGMO software package [ESA-ACT, 2012], which is described in more detail in Chapter 6. A big advantage is that by accessing the code directly, all input variables can be obtained and also any intermediate calculation variables can be checked. Especially since other parameters than the ΔV are equally critical, this is considered very valuable.

The GTOP database includes an MGA trajectory model as well as an MGA-1DSM velocity formulation trajectory model in which swing-bys are assumed to be unpowered. This does not cover all the trajectory models that are to be tested. However by going into the code directly, sufficient benchmark cases can be obtained from this code to also test the full MGA-1DSM-VF trajectory model and the MGA-1DSM-PF trajectory model.

Each leg function will be subjected to at least one similar leg obtained from GTOP. Also various entire trajectories from GTOP will be compared with the results obtained from this trajectory class to verify its accuracy and computation speed. These verifications are discussed to greater detail in Sections 5.2 to 5.5.

5.2 MGA Trajectory Model

This section discusses the Multiple Gravity Assist (MGA) trajectory model. Section 5.2.1 gives an overall description of this trajectory model. Section 5.2.2 discusses the verification of the departure leg module and Section 5.2.3 discusses the verification of the swing-by leg module. Finally the entire trajectory model is verified in Section 5.2.4.

5.2.1 Description

The MGA trajectory model has been employed very often in high-thrust trajectory design. The exact implementation of the MGA model differs because some models assume powered swing-bys and others

assume unpowered swing-bys. Further differences are present in the implementation of the powered swing-by. These models allowed researchers and mission planners to study many different trajectories using numerical methods in a short amount of time, with reasonable accuracy. This was a large improvement because trajectory planning used to require a great deal of experience. Although by far not the first, a comprehensive application was done by Petropoulos et al. [2000]; the model was applied to examine many different trajectories including multiple gravity assists for a mission to Jupiter.
Also at TU Delft many students have used this model to design interplanetary trajectories. Examples can be found in the thesis reports of [Schlijper, 2003] and [Melman, 2007].

The MGA trajectory makes the following basic assumptions regarding the trajectory: [Olympio et al., 2007]

- Patched-conics is applied and perturbations are neglected.
- The planetary sequence is fixed.
- Impulses only possible at departure, arrival and each swing-by planet.
- There is less than one revolution for each transfer.
- Transfer directions are all counter-clockwise.

The MGA trajectory model employs the following variables to describe the trajectory: [Izzo et al., 2007]

- t_0: the starting date of the mission,
- T_i: the time of flight of all n interplanetary legs of the trajectory.

The visitation times of the planets are subsequently calculated using the following equation:

$$t_i = t_0 + \sum_{j=1}^{i-1} T_j \tag{5.1}$$

The positions of the planetary encounters are now easily obtained from the ephemeris data, the planetary sequence and the visitation times. Because perturbations are neglected and the patched-conics assumption is applied, Lambert targeting can be used to obtain the initial and final velocities of each leg. By neglecting multi-revolution transfers and fixing the transfer direction, this provides a unique solution for all legs.
The required departure maneuver is calculated for the first leg. For all swing-by legs the gravity-assist calculator is used to determine the ΔV that is required to patch the incoming and outgoing velocities at the planetary encounter. Typically a capture maneuver is added after the final leg, yielding the following different contributions to the final ΔV-budget:

- $|\Delta V_0|$: the velocity change required to escape the parking orbit around the starting planet and obtain the required initial velocity of the first leg.
- $|\Delta V_i|$: the velocity change required at the gravity assists, in case these are powered.
- $|\Delta V_n|$: the velocity change to obtain the required orbit at the final plane from the final velocity of the last leg.

This calculation scheme is represented schematically in Figure 5.3.
An impression of a typical trajectory that can be modelled using this trajectory model is given in Figure 5.4.
The main advantage of the MGA trajectory model is the small amount of variables needed to describe the trajectory. This generally allows for a relatively fast and robust optimization.

5.2.2 Departure Leg Verification

The first leg of the Cassini-1 trajectory is used as a benchmark for testing the departure leg class. This leg is a transfer from Earth to Venus, as can be seen also in Figure 5.4.

All required input and model parameters, such as planetary positions, gravitational parameters and visitation times, were extracted from the GTOP model with an accuracy of 15 digits. Subsequently this leg was simulated using the code developed in this thesis. A comparison of the associated ΔV-cost and

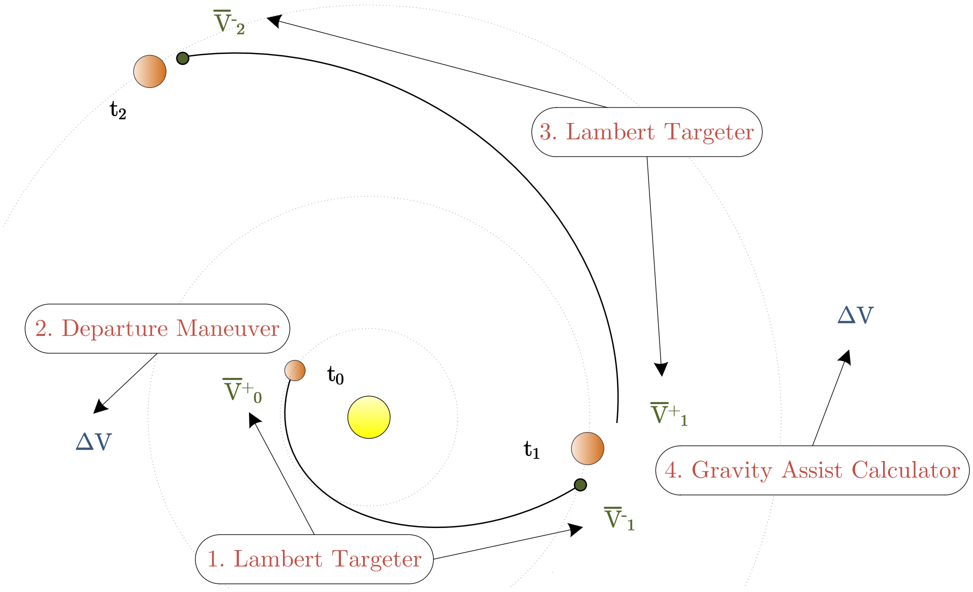


Figure 5.3: A schematic representation of the departure and swing-by legs in an MGA trajectory model.

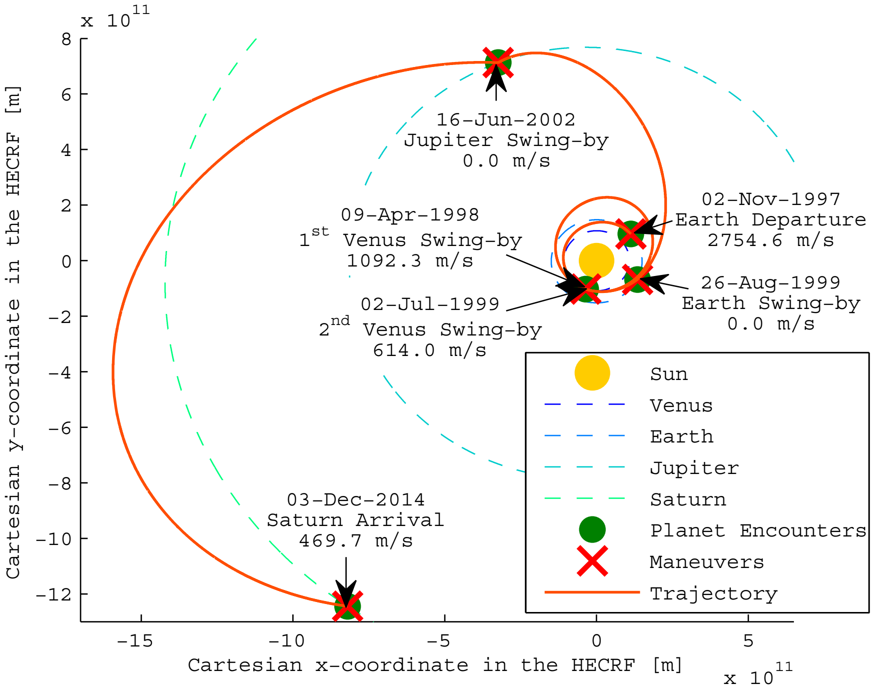


Figure 5.4: The best putative solution for the Cassini-1 trajectory [Vinko and Izzo, 2008], calculated and plotted using the trajectory class described in this thesis. Note that the purpose of this picture is to give the reader an impression of a typical trajectory modeled using the MGA trajectory model. The verification of the implementation of the trajectory model will be dealt with in subsequent sections.

the velocity when arriving at Venus between the GTOP software and the software programmed by the author is given in Table 5.2.

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting.

Table 5.2: Test of the MGA departure leg class against the first leg of the Cassini-1 trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result	MGA departure leg	Absolute	Relative
	[m/s]	[m/s]	difference [m/s]	difference [-]
ΔV-cost	2754.635987	2754.635834	$1.5{\cdot}10^{-4}$	$5.5{\cdot}10^{-8}$
V_x at Venus	34216.48275	34216.48351	$7.6{\cdot}10^{-4}$	$2.2{\cdot}10^{-8}$
V_y at Venus	-15170.14407	-15170.14264	$1.4{\cdot}10^{-3}$	$9.4{\cdot}10^{-8}$
V_z at Venus	395.792122	395.792115	$7.0{\cdot}10^{-6}$	$1.8{\cdot}10^{-8}$

5.2.3 Swing-by Leg Verification

The second leg of the Cassini-1 trajectory is used as a benchmark for testing the swing-by leg class. This leg is a transfer from Venus to Venus, as can be seen also in Figure 5.4.

After setting all the required input and model parameters with an accuracy of 15 digits, the simulation was run. A comparison of the associated ΔV-cost and the velocity when arriving at Venus between the GTOP software and the software programmed by the author is given in Table 5.3.

Table 5.3: Test of the MGA swing-by leg class against the second leg of the Cassini-1 trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result	MGA swing-by leg	Absolute	Relative
	[m/s]	[m/s]	difference [m/s]	difference [-]
ΔV-cost	1090.646229	1090.646311	$8.2{\cdot}10^{-5}$	$7.5{\cdot}10^{-8}$
V_x at Venus	37952.88446	37952.88462	$1.6{\cdot}10^{-4}$	$4.3{\cdot}10^{-9}$
V_y at Venus	-14096.96568	-14096.96573	$5.7{\cdot}10^{-5}$	$4.0{\cdot}10^{-9}$
V_z at Venus	-5753.512458	-5753.512483	$2.4{\cdot}10^{-5}$	$4.3{\cdot}10^{-9}$

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting and the gravity-assist propagation.

5.2.4 Full Model Verification

Also the trajectory class is subjected to a test case to verify its functionality and accuracy. The entire Cassini-1 trajectory is used for comparison of the results. This trajectory was modelled using the trajectory class described in this thesis. The results can be seen in Table 5.4. Note that the comparison was made for a version that uses the standard Tudat ephemeris and for a version that uses GTOP ephemeris.

Table 5.4: Test of the full Cassini-1 trajectory of GTOP with two versions of this trajectory modelled in this thesis' software. Values refer to the total ΔV that needs to be given. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Ephemeris	GTOP result	Trajectory class	Absolute difference	Relative difference
	[m/s]	[m/s]	with GTOP [m/s]	with GTOP [-]
GTOP ephemeris	4930.726868	4930.728448	$1.6{\cdot}10^{-3}$	$3.2{\cdot}10^{-7}$
Tudat ephemeris	n.a.	5635.955020	$7.0{\cdot}10^{2}$	$1.2{\cdot}10^{-1}$

As can be seen from Table 5.4, the difference with respect to the version using GTOP ephemeris is very small and can be deemed to originate from the iterative procedure involved in the Lambert targeting and the gravity assist.

However it is very interesting to note the sensitivity to the ephemeris data. The main reason for this was found to lie in the Lambert targeter for the Venus-Venus transfer. Because the location of Venus is almost exactly the same after one revolution, the plane determination function within the Lambert targeter is highly sensitive to small variations in the z-direction of the location of Venus. This effect can be seen in Figure 5.5: although the planetary positions are almost the same, the spacecraft trajectories differ greatly.

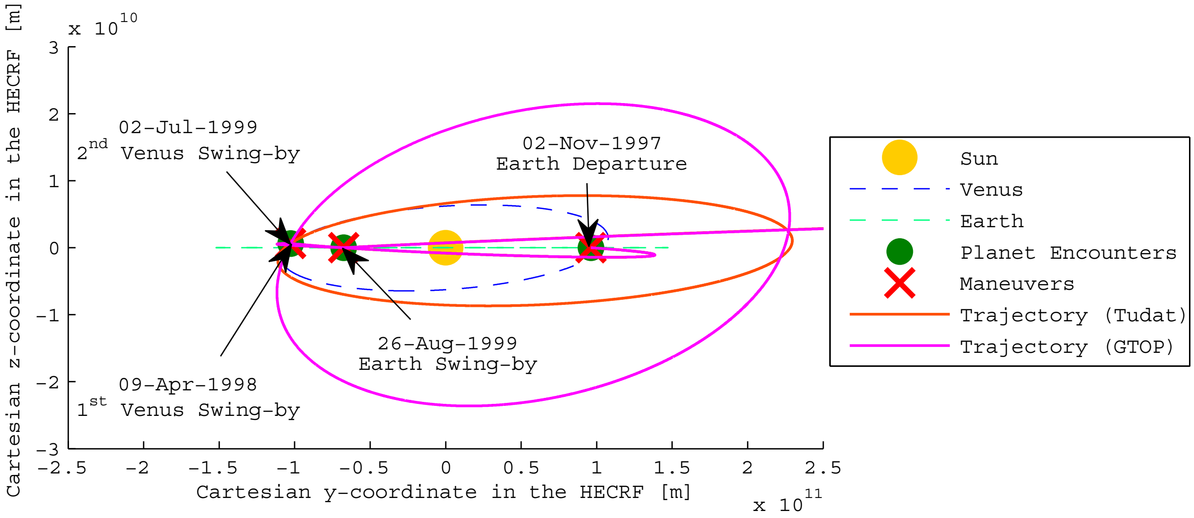


Figure 5.5: Comparison between the trajectories belonging to the best putative solution for the Cassini-1 problem, but calculated using different ephemeris. Zoomed on the Venus-Venus transfer in the y-z frame.

Also Figure 5.6 shows the entire trajectory if Tudat ephemeris is used, which can be compared to the results to the trajectory in Figure 5.4 where GTOP ephemeris is used. Note that the other data and visitation times required to model the trajectory have been kept constant and that the only difference is the ephemeris that is used for both.

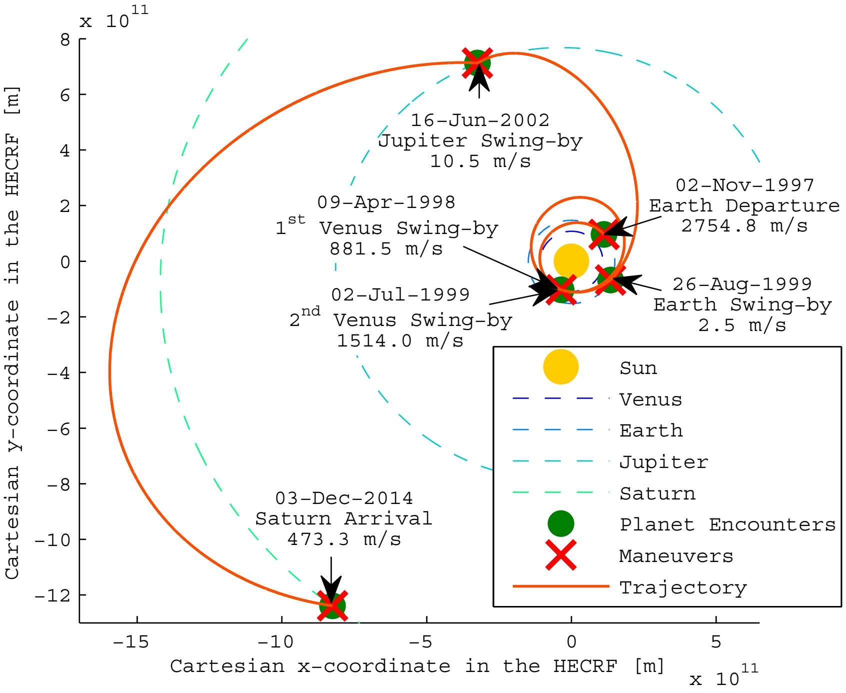


Figure 5.6: The best putative solution for the Cassini-1 problem implemented in the trajectory class described in this thesis utilizing Tudat ephemeris.

Finally the computational speed of this method is compared to that of the GTOP trajectory calculation method. Again the optimal trajectory of the Cassini-1 model is used for comparison. This trajectory was calculated 1 million times and the speed of computation is given in Table 5.5. The test was run on one core of an Intel(R) Core (TM) i7-2630QM @2.00 GHz processor.

Table 5.5: Comparison of the computational speed of GTOP and the trajectory class developed in this thesis.

Type of software	Computation time per trajectory [μs]
GTOP	53.8
Trajectory class	53.2

It can be concluded that the trajectory class in this thesis performs equally well as the GTOP method in terms of speed. This is remarkable because a more object-oriented approach was used in this thesis than what is used in GTOP, which often costs more overhead. The object-oriented approach manages similar performance because the objects can be reused for multiple computations. If a new trajectory object would be created each time, 103.9 μs are required instead of 53.2 μs.

5.3 MGA-1DSM-VF Trajectory Model

This section describes the Multiple Gravity Assist with 1 Deep Space Maneuver per leg using Velocity Formulation (MGA-1DSM-VF) trajectory model. Section 5.3.1 gives an overall description of this trajectory model. Section 5.3.2 discusses the verification of the departure leg module and Section 5.3.3 discusses the verification of the swing-by leg module. Finally the entire trajectory model is verified in Section 5.3.4.

5.3.1 Description

As a first note it is explicitly mentioned that the MGA-1DSM-VF trajectory model is NOT the same as the velocity formulation MGA-1DSM trajectory model often found in literature. Many different authors have written about a similar trajectory model, starting with Myatt et al. [2004] and Di Lizia et al. [2004], subsequently followed by Vasile and de Pascale [2006], Vinko et al. [2007a], Vasile et al. [2011] and Conway [2010]. However all these trajectory models, which are identical except for some variable definitions, assume only unpowered swing-bys are possible. This thesis will investigate the gain of incorporating powered gravity assists. Furthermore it will test the powered gravity assist version of the MGA-1DSM-VF model with the MGA-1DSM-PF model, which uses a powered gravity assist normally.

This MGA-1DSM-VF trajectory model is based on the same principle as described in [Vasile and de Pascale, 2006], but includes the option to perform a powered swing-by. To this end the powered swing-by calculator described in Section 4.4.3 is used. This trajectory model can hence be considered to be more complete than the trajectory model described in other sources. Variable naming is chosen similar to the naming used in [Vinko and Izzo, 2008]

The following assumptions are made in the model:

- Patched-conics is applied and perturbations are neglected.
- The planetary sequence is fixed.
- A maximum of 1 DSM is applied in each leg.
- The Lambert targeter only considers single-revolution counter-clockwise trajectories.

The MGA-1DSM-VF trajectory model employs the following variables to describe the trajectory:

- t_0: the starting date of the mission,
- T_i: the time of flight of all n interplanetary legs of the trajectory,
- η_i: the fraction of the time of flight at which a DSM takes place ($\eta \in [0, 1]$),

with the following additions for a departure leg:

- V_∞: the magnitude of the relative velocity with the departure planet,
- θ: the in-plane angle of the relative velocity with the departure planet,
- ϕ: the out-of-plane angle of the relative velocity with the departure planet,

and the following additions for a swing-by leg:

- r_p: the pericenter radius of the gravity assist at the concerned planet,
- b_{incl}: the rotation angle of the gravity assist,
- ΔV: the magnitude of the ΔV applied at pericenter of the swing-by.

The visitation times of the planets are calculated using Equation 5.1, similar to the MGA trajectory model. Using these visitation times the planetary positions can be obtained.
Next the heliocentric velocities after the departure planet are determined. For swing-by legs this is done by propagating the gravity assist using the powered gravity-assist propagation function described in Section 4.4.3. For the departure legs this velocity is determined by calculating the relative velocity vector $\boldsymbol{V}_0$ with respect to the departure planet and subsequently adding the velocity of the departure planet to this velocity. Equation 5.2 is used to determined $\boldsymbol{V}_0$:

$$\boldsymbol{V}_0 = \begin{bmatrix} \cos\theta\cos\phi \\ \sin\theta\cos\phi \\ \sin\phi \end{bmatrix} V_\infty \tag{5.2}$$

The x-direction of $\boldsymbol{V}_0$ is defined to be aligned with the heliocentric velocity of the departure planet, the z-direction is chosen normal to the orbital plane and finally the y-direction completes the coordinate frame.

After having determined the heliocentric velocity after the departure planet, a Kepler propagator is used to propagate the trajectory up to the moment of application of the DSM. The moment of application of a DSM is calculated using Equation 5.3:

$$t_{dsm,i} = t_0 + \sum_{j=1}^{i-1} T_j + \eta_i T_i \tag{5.3}$$

After the DSM location has been determined, a Lambert targeter is utilized to find the required heliocentric velocity after the DSM and the heliocentric velocity when arriving at the next planet. The ΔV that is required for the DSM is now obtained from the difference between the velocity before the moment of application of the DSM and the required heliocentric velocity after the DSM.

In this way the following different ΔV contributions can be identified for a typical MGA-1DSM-VF trajectory:

- $|\Delta V_0|$: the magnitude of the departure maneuver to obtain the required excess velocity with respect to the departure planet.
- $|\Delta V_{dsm,i}|$: the magnitude of the maneuver at the DSM location to match the final velocity of the first part of the leg with the required initial velocity of the second part of the leg.
- $|\Delta V_i|$: the velocity change at the gravity assists, in case these are powered.
- $|\Delta V_n|$: the velocity change to obtain the required orbit at the final plane from the final velocity of the last leg.

This calculation scheme is represented schematically in Figure 5.7.
An example of a trajectory that can be modelled using this trajectory model can be found in Figure 5.8.

An advantage of the VF is that the options for the trajectories can easily be bounded to include only feasible trajectories. The pericenter radius is used as decision variable and hence does not need to be constrained otherwisely. Furthermore also the ΔV at the gravity assist can be limited to the largest value that is reasonably expected in an optimal trajectory. In this way the propagation of the gravity assist already prunes a large number of practically infeasible trajectories from the search space.
A disadvantage of the VF model is that the final leg is strongly dependent on the first leg. In contrast to the PF model, the legs depend on all previous legs and the velocities therein. The velocities at the end of a leg are namely propagated into the next leg.

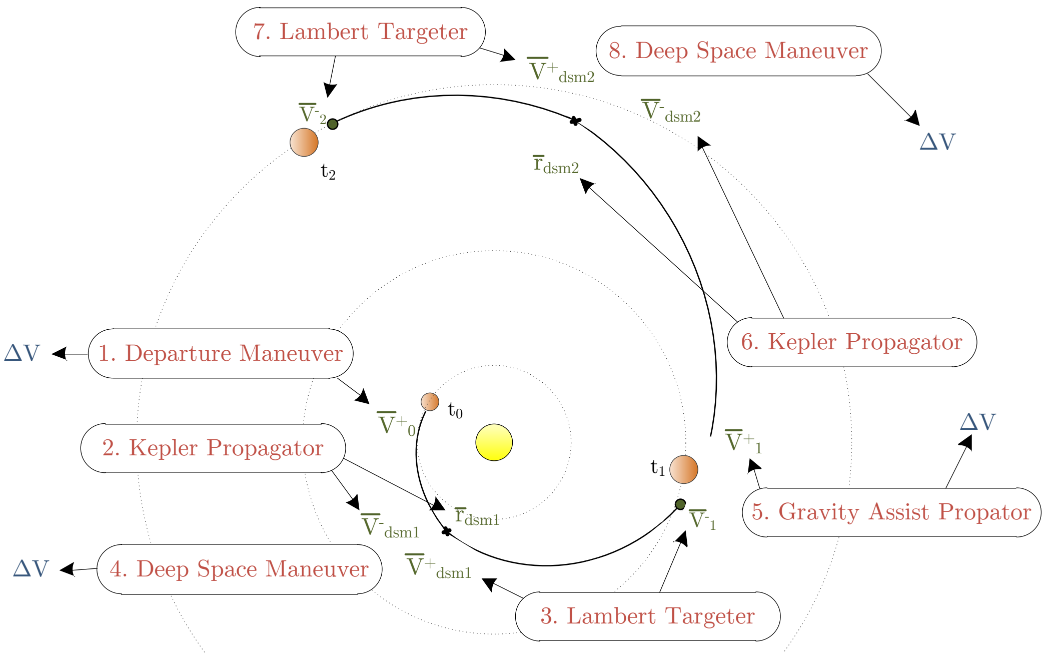


Figure 5.7: A schematic representation of the departure and swing-by legs in an MGA-1DSM-VF trajectory model.

5.3.2 Departure Leg

The first leg of the ideal Messenger-easy trajectory is used as a benchmark for testing the departure leg class. This leg is a transfer from Earth to Earth, as can be seen also in Figure 5.8.
After setting all the required input and model parameters with an accuracy of 15 digits, the simulation was run. A comparison of the associated ΔV-cost and the velocity when arriving at Earth between the GTOP software and the software programmed by the author is given in Table 5.6.

Table 5.6: Test of the MGA-1DSM-VF departure leg class against the first leg of the Messenger trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result [m/s]	MGA-1DSM-VF departure leg [m/s]	Absolute difference [m/s]	Relative difference [-]
ΔV-cost	2319.795886	2319.796269	$3.8\cdot10^{-4}$	$1.7\cdot10^{-7}$
V_x at Earth	17969.31663	17969.31722	$5.9\cdot10^{-4}$	$3.3\cdot10^{-8}$
V_y at Earth	-23543.69159	-23543.69165	$5.3\cdot10^{-5}$	$2.3\cdot10^{-9}$
V_z at Earth	6.383847	6.383847	$7.7\cdot10^{-8}$	$1.2\cdot10^{-8}$

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting.

5.3.3 Swing-by Leg

The fourth leg of the ideal Messenger-easy trajectory is used as a first benchmark to test the swing-by leg class. This leg was selected over the second and third leg because those involve multiple-revolution swing-bys. These cannot be modeled in the position formulation class using only single-revolution Lambert targeters. Taking the fourth leg as a benchmark for both reduced the number of tests that had to be setup. The fourth leg is a transfer from Venus to Mercury, as can be seen also in Figure 5.8.

A comparison of the associated ΔV-cost and the velocity at Mercury between the GTOP software and

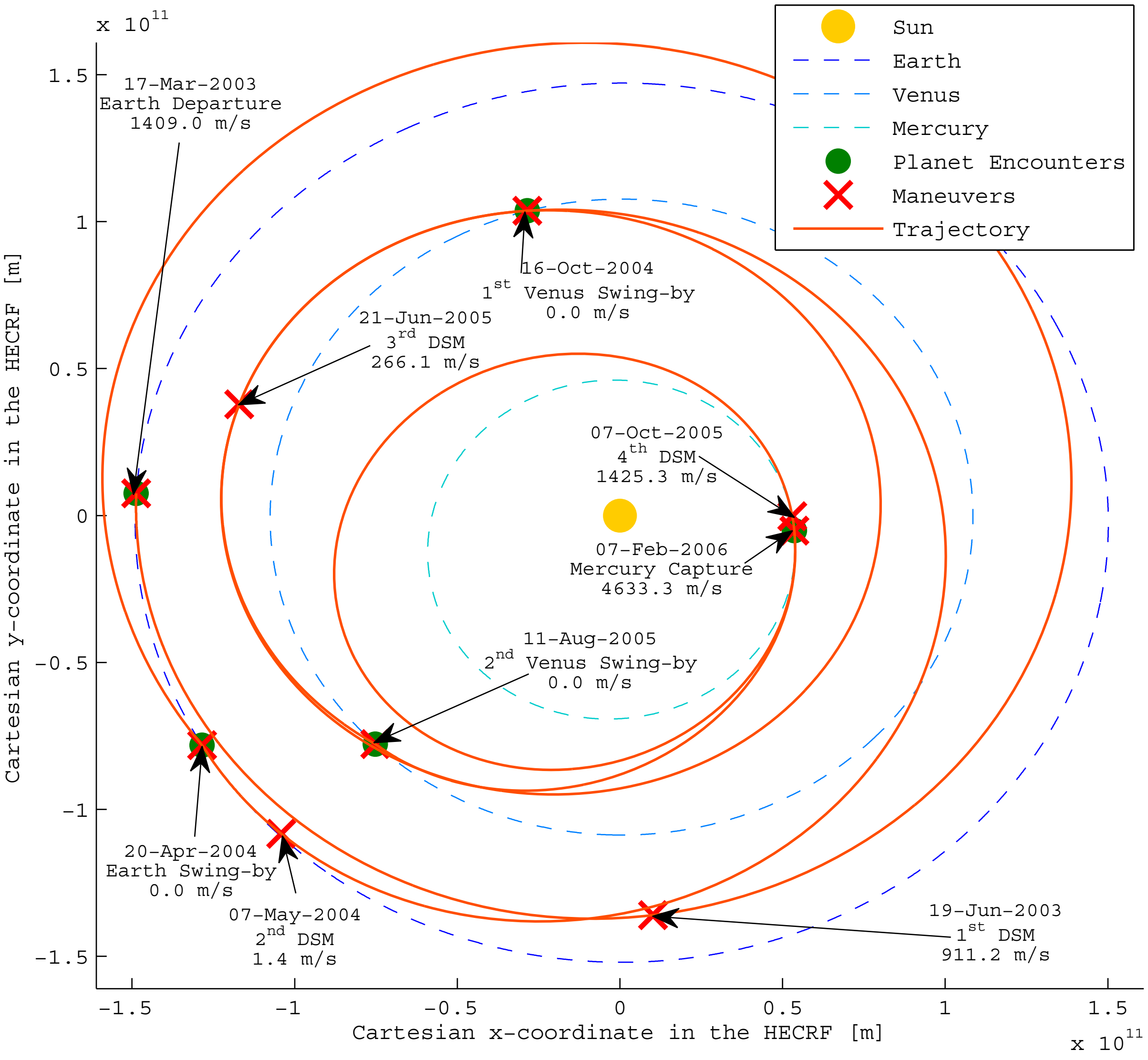


Figure 5.8: The best putative solution for the Messenger-easy problem as of 2008 [Vinko and Izzo, 2008], calculated and plotted using the trajectory class described in this thesis.

the software programmed by the author is given in Table 5.7.

Table 5.7: Test of the MGA-1DSM-VF swing-by leg class against the fourth leg of the Messenger trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result [m/s]	MGA-1DSM-VF swing-by leg [m/s]	Absolute difference [m/s]	Relative difference [-]
ΔV-cost	1415.440206	1415.440408	$2.0 \cdot 10^{-4}$	$1.4 \cdot 10^{-7}$
V_x at Mercury	-5080.636241	-5080.636207	$3.4 \cdot 10^{-5}$	$6.7 \cdot 10^{-9}$
V_y at Mercury	55179.12059	55179.12038	$2.1 \cdot 10^{-4}$	$3.8 \cdot 10^{-9}$
V_z at Mercury	3549.418322	3549.418307	$1.5 \cdot 10^{-5}$	$4.2 \cdot 10^{-9}$

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting.

Note that the swing-by in the test case is explicitly specified to be unpowered for the MGA-1DSM-VF test. Hence another test is required to test the powered swing-by functionality in the swing-by leg of the MGA-1DSM-VF trajectory model.
To obtain a test case for this, the second leg of the ideal Cassini-1 trajectory is used. The associated input parameters for the swing-by were obtained by reverse-engineering them based on the known swing-by characteristics. η_{dsm} is irrelevant because no actual DSM is supposed to be present.

A comparison of the associated ΔV-cost and the velocity at Mercury between the GTOP software and the software programmed by the author is given in Table 5.8.

Table 5.8: Test of the MGA-1DSM-VF swing-by leg class against the second leg of the Cassini-1 trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result [m/s]	MGA-1DSM-VF swing-by leg [m/s]	Absolute difference [m/s]	Relative difference [-]
ΔV-cost	1090.646229	1090.646729	$5.0 \cdot 10^{-4}$	$4.6 \cdot 10^{-7}$
V_x at Venus	37952.88446	37952.88479	$3.3 \cdot 10^{-4}$	$8.8 \cdot 10^{-9}$
V_y at Venus	-14096.96568	-14096.96533	$3.4 \cdot 10^{-4}$	$2.5 \cdot 10^{-8}$
V_z at Venus	-5753.512458	-5753.512489	$3.0 \cdot 10^{-5}$	$5.3 \cdot 10^{-9}$

Again it can be concluded that the differences are negligable. They probably originate from the iterative procedure involved in the Lambert targeting and the reverse-engineering to obtain the variables.

Also it is interesting to note that the error in the ΔV is higher than was calculated using the MGA trajectory model in Table 5.3. This is no surprise though. A small error will namely be present after the gravity-assist propagator, which will be propagated by the Kepler propagator. Hence a small ΔV will have to be applied to correct for this.

5.3.4 Full Model Verification

The full trajectory model is subjected to two different test cases. The first one is the Messenger-easy ideal trajectory, which can be seen in Figure 5.8. The second test case is the ideal Cassini-2 trajectory, which can be seen in Figure 5.9.

A comparison in ΔV-cost between the two models can be seen in Table 5.9.

Table 5.9: Comparison of the MGA-1DSM-VF trajectory model described in this thesis with two GTOP trajectories. Values refer to the total ΔV that needs to be given. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Problem	GTOP result [m/s]	Trajectory class [m/s]	Absolute difference [m/s]	Relative difference [-]
Messenger-easy	8630.832562	8630.833478	$9.2 \cdot 10^{-4}$	$1.1 \cdot 10^{-7}$
Cassini-2	8383.190143	8383.229637	$3.9 \cdot 10^{-2}$	$4.7 \cdot 10^{-6}$

It is concluded from Table 5.9 that the differences are again very small. They may originate from the iterative procedure involved in the Lambert targeting and the subsequent propagation of the error by the gravity assist and Kepler propagators. Note that this error propagation may also explain why Cassini2 is less accurate. It namely consists of more legs.

Finally the computational speed of this method is compared to that of the GTOP trajectory calculation method. Again both Messenger-easy and Cassini-2 are used for comparison. Both trajectories were calculated 1 million times and the speed of computation is given in Table 5.10. The test was run on one

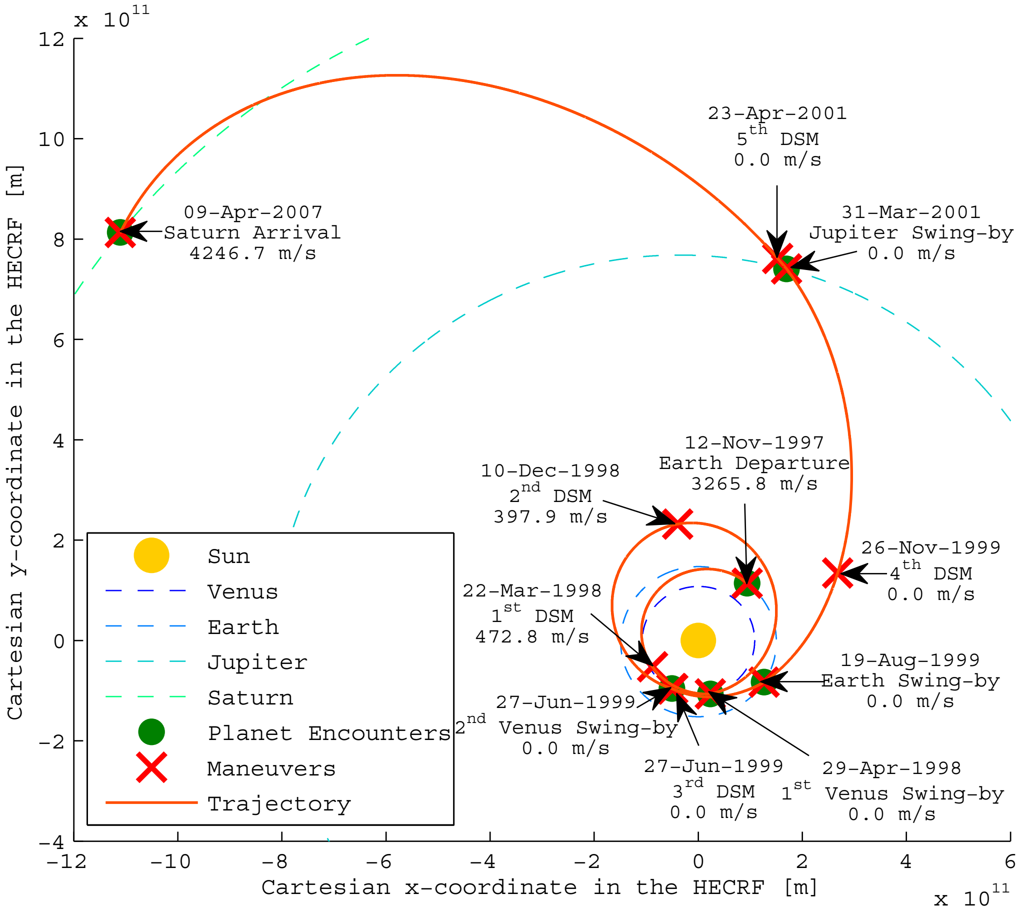


Figure 5.9: The best putative solution for the Cassini-2 problem as of 2009 [Vinko and Izzo, 2008] [Schlüter et al., 2011], calculated and plotted using the trajectory class described in this thesis.

core of an Intel(R) Core (TM) i7-2630QM @2.00 GHz processor.

Table 5.10: Comparison of the computational speed of GTOP and the trajectory class with MGA-1DSM-VF legs developed in this thesis.

	Computation time per trajectory [μs]	
Type of software	Messenger-easy	Cassini-2
GTOP	46.9	58.3
Trajectory class	53.5	63.1

From Table 5.10 it can be concluded that the trajectory class in this thesis is slightly slower than the GTOP method. This may partially be caused by the larger computational effort required for propagating a powered gravity assist when compared to an unpowered gravity assist. The computational speeds are very comparable still in spite of the more general and object-oriented approach used in this thesis.

5.4 MGA-1DSM-PF Trajectory Model

This section describes the Multiple Gravity Assist with 1 Deep Space Maneuver per leg using Position Formulation (MGA-1DSM-PF) trajectory model. Section 5.4.1 gives an overall description of this trajectory model. Subsequently Section 5.4.2 and Section 5.4.3 describe the departure and swing-by leg verification respectively. Finally Section 5.4.4 discusses the verification of the entire trajectory model.

5.4.1 Description

The position formulation is encountered less frequently in literature than the popular velocity formulation. This formulation was used by [Becarra et al., 2007] and is also described by Vasile and Ceriotti in [Conway, 2010]. A formulation similar to [Becarra et al., 2007] is used for this trajectory model. Only the definition for the dimensionless radius of the DSM is changed to allow for legs to have the same departure and arrival planet.

The following assumptions are made in the model:

- Patched-conics is applied and perturbations are neglected.
- The planetary sequence is fixed.
- A maximum of 1 DSM is applied in each leg.
- The Lambert-targeter only considers single-revolution counter-clockwise trajectories.

The MGA-1DSM-PF trajectory model employs the following variables to describe the trajectory:

- t_0: the starting date of the mission;
- T_i: the time of flight of all n interplanetary legs of the trajectory;
- η_i: the fraction of the time of flight at which a DSM takes place ($\eta \in [0, 1]$);
- r_i: the dimensionless distance of the DSM to the Sun, which is scaled to the distance of the departure planet to the Sun;
- θ_i: in-plane angle, defined as the angle between the position of the departure planet and the projection of the DSM location in the orbital plane of the departure planet;
- ϕ_i: out-of-plane angle, defined as the angle between the DSM location and the orbital plane of the departure planet.

The definition of the last three variables is represented schematically in Figure 5.10.

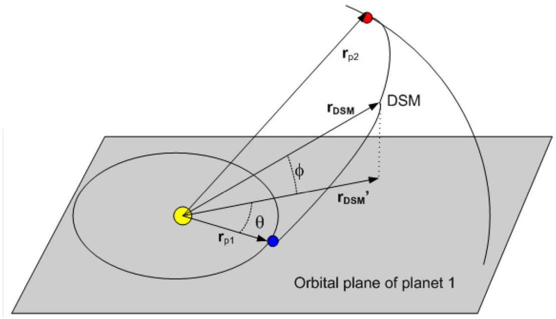


Figure 5.10: Schematic representation of the definition of various variables in the MGA-1DSM-PF trajectory model. [Becarra et al., 2007]

The visitation times of the planets are again calculated using Equation 5.1, similar to the other two trajectory models. These visitation times are subsequently used to obtain the planetary positions. The moment of application of a DSM is calculated using Equation 5.3, similar to the MGA-1DSM-VF trajectory model.

Next the DSM locations are calculated using the departure planet position, r_i, θ_i and ϕ_i. Now that the locations of the DSMs and the planetary positions are known, Lambert targeters can be used to find the required velocities before and after each planetary encounter and DSM. Using the initial heliocentric velocity in the departure leg, the ΔV that is required for the departure maneuver can be calculated. Similarly for the swing-by legs the required ΔV for the gravity assist can be determined using the gravity-assist calculator. Finally the ΔV that is required for the DSM can be calculated by taking the difference between the incoming and outgoing velocities at the DSM location.

In this way the following different ΔV contributions can be identified for a typical MGA-1DSM-PF trajectory:

- $|\Delta V_0|$: the magnitude of the departure maneuver to obtain the require excess velocity with respect to the departure planet.
- $|\Delta V_{dsm,i}|$: the magnitude of the maneuver at the DSM location to match the final velocity of the first part of the leg with the required initial velocity of the second part of the leg.
- $|\Delta V_i|$: the velocity change that is required by the gravity assists, if any.
- $|\Delta V_n|$: the velocity change to obtain the required orbit at the final plane from the final velocity of the last leg.

This calculation scheme is also represented schematically in Figure 5.11.

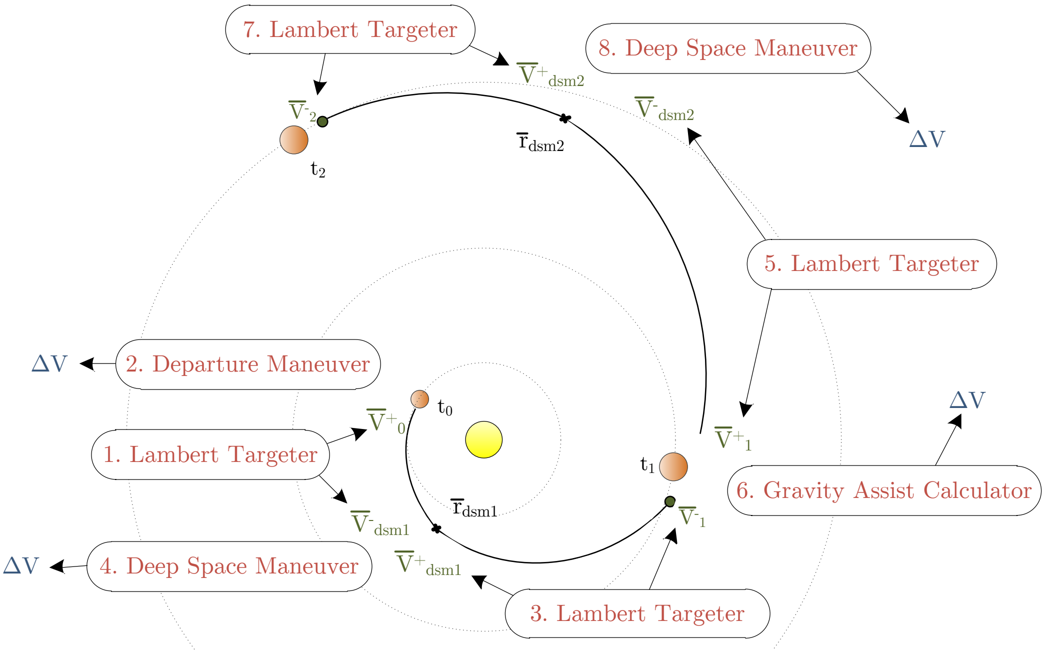


Figure 5.11: A schematic representation of the departure and swing-by legs in an MGA-1DSM-PF trajectory model.

In principle the same trajectories can be modelled by the MGA-1DSM-PF model as can be modelled by the MGA-1DSM-VF trajectory model. An example of a trajectory that is modelled by the MGA-1DSM-PF model can be seen in Figure 5.9. One difference between the models is that the MGA-1DSM-VF trajectory model allows for multiple-revolution transfers in the first part of the transfer, because a Kepler propagator is used instead of a Lambert targeter. If multiple-revolution Lambert targeting would be added to the PF model, this problem could be overcome. This is not part of this thesis work though.

An advantage of using the MGA-1DSM-PF trajectory model is that the final legs do not depend on the velocities in the previous legs. In the MGA-1DSM-VF trajectory model the last leg is dependent on all parameters from the previous legs, because the final velocities are propagated into the next leg. In the MGA-1DSM-PF trajectory model the last leg is only dependent on the visitation times and the parameters belonging to the last leg. This can be a big advantage, especially when using an incremental approach to solve the optimization problem. This was also the main reason Becarra et al. [2007] used the position formulation.
A disadvantage of the MGA-1DSM-PF trajectory model is that many specified positions at which the DSM takes place are practically infeasible in terms of ΔV. Still most of them can not be pruned easily without compromising the set of feasible DSM locations. The constraints on the gravity assist parameters that are used in the VF variant are more useful in that respect.

5.4.2 Departure Leg

The first leg of the ideal Messenger-easy trajectory is used as a benchmark for testing the departure leg class. This Earth-Earth transfer can be seen in Figure 5.8. All parameters were extracted from GTOP with an accuracy of 15 digits. Note that the parameters for the DSM location were reverse-engineered. A comparison of the associated ΔV-cost and the velocity when arriving at Earth between the GTOP software and the software programmed by the author is given in Table 5.11.

Table 5.11: Test of the MGA-1DSM-PF departure leg class against the first leg of the Messenger trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result	MGA-1DSM-PF	Absolute	Relative
	[m/s]	departure leg [m/s]	difference [m/s]	difference [-]
ΔV-cost	2319.795886	2319.796269	$3.8 \cdot 10^{-4}$	$1.7 \cdot 10^{-7}$
V_x at Earth	17969.31663	17969.31722	$5.9 \cdot 10^{-4}$	$3.3 \cdot 10^{-8}$
V_y at Earth	-23543.69159	-23543.69165	$5.3 \cdot 10^{-5}$	$2.3 \cdot 10^{-9}$
V_z at Earth	6.383847	6.383847	$7.7 \cdot 10^{-8}$	$1.2 \cdot 10^{-8}$

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting.

5.4.3 Swing-by Leg

The fourth leg of the ideal Messenger-easy trajectory is used as a first benchmark to test the swing-by leg class. This leg is a transfer from Venus to Mercury, as can also be seen in Figure 5.8.

After setting all the required input and model parameters with an accuracy of 15 digits, the simulation was run. Again the parameters regarding the DSM location were reverse-engineered. A comparison of the associated ΔV-cost and the velocity at Mercury between the GTOP software and the software programmed by the author is given in Table 5.12.

Table 5.12: Test of the MGA-1DSM-PF swing-by leg class against the fourth leg of the Messenger trajectory of GTOP. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Variable	GTOP result	MGA-1DSM-PF	Absolute	Relative
	[m/s]	swing-by leg [m/s]	difference [m/s]	difference [-]
ΔV-cost	1415.440206	1415.440408	$2.0 \cdot 10^{-4}$	$1.4 \cdot 10^{-7}$
V_x at Mercury	-5080.636241	-5080.636207	$3.4 \cdot 10^{-5}$	$6.7 \cdot 10^{-9}$
V_y at Mercury	55179.12059	55179.12038	$2.1 \cdot 10^{-4}$	$3.8 \cdot 10^{-9}$
V_z at Mercury	3549.418322	3549.418307	$1.5 \cdot 10^{-5}$	$4.2 \cdot 10^{-9}$

It can be seen that all differences are negligible. They probably originate from the iterative procedure involved in the Lambert targeting.

5.4.4 Full Model Verification

To verify the MGA-1DSM-PF trajectory model as a whole, the entire Cassini-2 trajectory was reverse-engineered to obtain the correct DSM location parameters. The resulting ΔV-cost was subsequently compared to the GTOP model and the results can be seen in Table 5.13.
It is concluded from Table 5.9 that the differences are again negligible. They may originate from the iterative procedure involved in the Lambert targeting.

Finally the computational speed of this model is compared to that of the GTOP trajectory calculation method and to that of the MGA-1DSM-VF trajectory model. The Cassini-2 trajectory was used for

Table 5.13: Comparison of the full Cassini-2 trajectory of GTOP with the MGA-1DSM-PF model described in this thesis. Values refer to the total ΔV that needs to be given. Note that it concerns a reconstruction of the GTOP trajectory and no optimization is performed.

Problem	GTOP result [m/s]	Trajectory class [m/s]	Absolute difference [m/s]	Relative difference [-]
Cassini-2	8383.190143	8383.190931	$7.9 \cdot 10^{-4}$	$9.4 \cdot 10^{-8}$

comparison. The trajectories were calculated 1 million times and the speed of computation is given in Table 5.14. The test was run on one core of an Intel(R) Core (TM) i7-2630QM @2.00 GHz processor.

Table 5.14: Comparison of the computational speed of the MGA-1DSM-PF model against GTOP and the MGA-1DSM-VF model for the best putative Cassini-2 trajectory [Izzo and Vinko, 2012].

Type of software	Computation time per trajectory [μs]
GTOP	58.3
MGA-1DSM-VF model	63.1
MGA-1DSM-PF model	85.7

From Table 5.14 it can be concluded that the position formulation is significantly slower than the velocity formulation. The most probable cause for that is the fact that a Lambert targeter is used in the first part of each leg, whereas the velocity formulation models use a Kepler propagator.

5.5 Capture Leg

This leg class is mainly included to allow for modelling of sample-return missions or mission that may visit several targets. Both sample return missions to Mars and missions visiting multiple asteroids are frequently analyzed by mission planners. [Conway, 2010]
This leg type is also present at the end of a ‘normal trajectory’, as it will basically calculate the capture maneuver required. Furthermore the user may specify a certain stay time at the target. During this time the position of the spacecraft will be fixed to that of the body in question. Subsequently a departure maneuver may be used to continue the heliocentric part of the trajectory.
An example of a trajectory designed and optimized using this class can be seen in Figure 5.12. This is the optimized trajectory of an Earth-Mars sample return mission, that was set up by the author for the course on Tudat.

The capture leg class was subjected to the same test beds that were used to verify the escape/capture function described in Section 4.6. Furthermore the accuracy has been tested indirectly, because a capture maneuver was present in all trajectories that were verified in Sections 5.2.4, 5.3.4 and 5.4.4.

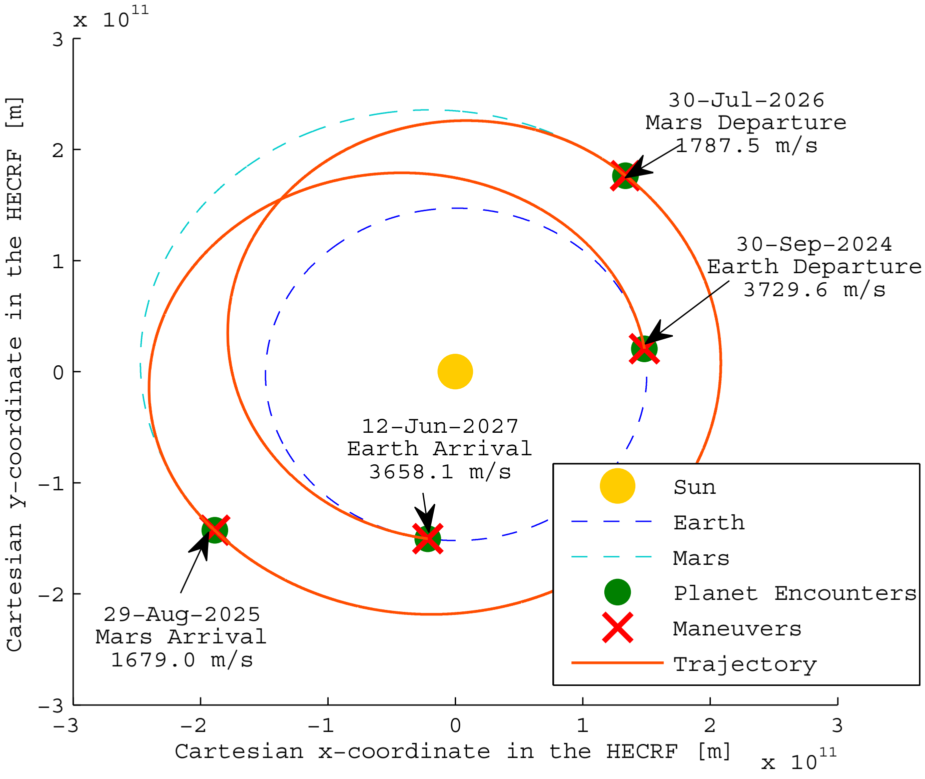


Figure 5.12: Example application of the capture leg class. This trajectory represents the optimized trajectory for the Earth-Mars sample return mission that is part of the Tudat course.

Chapter 6

Existing Optimization Software

Chapters 2 and 3 discussed that at least DE, GA, and PSO need to be investigated thoroughly in this thesis. Furthermore it is deemed useful if also local optimization methods could be employed.
No optimization tools were available within Tudat at the start of this MSc thesis. Hence an investigation was started to find the required optimization software. Various optimization toolboxes are available online and many useful algorithms can be gathered online as well. This chapter deals with selecting the software that will be used in this thesis.

To this end Section 6.1 deals with the requirements on this external software. Subsequently Section 6.2 gives an overview of the available software and deals with the selection between the software. Finally Section 6.3 describes the software of choice and the methods available therein.

6.1 Requirements of Software

The optimization toolbox should meet certain requirements, which are discussed below:

The software toolbox should provide methods that can solve the trajectory problems. This means it should provide algorithms to solve problems that are:

- multi-dimensional
- continuous
- box-constrained
- single-objective

The software toolbox should be easy to use and apply to the trajectory problems that it will need to solve. To this end, especially the problem definition should be clear and unambiguous.

It should be a fast optimization toolbox, that is preferably written in C++. Alternatively it can be written in a different language, but in that case a proper wrapper for the corresponding methods/toolbox should be available. Clearly this is not preferred due to the corresponding overhead.

Also it should have a very general set-up together with a very well thought-out architecture. It should allow to test many different algorithms, but also allow to implement new methods in the same architecture.

Also the toolbox should give a good insight into how the different algorithms are implemented exactly. Typically many variants are available. The choice for the implementation can have big implications on the functionality and performance of these methods. Hence it is important that the source code can be accessed and is written in such a way that the implementation can be checked and adjusted if desired.

It should provide a wide variety of algorithms. Following the thesis goals specified in Chapter 3 at least the following methods should be included:

- Differential Evolution (DE)

- Particle Swarm Optimization (PSO)
- Genetic Algorithm (GA)

Preferably also local optimization techniques should be available, such as Sequential Quadratic Programming (SQP) and Nelder-Mead (NM)

6.2 Overview of Available Software

Various available optimization software toolboxes were examined, that are available online. The most relevant findings are summarized in Table 6.1. Other findings of the author have been summarized on the Tudat wiki page, which can be accessed at `http://tudat.tudelft.nl/projects/tudat/wiki/Optimization`. These include toolboxes that require paid licenses not available at the TU Delft, or toolboxes that focus on other types of optimization, such as multi-objective or combinatorial optimization.

Table 6.1: Overview of available optimization toolboxes.

Toolbox	Local Techniques	Global techniques	Additional information
GSL	NM, various others	SA, MC	A scientific library that provides many other techniques as well. [GNU, 2012]
IPOPT	Single technique	None	Good and efficient local optimization technique. [Wächter and Biegler, 2006]
METSlib	Single method	SA, few others	Non-mature toolbox, few and simple techniques only. [Maischberger, 2011]
NLOPT	NM, SQP, many others	Various simple	Good toolbox providing many useful local optimization techniques, but less useful global techniques. [Johnson, 2012]
OptSolve++	Various	None	Various interesting local techniques. [Tech-X, 2012]
OPT++	Various	None	Provides various older local optimization methods. [Meza, 1994]
PaGMO	NM, SQP, many others	DE, PSO, ASA, GA, many others	Good well-structured toolbox. Well thought-out architecture. Includes many techniques. [Biscani et al., 2010]
SNOPT	Single technique	None	Very efficient SQP algorithm. [Murray, 1997]

Of the investigated software toolboxes, only PaGMO meets all requirements. Although many optimization toolboxes exist that are focused on local optimization, very few of those also provide good support for global optimization algorithms. If this support is provided, it is typically very limited.
PaGMO on the other hand has a very well thought-out architecture. It is very structured and it includes many algorithms. The code is set up such that it is also very easy to define a new algorithm or a different algoritm.
PaGMO also includes wrappers for various other toolboxes. IPOPT and SNOPT are included and also various methods from the GSL and NLOPT toolbox are included.

For these reasons, PaGMO is chosen as an optimization toolbox.

6.3 Description of PaGMO Optimization Toolbox

This section describes the Parallel Global Multi-objective Optimizer (PaGMO) in more detail. Section 6.3.1 gives a brief history of the toolbox. Section 6.3.2 gives a brief overview of the functionality of PaGMO. Subsequently Section 6.3.3 gives an overview of the algorithms that are available in PaGMO. Finally Section 6.3.4 discusses implementation issues with PaGMO.

6.3.1 History

Soon after the creation of the Advanced Concepts Team (ACT) within the European Space Agency (ESA), projects related to interplanetary trajectory optimization were investigated. This started with interesting investigations such as [Di Lizia et al., 2004] and [Myatt et al., 2004]. Many publications followed in which also the addition of DSMs to the trajectory was investigated.
Soon afterwards the need to have a good optimization toolbox was identified within ESA. A toolbox was developed by Izzo and Vinko, called Distributed Global Multi-objective Optimizer (DiGMO). This toolbox included some basic optimization algorithms, as well as the option to distribute the workload over multiple processors. This resulted in various space-related publications, such as [Vinko et al., 2007a] and [Vinko and Izzo, 2008] in which interplanetary trajectories were optimized. It was also used for optimization of various other space-related subjects within ESA. [ESA-ACT, 2012]
An initial version of Parallel Global Multi-Objective Optimizer (PaGMO) was presented by [Izzo et al., 2009]. This initial version was subsequently restructured to allow for a generic framework to allow for massively parallel global optimization. This version of PaGMO is presented in [Biscani et al., 2010]. The code was also published as open source code. [ESA-ACT, 2012]

PaGMO has subsequently been improved upon by various people. It was accepted for the Google Summer of Code Project 2010. After being rejected for the Google Summer of Code 2011, an ESA Summer of Code was held in which people worked on improving PaGMO among other projects. [ESA-ACT, 2012]
The initial programming language for PaGMO was C++. Also a Python version was subsequently built, called PyGMO. [Izzo, 2012]

6.3.2 Overview of Functionality

PaGMO provides many functionalities related to optimization. It is a toolbox which allows the user to define a problem of his own and then use any optimization technique to help find the minimum of that problem. Many types of problems and optimization techniques are available. PaGMO supports: [ESA-ACT, 2012]

- both single- and multi-objective optimization,
- unconstrained, box-constrained, equality constrained and inequality constrained problems,
- combinatorial and continuous variables, and problems consisting of both,
- many global and local optimization algorithms,
- branching into different topologies,
- multi-core and parallel computing.

Also the PaGMO code has been written in a very structured way. It gives a good insight into how the different algorithms are implemented exactly. It provides a good architecture in which new algorithms can be written easily. A drawback is that no framework is included for sampling methods yet.

Using PaGMO

To use PaGMO, the user must first create a problem definition. To do this, the dimension of the decision variable vector must be specified, along with the type of variables that are included (continuous versus combinatorial). Also the corresponding bounds and constraints need to be specified by the user. The computation scheme and the corresponding objective functions must be implemented and constraint functions have to be specified, if any.

Subsequently the user initializes a population. This is done automatically by PaGMO if the user specifies the corresponding problem and the number of individuals.

Next the user specifies the optimization algorithm that needs to evolve the population. The tuning parameters and the number of generations that the optimization technique should optimize are given here. Based on this an algorithm object is generated, which is linked to a certain population. Because of the general definition of the population it is also possible to optimize the population with one algorithm first and then pass the population to another algorithm.

Finally the computation can be parallelized and branched into different islands and in different topologies. Different algorithms can work on different islands (processor cores). Different migration settings can be used.

An overview is given in Figure 6.1.

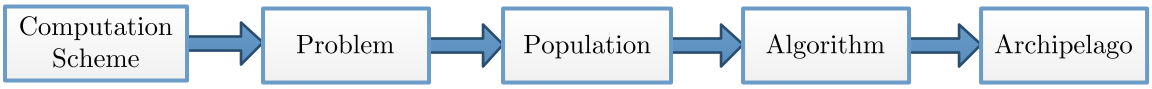


Figure 6.1: A schematic overview of the PaGMO structure.

6.3.3 Overview of Included Algorithms

The information in this section is based on the information and code available on [ESA-ACT, 2012] as of March 2012. It is noted that a newer version was released in the meantime, but the author was so far in his thesis that he sticked to the old version. More information on the techniques and algorithms can be found in [ESA-ACT, 2012].

PaGMO includes a large number of optimization techniques. Many global optimization methods are written by PaGMO programmers themselves. No local optimization techniques are written in PaGMO. Instead PaGMO includes wrappers for various algorithms available in four different optimization toolboxes. These toolboxes are open-source projects, except for SNOPT. SNOPT is a commercial toolbox for which TU Delft has a license.
Table 6.2 gives an overview of the global optimization algorithms included in PaGMO. Table 6.3 gives an overview of the local optimization algorithms included in PaGMO.

Table 6.2: Overview of the global optimization techniques in PaGMO [ESA-ACT, 2012]. Note that CMAES stands for Covariance Matrix Adaptation-Evolutionary Strategy.

Optimization technique	Discrete Problems	Continuous problems	Single-objective	Multi-objective	Notes
Adaptive Simulated Annealing	No	Yes	Yes	No	
Artificial Bee Colony	No	Yes	Yes	No	
Ant Colony Optimization	Yes	No	Yes	No	
CMAES	No	Yes	Yes	No	
Compass Search	No	Yes	Yes	No	
Differential Evolution	No	Yes	Yes	No	
Firefly Algorithm	No	Yes	Yes	No	
Genetic Algorithm	Yes	Yes	Yes	No	
Improved Harmony Search	Yes	Yes	No	Yes	
Monte Carlo	Yes	Yes	Yes	Yes	
Multi-Start	No	Yes	Yes	Yes	needs local optimizer
Monotonic Basin Hopping	No	Yes	Yes	Yes	needs local optimizer
Nondominated Sorting GA	Yes	Yes	No	Yes	
Particle Swarm Optimization	No	Yes	Yes	No	

6.3.4 Implementing PaGMO

Importing PaGMO turned out to be quite a difficult task under Windows. Although it is very straightforward on Linux/Apple, library conflicts made it a much more difficult task under Windows. The main

Table 6.3: Overview of the local optimization techniques in PaGMO [ESA-ACT, 2012]. The asterisk (*) means that the algorithm was not available yet in the version of the author.

Toolbox	Algorithm	Notes
GSL	Broyden-Fletcher-Goldfarb-Shanno	gradient method, 2 variants
GSL	Fletcher-Reeves	gradient method
GSL	Nelder-Mead	derivative free method, 3 variants
GSL	Polak-Ribiere	gradient method
IPOPT	IPOPT	interior point gradient method
NLOPT	Augmented Lagrangian *	incorporating constraints for other methods
NLOPT	BOBYQA	derivative free method
NLOPT	COBYLA	derivative free method
NLOPT	Method of Moving Asymptotes *	gradient method
NLOPT	Sbplx	derivative free method, variant on Nelder-Mead
NLOPT	SLSQP *	gradient method
SNOPT	SNOPT	gradient method

problem here is that Tudat uses static Boost libraries and PaGMO requires dynamic libraries, which causes a conflict. Somehow Windows does not load the dynamic libraries for PaGMO, and static libraries for the rest of the project. A way was found to work around this issue by manually editing the build files after the project was created. The method of doing so has been added to the Tudat site [Tudat, 2012].

Furthermore it proved difficult to load the external libraries correctly for the local optimization methods. Loading them in PaGMO was comparatively easy, but subsequently making sure that they were loaded correctly if PaGMO is loaded as an external library proved more difficult. This may also be caused by the lack of experience of the author in doing so. Eventually the GSL and NLOPT libraries were successfully linked. The time spent on doing so was very significant. The author expected that another very significant time investment would be required to also link the IPOPT and SNOPT libraries. Hence these were not linked anymore. Tips and tricks on linking the GSL and NLOPT libraries have been added to the Tudat site [Tudat, 2012].

Finally it was noted that at least under the version used by the author and on the problems created by the author, some algorithms crashed often. All three variants of the NM algorithm in GSL crashed often, as well as one variant of the GSL BFGS algorithm. The other GSL algorithms crashed sometimes as well, although less frequently. Finally one neighborhood type in PSO was also found to be unstable. It is unclear what causes this instability and it may possibly be fixed already. The vast majority of the native PaGMO algorithms worked perfectly.

Part III

Algorithm Tuning

Chapter 7

Algorithm Tuning Procedure

This chapter explains the tuning procedure that will be used in this thesis and the performance criteria used therein. To this end Section 7.1 gives a short introduction to performance comparisons related to interplanetary trajectory optimization. Section 7.2 discusses the importance of proper tuning. Subsequently Section 7.3 discusses the performance criteria that will be used throughout this thesis. Finally Section 7.4 will test the accuracy of the performance criteria used in this thesis.

7.1 Introduction to Performance Comparison

Due to various reasons it is typically very difficult to compare different optimization techniques against each other. Different publications may employ:

- different test problems,
- different testing schemes,
- different performance criteria,
- different or unknown tuning parameters for the techniques employed,
- different trajectory models,
- different ephemeris data,
- different processors,

and various other aspects may also complicate the comparison between different techniques.

Comparing between different optimization techniques is not straightforward because of these issues. To correctly compare different methods, they have to be applied on the same problem, subjected to the same and rigorous testing scheme, the same trajectory models employing the same ephemeris data etc. Even then the correctness of the comparison may be doubted.
Many of these problems are overcome by the development of the standardized ESA trajectory problems, GTOP. Using GTOP, comparisons can be made between different optimization algorithms that use the same trajectory models and same ephemeris data. Only problems that have a necessity to use a different trajectory model are still relatively difficult to compare with other techniques. [Vinko and Izzo, 2008]

Still many factors determine if different results may be compared fairly, even if GTOP instances are used. One important aspect is how well the optimization algorithms have been tuned. Another important aspect is the robustness of the performance criteria that are used. Section 7.2 discusses the importance of proper tuning and Section 7.3 discusses various performance criteria that are used throughout literature.

7.2 Effects of Tuning Parameters

It is long known that the tuning parameters of various optimization algorithms have a very strong effect on their performance. Olds et al. [2007] did a very interesting study on the sensitivity of DE to its tuning parameters when applied to interplanetary trajectory optimization. They tested a large number of different settings of a basic version of DE and applied them to four different interplanetary trajectory problems. For each setting 1000 tests were done. The time to calculate the solution with a 95% reliability was calculated and used as a measure for performance. Some of the results can be seen in Table 7.1.

Table 7.1: Run time [min] to reliably optimize the Cassini mission with 1 DSM as a function of various tuning parameters of DE. [Olds et al., 2007]

	$n_p = 20$			$n_p = 36$		
F	$CR = 0.4$	$CR = 0.6$	$CR = 0.8$	$CR = 0.4$	$CR = 0.6$	$CR = 0.8$
0.2	23.2	no sol.	no sol.	14.3	354.1	49.9
0.4	3.7	3.3	no sol.	13.1	5.4	1.8
0.6	19.1	2.8	1.4	195.5	9.1	1.8
0.8	160.2	3.0	1.4	no sol.	206.8	3.0
1.0	267.3	109.6	no sol.	no sol.	631.5	24.2
rand[-1,1]	4.9	2.4	1.4	9.3	2.1	1.3

It can be seen in Table 7.1 that the performance is highly dependent on the tuning parameters of DE: F, n_p and CR. Note that the definition of these tuning parameters will be discussed in Chapter 9 and that this section just points out the influence of these parameters on the performance. For some settings it is possible to find the solution to the problem in slightly over 1 minute, whereas for other settings this takes multiple hundred minutes, or no reliable solution can be found at all. Also the performance is highly sensitive to the tuning parameters. Changing the population size can mean that the optimum that was found in 1.8 minutes cannot reliably be found anymore. Changing the crossover rate by 0.2 can mean a solution found in 3.3 minutes cannot be reliably found anymore either. Finally also changing the scale factor by 0.2 can mean that a solution that was found in 1.4 minutes cannot be reliably found anymore.

The authors managed to find a relatively good setting for the problems considered. Still this setting is a trade-off, and for various problems a different setting can be found that outperforms this setting by a factor 2. Given that the problems are relatively similar, this result is remarkable. It is noted also that the problems considered in [Olds et al., 2007] are much easier than the problems considered in this thesis. The problems namely included one DSM in the entire trajectory. It is possible that the results differ even more on the difficult problems considered in this thesis.

The importance of tuning is also stressed by Price et al. [2005]. They investigated the ideal tuning parameter regime for a large number of different problems. An example of three ideal regimes can be seen in Figure 7.1. It can be seen that the location of these ideal regimes may differ strongly. Also it is noted that the ideal tuning regimes also depend on the strategy used by DE.

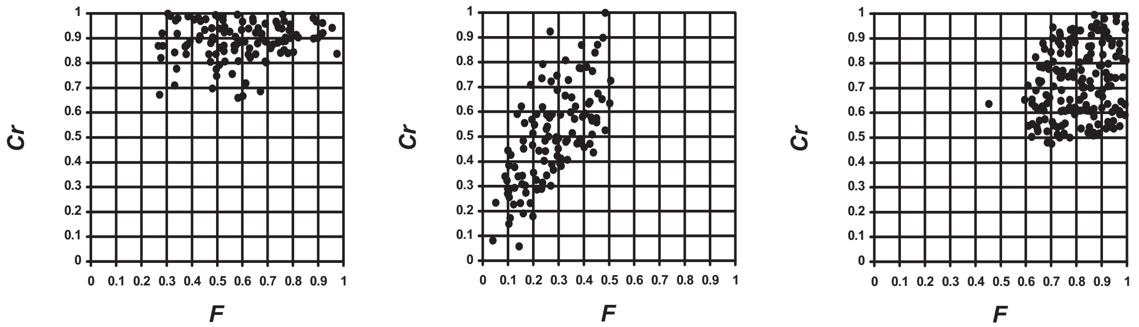


Figure 7.1: Ideal tuning regimes of a certain DE strategy for the 10D Rotated Whitley, Griewank and Rosenbrock problems. The density of the dots is a measure for the performance of a certain regime. Adapted from [Price et al., 2005].

This strong dependence on the tuning parameters is something that is easily overlooked by authors that benchmark the performance of their newly developed algorithm. These algorithms have typically been tuned perfectly and are subsequently compared to other algorithms that have not been tuned at all, or only tuned slightly. This makes a fair comparison impossible. The fairest comparisons in this respect are found in [Vasile et al., 2008], [Vasile et al., 2009], [Vasile et al., 2010] and [Vasile et al., 2011]. Still the tuning process used therein is not considered fully adequate.

7.3 Performance Criteria

When comparing different algorithms and different settings of tuning parameters, it is critical to define a proper performance criterion. This performance criterion should be a good measure of the performance of the algorithm on the optimization problems. Furthermore the performance criterion should be accurate and repeatable.

In this respect it has to be noted that all techniques that have proven to be able to solve complex MGA-1DSM problems are partly stochastic. It will hence not be possible to obtain a 100% chance on finding the minimum. Instead, one wants to maximize the chance of finding the global minimum using the selected method.

Given the stochastic nature, the performance criterion should consist of two separate parts: a measure of performance and the computational cost required. Any sampling scheme will namely be able to find the global optimum with an infinite amount of computational resources.

First Section 7.3.1 will discuss the computational cost. Subsequently Section 7.3.2 discusses the measure for performance that will be used. Finally Section 7.3.3 discusses the implementation of the performance measure.

7.3.1 Computational Cost

Two different methods are typically used to define the computational cost:

- computational time,
- number of function evaluations.

Computational time is the most representative indicator for the performance of the optimization technique. This measure also takes into account the computational effort of the optimization technique itself. On the other hand computational time is much less comparable across different publications and across different machines. Also it is dependent on the CPU usage of other processes on the computer.

Most nature-inspired metaheuristics do not require a large amount of computational resources when compared to the computations required by the trajectory models [Myatt et al., 2004]. Hence many publications have used the number of function evaluations as a measure for computational cost. This namely allows proper comparisons between different studies and is independent of the computer used to run the simulations.

This thesis will also use the number of evaluations as a main indicator for the computational cost. The simulations will be run on more than one computer and also on computers that will be used for other purposes in the meantime. Also the number of evaluations allows the fairest comparison with existing publications and future research. The Cassini2 DSM VF problem, which is described in Section 8.2, has been optimized for 10^5 evaluations using various optimization algorithms to verify the low dependence on the algorithms. The resulting computation time for a selection of algorithms is given in Table 7.2.

Table 7.2: Average computation time required for 1 evaluation for different algorithms during the optimization of the Cassini2 DSM VF problem. Based on the average of 10^5 evaluations.

Algorithm	DE, $n_{pop} = 40$	DE $n_{pop} = 400$	PSO	GA	SBPLX
Computation time [μs]	115	113	115	121	119

Table 7.2 shows that the computation times are very similar. Hence the number of evaluations can be used for comparison between algorithms instead of the computational time.

7.3.2 Measure for Performance

Various methods are typically used to define performance:

- best value obtained,
- best, mean and variance of a large number of samples,
- average success-rate of a large number of samples,
- computational cost for achieving a certain probability on success.

As discussed earlier, the repeatability of the result is an important aspect for measuring the performance. Obviously simply giving the best value obtained gives little information on the repeatability of a result. Furthermore it does not allow for differentiating between algorithms that both obtained the best value.

Also the best, mean and variance does not provide a robust way of measuring the performance. An algorithm that sometimes does not find solutions at all, but converges to the best putative solution comparatively frequently, is more useful in practice than an algorithm that consequently finds a suboptimal solution. A best, mean and variance representation does hence not measure the aspects that define the usefulness of an algorithm in practice.

A real-life example can be seen in Figures 7.2 and 7.3. Both are histograms of the final solution obtained after 1000 attempts at optimizing the Cassini2 MGA problem, which will be described in Section 8.3.5. The algorithm in Figure 7.2 often does not find a good solution at all, but does find the best solution at a rate of about 6%. Because the algorithm gets stuck far from the best solution, the average fitness value is comparative high. On the other hand, the algorithm in Figure 7.3 often finds a good solution, but only finds the actual solution once. Because of that, its mean value would be much better than that of the other algorithm. In practice the first algorithm is more useful because it has a larger chance on finding the actual optimum. This shows how a best-mean-variance representation may be misleading.

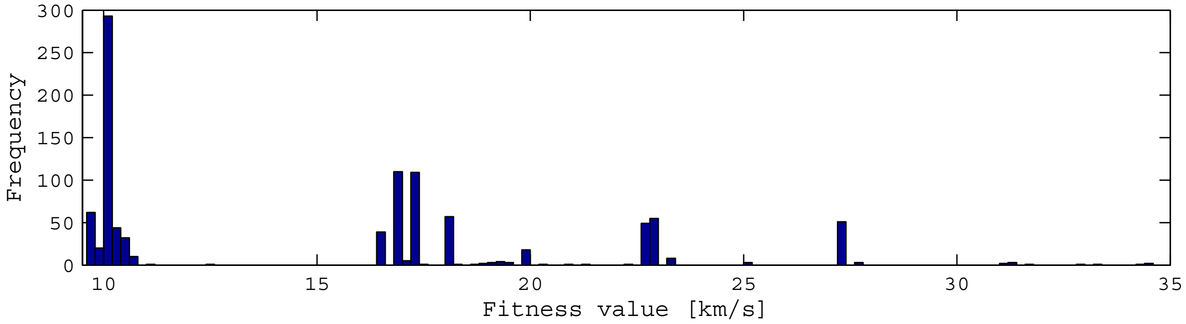


Figure 7.2: Histogram of 1000 optimization runs on the Cassini2 MGA problem using DE4 with population size of 14.

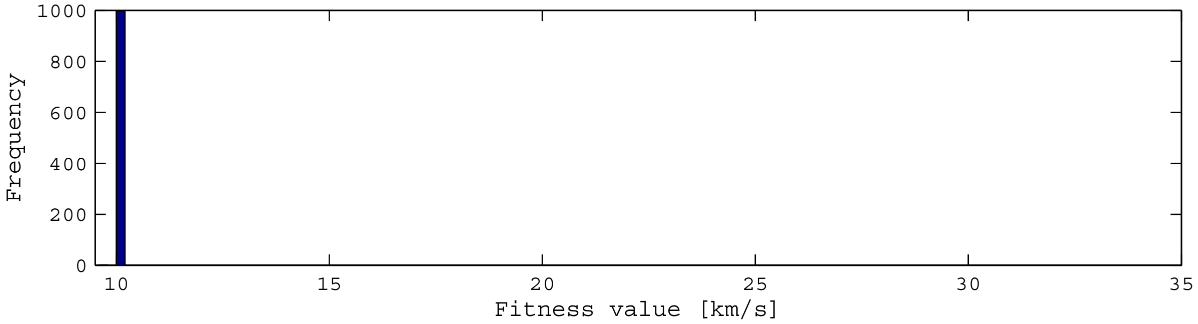


Figure 7.3: Histogram of 1000 optimization runs on the Cassini2 MGA problem using DE2 with population size of 100. It is noted that only 1 run managed to find the optimum of this run.

Another method is to define a measure for success for a certain optimization run. Typically this criterion may be a function value within a certain margin of the global minimum. A success rate can be determined by performing a large number of optimization runs. This success rate is a very effective representation of the algorithm's performance and repeatability. It addresses the issues raised in best, mean and variance representation. One is most interested in the reliability of the algorithm to provide a good solution and the amount of iterations required for that. One issue is that this method does not yet take into account the option to restart algorithms. An algorithm that converges in 1000 evaluations with 40% success rate is more useful than an algorithm than needs 100000 evaluations to obtain 80% success rate, as will be further explained in the next paragraph.

A fourth method is to calculate the best efficiency of a certain algorithm to find the best putative solution, depending on the number of function evaluations. The accompanying success rate and number of evaluations can subsequently be used to calculate the number of evaluations to obtain a certain chance of finding the best putative solution. Olds et al. [2007] required a 95% chance of finding the best putative solution.
This thesis will use the same requirement and the performance measure will be the number of evaluations to obtain at least 95% chance on finding the solution, $n_{95\%}$. In case a higher success rate is obtained during a trial, $n_{95\%}$ is set to the number of evaluations of that trial, n. In case the success rate was less than 95%, Equation 7.1 can be used to calculate $n_{95\%}$ [Olds et al., 2007].

$$n_{95\%} = n \frac{\log(1 - 0.95)}{\log(1 - p_{success})} \tag{7.1}$$

7.3.3 Implementation of the Performance Measure

An overview of the practical implementation of the performance measure when comparing l algorithms is given in Algorithm 7.1. It is also discussed in words in the next paragraph.

Algorithm 7.1: computes the performances of algorithms A_1 to A_l, as used in the tuning process in this thesis.

```
for $A_i = A_1{:}A_l$
    for $j = 1 : m_{runs}$
        for $k = 0 : n_{readout} : n_{max}$
            Optimize problem using $A_i$ for $n_{readout}$ evaluations
            Store best fitness to fitness archive: $F_{j,k} = f_{best}$
        end for
    end for
    for $k = 0 : n_{readout} : n_{max}$
        Calculate success rate $p_{success,k}$ from $F_{j,k}$
        Calculate $n_{95\%,k}$ from $p_{success,k}$ using Equation 7.1
    end for
    Store best $n_{95\%,k}$ to performance archive $P_i$
end for
```

For each algorithm A_i whose performance is to be tested, m_{runs} optimization runs are performed. These optimization runs will last for n_{max} evaluations, with readouts every $n_{readout}$ evaluations. During the readouts the best fitness of the population is stored into an archive. After the optimization runs have been performed, the success rates at all readouts is calculated. The $n_{95\%}$ is subsequently calculated at each readout and the best $n_{95\%}$ is stored, representing the performance of the algorithm.
It is noted here that in some figures in this thesis, n_{eval} or simply n is used to refer to $n_{95\%}$. This will be indicated in the caption.

Similar to other methods, also this algorithm has some flaws with respect to the ideal situation. Four of those issues are discussed below.

First of all a proper definition of the success margin definition determines the applicability of this method. In this thesis a margin of 50 m/s will be used to distinguish between optimal and suboptimal values. Experimentally it was determined that problems often did not have suboptima within this threshold. It is further argued that, in practice, a suboptimum within 50 m/s of the actual optimum is just as useful as the best optimum. Differences caused by the assumptions used in the trajectory models namely amount to similar deviations [Conway, 2010]. A success margin varying from 50 to 150 m/s was used by [Vasile et al., 2008], and their subsequent publications. Olds et al. [2007] used success margins varying from 33 to 300 m/s, depending on the problem.

A second issue is that the solution needs to be known. It is noted that in practice, one can never be sure that a certain complex interplanetary trajectory is optimal when making use of metaheuristics. Instead one wants to be as sure as possible that a certain best putative solution is also the actual solution. There are various ways to cope with that during the tuning phase. One method is to use problems that have been optimized by others frequently, because of which the probability is high that the best putative solution is also the actual solution. Another method is to optimize a certain problem very often and subsequently checking that the best putative solution is indeed found often by different optimization techniques.

This thesis primarily uses problems that have been optimized by many others. Problems that were newly defined for this thesis were subjected to a large number of preliminary runs. Also the final results of all actual runs were checked to ensure that no improvement was found.

A third issue is that $n_{95\%}$ can only be determind accurately after a large number of optimization runs have been performed. This means that the most complex type of problems, cannot be subjected to this success representation. For problems of lower complexity, the number of optimization runs is a trade-off between the desired accuracy and the computational cost of doing so. If a larger amount of computational effort is dedicated to testing a single algorithm, less different algorithms may be compared. On the other hand if not enough runs are performed, the reliability will decrease and trade-offs between different settings can be made with less accuracy. Olds et al. [2007] used 1000 runs per setting. Vasile et al. [2008] and subsequent publications used 200 runs per setting instead.

This thesis will adopt 200 runs per setting as well. Only for 2 problems with an exceptional low success rate, a larger number of runs will be used. Also this thesis will check for success once every 200 evaluations, for practical reasons. For the very complex problems, this check is only performed once every 2000 evaluations. The accuracy of the results that are obtained in this way will be investigated more thoroughly in Section 7.4.

Finally an important note is that Algorithm 7.1 does not take stopping criteria into account. Instead it simply assumes that the algorithm is run for the same number of evaluations in each run and is stopped at the 'ideal' number of evaluations. This gives an indication of the 'ideal' $n_{95\%}$. This decision was made explicitly, as wrongly defining stopping criteria would have a large effect on the $n_{95\%}$ and could have led to wrong conclusions. Given the large number of problems and algorithms that will be tested, such mistakes are not unlikely. Also most comparisons are made between different settings of the same algorithm. Typically this will mean that they all suffer comparably from the fact that the 'ideal' $n_{95\%}$ is calculated. Only when comparing between different algorithms, one needs to take into account that some algorithms have more effective means of defining a stopping criterium. By the time these comparisons will be made in Chapter 14, the difference between algorithms was already so large that this effect could be deemed negligable.

Furthermore it is noted that the difference with reality is not very large for some algorithms. Especially DE, and to a lesser extent also PSO, have a large tendency to 'collapse' to a very good solution once it has been found. It was already noted by Vasile et al. [2008] that a very effective stopping criterion was to stop once the average distance between individuals in the decision space drops below a certain threshold. Note that all variables in the decision space need to be scaled from 0 to 1 for fair distance calculation. Examples of this phenomenon are given in Figure 7.4 for the EVEJS MGA problem, which is further described in Section 8.3.3, in Figure 7.5 for the Cassini2 DSM In Second Leg Only VF problem, which is further described in Section 8.3.6, and finally in Figure 7.6 for the Cassini2 MGA problem, which is further described in Section 8.3.5.

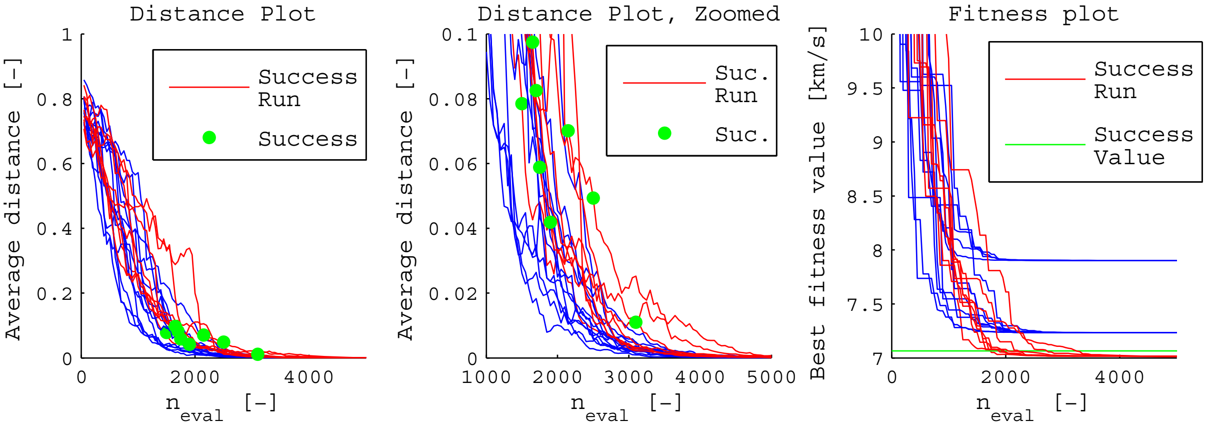


Figure 7.4: Example of convergence of 20 runs with the same algorithm on the EVEJS MGA problem. The average distance between individuals is plotted on the left. The middle plot is a zoomed version of that. Finally the plot on the right gives the fitness values. The green dots specify the moment at which success is achieved.

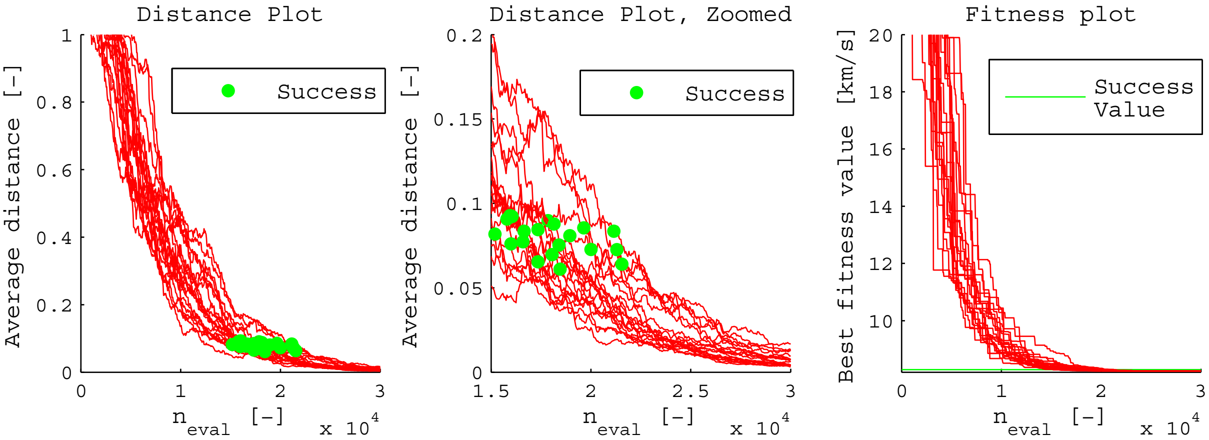


Figure 7.5: Example of convergence of 20 runs with the same algorithm on the Cassini2 DSM In Second Leg Only VF problem. The average distance between individuals is plotted on the left. The middle plot is a zoomed version of that. Finally the plot on the right gives the fitness values. The green dots specify the moment at which success is achieved.

It can be seen in all three plots that the convergence typically occurs after about 50% additional evaluations have been performed. Furthermore it can be seen that the average distance rapidly approaches 0 after success is achieved, meaning that the loss of choosing a certain average distance as a stopping criterion is not extremely high. Actually the ‘real’ performance may be better than the ‘ideal’ performance in case a very good stopping criterion is chosen. Algorithm 7.1 will namely select 5000 evaluations as the number of evaluations for the ideal $n_{95\%}$ for the Cassini2 MGA problem. On the other hand in case 0.01 would be used as a stopping criterion, various optimization runs would stop earlier, between 2000 and 4000 evaluations, thereby saving computational effort.

In reality one may want to specify a stopping criterion that is slightly safer in case the user is not familiar with the problem and the convergence of the algorithm on the problem. This will result in slightly worse performance, but one wants to be sure that premature convergence is avoided. Given that DE typically collapses very quickly afterwards, the required number of evaluations will not be factors higher.

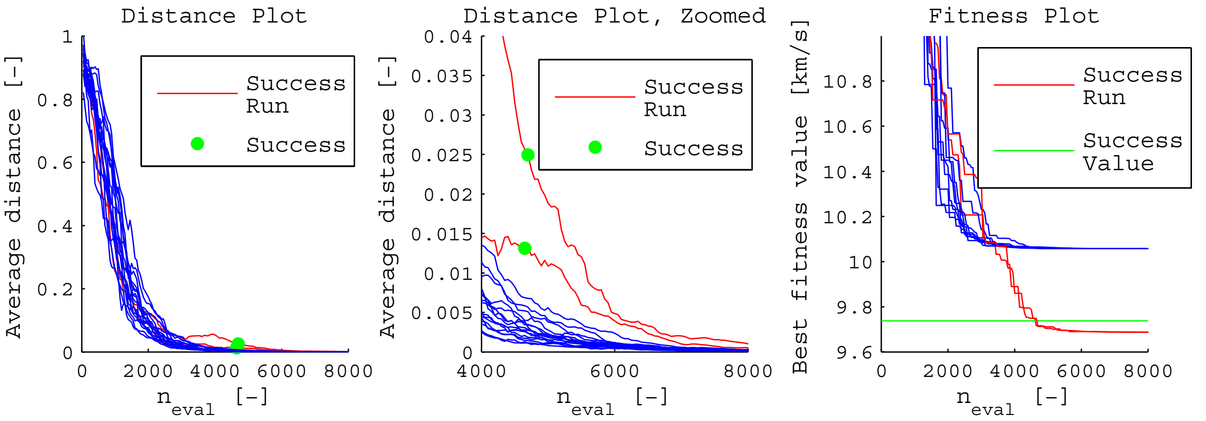


Figure 7.6: Example of convergence of 20 runs with the same algorithm on the Cassini2 MGA problem. The average distance between individuals is plotted on the left. The middle plot is a zoomed version of that. Finally the plot on the right gives the fitness values. The green dots specify the moment at which success is achieved.

7.4 Accuracy of Performance Criterion

This section will estimate the accuracy of the performance criterion used in this thesis. To get an estimate of this accuracy, various Monte Carlo analyses were performed. The performance of various algorithms on various problems was measured using the proposed performance criterion 100 times. The spread in the outcome gives a good indication of the accuracy and repeatability of the results obtained using this performance criterion.
Before discussing the results of this analysis, it is noted explicitly that all results in this section are purely meant to estimate the accuracy of the performance criterion. To do so various optimization algorithms and tuning parameter settings are tested. The goal is explicitly not to compare the actual performance of these algorithms. That will be done in the following chapters.

First of all the EMJS MGA problem, which will be further discussed in Section 8.3.2, was used for this test. Note that this thus means that a total of 100·200 independent optimization runs were performed per setting. The results are given in a box-and-whisker plot in Figure 7.7. Similarly the mean, median and standard deviation, along with the precise optimizer settings that were used are given in Table 7.3.

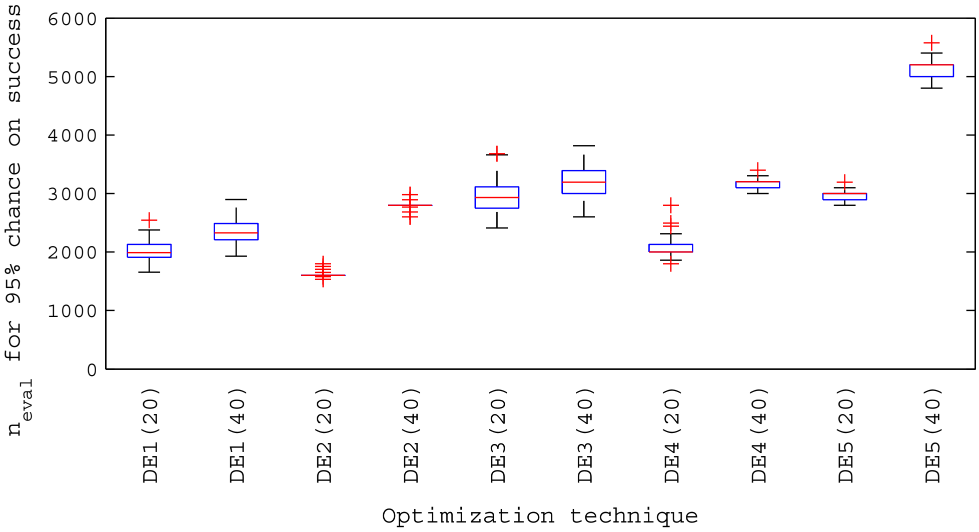


Figure 7.7: Box-and-whisker plot of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the EMJS MGA problem. See text for explanation.

Table 7.3: The mean, median and standard deviation of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the EMJS MGA problem.

Name	F	CR	$strat$	n_{pop}	median	mean	σ	σ/mean [%]
DE1(20)	0.8	0.8	1	20	1991	2015	181	9.0
DE1(40)	0.8	0.8	1	40	2328	2346	204	8.7
DE2(20)	0.8	0.8	2	20	1600	1629	63	3.9
DE2(20)	0.8	0.8	2	40	2800	2798	57	2.0
DE3(20)	0.8	0.8	3	20	2932	2946	258	8.8
DE3(20)	0.8	0.8	3	40	3194	3197	250	7.8
DE4(20)	0.8	0.8	4	20	2000	2067	138	6.7
DE4(40)	0.8	0.8	4	40	3200	3165	109	3.4
DE5(20)	0.8	0.8	5	20	3000	2953	96	3.3
DE5(40)	0.8	0.8	5	40	5200	5126	137	2.7

It is noted that in the box-and-whisker plot the median is plotted with a red line, the 25th and 75th percentile are given by the region in the box and 1.5 times the interquartile range (range between 25th and 75th percentile) are indicated by the outer edge of the whisker. In case only values are closer together, the outermost value is indicated by the outer edge of the whisker instead. Values outside this range are plotted separately with the red pluses and are considered outliers. Note that if the distribution is normal, 99.3 % of the samples are within these whiskers. Finally it is noted that the whiskers are plotted with large dashed lines by default, which have no further meaning.

From Table 7.3 and Figure 7.7 it is concluded that the accuracy of the performance criterion is quite high. The standard deviation is typically 2 to 10% and a large number of samples is within a box of 2 to 5% around the actual performance. The extreme outliers appear to be at 20% from the actual performance.

A similar analysis is performed on a different problem, namely the Cassini2 Last Leg VF problem described in 8.4.2. The results of this analysis are given in Figure 7.8. The mean, median, standard deviation, and tuning parameter settings are given in Table 7.4 for the DE cases, in Table 7.5 for the GA cases and in Table 7.6 for the PSO cases.

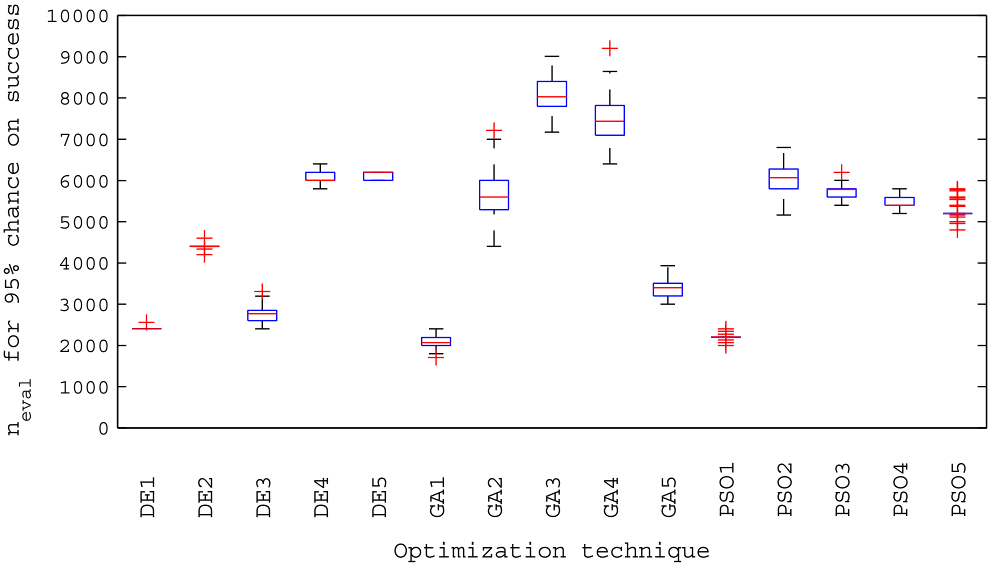


Figure 7.8: Box-and-whisker plot of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE, GA and PSO settings were used and the analysis was performed on the Cassini2 Last Leg VF problem. See text for explanation.

Table 7.4: The mean, median and standard deviation of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the Cassini2 Last Leg VF problem.

Name	F	CR	$strat$	n_{pop}	median	mean	σ	σ/mean [%]
DE1	0.8	0.4	1	40	2400	2402	16	0.7
DE2	0.4	0.6	2	80	4400	4382	62	1.4
DE3	0.4	0.2	3	20	2769	2756	176	6.4
DE4	1.0	0.6	4	50	6000	6062	101	1.7
DE5	0.6	1.0	5	60	6200	6134	94	1.5

Table 7.5: The mean, median and standard deviation of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various GA settings were used and the analysis was performed on the Cassini2 Last Leg VF problem.

Name	n_{pop}	CR	m	m_{type}	m_w	sel_{type}	median	mean	σ	σ/mean [%]
GA1	25	0.4	0.02	0	0.2	1	2066	2068	130	6.3
GA2	25	1.0	0.02	1	0.05	0	5600	5680	499	8.8
GA3	50	0.6	0.05	0	0.05	1	8028	8107	465	5.7
GA4	75	0.6	0.1	1	0.05	0	7437	7502	505	6.7
GA5	100	1.0	0.2	0	0.2	0	3400	3379	165	4.9

Table 7.6: The mean, median and standard deviation of $n_{95\%}$ for random 100 independent samples consisting of 200 runs. Various PSO settings were used and the analysis was performed on the Cassini2 Last Leg VF problem. Note that n_{type} and n_{param} stand for $neigh_{type}$ and $neigh_{param}$ respectively.

Name	n_{pop}	ω	η_1	η_2	v_{coeff}	var	n_{type}	n_{param}	median	mean	σ	σ/mean [%]
PSO1	35	0.6	3.9	1.5	0.4	5	2	18	2200	2207	62	2.8
PSO2	50	0.4	3.4	2.4	0.9	5	2	2	6069	6088	301	4.9
PSO3	60	0.7	1.4	1.9	0.8	3	3	13	5784	5750	166	2.9
PSO4	120	0.4	1.5	2.4	0.8	1	3	1	5400	5489	117	2.1
PSO5	190	0.5	1.4	1.9	0.2	1	1	18	5200	5217	180	3.4

It is noted that various instances of DE and PSO have an exceptional high accuracy on this problem. This is caused by the fact that the readout frequency is each 200 evaluations and the fact that both obtain a success rate of at least 95% at the same readout interval. Still it is noted that the standard deviation is within 10% of the actual value typically. Also the spread is again within 20 to 25 % of the actual performance.

This analysis is also performed for a more difficult problem, namely the Cassini2 DSM In Second Leg VF problem described in Section 8.3.6. Some pseudo-random settings with good performance were taken from the tuning process because taking random values would result in a very large difference in performance between different algorithm settings. The results are given in Figure 7.9. Similarly the mean, median and standard deviation, along with the precise optimizer settings that were used are given in Table 7.7.

Also for this problem the results are considered accurate. For some algorithms the standard deviation has risen to about 15 %. The main difference between these runs (DE4(35) and DE1(60)) is that they have a comparably low success rate of only 30 to 40%. As can also be derived from Equation 7.1, $n_{95\%}$ is highly sensitive to low success rates. Optimization algorithms that perform well, solve most problems with a success rate of at least 30 %. In these cases this sensitivity is limited to an accuracy that is still reasonable for comparing different settings.
The problems presented so far are also typically solved with a success rate that is equal to or higher than 30 %. The only exception to this are the Cassini1 and Cassini2 MGA problems. These problems are typically solved with an success rate of only 5 or 10%. Hence one way to cope with this is to increase the number of independent runs. In this thesis the number of independent runs was increased to 1000

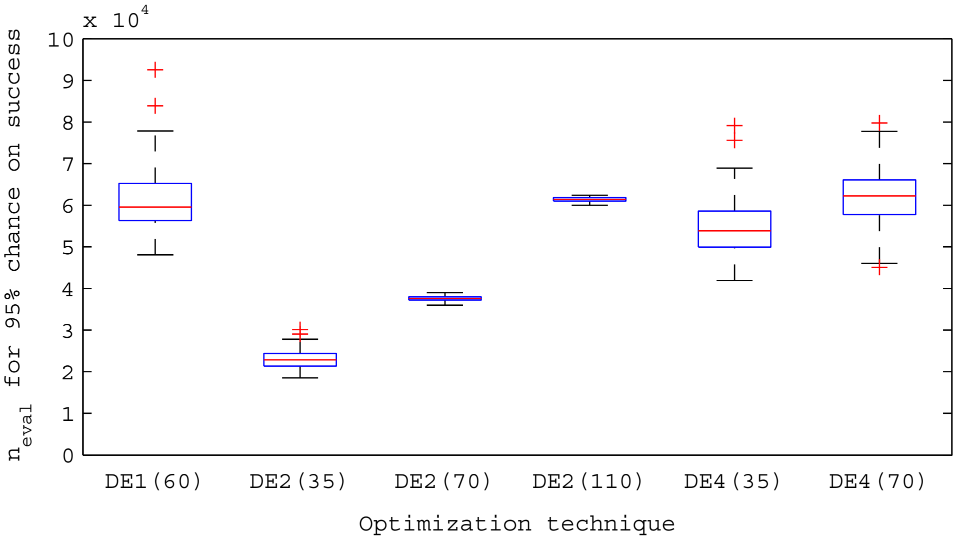


Figure 7.9: Box-and-whisker plot of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the Cassini2 DSM In Second Leg VF problem. See text for explanation.

Table 7.7: The mean, median and standard deviation of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the Cassini2 DSM In Second Leg VF problem.

Name	F	CR	$strat$	n_{pop}	median $[\cdot 10^3]$	mean $[\cdot 10^3]$	σ $[\cdot 10^3]$	σ/mean [%]
DE1(60)	0.7	0.9	1	60	59.5	61.1	7.6	12.4
DE2(35)	0.7	0.97	2	35	22.8	23.1	2.4	10.4
DE2(70)	0.7	0.97	2	70	37.6	37.6	0.5	1.3
DE2(110)	0.7	0.97	2	110	61.4	61.4	0.6	1.0
DE4(35)	0.5	0.94	4	35	53.9	55.0	7.6	13.8
DE4(70)	0.5	0.94	4	70	62.3	61.8	6.9	11.1

runs for these problems. The results of the Monte Carlo analysis for the Cassini2 MGA problem using both 200 runs and 1000 runs are presented in Figure 7.10. The mean, median and standard deviation, along with the precise optimizer settings that were used are given in Table 7.8. Note that for plotting reasons, settings were selected with roughly the same $n_{95\%}$.

Table 7.8: The mean, median and standard deviation of $n_{95\%}$ for 100 independent samples. Various DE settings were used and the analysis was performed on the Cassini2 MGA problem. Settings were run for both 200 runs and 1000 runs for comparison of the resulting accuracy.

Name	F	CR	$strat$	n_{pop}	# of runs	median	mean	σ	σ/mean
						$[\cdot 10^3]$	$[\cdot 10^3]$	$[\cdot 10^3]$	[%]
DE1(200)	0.6	0.9	1	60	200	215	216	54	25
DE1(1000)	0.6	0.9	1	60	1000	216	218	24	11
DE3(200)	0.9	0.9	3	20	200	213	217	50	23
DE3(1000)	0.9	0.9	3	20	1000	221	223	26	12
DE4(200)	0.5	0.94	4	20	200	208	215	53	25
DE4(1000)	0.5	0.94	4	20	1000	214	216	28	13

It can be seen in Table 7.8 that the standard deviation is quite considerable in case only 200 runs are performed for these problems. Also Figure 7.10 shows that the spread is so large that a successful comparison between algorithms can not be made anymore. Increasing the number of runs to 1000 reduces the standard deviation to about 12 % for these algorithms. This allows for a better comparison between

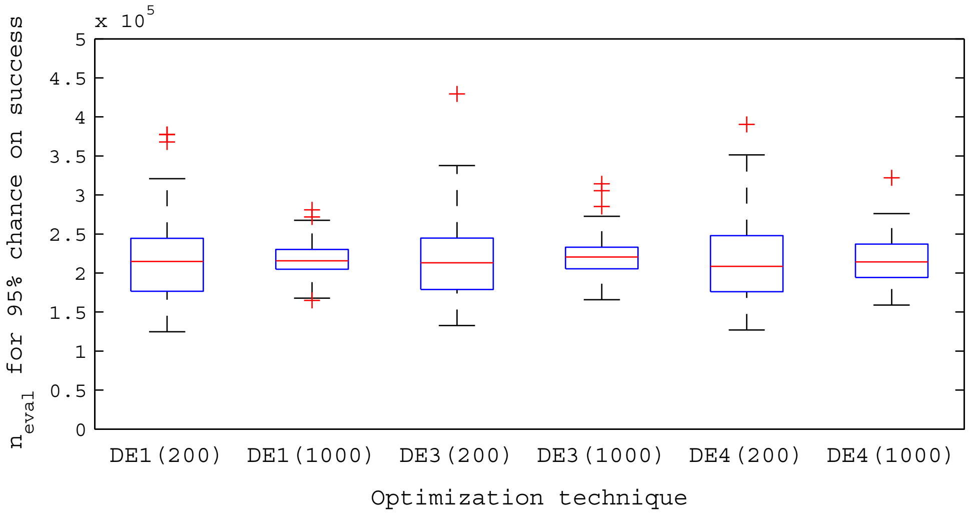


Figure 7.10: Box-and-whisker plot of $n_{95\%}$ for 100 independent samples consisting of 200 runs. Various DE settings were used and the analysis was performed on the Cassini2 DSM In Second Leg VF problem. See text for explanation.

algorithms.

Conclusion

In general it is noted that the performance criterion is accurate enough to allow it to be used in comparing different algorithm settings in the tuning phase. The performance is largely dependent on the success rate and additional runs should be performed in case the success rate is very low.

Chapter 8

Tuning Problem Description

This chapter describes the problems that will be used to tune and compare the different optimization techniques in the following chapters. An introduction is given in Section 8.1. Subsequently the full-difficulty Cassini2 problems are discussed in Section 8.2. Section 8.3 discusses the trajectory model simplification problems. Section 8.4 discusses the problems that optimize specific legs of the trajectory. The problems with narrowed bounds are discussed in Section 8.5. Finally an overview of all problems is given in Section 8.6.

8.1 Introduction

This section gives an introduction into the set-up of the tuning problems. Section 8.1.1 first discusses the criteria that were defined for these tuning problems. Subsequently Section 8.1.2 deals with the selection of a certain GTOP trajectory.

8.1.1 Tuning Problem Criteria

To allow for tuning these optimization techniques, these problems need to meet the following two requirements:

- Known optimum: The optimum of the problems must be known in order to measure the performance of the optimization techniques in finding the optimum.
- Resemblence: The problems should resemble the real problems as much as possible, while still being simple enough to allow for repeated optimization using a large number of optimization techniques and settings.

The requirement to know the optimum value of the tuning problems can be difficult to meet. This can be coped with in two ways: by using existing problems of which the optimum is known or by performing enough simulations such that best putative solution can be considered to be the actual solution to the problem.
For this thesis, the GTOP database offers a unique opportunity to define problems of which the optimum can be considered to be known. Most problems in GTOP have namely been optimized using a wide variety of optimization techniques. Save for the TandEM problems and the Messenger-Full problem, the optima have been stable for at least three years [Izzo and Vinko, 2012]. Hence many of the problems in this thesis draw inspiration from GTOP problems.

The resemblance requirement is very important because the performance of optimization techniques is problem-dependent. Of course a full-difficulty problem resembles the actual optimization problems perfectly, but it will not allow for repeated optimization by a large number of different optimization techniques and settings. Because the latter is essential for successful tuning, the problems are simplified. By simplifying the problem in three different ways and subsequently combining the results of the different simplification methods, a result is striven for that matches the actual performance as much as possible. First of all the problem is simplified by specifying some or all legs to be MGA legs instead of MGA-1DSM legs. The second way is by optimizing only a subset of all parameters and using the best putative values for the other parameters. This way only one or two legs of the entire trajectory are optimized and optimal values are used for the other legs. Finally the third way is to narrow the bounds of the search space. Furthermore it is noted that all tuning problems consist of optimizing a similar mission in terms of target planet and transfer times.

8.1.2 Selection of Trajectories

The GTOP database consists of a wide variety of problems, as can be seen in Table 8.1.

Table 8.1: Overview of the GTOP database. [Izzo and Vinko, 2012]

Problem	Target	Trajectory Model	Optimization Criterion	# of Vars	# of Sub-problems
Cassini1	Saturn	MGA	ΔV	6	1
GTOC1	Asteroid	MGA	different	8	1
SAGAS	Edge of solar system	MGA-1DSM-VF-UNP	Time	12	1
Messenger	Mercury	MGA-1DSM-VF-UNP	ΔV	18	1
TandEM	Saturn	MGA-1DSM-VF-UNP	Final mass	18	48
Rosetta	Asteroid	MGA-1DSM-VF-UNP	ΔV	22	1
Cassini2	Saturn	MGA-1DSM-VF-UNP	ΔV	22	1
Messenger-Full	Mercury	MGA-1DSM-VF-UNP	ΔV	26	1

A number of problems in GTOP are trajectories to Saturn. This is deemed a very large advantage as it allows to tune the optimization algorithm properly and subsequently apply the optimization algorithm to a wide variety of problems. This was also one of the reasons why this thesis focuses on trajectories to Saturn, as discussed in Chapter 3.
This identifies Cassini1, Cassini2 and the TandEM problems as useful problems for this thesis. The Messenger problems are not considered because the search space is likely to be very different. Furthermore these trajectories are likely to include multiple-revolution transfers, which are not possible with all trajectory models. The Rosetta mission is also deemed to have a different search space, because the goal is to reach an asteroid that is in a significantly different orbit than Saturn. GTOC1 and SAGAS are not used because their objective is very different.

In contrast to most of the other problems, the decision vectors corresponding to the TandEM trajectories are not posted online. This is considered a considerable disadvantage in setting up the tuning problems. Hence the Cassini1 and Cassini2 trajectories are used as inspiration in setting up the tuning problems.

Before going into details on the way the tuning problems were set up, it is important to realize that the best putative solution of GTOP is not necessarily the optimum that can be found with this thesis software. More general trajectory models are used in this thesis, which include both powered swing-bys and DSMs. This allows for better trajectories to be found than is possible with GTOP. During the preliminary runs of the tuning problems, improvements have been found over the GTOP solutions. This is further discussed in Chapter 16. Also this difference in trajectory model means that the problem dimension increases from 22 to 26 variables.

8.2 Cassini2 Full Difficulty

First of all it is noted that this problem is too difficult for the early tuning phase of course. However, because the other problems are derived from this problem, it is discussed first. The best putative solution for the GTOP trajectory model was already shown in Figure 5.9 and requires a total ΔV of 8383.2 m/s. Over the course of this thesis, a better solution was found for the trajectory models in this thesis. This trajectory can be seen in Figure 8.1. The improved trajectory requires a total ΔV of 8237.9 m/s.

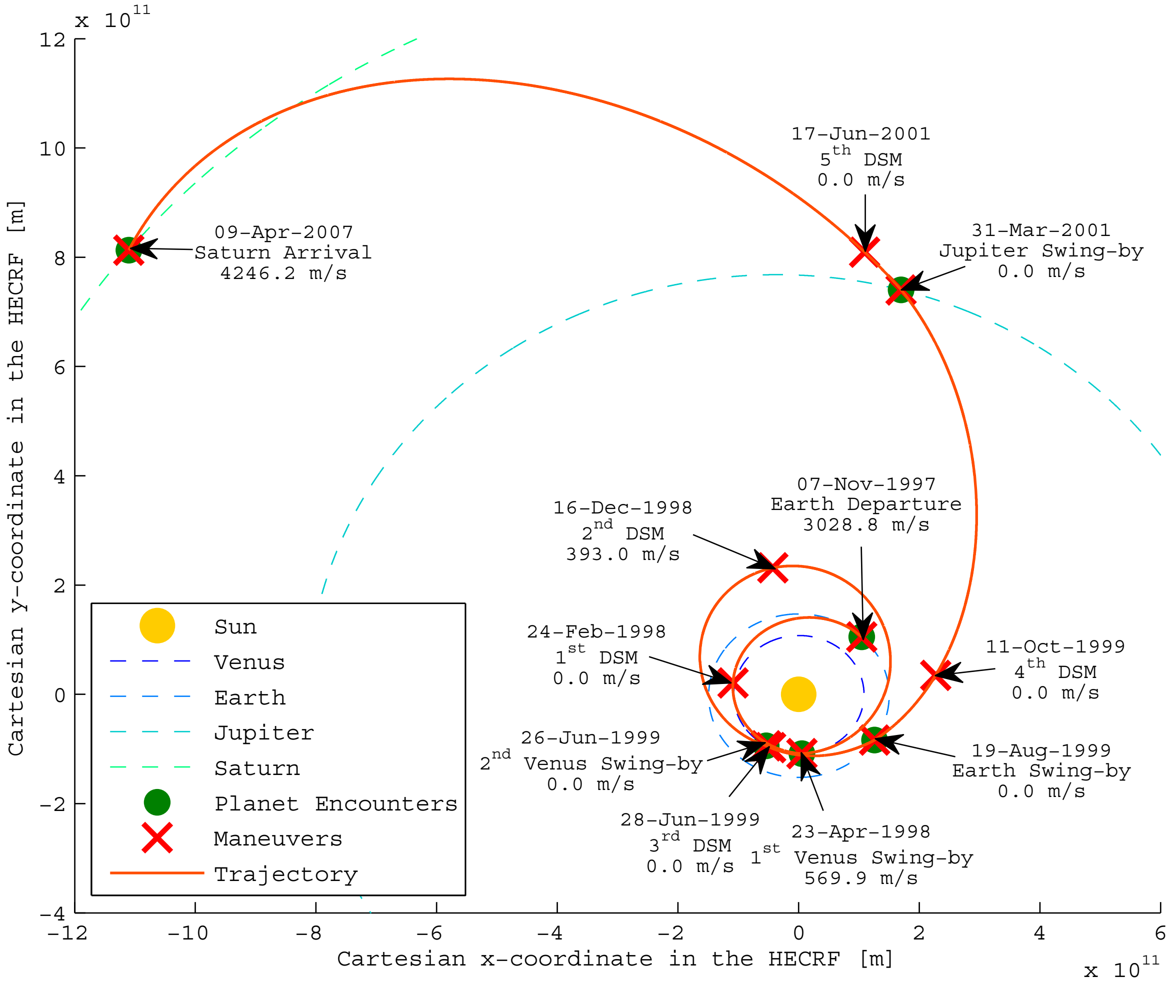


Figure 8.1: The best putative trajectory for the Cassini2 MGA-1DSM-VF problem.

Two implementations of this problem were made. The decision vector, the corresponding bounds and the best putative solutions for the PF version of the problem are given in Table 8.2. Similarly the decision vector, the corresponding bounds and the best putative solutions for the VF version of the problem are given in Table 8.3. It is noted here that both the best putative GTOP solution as well as the best putative solution according to this thesis are given for both trajectory models. The best putative GTOP solution was obtained by applying a local optimization technique to the GTOP solution transcribed to these trajectory models.

It must be noted that the determination of the bounds for the PF variant of the trajectory is subjective. The PF trajectory model employs a different set of variables to describe the trajectory and therefore it is impossible to include the exact same set of possible trajectories. An attempt was made to include the same trajectories as the VF variant, although that is impossible in practice. With just the bounds given in Table 8.2 a solution can be found which requires only 7958 m/s. The excess velocity at the Earth is namely not bound in the PF trajectory. The fact that it should be at least 3000 m/s in the VF trajectory prevents better solutions from being found. Table 8.2 gives the best putative solution for the PF trajectory in case this same constraint is added. The narrowed bounds problems were also set up around this solution.

Another issue is that it was troublesome to set proper bounds on the ΔV for the VF variant of the trajectory. In theory the ΔV may namely be negative or positive, otherwisely said decelerating the

Table 8.2: The decision vector bounds and the putative solution for the Cassini2 Full PF problem. ** note that this concerns the best putative solution with a minimum excess velocity at the Earth of 3000 m/s, as discussed in the text.

Decision	Corresponding	Lower	Upper	Best Putative	Best Putative	
Vector Entry	Variable	Bound	Bound	GTOP Solution	Overall Solution**	Units
x(1)	t_0	-1000	0	-779.62980	-785.10814	MJD2000
x(2)	T_1	100	400	167.93761	167.34121	days
x(3)	T_2	100	500	424.03220	428.97894	days
x(4)	T_3	30	300	53.30487	54.14440	days
x(5)	T_4	400	1600	589.76790	589.92251	days
x(6)	T_5	800	2200	2199.96191	2199.99999	days
x(7)	η_1	0.01	0.90	0.772878	0.706525	-
x(8)	r_1	0.5	1.5	0.691818	0.724324	D_{pl}
x(9)	θ_1	0	2π	2.798174	2.425792	rad
x(10)	ϕ_1	$-\pi/12$	$\pi/12$	0.009588	0.014523	rad
x(11)	η_2	0.01	0.90	0.531757	0.553077	-
x(12)	r_2	0.5	2.5	2.162443	2.163471	D_{pl}
x(13)	θ_2	0	2π	3.095803	3.278860	rad
x(14)	ϕ_2	$-\pi/12$	$\pi/12$	-0.000001	0.000152	rad
x(15)	η_3	0.01	0.90	0.010789	0.090170	-
x(16)	r_3	0.5	2.5	0.999310	0.997426	D_{pl}
x(17)	θ_3	0	2π	0.019932	0.169589	rad
x(18)	ϕ_3	$-\pi/12$	$\pi/12$	0.001115	0.009412	rad
x(19)	η_4	0.01	0.90	0.167389	0.166311	-
x(20)	r_4	0.8	6.0	1.979967	1.971959	D_{pl}
x(21)	θ_4	0	2π	1.046060	1.045411	rad
x(22)	ϕ_4	$-\pi/12$	$\pi/12$	-0.008337	-0.008353	rad
x(23)	η_5	0.01	0.90	0.010426	0.093588	-
x(24)	r_5	0.8	2.5	1.022312	1.187055	D_{pl}
x(25)	θ_5	0	2π	0.027977	0.215349	rad
x(26)	ϕ_5	$-\pi/12$	$\pi/12$	0.000187	0.001433	rad

spacecraft or accelerating the spacecraft. If however the bounds also included the option to decelerate, the spacecraft would sometimes be caught in orbit around the swing-by planet. This is of course unwanted and would cause the trajectory to crash. Hence only positive ΔVs were permitted. This is not likely to be a problem in this simulation, because the goal is to *increase* the heliocentric velocity to end up at Saturn. A solution to this problem is needed for trajectories going to the inner planets though.

8.3 Trajectory Model Simplification Problems

This section discusses the problems that simplify the trajectory model. Main inspiration for these problems is the Cassini2 trajectory. Hence three different problems were derived from the Cassini2 trajectory. First of all a problem was set up in which all MGA-1DSM legs were replaced by MGA legs. Also a case was set up in which all MGA-1DSM legs were replaced by MGA legs, except for the second leg. Preliminary runs namely indicated that the second leg was the only leg in which a DSM was actually present in case powered swing-bys are included. This case was set up both for the PF and for the VF trajectory model.
A fourth problem was derived from the Cassini1 problem in GTOP. Among others Vasile et al. [2008] show that the Cassini1 problem is already a difficult optimization problem that many optimization techniques struggle with. Since the Cassini2 MGA problem and the Cassini2 MGA problems with a DSM in the second leg are likely to be of at least equal difficulty, simpler MGA problems were also set up.

A useful set of optimization problems was found in [Myatt et al., 2004]. They describe four MGA trajec-

Table 8.3: The decision vector bounds and the best putative solution for the Cassini2 Full VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative GTOP Solution	Best Putative Overall Solution	Units
x(1)	t_0	-1000	0	-779.62980	-785.10814	MJD2000
x(2)	T_1	100	400	167.93761	167.34121	days
x(3)	T_2	100	500	424.03220	428.97894	days
x(4)	T_3	30	300	53.30487	54.14440	days
x(5)	T_4	400	1600	589.76790	589.92251	days
x(6)	T_5	800	2200	2199.96191	2199.99999	days
x(7)	η_1	0.01	0.90	0.772878	0.65302	-
x(8)	V_∞	3000	5000	3265.804	3028.80199	m/s
x(9)	θ	0	2π	3.320288	3.26708	rad
x(10)	ϕ	$-\pi/2$	$\pi/2$	0.237444	0.20197	rad
x(11)	η_2	0.01	0.90	0.531757	0.55174	-
x(12)	$b_{incl,2}$	$-\pi$	π	-1.593311	-1.52221	rad
x(13)	$r_{p,2}$	1.05	6	1.35685	1.20565	R_{pl}
x(14)	ΔV_2	0	2000	0	569.86175	m/s
x(15)	η_3	0.01	0.90	0.010789	0.03933	-
x(16)	$b_{incl,3}$	$-\pi$	π	-1.959572	-1.97206	rad
x(17)	$r_{p,3}$	1.05	6	1.05000	1.12068	R_{pl}
x(18)	ΔV_3	0	2000	0	0	m/s
x(19)	η_4	0.01	0.90	0.167389	0.09044	-
x(20)	$b_{incl,4}$	$-\pi$	π	-1.55480	-1.54645	rad
x(21)	$r_{p,4}$	1.15	6.5	1.30685	1.283713	R_{pl}
x(22)	ΔV_4	0	2000	0	0	m/s
x(23)	η_5	0.01	0.90	0.010426	0.03573	-
x(24)	$b_{incl,5}$	$-\pi$	π	-1.513432	-1.51344	rad
x(25)	$r_{p,5}$	1.7	291	69.8134	70.00169	R_{pl}
x(26)	ΔV_5	0	2000	0	0	m/s

tory problems to Saturn with increasing difficulty. The most difficult problem is similar to the Cassini1 problem, and is therefore discarded. The other three trajectories are an Earth-Jupiter-Saturn (EJS), Earth-Mars-Jupiter-Saturn (EMJS) and Earth-Venus-Earth-Jupiter-Saturn (EVEJS) transfer.
These three trajectories have been adopted in this thesis. The bounds have been changed slightly to make them consistent with the bounds of the Cassini1 problem, such that a comparison can be made between them. For similar reasons, and because it is unknown which ephemeris was used in [Myatt et al., 2004], GTOP ephemeris is used for all trajectory problems. Also other constants, such as the gravitational parameters, minimum pericenter radii and planetary radii were assumed similar to GTOP. Finally it is noted that although the EJS, EMJS and EVEJS MGA problems are similar to those in [Myatt et al., 2004], the solutions will typically be different. Myatt et al. [2004] assumed unpowered gravity assists, whereas this thesis includes the option to perform a powered gravity assist.

In total a set of 7 problems of increasing difficulty is set up. An overview of the trajectories is given in Table 8.4.

8.3.1 EJS MGA Problem

This problem is a transfer from Earth to Saturn with only one swing-by along Jupiter. The objective is to minimize the ΔV for a mission from the SoI of the Earth to a capture orbit around Saturn with a pericenter radius of 108950 km and an eccentricity of 0.98.
It is inspired by a similar trajectory described in [Myatt et al., 2004]. The decision vector of this problem has the lowest number of variables of the tuning problems considered in this thesis. The decision vector, along with the bounds and the best putative solution are given in Table 8.5. The best putative trajectory is visualized in Figure 8.2.

Table 8.4: Overview of the trajectory model simplification problems that were set up for tuning purposes. Note that all problems share the objective of minimizing the total ΔV, but sometimes the total ΔV for capture in a certain orbit is optimized and otherwise the total ΔV for rendezvous with the final planet is optimized. The first one is the objective for the Cassini1 problem in GTOP, whereas the second objective is the objective for the Cassini2 problem.

	Planetary	Vari-	MGA	DSM	Objective	Relevant
Problem Name	Sequence	ables	Legs	Legs	Type	Section
EJS	E-J-S	3	2	0	capture	8.3.1
EMJS	E-M-J-S	4	3	0	capture	8.3.2
EVEJS	E-V-E-J-S	5	4	0	capture	8.3.3
Cassini1	E-V-V-E-J-S	6	5	0	capture	8.3.4
Cassini2 MGA	E-V-V-E-J-S	6	5	0	rendezvous	8.3.5
Cassini2 DSM In Second Leg PF	E-V-V-E-J-S	10	4	1	rendezvous	8.3.6
Cassini2 DSM In Second Leg VF	E-V-V-E-J-S	10	4	1	rendezvous	8.3.6

Table 8.5: The decision vector bounds and the best putative solution for the EJS MGA problem.

Decision	Corresponding	Lower	Upper	Best Putative	
Vector Entry	Variable	Bound	Bound	Solution	Units
x(1)	t_0	-1000	0	-177.85323	MJD2000
x(2)	T_1	400	2000	910.16501	days
x(3)	T_2	1000	6000	4422.78878	days

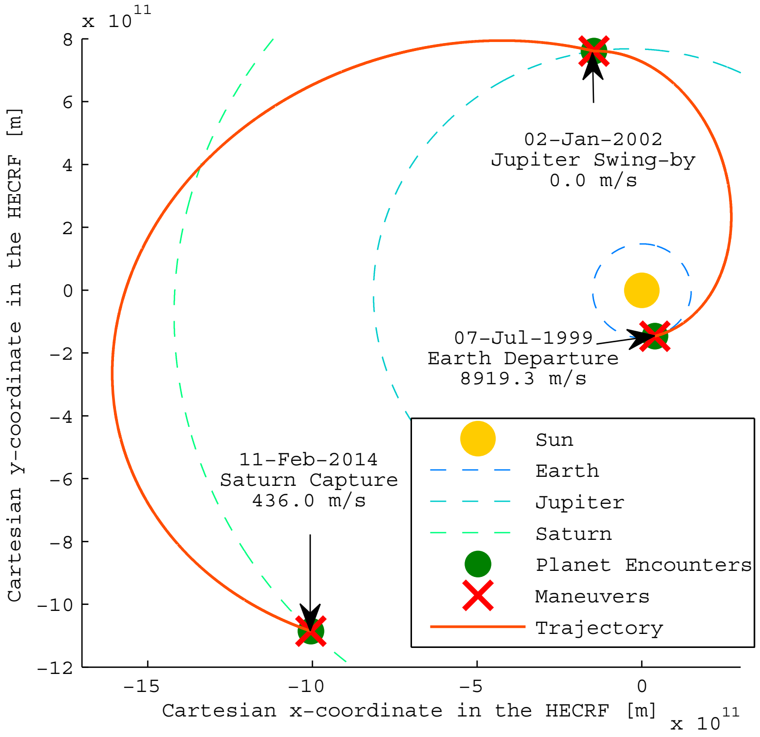


Figure 8.2: The best putative trajectory for the EJS MGA problem.

The best putative solution of 9355.2 m/s was obtained in preliminary optimization runs and was not improved upon in any of the optimization runs performed in this thesis. It is slightly higher than the optimum of 9351.79 m/s that is found in [Myatt et al., 2004]. This may very well be caused by ephemeris differences, especially since the decision vector is also very close to the decision vector in [Myatt et al., 2004]. Given that the best solution does not employ a powered swing-by, the solutions should be comparable.

8.3.2 EMJS MGA Problem

This problem is a similar transfer as EJS, only this time a swing-by is performed at both Mars and Jupiter. The objective is also the same. The problem is inspired by a similar trajectory with different bounds described in [Myatt et al., 2004]. The decision vector, along with the bounds and the best putative solution belonging to the EMJS MGA problem are given in Table 8.6. The best putative trajectory is visualized in Figure 8.3.

Table 8.6: The decision vector bounds and the best putative solution for the EMJS MGA problem.

Decision	Corresponding	Lower	Upper	Best Putative	
Vector Entry	Variable	Bound	Bound	Solution	Units
x(1)	t_0	-1000	0	-840.26891	MJD2000
x(2)	T_1	100	1000	668.63519	days
x(3)	T_2	400	2000	1164.18805	days
x(4)	T_3	1000	6000	5765.75449	days

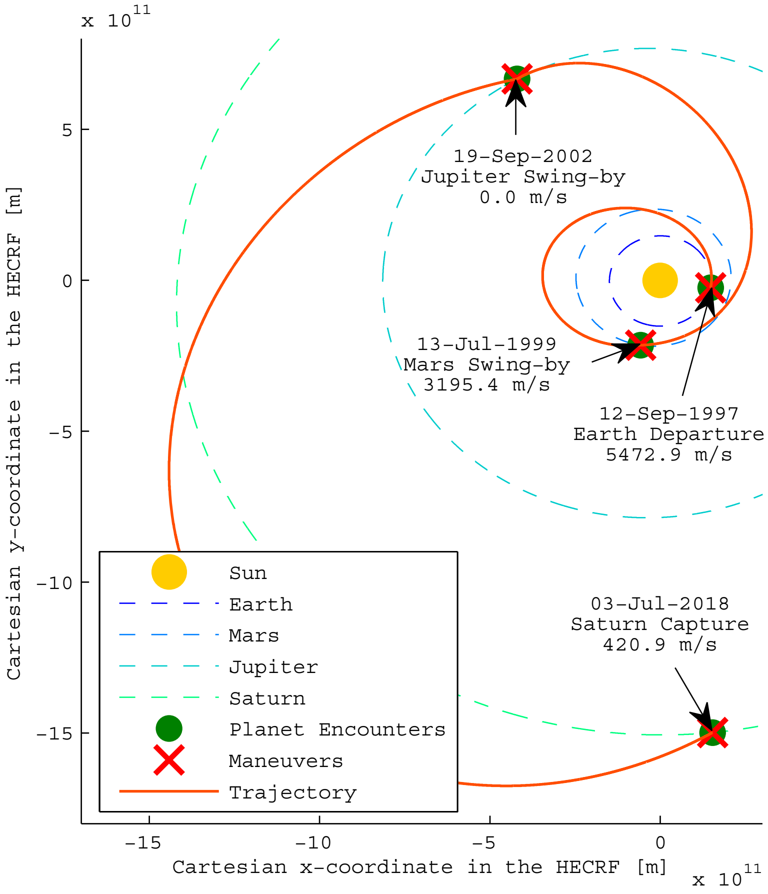


Figure 8.3: The best putative trajectory for the EMJS MGA problem.

The best putative solution of 9089.2 m/s was obtained in preliminary optimization runs and was not improved upon in any of the optimization runs performed in this thesis. In this case the swing-by at Mars is powered, causing a better solution than the 9185.97 m/s that is found in [Myatt et al., 2004].

8.3.3 EVEJS MGA Problem

Also the EVEJS MGA problem is inspired by a similar problem with slightly different bounds in [Myatt et al., 2004]. For this problems consecutive swing-bys are performed at Venus, Earth and Jupiter. The objective remains the same. The decision vector, along with the bounds and the best putative solution belonging to the EMJS MGA problem are given in Table 8.7. The best putative trajectory is visualized in Figure 8.4.

Table 8.7: The decision vector bounds and the best putative solution for the EVEJS MGA problem. Note that the optimal value for T_4 is at the boundary of the search space. A better solution is hence likely to be found at a larger value of T_4. Because the purpose of this problem is just the tuning of optimization algorithms, the bounds are left as is.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	t_0	-1000	0	-237.95767	MJD2000
x(2)	T_1	30	400	163.02744	days
x(3)	T_2	30	400	274.26495	days
x(4)	T_3	400	2000	1106.05043	days
x(5)	T_4	1000	6000	6000.00000	days

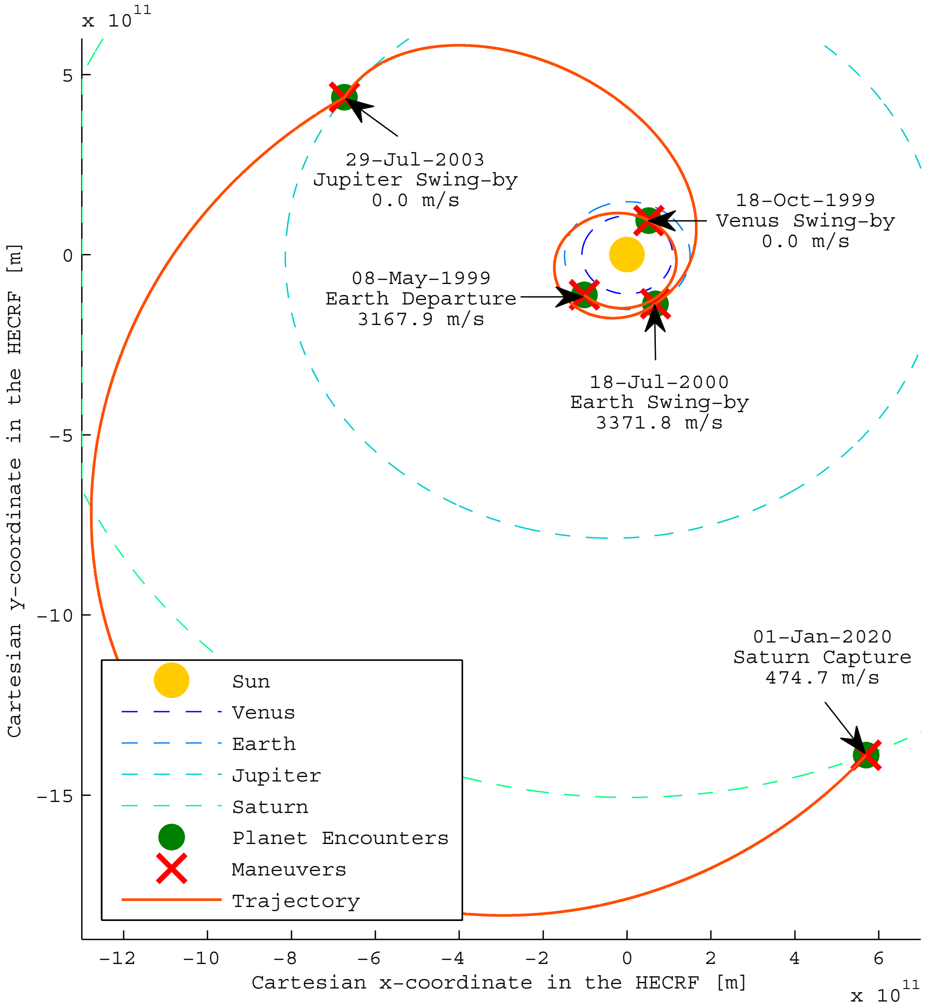


Figure 8.4: The best putative trajectory for the EVEJS MGA problem.

The best putative solution of 7014.4 m/s was obtained in preliminary optimization runs and was not improved upon in any of the optimization runs performed in this thesis. In this case the optimum is significantly lower than the optimum of 7548.36 m/s that is found in [Myatt et al., 2004].

8.3.4 Cassini1 MGA Problem

This problem is the same the Cassini1 problem from GTOP. It is modelled using the trajectory class in this thesis, but the same ephemeris, values for constants and the same objectives were used. The Cassini1 problem is a transfer from Earth to Saturn with swing-bys along Venus, Venus, Earth and Jupiter. The objective is the same as that of the EJS, EMJS and EVEJS problems. The decision vector, along with the bounds and the best putative solution belonging to the Cassini1 problem are given in Table 8.8. The best putative trajectory was already visualized in Figure 5.4.

Table 8.8: The decision vector bounds and the best putative solution for the Cassini1 MGA problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	t_0	-1000	0	-789.81170	MJD2000
x(2)	T_1	30	400	158.30203	days
x(3)	T_2	100	470	449.38587	days
x(4)	T_3	30	400	54.74897	days
x(5)	T_4	400	2000	1024.36206	days
x(6)	T_5	1000	6000	4552.30797	days

The best putative solution of 4930.7 m/s was found in 2006 by Stickel [Izzo and Vinko, 2012]. It is however noted that this problem appears to be considerably difficult to optimize. A very strong suboptimal solution is namely present at 5303.4 m/s, as is also further discussed in Chapter 13. Vinko and Izzo [2008] did not find a better solution than that with any optimization technique in their benchmark. In [Vasile et al., 2008] various optimization techniques do find the best putative solution after a large number of function evaluations and with a small success rate.

8.3.5 Cassini2 MGA Problem

The Cassini2 MGA problem was inspired by the Cassini2 problem from GTOP. It has similar bounds and objective, but the trajectory model has been changed to an MGA trajectory model. Hence the problem has similar dimension and probably also similar difficulty as the Cassini1 problem. The bounds differ and the objective is different though. The Cassini2 problem is a transfer from Earth to Saturn, with swing-bys along Venus, Venus, Earth and Jupiter. The objective is different from Cassini1. For Cassini2 the objective is to rendezvous with Saturn instead of to be captured in orbit around Saturn. This, along with the tighter bounds on the transfer times, causes a significantly higher objective value. The decision vector, along with the bounds and the best putative solution belonging to the Cassini2 problem are given in Table 8.9. The best putative trajectory is visualized in Figure 8.5.

Table 8.9: The decision vector bounds and the best putative solution for the Cassini2 MGA problem. Note that again the optimal solution to one of the variables is located at the boundary of the search space. Again this is left as is, because it is part of the problem for all optimization algorithms.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	t_0	-1000	0	-792.6415175	MJD2000
x(2)	T_1	100	400	159.0245571	days
x(3)	T_2	100	500	449.3855767	days
x(4)	T_3	30	300	50.75432599	days
x(5)	T_4	400	1600	589.3206427	days
x(6)	T_5	800	2200	2200.00000	days

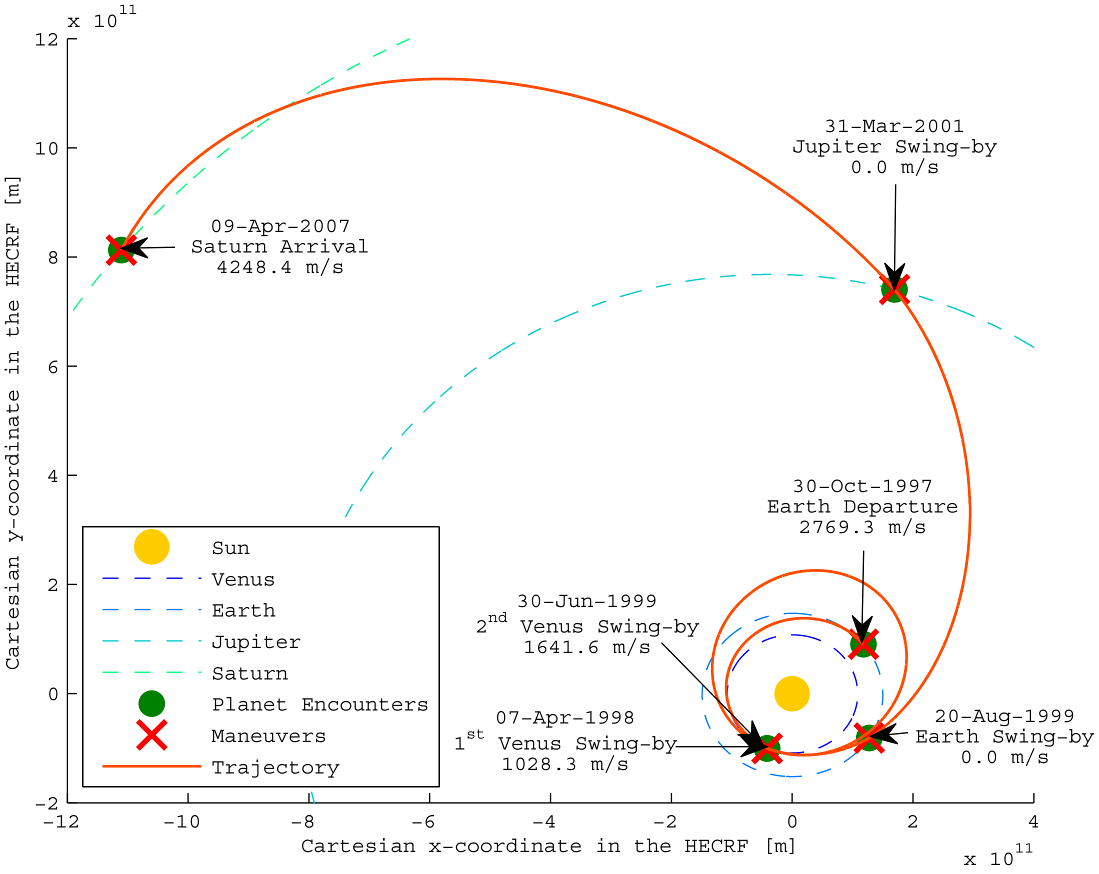


Figure 8.5: The best putative trajectory for the Cassini2 MGA problem.

The best putative solution of 9687.5 m/s was found after a large set of preliminary runs. It was not improved upon in any simulation. Similar to the Cassini1 problem, a strong suboptimal basin of attraction causes many optimization techniques to converge to a different solution of 10057.8 m/s.

8.3.6 Cassini2 DSM in Second Leg Only Problems

For this problem all the legs except the second leg are replaced by legs of an MGA trajectory model. This reduces the dimension of the decision vector from 26 variables to only 10. The second leg was chosen because it appeared to be the only leg in which a DSM was optimal.
The decision vector corresponding to these problems, along with the bounds and the best putative solution belonging to these problems can be seen in Tables 8.10 and 8.11. The best putative trajectory is visualized in Figure 8.6.

Both the VF and the PF version of the problem find a solution of 8237.9 m/s. This is a better solution than both the best putative GTOP solution. This solution was found in the preliminary runs and was not improved upon in any simulation performed afterwards.

8.4 Specific Leg Problems

This section describes a completely different type of problems. The sensitivity of the final trajectories is mimicked by employing the full Cassini2 trajectory. To simplify the problem, the variables corresponding to various legs are taken equal to the best putative GTOP solution. The basic idea is that these problems will share the same sensitivity to all optimization parameters as the final problem will have, although they will be much easier to optimize.

Table 8.10: The decision vector bounds and the best putative solution for the Cassini2 DSM in Second Leg Only PF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	t_0	-1000	0	-785.09294	MJD2000
x(2)	T_1	100	400	167.39517	days
x(3)	T_2	100	500	428.97047	days
x(4)	T_3	30	300	54.09923	days
x(5)	T_4	400	1600	589.91434	days
x(6)	T_5	800	2200	2200.00000	days
x(7)	η_2	0.01	0.90	0.553255	-
x(8)	r_2	0.5	2.5	2.163380	D_{pl}
x(9)	θ_2	0	2π	3.279187	rad
x(10)	ϕ_2	$-\pi/12$	$\pi/12$	0.000151	rad

Table 8.11: The decision vector bounds and the best putative solution for the Cassini2 DSM in Second Leg Only VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	t_0	-1000	0	-785.09294	MJD2000
x(2)	T_1	100	400	167.39517	days
x(3)	T_2	100	500	428.97047	days
x(4)	T_3	30	300	54.09923	days
x(5)	T_4	400	1600	589.91434	days
x(6)	T_5	800	2200	2200.00000	days
x(7)	η_2	0.01	0.90	0.55174	-
x(8)	$b_{incl,2}$	$-\pi$	π	-1.52221	rad
x(9)	$r_{p,2}$	1.05	6	1.20565	R_{pl}
x(10)	ΔV_2	0	2000	569.86175	m/s

An overview of the problems that were set up in this way can be seen in Table 8.12. It is noted that for all problems the objective is the same as for the actual Cassini2 trajectory, namely optimizing ΔV for rendezvous with Saturn. Also the planetary sequence is the same: Earth-Venus-Venus-Earth-Jupiter-Saturn. Finally it is noted that the constants that are needed are all taken from GTOP.

Table 8.12: Overview of the specific leg problems that were set up for tuning purposes. All problems were set up both for a position formulation and a velocity formulation model.

Problem Name	# of Variables	Relevant Section
Cassini2 First Leg PF/VF	5	8.4.1
Cassini2 Last Leg PF/VF	5	8.4.2
Cassini2 First and Last Leg PF/VF	10	8.4.3
Cassini2 Last Two Legs PF/VF	10	8.4.4

8.4.1 Cassini2 First Leg Problems

For these trajectory problems all variables of the best putative GTOP solution of the Cassini Full PF and VF problems were assumed, except for the variables specific to the first leg. This leaves five variables that are to be optimized by the optimization techniques.

The decision vector corresponding to these problems, along with the bounds and the best putative solution belonging to these problems, can be seen in Tables 8.13 and 8.14.

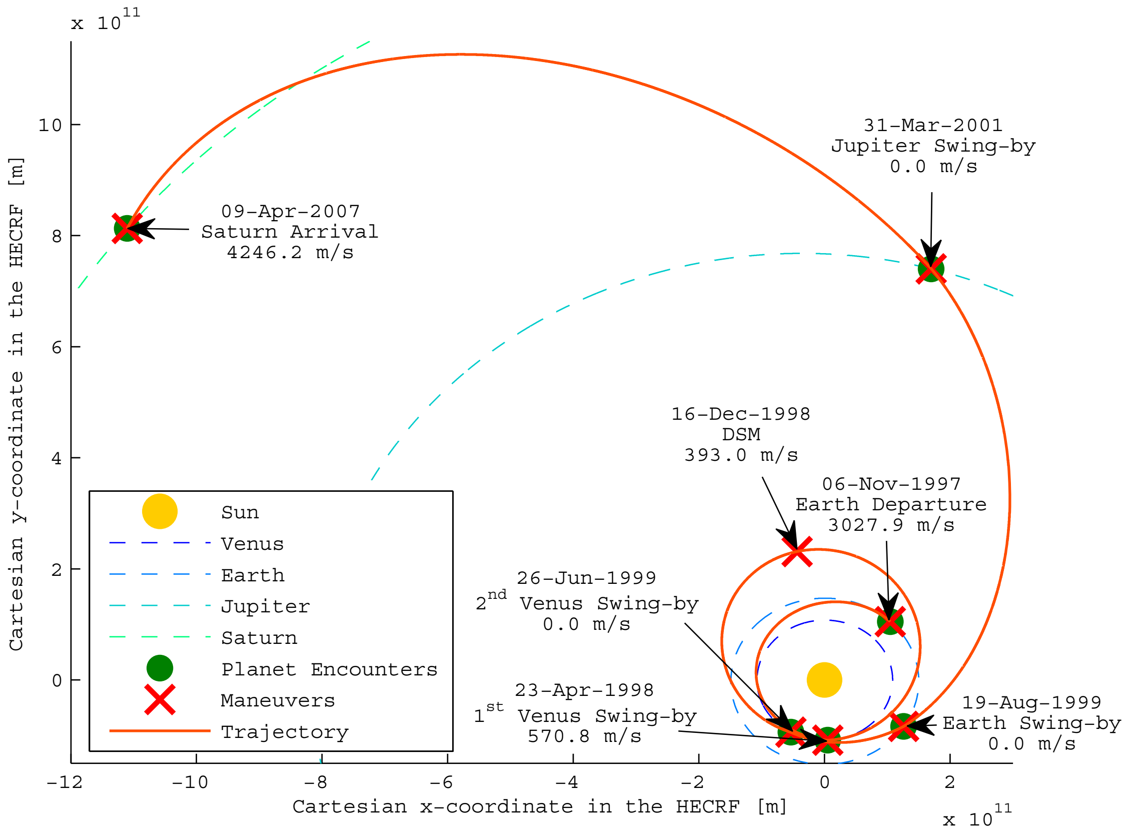


Figure 8.6: The best putative trajectory for the Cassini2 DSM in Second Leg Only problem.

Table 8.13: The decision vector bounds and the best putative solution for the Cassini2 First Leg PF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_1	100	400	167.93761	days
x(2)	η_1	0.01	0.90	0.604561	-
x(3)	r_1	0.5	1.5	0.761691	D_{pl}
x(4)	θ_1	0	2π	1.970464	rad
x(5)	ϕ_1	$-\pi/12$	$\pi/12$	0.030072	rad

Table 8.14: The decision vector bounds and the best putative solution for the Cassini2 First Leg VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_1	100	400	167.93761	days
x(2)	η_1	0.01	0.90	0.772878	-
x(3)	V_∞	3000	5000	3265.804	m/s
x(4)	θ	0	2π	3.320288	rad
x(5)	ϕ	$-\pi/2$	$\pi/2$	0.237444	rad

It is very interesting to note that during the preliminary runs a better solution than the GTOP solution was found for the PF variant of the problem. It improved the decision vector to a ΔV of 8283.8 m/s. This trajectory can be seen in Figure 8.7. Note that this trajectory could not be found for the VF variant, because of the dependency of the last legs on the first legs. This issue is discussed in greater detail in Chapter 16. The VF variant has a best putative solution of 8383.2 m/s, similar to the Cassini2 problem.

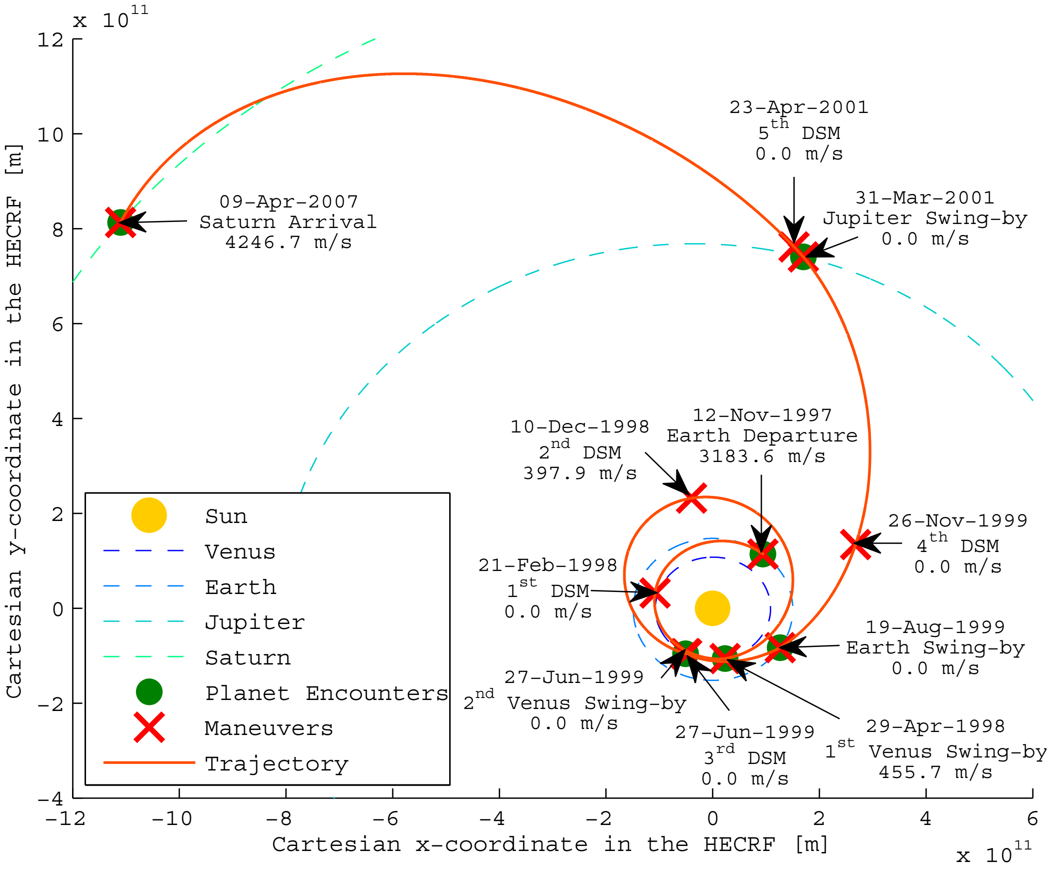


Figure 8.7: The best putative trajectory for the Cassini2 First Leg PF problem.

8.4.2 Cassini2 Last Leg Problems

For these trajectory problems all variables of the best putative GTOP solution of the Cassini Full PF and VF problems were assumed, except for the variables specific to the last leg. This again leaves five variables that are to be optimized by the optimization techniques.
The decision vector corresponding to these problems, along with the bounds and the best putative solution belonging to these problems, can be seen in Tables 8.15 and 8.16.

Table 8.15: The decision vector bounds and the best putative solution for the Cassini2 Last Leg PF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_5	800	2200	2199.96191	days
x(2)	η_5	0.01	0.90	0.010426	-
x(3)	r_5	0.8	2.5	1.022312	D_{pl}
x(4)	θ_5	0	2π	0.027977	rad
x(5)	ϕ_5	$-\pi/12$	$\pi/12$	0.000187	rad

The best putative solutions of 8383.2 m/s that were found for these problems are the same as the GTOP solution.

Table 8.16: The decision vector bounds and the best putative solution for the Cassini2 Last Leg VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_5	800	2200	2199.96191	days
x(2)	η_5	0.01	0.90	0.010426	-
x(3)	$b_{incl,5}$	$-\pi$	π	-1.513432	rad
x(4)	$r_{p,5}$	1.7	291	69.8134	R_{pl}
x(5)	ΔV_5	0	2000	0	m/s

8.4.3 Cassini2 First and Last Leg Problems

For these trajectory problems all variables of the best putative GTOP solution of the Cassini Full PF and VF problems were assumed, except for the variables specific for the first and the last leg. This leaves ten variables that are to be optimized by the optimization techniques.
The decision vector corresponding to these problems, along with the bounds and the best putative solution belonging to these problems, can be seen in Tables 8.17 and 8.18.

Table 8.17: The decision vector bounds and the best putative solution for the Cassini2 First and Last Leg PF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_1	100	400	167.93761	days
x(2)	T_5	800	2200	2199.96191	days
x(3)	η_1	0.01	0.90	0.604561	-
x(4)	r_1	0.5	1.5	0.761691	D_{pl}
x(5)	θ_1	0	2π	1.970464	rad
x(6)	ϕ_1	$-\pi/12$	$\pi/12$	0.030072	rad
x(7)	η_5	0.01	0.90	0.010426	-
x(8)	r_5	0.8	2.5	1.022312	D_{pl}
x(9)	θ_5	0	2π	0.027977	rad
x(10)	ϕ_5	$-\pi/12$	$\pi/12$	0.000187	rad

Table 8.18: The decision vector bounds and the best putative solution for the Cassini2 First and Last Leg VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_1	100	400	167.93761	days
x(2)	T_5	800	2200	2199.96191	days
x(3)	η_1	0.01	0.90	0.772878	-
x(4)	V_∞	3000	5000	3265.804	m/s
x(5)	θ	0	2π	3.320288	rad
x(6)	ϕ	$-\pi/2$	$\pi/2$	0.237444	rad
x(7)	η_5	0.01	0.90	0.010426	-
x(8)	$b_{incl,5}$	$-\pi$	π	-1.513432	rad
x(9)	$r_{p,5}$	1.7	291	69.8134	R_{pl}
x(10)	ΔV_5	0	2000	0	m/s

Similar to the Cassini2 First Leg problems, the PF variant found a better solution than the best putative GTOP solution. This solution is equal to the solution of 8283.8 m/s shown in Figure 8.7. Again the VF variant only came up with the best putative GTOP solution of 8383.2 m/s.

8.4.4 Cassini2 Last Two Leg Problems

For these trajectory problems all variables of the best putative GTOP solution of the Cassini Full PF and VF problems were assumed, except for the variables specific to the last two legs. This again leaves ten variables that are to be optimized by the optimization techniques.
The decision vector corresponding to these problems, along with the bounds and the best putative solution belonging to these problems, can be seen in Tables 8.19 and 8.20.

Table 8.19: The decision vector bounds and the best putative solution for the Cassini2 Last Two Legs PF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_4	400	1600	589.76790	days
x(2)	T_5	800	2200	2199.96191	days
x(3)	η_4	0.01	0.90	0.167389	-
x(4)	r_4	0.8	6.0	1.979967	D_{pl}
x(5)	θ_4	0	2π	1.046060	rad
x(6)	ϕ_4	$-\pi/12$	$\pi/12$	-0.008337	rad
x(7)	η_5	0.01	0.90	0.010426	-
x(8)	r_5	0.8	2.5	1.022312	D_{pl}
x(9)	θ_5	0	2π	0.027977	rad
x(10)	ϕ_5	$-\pi/12$	$\pi/12$	0.000187	rad

Table 8.20: The decision vector bounds and the best putative solution for the Cassini2 Last Two Legs VF problem.

Decision Vector Entry	Corresponding Variable	Lower Bound	Upper Bound	Best Putative Solution	Units
x(1)	T_4	400	1600	589.76790	days
x(2)	T_5	800	2200	2199.96191	days
x(3)	η_4	0.01	0.90	0.167389	-
x(4)	$b_{incl,4}$	$-\pi$	π	1.554796	rad
x(5)	$r_{p,4}$	1.15	6.5	1.30685	R_{pl}
x(6)	ΔV_4	0	2000	0	m/s
x(7)	η_5	0.01	0.90	0.010426	-
x(8)	$b_{incl,5}$	$-\pi$	π	-1.513432	rad
x(9)	$r_{p,5}$	1.7	291	69.8134	R_{pl}
x(10)	ΔV_5	0	2000	0	m/s

The best putative solutions that were found for these problems are the same as the GTOP solution of 8383.2 m/s.

8.5 Narrowed Bounds Problems

The narrowed bounds problems employ yet another way to simplify the problems. In these problems the bounds of all variables are reduced to a small region around the best putative solution. The basic idea is that this type of problems incorporates both the sensitivity of the actual problem, as well as the dependency of the variables of all legs. The optimization techniques are significantly aided in their search because of the reduced bounds, allowing for an optimization that is fast enough to allow it to be used in the tuning phase.

By the time these problems were set up, the author was aware of the location of the best putative solution to the Cassini2 problem for trajectory models including powered swing-bys. Hence these problems were set up around these solutions instead of the best putative GTOP solutions.

Two variants of the narrowed bounds were written. A first one in which the bounds of all variables were reduced to 10 % of the search space around the best putative solution. Subsequently a second one in which the bounds were reduced to only 1 % of the search space around the best putative solution. This reduces the search space to only 10^{-26} and 10^{-52} of the original search space, because the problem consists of 26 variables. This large reduction in search space was required to allow these problems to be used in the tuning phase.

These problems are further referred to as the Cassini2 Narrowed Bounds 0.1 PF/VF and Cassini2 Narrowed Bounds 0.01 PF/VF problems.

8.6 Overview of Tuning Problems

In total 19 tuning problems have been set up. Seven of them simplify the trajectory model to allow them to be optimized easily, eight of them optimize a subset of the legs of the entire problem, and four of them have significantly reduced bounds. An overview of all problems is given in Table 8.21.

Table 8.21: Overview of the tuning problems.

		# of Legs		Best Putative
Problem Name	Type	MGA / DSM	# of vars	Solution (m/s)
EJS MGA	Model simplification	2 / 0	3	9355.2
EMJS MGA	Model simplification	3 / 0	4	9089.2
EVEJS MGA	Model simplification	4 / 0	5	7014.4
Cassini1 MGA	Model simplification	5 / 0	6	4930.7
Cassini2 MGA	Model simplification	5 / 0	6	9687.5
Cassini2 DSM In Second Leg PF	Model simplification	4 / 1	10	8237.9
Cassini2 DSM In Second Leg VF	Model simplification	4 / 1	10	8237.9
Cassini2 First Leg PF	Specific Legs	0 / 1 of 5	5	8283.8
Cassini2 First Leg VF	Specific Legs	0 / 1 of 5	5	8383.2
Cassini2 Last Leg PF	Specific Legs	0 / 1 of 5	5	8383.2
Cassini2 Last Leg VF	Specific Legs	0 / 1 of 5	5	8383.2
Cassini2 First And Last Leg PF	Specific Legs	0 / 2 of 5	10	8283.8
Cassini2 First And Last Leg VF	Specific Legs	0 / 2 of 5	10	8383.2
Cassini2 Last Two Legs PF	Specific Legs	0 / 2 of 5	10	8383.2
Cassini2 Last Two Legs VF	Specific Legs	0 / 2 of 5	10	8383.2
Cassini2 Narrowed 0.01 PF	Narrowed bounds	0 / 5	26	8237.9
Cassini2 Narrowed 0.01 VF	Narrowed bounds	0 / 5	26	8237.9
Cassini2 Narrowed 0.1 PF	Narrowed bounds	0 / 5	26	8237.9
Cassini2 Narrowed 0.1 VF	Narrowed bounds	0 / 5	26	8237.9

Chapter 9

Tuning Differential Evolution

This chapter discusses the tuning process that was used for DE. Section 9.1 describes the version of DE that is implemented in PaGMO. Subsequently the tuning process is described. Due to the large number of options, this tuning process is split up into three phases. The first phase intends to prune settings that perform bad from the total set of options. The second phase is then used to find the settings that appear to perform best. Finally a third tuning phase was used to determine the ideal population size. Section 9.2 discusses the first phase, Section 9.3 discusses the second phase and Section 9.4 discusses the third phase. Finally conclusions are drawn in Section 9.5.

9.1 DE in PaGMO

This thesis uses the initial DE algorithm that was present in PaGMO as of March 2012. During the course of the thesis project, the author found out that newer versions had also been implemented. These have not been tested anymore though.

The DE algorithm has four main tuning parameters. These are the scale factor (F), the cross-over rate (CR), the population size (n_{pop}) and the strategy ($strat$). The scale factor determines the amplification of the difference vector that is formed. In PaGMO the domain of F is limited to [0,1]. The cross-over rate determines the chance of a variable to be included in the mutant vector and its domain is limited to [0,1]. The population size obviously determines the number of individuals that are included. A population of at least 6 individuals is required by PaGMO.
The strategy requires a more thorough explanation as it determines the exact way in which the mutant vector is formed. Typically the following notation is used in literature DE/x/y/z. In here x determines whether the mutant vector will be added to a random individual (x =rand) or the best individual (x =best) in the search for a better solution. Sometimes these strategies are also referred to as exploratory (rand) and convergence (best) strategies. The y subsequently determines the number of difference vectors used. Finally z determines the cross-over scheme that is used. A binomial scheme performs independent binomial experiments for each parameter. The exponential scheme starts with a random parameter and then continues including more parameters until the random value r is lower than CR. In this way a chain of subsequent variables is obtained that form the mutant vector. [Storn and Price, 1997]

The first five strategies in PaGMO are considered here. An overview of the corresponding strategies is given in Table 9.1.

It is noted that only exponential cross-over schemes are included. The fact that PaGMO also includes binomial cross-over schemes was only found out when the tuning was already done. Hence the binomial cross-over schemes are not tuned in this thesis anymore and are left for future students to investigate. Compared to other effects, Storn and Price [1997] note that the effect of the cross-over scheme is likely much smaller than the other differences between strategies and tuning parameters. It is noted explicitly as a warning that the optimal values for CR are typically very different between both schemes, especially

Table 9.1: Overview of the DE strategies that will be investigated in this thesis.

Strategy Number	Notation in Literature	Mutation vector scheme
1	DE/best/1/exp	$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1} - x_{r2})$
2	DE/rand/1/exp	$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2} - x_{r3})$
3	DE/rand-to-best/1/exp	$v_{i,G+1} = x_{i,G} + F \cdot (x_{best,G} - x_{i,G}) + F \cdot (x_{r1} - x_{r2})$
4	DE/best/2/exp	$v_{i,G+1} = x_{best,G} + F \cdot (x_{r1} + x_{r2} - x_{r3} - x_{r4})$
5	DE/rand/2/exp	$v_{i,G+1} = x_{r1,G} + F \cdot (x_{r2} + x_{r3} - x_{r4} - x_{r5})$

when optimizing problems of higher dimension.

9.2 First Tuning Phase

The following seven problems were selected for the initial tuning phase because they were deemed to be the least complex problems:

- Trajectory model simplification problems:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA.
- Specific leg problems:
  - Cassini2 First Leg PF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF.

All problems were optimized with a maximum of 20000 evaluations. To help identify the good regions of performance, two different approaches were used:

- Grid search: the four main tuning parameters are discretized with five values per parameter. This yields 625 settings of which the performance will be measured. The values that were taken into account are given in Table 9.2.
- Random search: random values were taken within a certain range for all four main tuning parameters of DE. This was repeated 500 times. The corresponding ranges can be seen in Table 9.2.

Table 9.2: The set/range of values that was used in the first DE tuning phase.

Parameter	Values used in grid	Range for random samples
F	0.2, 0.4, 0.6, 0.8, 1.0	0.0 - 1.0
CR	0.2, 0.4, 0.6, 0.8, 1.0	0.0 - 1.0
n_p	6, 10, 20, 40, 80	6 - 80
str	1, 2, 3, 4, 5	1 - 5

First the results of the grid search will be discussed in Section 9.2.1. Subsequently the random search will be discussed in Section 9.2.2. Finally the conclusions of the first tuning phase are presented in Section 9.2.3.

9.2.1 Grid Search

The full results of the grid search are not displayed here because of the size it would require. Instead they can be found in Appendix A. As a typical example, the results of the Cassini2 Last Leg PF problem are given in Figure 9.1.

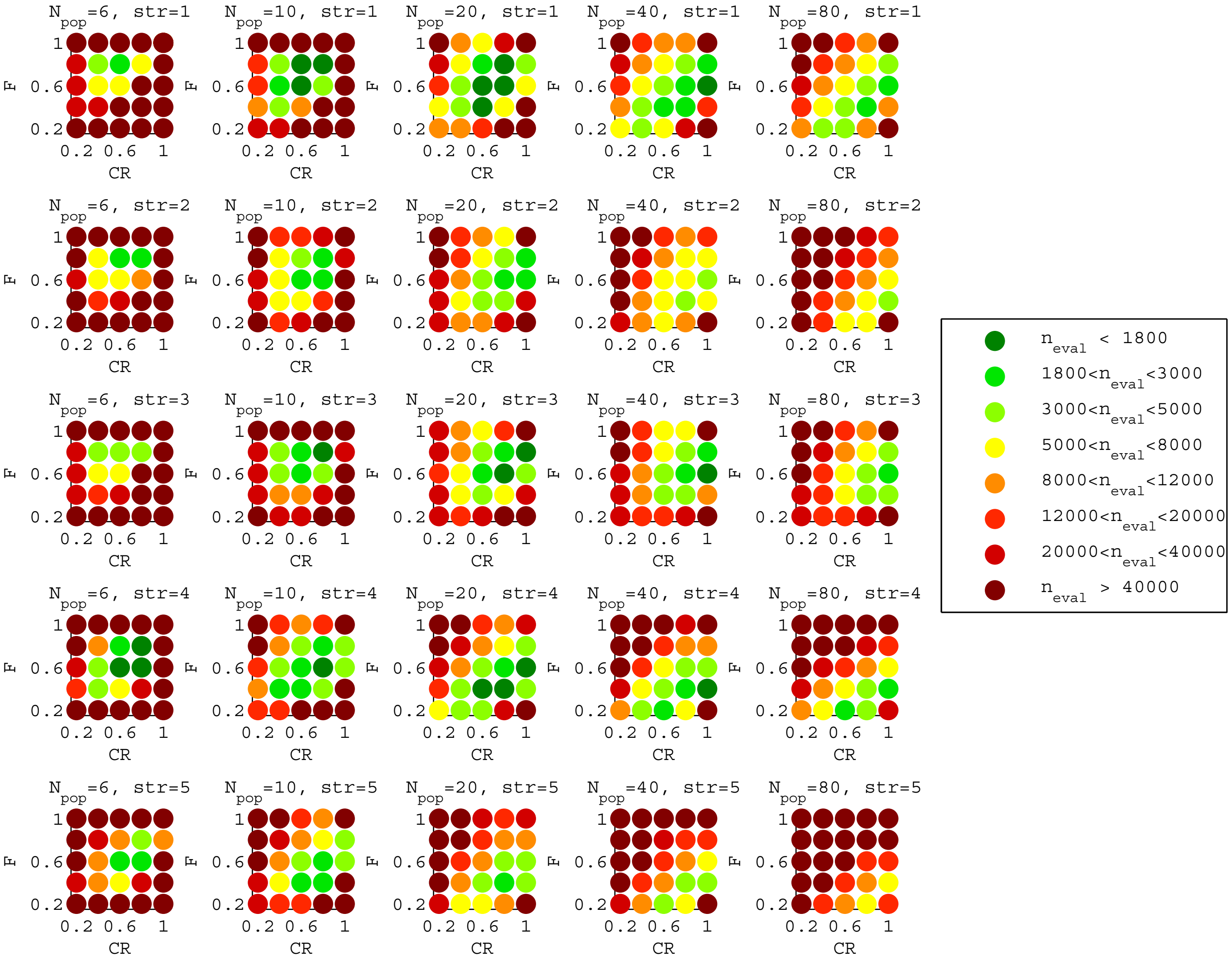


Figure 9.1: The results of the first phase tuning process of DE on the Cassini2 Last leg PF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

One of the first conclusions that can be drawn from the results is that DE is indeed very sensitive to its tuning parameters. Performance sometimes changes an order of magnitude or more if only one parameter is changed by one step. Still at the same time the plots do seem to indicate regions that typically have a better performance than other regions.
The big challenge is now to condense all this information to define the best performing regions in these plots. There is no perfect way to do so, taking into account all problem differences, parameter dependencies and other imperfections. The following paragraph describes the heuristic that was used.

First of all the population size tuning was left out for now. The population size is expected to be the most sensitive to problem scaling and hence expected to be the least stable across different problems. Although it is also very sensitive, the sensitivity of the results to the population size appeared to be less than that of other parameters. As can be seen in Figure 9.1, multiple population size entries typically perform well and are colored green for good settings of F, CR and $strat$. Note that each different entry corresponds to doubling the population size. In contrast, a change of 0.2 in F or CR often results in a significant change of performance. Hence it was decided to condense the information of the different population sizes into one. This is done by simply taking the best performing population size for each combination of the other tuning parameters. The third phase will determine the ideal population size.

Next a scoring mechanism was defined. Since the main interest is in the regions performing the best, only well-performing regions will be assigned points. Each setting of F, CR and str was compared to the performance of the best sample. A score is then assigned to each setting on a scale from 0 to 1, where 1 corresponds to the performance of the best sample and 0 to three times the number of evaluations of the best sample. Samples with larger number of evaluations are also given a score of 0. An overview of this scoring mechanism is given in Algorithm 9.1 below.

Algorithm 9.1: Scoring mechanism for computing performance of different tuning settings belonging to a certain algorithm. Note that n is written here for $n_{95\%}$.

```
Determine best setting $x_{best}$ and corresponding best number of evaluation $n_{best}$
Set $n_{threshold} = 3 \cdot n_{best}$
for $i = 1 : m_{settings}$
    if $n_i < n_{threshold}$
        Set score: $s_i = \frac{n_{threshold} - n_i}{n_{threshold} - n_{best}}$
    else
        Set score: $s_i = 0$
    end if
end for
```

This was done for all problems, yielding matrices with performances relative to the best solution for each problem.

Figures 9.2 and 9.3 show the results of the previous heuristic for the MGA problems and the MGA-1DSM problems respectively.

One remark has to be made on the results in Figure 9.2 and Figure 9.3. The experiments were namely only conducted with a maximum of 20,000 function evaluations. This poses no problems in applying Algorithm 9.1 for five out of the seven problems. The best solutions of these problems namely require 2000 evaluations or less. The EVEJS MGA and the Cassini2 First Leg VF problems require around 8000 evaluations though. Algorithm 9.1 is still applied, even though a fully accurate distinction between 20,000 and 24,000 can not be made reliably. Algorithms that converge in this range may namely be given a lower score than would have been the case if 24,000 evaluations were performed. This effect is deemed relatively small, because of the low score that would be assigned to these samples anyway. Also it is noted that most simulations of the EVEJS MGA problem had already converged, but to different minima. In hindsight the simulations should have used a larger number of evaluations of course. This will be done in phase 2 of the tuning process for a selection of the tuning parameters.

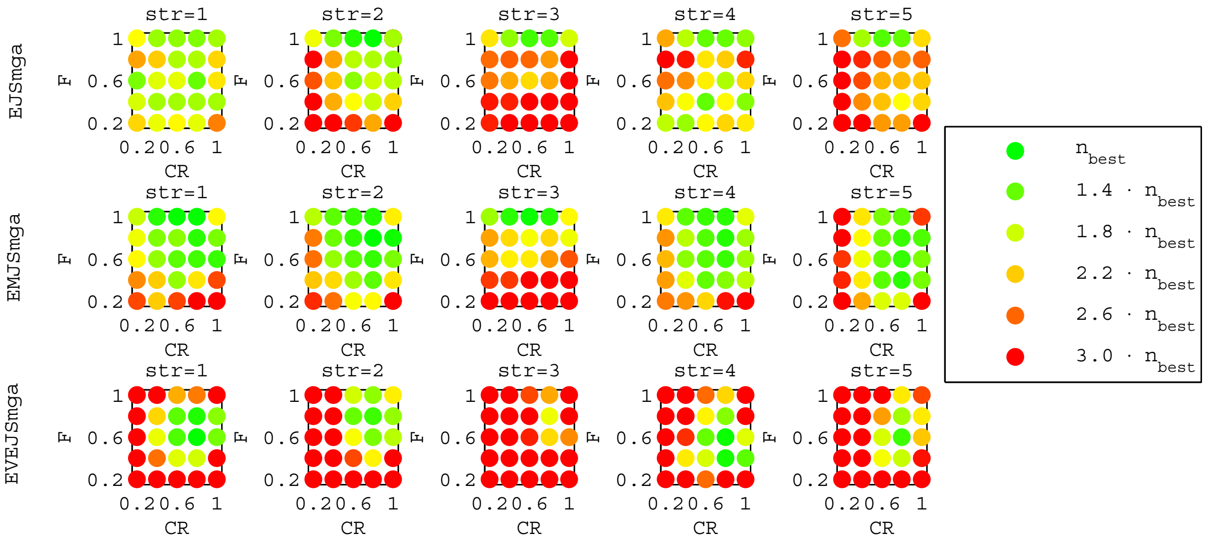


Figure 9.2: The condensed results of the first phase grid search for the MGA problems. The results are given in a continuous way with scores ranging from the best number of evaluations for that problem (green) to 3 times the best number or more(red). See text for explanation.

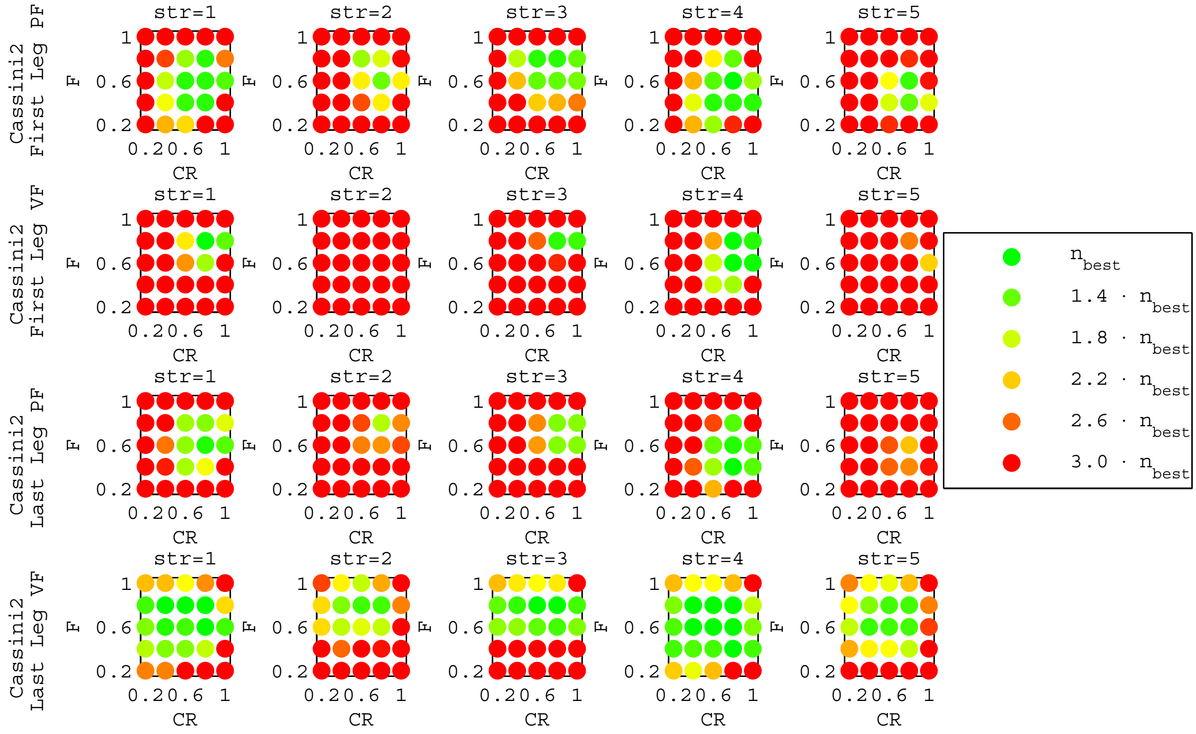


Figure 9.3: The condensed results of the first phase grid search for the MGA problems. The results are given in a continuous way with scores ranging from the best number of evaluations for that problem (green) to 3 times the best number or more(red). See text for explanation.

The next step is to combine the results of the different problems in one picture. To do so the average performance is calculated for the trajectory model simplification problems and the specific leg problems separately. Subsequently the average overall performance is computed by taking the average of these average performances. The specific leg performance is weighed double in calculating this final average. These problems are deemed to represent the search space of the final problems the best because of the fact that no DSMs are present in the trajectory simplification problems. Also the specific leg problems share the same sensitivity as the final problems. The results of this step can be seen in Figure 9.4.

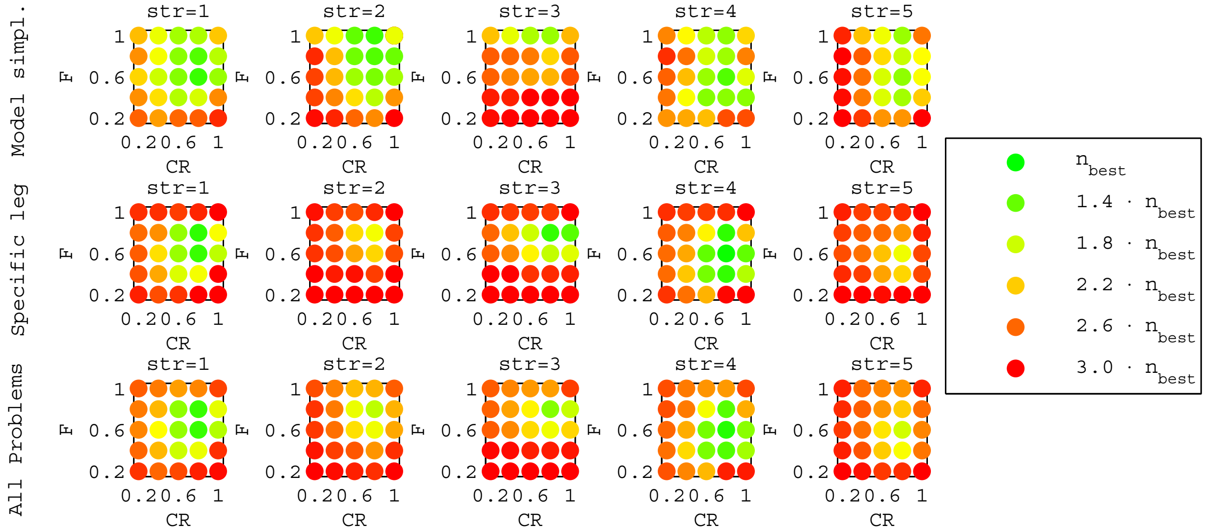


Figure 9.4: The average results of the first phase grid search. Note that the first row contains the condensed information of Figure 9.2 and the second row that of Figure 9.3. See text for further explanation.

This picture seems to clearly indicate the most promising regions in the F and CR spectrum for different strategies. Also it appears that strategy 1 and 4 are the most promising strategies and strategy 2 and 5 are the least promising. Before selecting the regimes for the second tuning phase, the random search results are analyzed first.

9.2.2 Random Search

Also for the random search the full results are not displayed here because of the size it would require. Instead they can be found in Appendix A. As a typical example, the results of the Cassini2 Last Leg PF problem are given here in Figure 9.5. It is noted explicitly that only the best 100 samples of a total of 500 samples are depicted in this plot.
To further analyze the results, they are once again processed. First the four different tuning parameters are discretized. Secondly all points are evaluated and scored using Algorithm 9.1. These scores are subsequently grouped based on the regime of the tuning parameter to which they belong. This gives scores to the different tuning parameter regimes. The fraction of weighed best solutions that is present in each regime is finally obtained by dividing by the total score.

This was done for all problems and all tuning parameters, resulting in Figures 9.6 to 9.9. Note that the population size is also analyzed here. Although it will be left out until Section 9.4 it is important that the next phase utilizes reasonably suited values for the population size. Hence it is also included here to get an estimate of the best population size.

Figure 9.6 seems to indicate that the ideal population size is slightly below 20.

Figure 9.7 seems to indicate that F should be chosen larger than 0.4, but does not help in distinguishing in the regime from 0.4 to 1.0.

Figure 9.8 indicates that the CR should be chosen larger than 0.6.

Figure 9.9 seems to indicate that strategy 1 is the best choice, followed by strategy 4. Strategy 5 seems to perform worst. The difference is not very clear on all problems though.

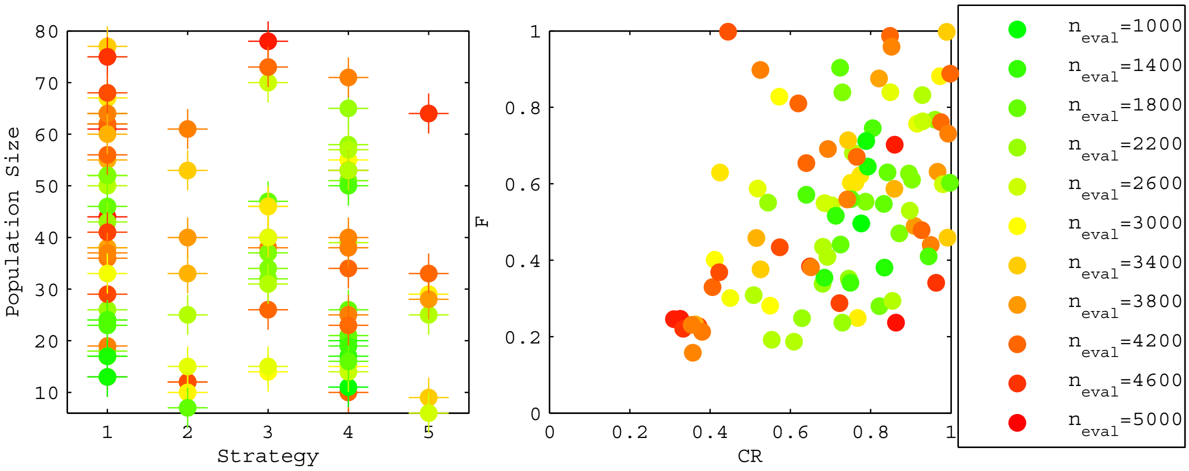


Figure 9.5: The results of the first phase tuning process of DE on the Cassini2 Last leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum ($n_{95\%}$ or n_{eval}). The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected.

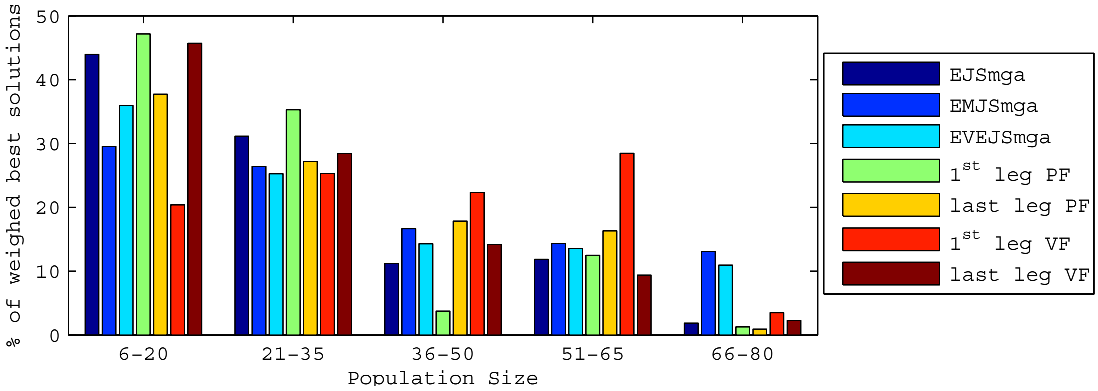


Figure 9.6: The distribution of good results over different regimes of population sizes. See text for explanation.

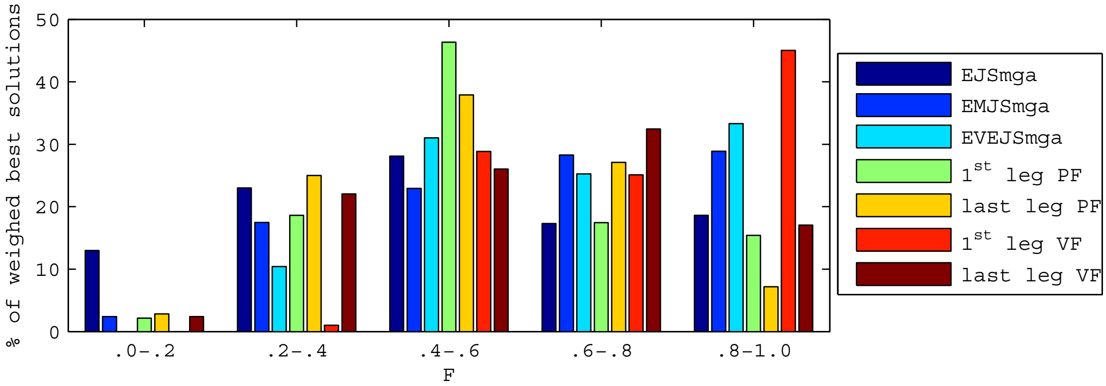


Figure 9.7: The distribution of good results over different regimes of F. See text for explanation.

Apart from this, also the effects of the different tuning parameters was analyzed for each strategy. It can

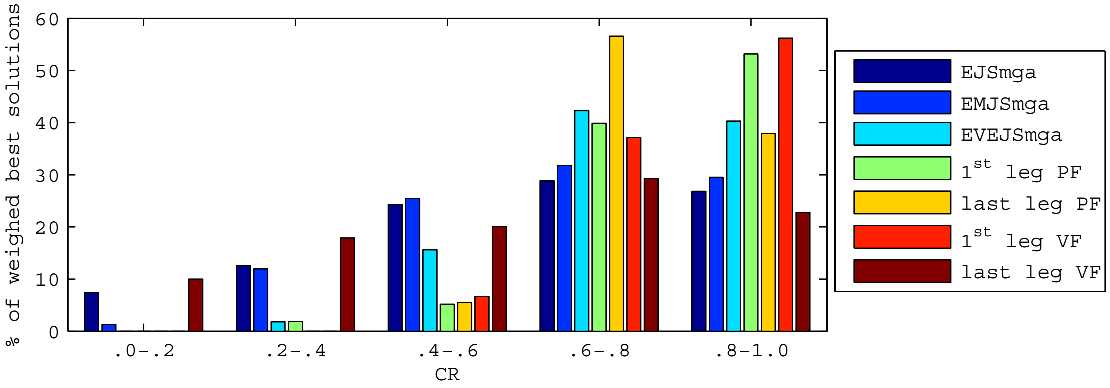


Figure 9.8: The distribution of good results over different regimes of CR. See text for explanation.

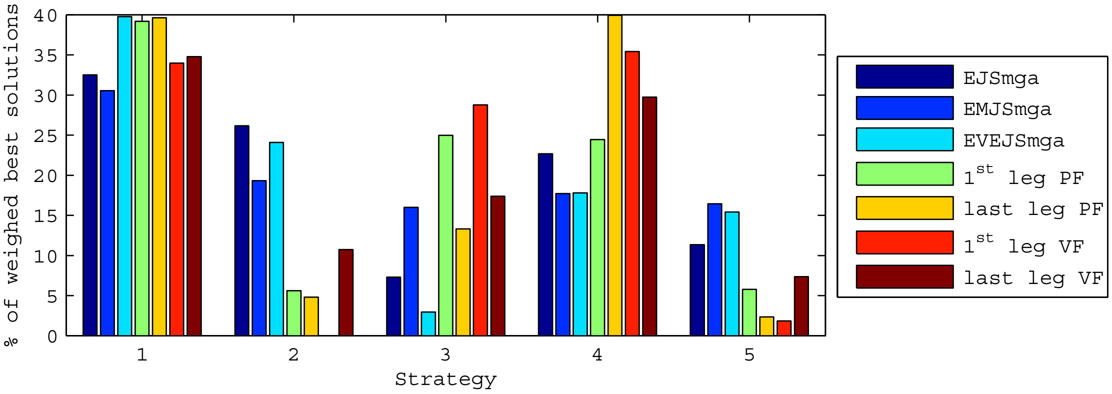


Figure 9.9: The distribution of good results over the different strategies. See text for explanation.

namely be expected that different regimes perform best for the different strategies, as is also indicated by the results of the grid search. To do so a Algorithm 9.1 was applied again, but now treating each strategy as a separate algorithm. Subsequently the results of the trajectory model simplification and the specific legs problems are averaged out. Finally these results are added, again weighing the specific leg problems twice.
This results in an average performance of different tuning parameter regimes for each strategy. The results for the population size can be seen in Figure 9.10, the results for F can be seen in Figure 9.11 and finally the results for CR can be seen in Figure 9.12. It has to be noted that for strategy 3, the EVEJS MGA problem was not considered and for strategies 2 and 5 the Cassini2 First Leg VF problem was not considered. This was due to the fact that the results would lie for a considerable part in the spectrum above 20.000 evaluations, which was not considered accurate anymore.

It can be seen in Figure 9.10 that the regime with population size between 6 and 20 is most favourable for most strategies. Only strategy 3 favours a slightly larger population. Given that the difference with the 21-35 regime is not very big, it is assumed that a population size of 15-20 would in general be optimal. This result is also considerably stable along the different problems, as can be seen in Figure 9.6. This conclusion is also supported by the results of the grid search, which can be seen in Appendix A.

It can be seen in Figure 9.11 that the good regime for F is strongly dependent on the strategy that is chosen. Strategies 4 and 5 favour a much lower F than the other three strategies. This result can also be seen in the results of the grid search, although the difference was smaller in the grid search.
The fact that optimal regimes are so different means that it is important to define separate tuning regimes for each strategy.

As can be seen in Figure 9.12, the difference between good CR values is much smaller between different strategies. The good regime appears to be in between 0.6 and 1.0 for all strategies. This is also in

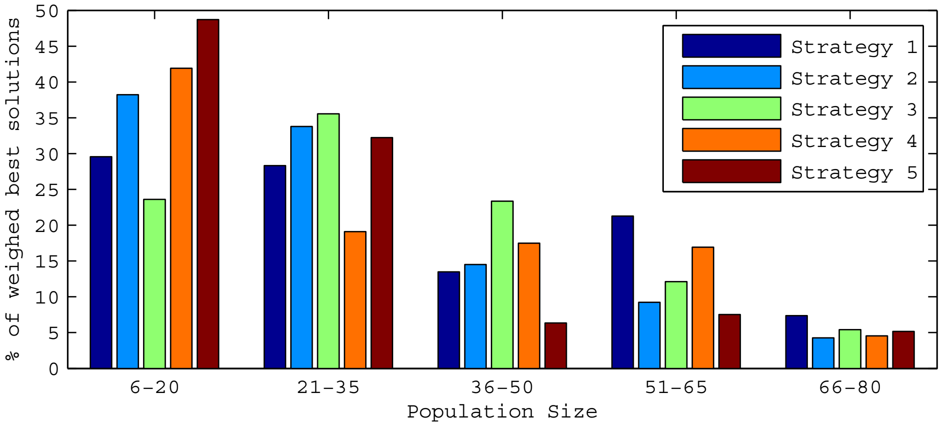


Figure 9.10: The distribution of good results over different regimes of population sizes for the different strategies. See text for explanation.

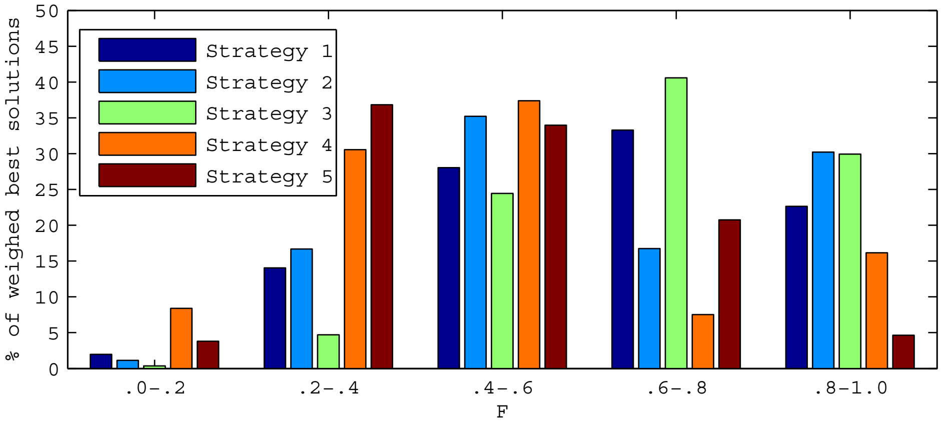


Figure 9.11: The distribution of good results over different regimes of F for the different strategies. See text for explanation.

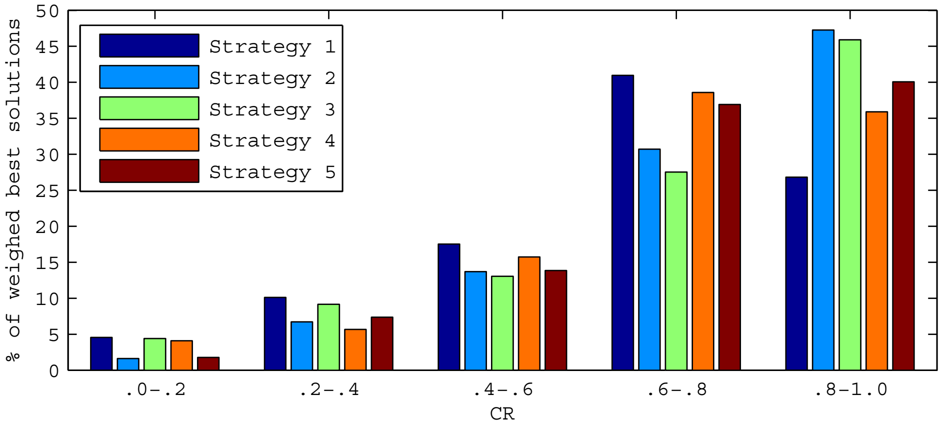


Figure 9.12: The distribution of good results over different regimes of CR for the different strategies. See text for explanation.

accordance with the results of the grid search.

9.2.3 Conclusions First Tuning Phase

The following conclusions are drawn based on the results of the previous sections to define the new tuning regimes:

- To help in analyzing the results, the population size was left out for now. It is likely to be very problem-dependent and leaving it out allows to sample other variables more accurately. Two population sizes will be considered for phase 2. The population size will be tuned separately afterwards.
- The ideal population size appeared to be slightly below 20. The problem difficulty of the second phase will probably be double that of the first phase, as the number of parameters is doubled for most problems. Hence populations of 20 and 60 are considered to take that into account.
- Tuning regimes are defined separately for each strategy because especially the ideal regime for F depends on the strategy used.
- The grid search provided good insight in the performance regimes. The results were still very sensitive to the steps in F and CR. Hence a smaller step size of 0.1 will be used in the grid for the next phase.
- The performance of different strategies differed significantly. This trend was considered to be too problem-dependent to prune strategies already. However a larger tuning regime will be assigned to the strategies that performed well, as these are most promising. This applies to strategy 1 especially and to strategy 4 as well.
- The tuning regimes that were defined in this way can be seen in Table 9.3.

Table 9.3: The set of values that will be used in the second DE tuning phase. Population sizes of 20 and 60 will be used.

Strategy	Values of F used in grid	Values of CR used in grid
1	0.5, 0.6, 0.7, 0.8, 0.9	0.5, 0.6, 0.7, 0.8, 0.9
2	0.5, 0.6, 0.7, 0.8, 0.9	0.7, 0.8, 0.9
3	0.6, 0.7, 0.8, 0.9	0.7, 0.8, 0.9
4	0.3, 0.4, 0.5, 0.6, 0.7	0.6, 0.7, 0.8, 0.9
5	0.3, 0.4, 0.5, 0.6	0.7, 0.8, 0.9

9.3 Second Tuning Phase

Thirteen problems are considered for the second phase. An overview of these problems, including the number of evaluations they were run for, is given in Table 9.4.

This time the required number of evaluations was determined using preliminary runs. The number evaluations they were run for is significantly larger than the best $n_{95\%}$. One exception to this is the Cassini2 MGA problem. The main reason is that this problem is only optimized using a very low success rate. Hence instead the simulations were run for a number of evaluations that was significantly larger than the number of evaluations after which the preliminary runs had all converged. This was 10^5 for all strategies except strategy 3, which was run for $4 \cdot 10^5$. The Cassini2 MGA problem was also run 1000 times instead of 200 to cope with the low success rate. The Cassini1 problem was not included in this phase because of the large computational effort to do so and the fact that the problem is very similar to the Cassini2 MGA problem.
Furthermore it is noted that the Cassini2 First Leg VF problem and the EVEJS MGA problem were already present in the first phase tuning. They are included again because they were found to be very difficult to optimize in this first phase.

In this second phase a grid search was performed, using the parameters listed in Table 9.3. The simulations were performed for two different population sizes. The best population size was selected for each sample in the results presented in this section. First the results of the parameter reduction problems are

Table 9.4: The problems included in the second tuning phase of DE, along with the maximum number of evaluations they were run for and the resulting best $n_{95\%}$. Note that all problems were run for much more evaluations than the best $n_{95\%}$, meaning that the results can be used to differentiate between a large spectrum of different performances.

Problem	Problem Type	Maximum number of evaluations [10^3]	Resulting best $n_{95\%}$ [10^3]
Cassini2 First Leg VF	Specific legs	80	8.2
Cassini2 First And Last Leg PF	Specific legs	40	3.4
Cassini2 First And Last Leg VF	Specific legs	120	20
Cassini2 Last Two Legs PF	Specific legs	120	10.0
Cassini2 Last Two Legs VF	Specific legs	40	3.3
EVEJS MGA	Model simplification	80	7.6
Cassini2 MGA	Model simplification	100/400 (*)	140
Cassini2 DSM In Second Leg PF	Model simplification	600	73
Cassini2 DSM In Second Leg VF	Model simplification	150	27
Cassini2 Narrowed 0.01 PF	Narrowed bounds	100	8.8
Cassini2 Narrowed 0.01 VF	Narrowed bounds	100	4.0
Cassini2 Narrowed 0.1 PF	Narrowed bounds	300	116
Cassini2 Narrowed 0.1 VF	Narrowed bounds	200	57

given, subsequently those of the trajectory model simplification problems and finally those of the bound narrowing problems.

The results of the parameter reduction problems can be seen in Figure 9.13. Note that all results were scaled to the best result on that particular problem.

Figure 9.13 shows that strategy 4 performs best, followed by strategies 1 and 3. The region around F=0.5/0.6, CR=0.8/0.9 seems good for strategy 4 and the region around F=0.7/0.8 and CR=0.8/0.9 is a good region for strategy 1. The best region is slightly less stable for strategy 3. Strategies 2 and 5 seem less suited for this type of problems.

The results of the model simplification problems can be seen in Figure 9.14. Note that all results were scaled to the best result on that particular problem.

In this case the pattern is much less consistent. Strategy 4 shows reasonably stable performance around F=0.5-0.6 and CR=0.8-0.9 and strategy 1 also performs reasonably consistent at F = 0.7-0.9 and CR =0.8-0.9. Strategy 3 does not show good performance. It is especially interesting that strategy 2 suddenly scores best on the Cassini2 DSM In Second Leg Only problems. This is especially important as these problems likely resemble the final problems the most.

The results of the bound-narrowing problems can be seen in Figure 9.15. Again all results were scaled to the best result on that particular problem.

For the bound-narrowing problems a clear trend towards a high CR is visible. This leads to an assumption that the current maximum CR of 0.9 may not be high enough. Also it is very interesting to note that strategy 3 performs exceptionally well at this class of problems. Also strategy 4 performs well on this problem, followed by strategies 2, 5 and 1. It is noted especially that the tuning regimes that perform well are exceptionally small for these problems.

The performances are again calculated using Algorithm 9.1. These scores are then averaged for the three different classes of problems. Subsequently the average of the performance for these different classes is calculated. Problems and problem classes were given equal weight.

Figure 9.16 gives an overview of the results obtained in this way.

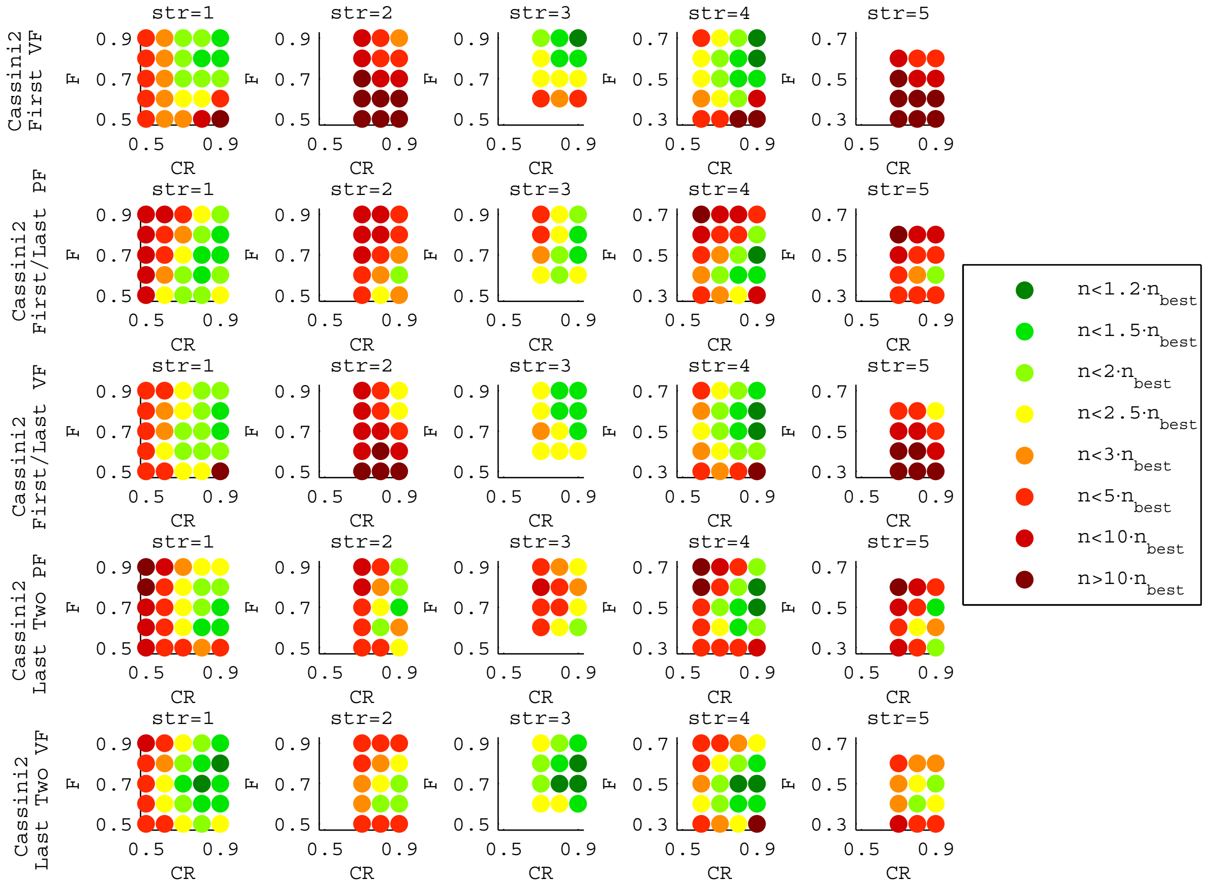


Figure 9.13: The performance ($n_{95\%}$ or n) of different settings of DE on the parameter reduction problems in the second tuning phase of the DE. See text for explanation.

From this figure, one point clearly performs best on average. Strategy 4, F=0.5, CR=0.9 has a score that is far better than all other combinations considered. This setting only requires 35% more evaluations on average than the best-performing setting on the specific problem. The worst performance of this setting required 97% more evaluations than the best-performing setting.

Because the performance on the actual problems may be slightly different, some other settings were also selected. Strategy 1, F=0.7, CR=0.9 is also selected, because it is the second best setting. It is also the only other setting that has found the solution within 3 times the number of evaluations of the best setting for all problems.

Furthermore it is noted that strategy 3 outperformed the other strategies in the narrowed bound problems and that strategy 2 clearly outperformed the other strategies in the Cassini2 DSM In Second Leg Only problems. Hence also these strategies were chosen for the final optimization, both with F=0.7 and CR=0.9.

Finally it is noted that the best settings are all at the upper limit of the values of CR that were tested. It may be so that a higher CR results in better performance. Hence the four best settings were analyzed again, but now with a CR of 0.94, 0.97 and 1.00. The results can be seen in Table 9.5. The scores in this table represent the performance with respect to the best setting for a specific problem, as calculated with Algorithm 9.1. Scores are given on a scale from 0% to 100%, where as usual 100% corresponds to the best setting and 0% to settings requiring 3 times the number of evaluations of the best, or more.

It is noted that the value for CR does not appear to have a strong correlation with the problem type. Many variations are visible though. The results of Table 9.5 allow to define a winner for all strategies.

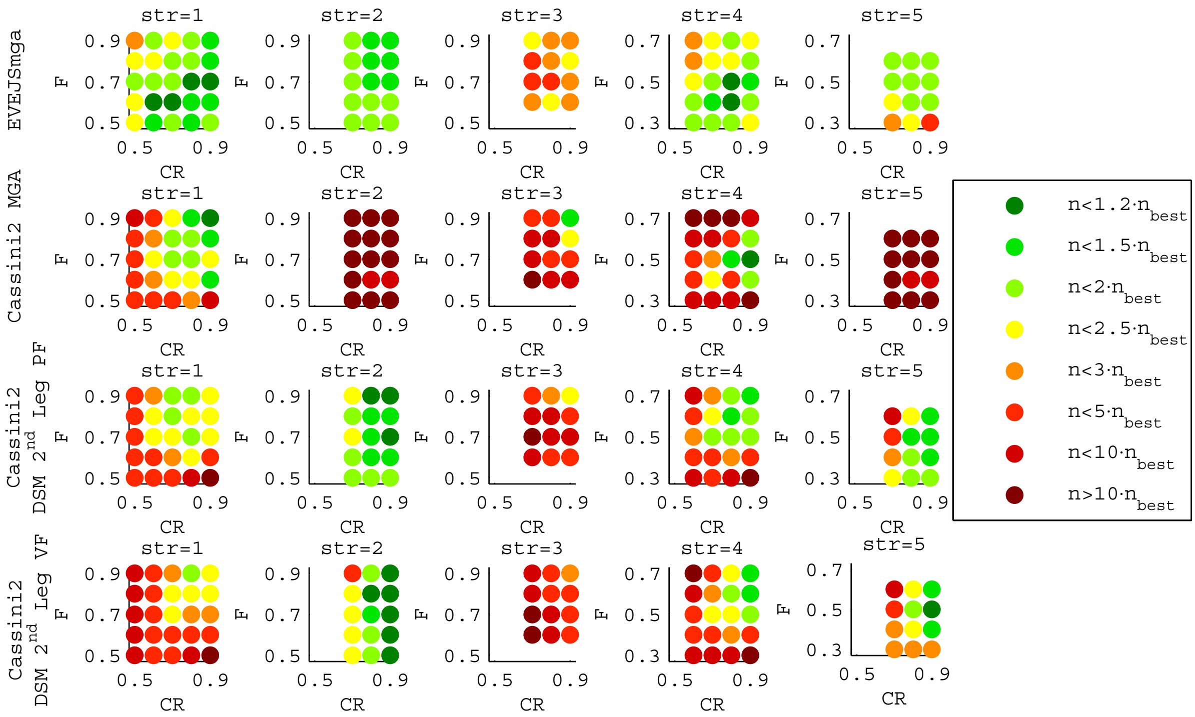


Figure 9.14: The performance ($n_{95\%}$ or n) of different settings of DE on the model simplification problems in the second tuning phase of the DE. See text for explanation.

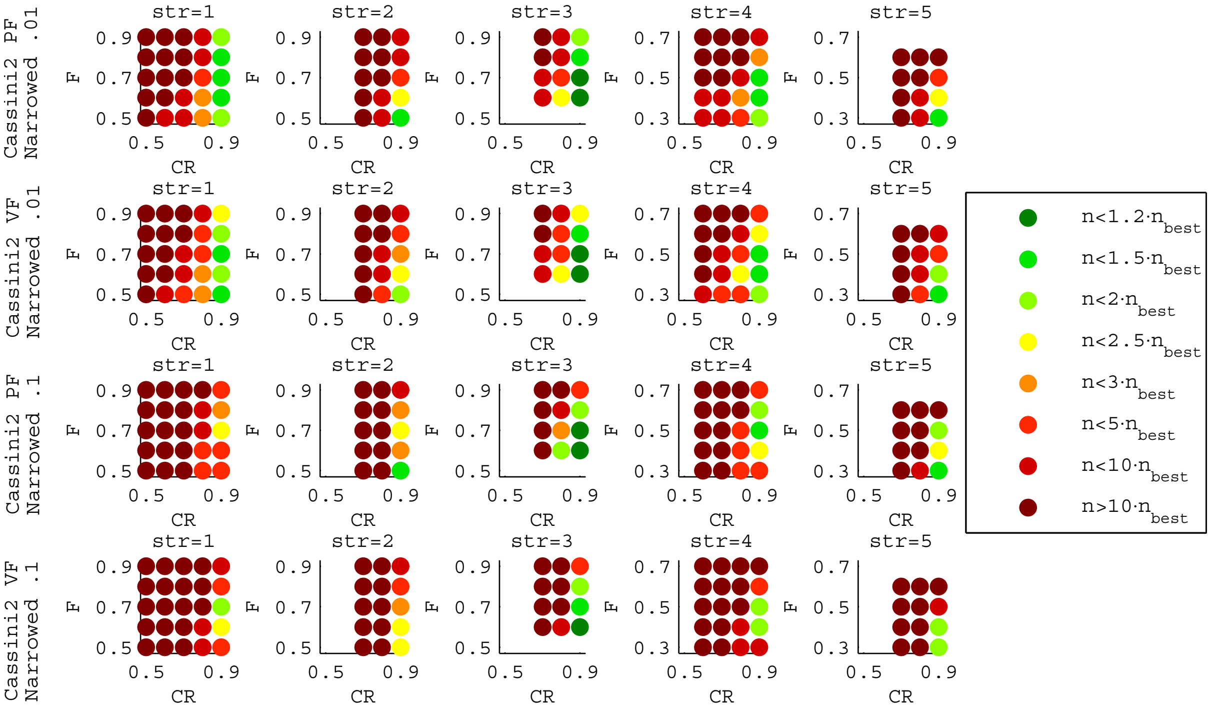


Figure 9.15: The performance ($n_{95\%}$ or n) of different settings of DE on the bound narrowing problems in the second tuning phase of the DE. See text for explanation.

Strategy 4 performs best at a CR of 0.94, which now is the best overall performing setting. Strategy 1 performs best at a CR of 0.90. A CR of 0.97 performs best for strategy 2. A CR of 0.9 is best for strategy 3, both overall and for the problems with narrowed bounds. From now on these are the settings

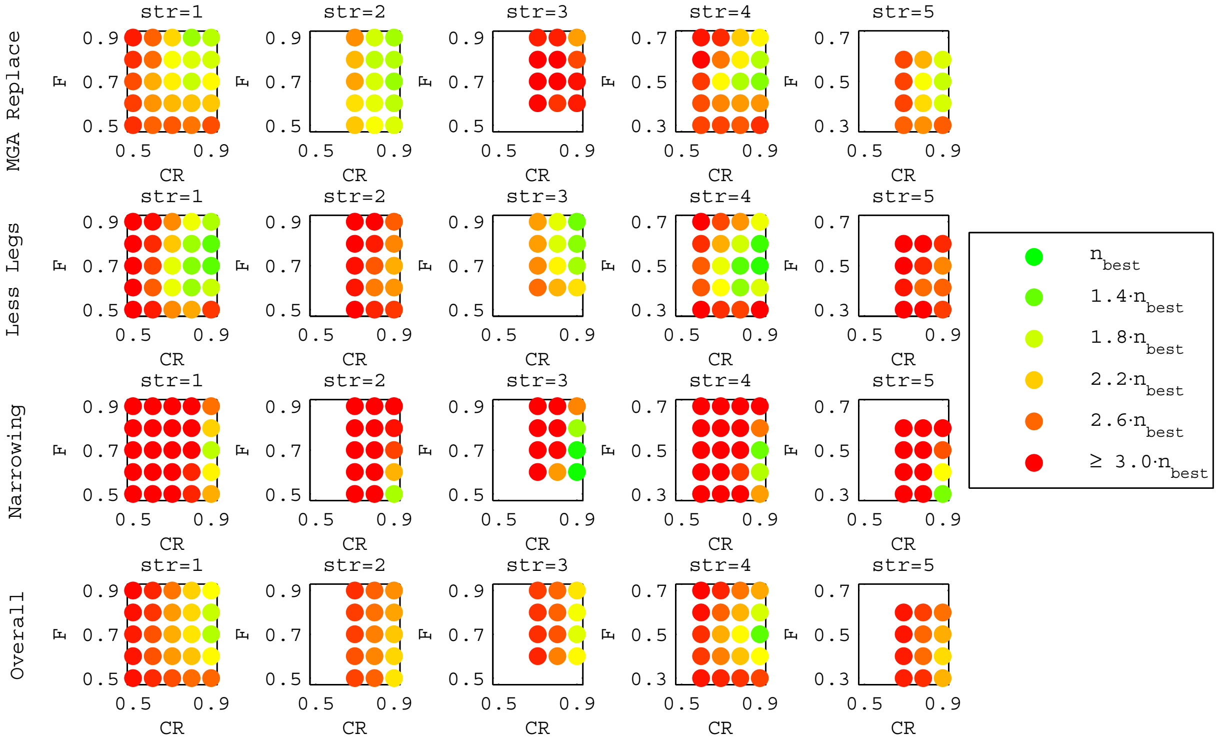


Figure 9.16: The average performance of different settings of DE on the problems in the second tuning phase of the DE. See text for explanation.

Table 9.5: The scores of various high CR values calculated using Algorithm 9.1 taking into account all values in the second phase. See text for explanation.

	Strategy 1				Strategy 2				Strategy 3				Strategy 4			
CR Problem	.9	.94	.97	1.0	.9	.94	.97	1.0	.9	.94	.97	1.0	.9	.94	.97	1.0
EVEJS MGA	94	93	69	53	87	78	83	56	16	46	27	29	83	83	100	89
Cassini2 MGA	48	51	53	93	0	0	0	0	0	0	0	0	100	99	71	48
DSM 2^{nd} PF	32	31	0	0	99	90	100	96	0	44	0	0	63	63	36	0
DSM 2^{nd} VF	23	34	0	0	95	84	92	98	0	19	0	0	52	57	53	0
MGA replace	**49**	**52**	**30**	**37**	**70**	**63**	**69**	**63**	**4**	**27**	**7**	**7**	**74**	**75**	**64**	**34**
1^{st} Leg VF	74	63	17	0	0	0	0	0	44	0	0	0	77	73	54	0
1^{st} + Last PF	84	70	71	25	21	44	59	0	82	62	0	0	100	85	56	74
1^{st} + Last VF	85	81	77	0	0	0	0	0	78	0	0	0	91	85	83	0
Last Two PF	66	92	87	38	73	71	28	0	23	0	0	0	81	97	100	98
Last Two VF	88	65	74	47	62	65	50	0	93	61	0	0	100	94	59	71
Less Legs	**80**	**74**	**65**	**22**	**31**	**36**	**27**	**0**	**64**	**25**	**0**	**0**	**90**	**87**	**70**	**49**
Narrow PF .01	84	90	87	0	0	60	92	0	98	98	64	0	79	99	88	0
Narrow VF .01	69	58	74	39	0	41	81	0	89	96	76	0	71	100	91	93
Narrow PF .1	32	27	17	0	46	21	99	0	99	0	0	0	79	84	76	0
Narrow VF .1	22	19	3	0	0	59	30	0	66	0	0	0	48	65	82	0
Narrowing	**52**	**48**	**45**	**10**	**11**	**45**	**76**	**0**	**88**	**48**	**35**	**0**	**69**	**87**	**84**	**23**
Overall	**60**	**58**	**47**	**23**	**38**	**48**	**57**	**21**	**52**	**33**	**14**	**2**	**78**	**83**	**73**	**35**

that will be used in the rest of this thesis. The different algorithms will be referred to as DE1, DE2, DE3 and DE4 and an overview is given in Table 9.6.

Table 9.6: The settings of the four versions of DE determined in the tuning process.

Name	F value	CR value	Strategy
DE1	0.7	0.90	1 (best/1/exp)
DE2	0.7	0.97	2 (rand/1/exp)
DE3	0.7	0.90	3 (rand-to-best/1/exp)
DE4	0.5	0.94	4 (best/2/exp)

9.4 Third Tuning Phase: Population Size

The next step is to determine the ideal population size. To that end the optimal settings that were just determined are subjected to a large set of problems with varying population sizes. In this way the best population size for each problem is then determined, such that a trend can be determined.
The following fifteen population sizes were used in this process: 6, 10, 14, 18, 24, 30, 36, 42, 50, 58, 66, 74, 84, 94, 104.

Figures 9.17, 9.18 and 9.19 give an overview of the typical performance at varying population size. These figures show the results for DE4, but the other optimizers showed a similar behavior. They are not given here, but can be seen in Appendix A. It is interesting to note that for many problems the best population size is evident from the figures. Only in some cases the best population size is ambiguous.
Note that the oscillatory behavior present for some problems is likely caused by the uncertainty in each measurement point. Also note that these graphs show that the guess of a population size of 20 or 60, as used in the second phase, is typically valid. Only for some problems this results in a significant loss of performance because neither population size performs well. Because this was averaged out over a large number of problems, this effect is deemed to be small.

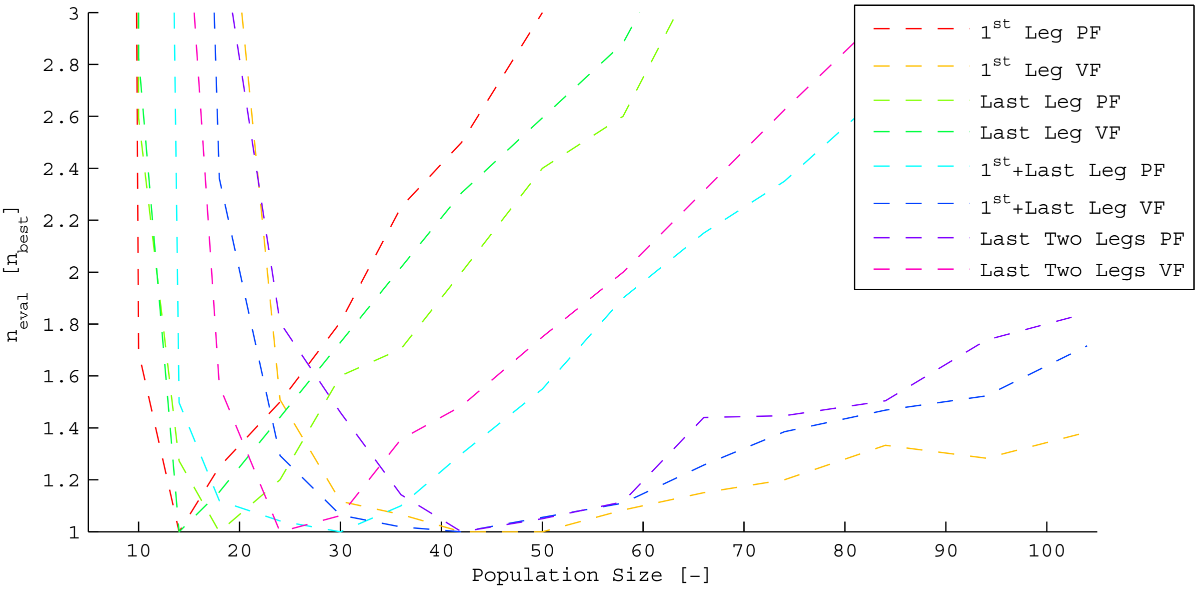


Figure 9.17: The performance of different population sizes of DE4 on the problems of optimizing certain legs of the entire problem. The performance is scaled to the best performing population size on a particular problem.

From these figures the best population size was determined for all problems. The results of this process can be seen in Table 9.7. Note that in case a clear winner was present, this population size was selected. In case the best result was a tie between two sizes, the average was taken. If the result was ambiguous the total spread of possible best population sizes is given in the table. In case two different values were optimal, but the area in between did not show stable performance, the best value is given and the second best value is added between brackets.

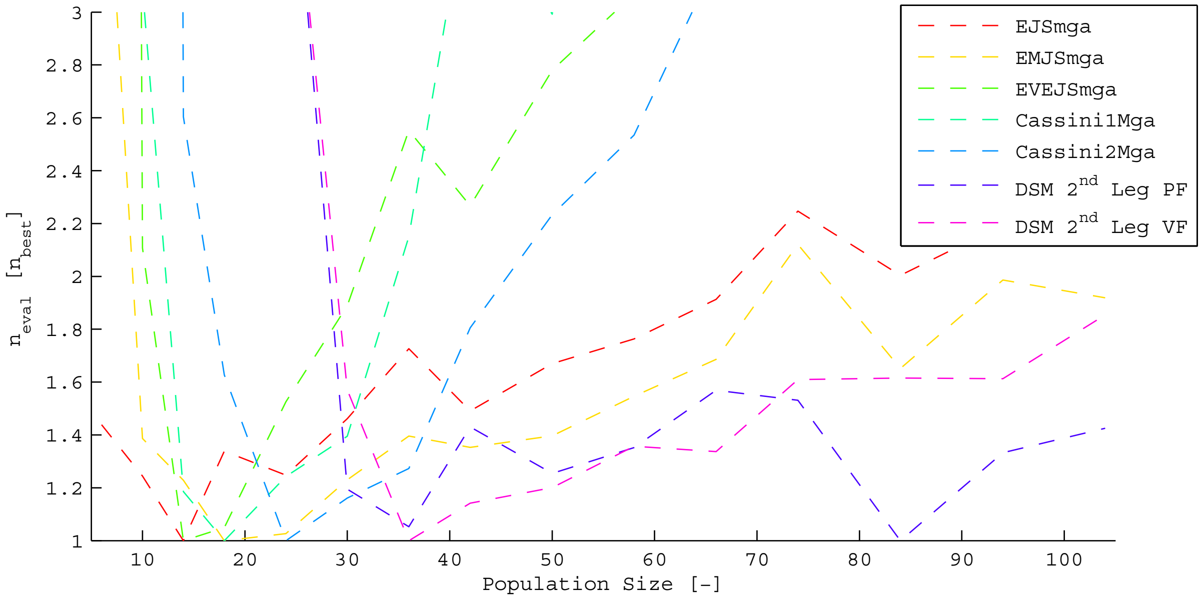


Figure 9.18: The performance of different population sizes of DE4 on the trajectory model simplification problems. The performance is scaled to the best performing population size on a particular problem.

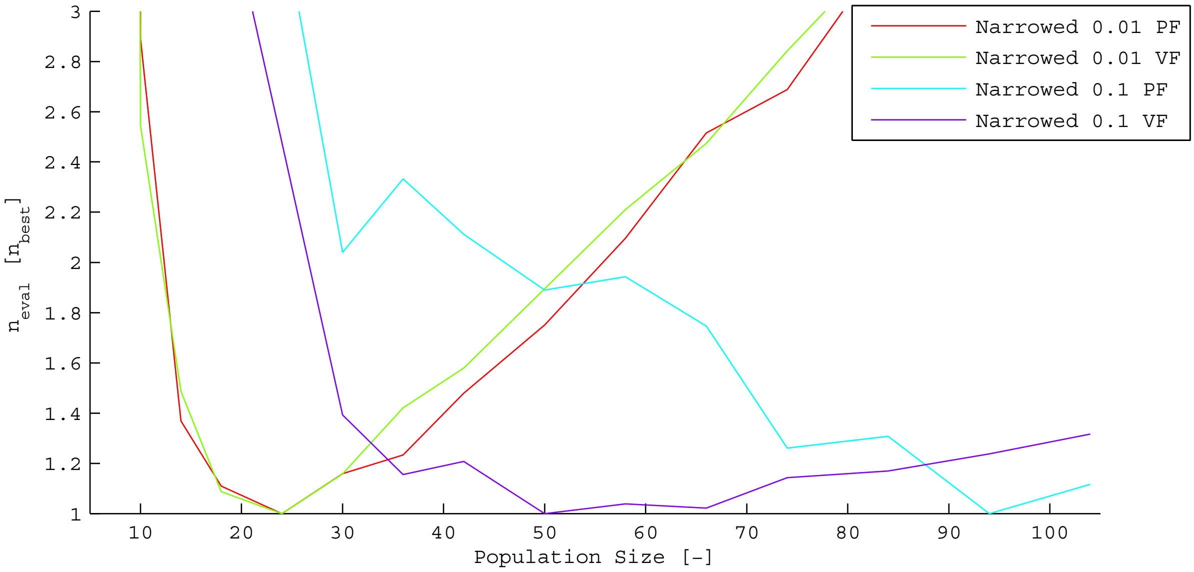


Figure 9.19: The performance of different population sizes of DE4 on the problems with narrowed search spaces. The performance is scaled to the best performing population size on a particular problem.

Subsequently the ideal population sizes are plotted against the number of variables for each DE strategy. In doing so the problems with narrowed bounds are not included, because the relation between the number of variables and the population size is not deemed representative for the actual problem.
The results can be seen in Figures 9.20 to 9.23. Also the boundaries at which the performance drops by more than 40% are plotted in these figures. Notice that some of the entries with 5, 6 or 10 variables were slightly offset to improve readability.

Using these plots a relation can be determined between the population size and the number of variables. A linear least-squares fit was made to obtain this relation for each strategy. These relations between

Table 9.7: The ideal population sizes for the four best versions of DE on a variety of problems. See text for further explanation.

		Ideal Population Size			
Problem	# of vars	DE1	DE2	DE3	DE4
EJS MGA	3	10	18-42	16	14
EMJS MGA	4	18	14-24	14-58	21
EVEJS MGA	5	18	24 (50)	18	16
Cassini1 MGA	6	18-30	16	18	18
Cassini2 MGA	6	36	16	24-36	24
DSM In 2^{nd} Leg PF	10	50	50-74	50	84 (36)
DSM In 2^{nd} Leg VF	10	58	42	36	36
First Leg PF	5	14	14	14	14
First Leg VF	5	46	24-36	30-58	46
Last Leg PF	5	16	16	16	14
Last Leg VF	5	18	16	18	18
First+Last PF	10	30	18	24	30
First+Last VF	10	30-58	33	36	42
Last Two PF	10	42-84	27	36	46
Last Two VF	10	24	21	27	27
Narrow 0.01 PF	26	18	18	14	24
Narrow 0.01 VF	26	21	16	16	24
Narrow 0.1 PF	26	94	54	66-84	94
Narrow 0.1 VF	26	42-74	36	36-58	50-66

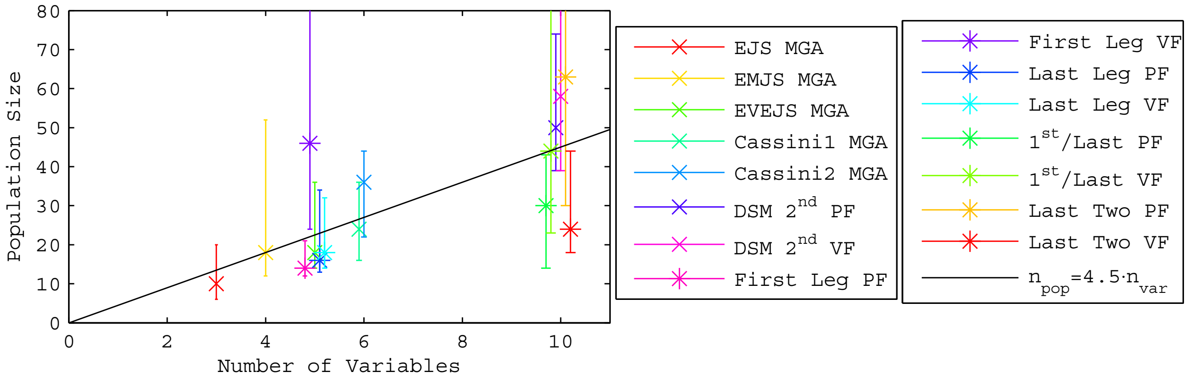


Figure 9.20: The ideal population sizes of DE1 for different problems. The bars indicate the region that requires within 40 % more evaluations than the best population size.

the population size (n_{pop}) and the number of variables (n_{var}) are given in Equations 9.1 to 9.4. The relations are also plotted in Figures 9.20 to 9.23.

$$n_{pop,DE1} = 4.5 \cdot n_{var} \tag{9.1}$$

$$n_{pop,DE2} = 10 + 2.2 \cdot n_{var} \tag{9.2}$$

$$n_{pop,DE3} = 13 + 2.1 \cdot n_{var} \tag{9.3}$$

$$n_{pop,DE4} = 3 + 3.7 \cdot n_{var} \tag{9.4}$$

As a means of verification of the above process, the ideal population size for the Cassini2 DSM VF problem was determined. The Cassini2 DSM PF problem turned out to be too difficult to allow for a similar process.

The Cassini2 DSM VF problem was optimized using the following population sizes for each strategy: 25, 32, 40, 48, 60, 75, 90, 110, 130, 160. The accompanying results were scaled to the best performing

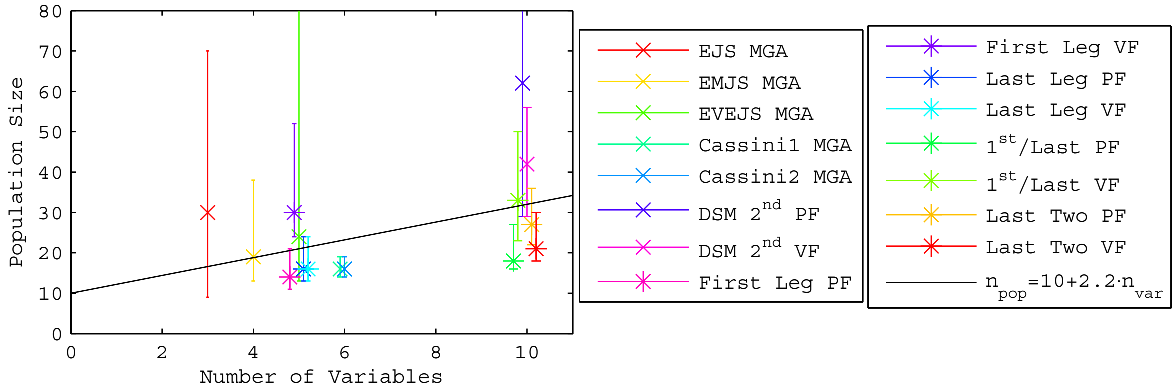


Figure 9.21: The ideal population sizes of DE2 for different problems. The bars indicate the region that requires within 40 % more evaluations than the best population size.

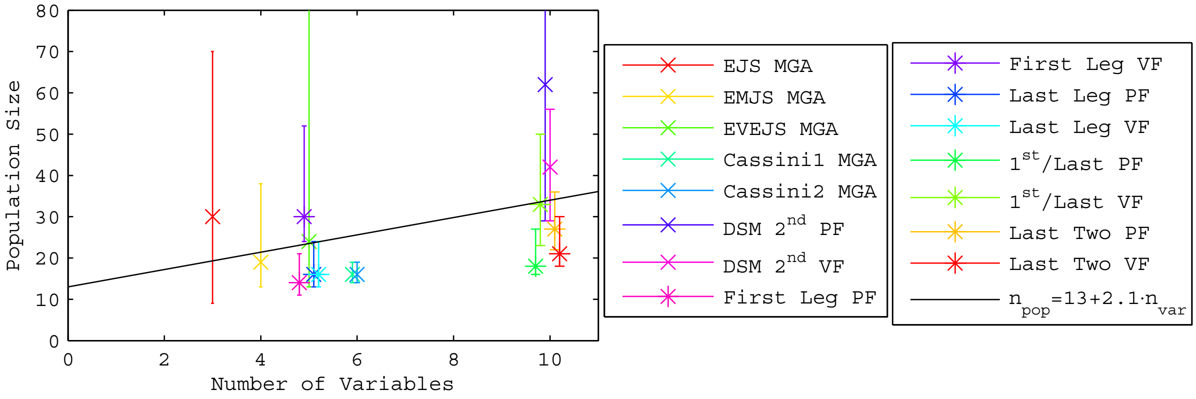


Figure 9.22: The ideal population sizes of DE3 for different problems. The bars indicate the region that requires within 40 % more evaluations than the best population size.

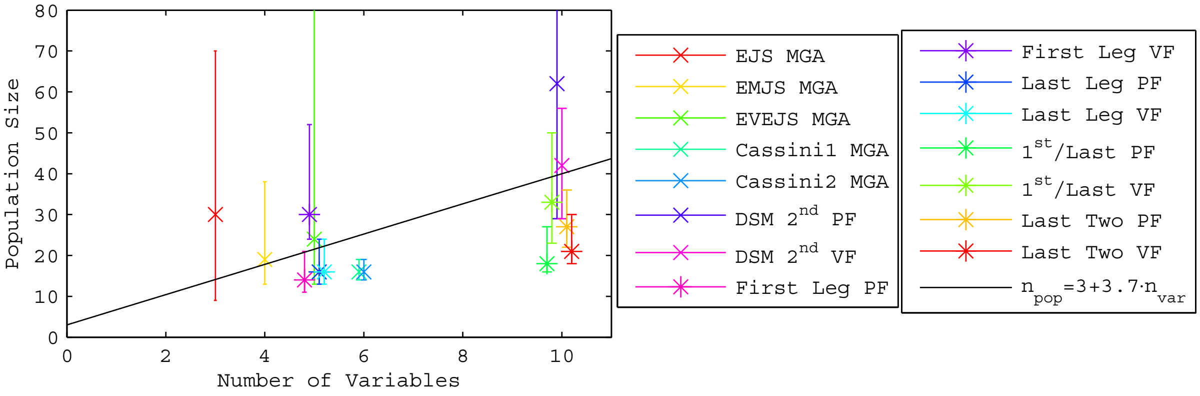


Figure 9.23: The ideal population sizes of DE4 for different problems. The bars indicate the region that requires within 40 % more evaluations than the best population size.

population size for each DE strategy and subsequently plotted in Figure 9.24. It is noted that DE3 did not manage to solve the problem reliably enough to allow the results to be plotted, whereas the other three strategies obtained fairly high success rates of 25 to 55%. It is also noted that the performance drop of DE2 after a population size of 130 is caused by the fact that not enough evaluations were performed to allow DE2 to converge for such a large population size.

It can be seen in Figure 9.24 that the statistical relations derived in Equations 9.1, 9.2 and 9.4 managed to predict the optimal population size quite accurately for the three different strategies considered. The

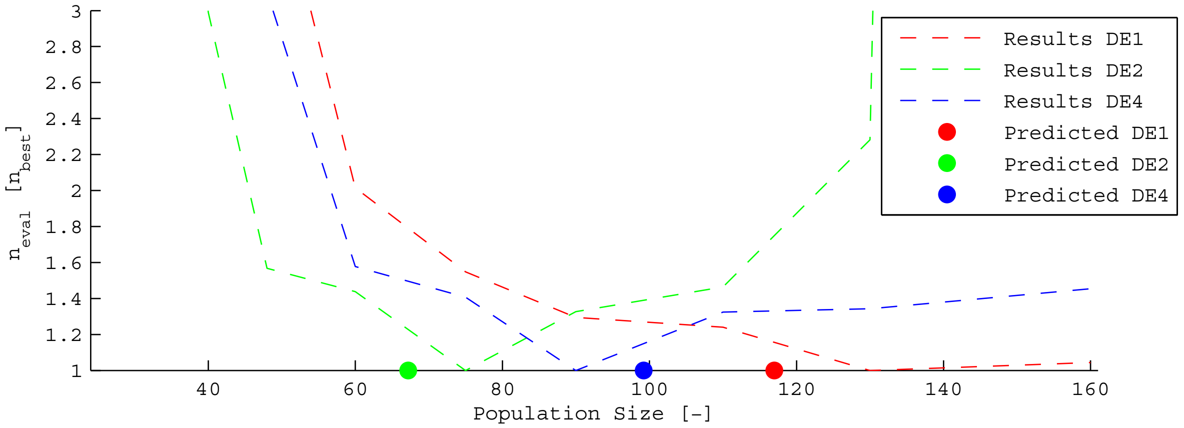


Figure 9.24: The performance of different population sizes for the DE1, DE2 and DE4 algorithms determined earlier on the Cassini2 DSM VF problem. Also plotted is the predicted location based on statistics of previous problems.

difference is well within the step-size considered in the grid. Also the apparent performance penalty is fairly low.

9.5 Conclusions

The previous sections have described the tuning process that was used to tune the DE. Although differences were present between problems, clear regions with good performance could be identified throughout the process. The test problems helped significantly in reducing the total search space and resulted in four different settings that performed properly. Two of those performed very well on all problems and two of those performed well on a particular kind of problems. A statistical relation for the ideal population size was also determined for these different settings.

Finally these settings along with the relation for the ideal population size were used to optimize the Cassini2 DSM VF problem. Three of the four settings (DE1, DE2 and DE4) yielded good results and also the statistical relation was very close to predicting the ideal population size. A comparison with other algorithms and other settings will be made in Chapter 14. An overview of the three tuning settings that performed well is given in Table 9.8.

Table 9.8: The settings of the four versions of DE determined in the tuning process.

Name	F value	CR value	Strategy	Population size
DE1	0.7	0.90	1 (best/1/exp)	$n_{pop,DE1} = 4.5 \cdot n_{var}$
DE2	0.7	0.97	2 (rand/1/exp)	$n_{pop,DE2} = 10 + 2.2 \cdot n_{var}$
DE3	0.7	0.90	3 (rand-to-best/1/exp)	$n_{pop,DE3} = 13 + 2.1 \cdot n_{var}$
DE4	0.5	0.94	4 (best/2/exp)	$n_{pop,DE4} = 3 + 3.7 \cdot n_{var}$

Chapter 10

Tuning Genetic Algorithm

This chapter discusses the tuning process that was used for GA. Section 10.1 describes the version of GA that is implemented in PaGMO. Subsequently the tuning process is described. Because of the large number of options, this tuning process is split up in four phases. The first phase intends to prune settings that perform badly from the total set of options. The second phase is then used to identify settings that perform well on a larger set of problems. A third phase is used to tune the mutation rate and width, and finally a fourth phase is used to determine the ideal population size. Section 10.2 discusses the first phase, Section 10.3 discusses the second phase, Section 10.4 discusses the third phase and finally Section 10.5 discusses the fourth phase. Finally Section 10.6 gives the conclusions of this chapter.

10.1 GA in PaGMO

Over the years, a very large number of different GA versions have been developed, that all have advantages and disadvantages over other versions. The authors of PaGMO decided to implement the basic version of the algorithm, along with some common mutation and selection strategies. They hardcoded the choice for floating point encoding, which is common for complex and sensitive problems.

Seven tuning parameters are present in the algorithm available to the author. An overview of these parameters is given in Table 10.1.

Table 10.1: Overview of the tuning parameters present in the GA implemented in PaGMO.

Parameter	Parameter Name	Domain, if applicable
n_p	Population size	At least 5
CR	Crossover probability/rate	In domain [0,1]
m	Mutation probability/rate	In domain [0,1], typically close to 0
n_{eli}	Number of elite individuals	At least 1
m_{type}	Mutation type	Gaussian or random
m_w	Mutation width	In domain [0,1], typically close to 0
sel_{type}	Selection type	Best-20% or roulette wheel

n_p, CR, m and n_{eli} are commonly used in GA and are not further described here. The mutation type requires more attention.

In random mutation, the mutation is performed fully at random. Also the size of the mutation is performed fully at random. In Gaussian mutation, the mutation is performed with a Gaussian distribution with a standard deviation equal to the mutation width.

Also the selection type deserves more attention. The best-20% selects the best 20% of the population and uses duplicates of them for the genetic operations. The roulette wheel selection randomly selects indviduals and subsequently uses a tournament selection to determine which individuals are used for the genetic operations for the next round.

All tuning parameters will be used in the tuning process, except for the number of elite individuals, which is kept at the default of 1.

10.2 First Phase

The following seven problems were selected for the initial tuning phase because they were deemed to be the least complex problems:

- Trajectory model simplification problems:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA.
- Specific leg problems:
  - Cassini2 First Leg PF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF.

All problems were optimized with a maximum of 20000 evaluations. To help identify the good regions of performance, two different approaches were used:

- Grid search: the tuning parameters are discretized with various values per parameter. The exact values considered can be seen in Table 10.2. A total of 768 settings are obtained in this way whose performance will be measured.
- Random search: random values were taken within a certain range for all six main tuning parameters of GA. This was repeated 500 times. The corresponding ranges can be seen in Table 10.2.

Table 10.2: The set/range of values that was used in the first GA tuning phase.

Parameter	Values used in grid	Range for random samples
n_p	25, 50, 100, 200	10-400
CR	0.4, 0.6, 0.8, 1.0	0.0 - 1.0
m	0.02, 0.05, 0.10, 0.20	0.0 - 0.4
m_{type}	0, 1	0 - 1
m_w	0.05, 0.10, 0.20	0.0 - 0.4
sel_{type}	0, 1	0 - 1

First the results of the grid search will be discussed in Section 10.2.1. Subsequently the random search will be discussed in Section 10.2.2. Finally the overall conclusions of the first phase are drawn in Section 10.2.3.

10.2.1 Grid Search

The full results of the grid search are not displayed here because of the size it would require. Instead they can be found in Appendix B. As a typical example, the results of the Cassini2 Last Leg VF problem are given here in Figure 10.1. Note that the same color scheme was used as was done in the DE grid search, to allow for comparison. Typically though, the difference between GA and DE is larger than in this plot. The difference in performance between different optimization techniques will be discussed in Section 14 and this difference is hence left untreated here.

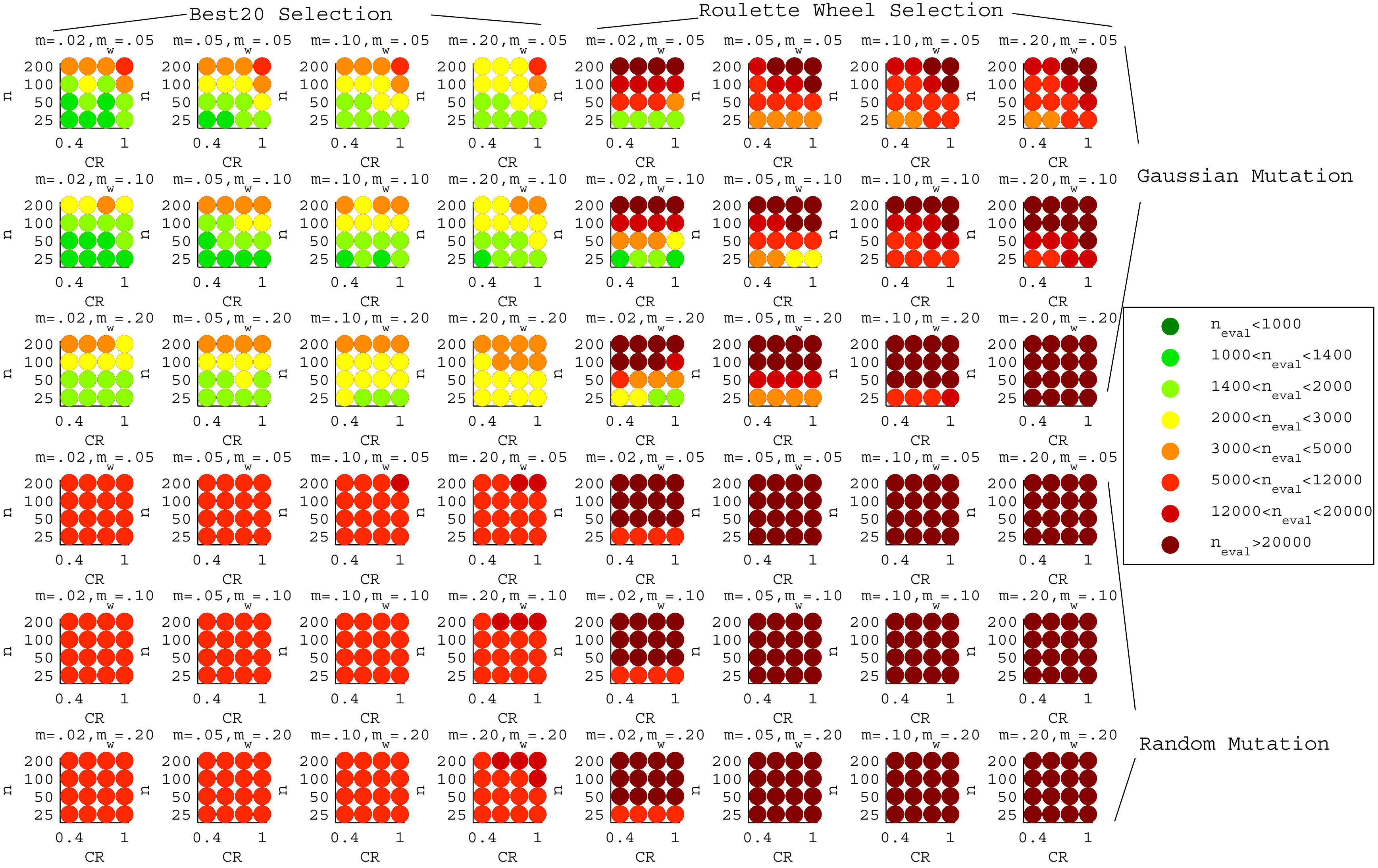


Figure 10.1: The results of the first phase tuning process of GA on the Cassini2 Last leg VF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. It is noted that the m_{width} is irrelevant for random mutation schemes. Also note that n_{eval} in the legend is identical to $n_{95\%}$.

It is noted that the EVEJSmga and Cassini2 First Leg VF problems were too difficult for GA and are hence not considered in this phase of the tuning process.
From the results of all five other problems, two clear conclusions can be drawn:

- Best 20 selection ($sel_{type} = 0$) significantly outperforms roulette wheel selection.
- Gaussian mutation ($m_{type} = 0$) significantly outperforms random mutation.

This was observed for all problems and almost all settings of the GA tuning parameters.
These observations significantly help in reducing the number of options that have to be considered. Similar to the DE tuning, the population size was deemed to be the most problem-dependent tuning parameter of GA. Although it does definitively have an influence on the outcome, its dependence is comparatively weak and likely to be problem dependent. Hence it will be optimized separately in a later phase and is taken out as a variable for now, similar to what was done in the DE tuning.
This leaves three tuning parameters: CR, m and m_w. For each problem the best performing sample out of the four different population sizes considered is taken. The results are subsequently scaled to the best performing sample using Algorithm 9.1, described in Section 9.2. Also the average performance is calculated, in which the average of the specific leg problems is weighed double with respect to the average of the trajectory simplification problems. The results of this process can be seen in Figure 10.2.

One remark has to be made with respect to Figure 10.2. Given the fact that the best result for the Cassini2 Last Leg PF was around 9000 evaluations and only 20000 evaluations were performed, the differentiation among the bad-performing samples is not fully accurate.

From Figure 10.2 various conclusions can be drawn:

- GA has a stable performance over a wide spectrum of m and CR.
- The mutation width should not be taken too large. Most problems favour an m_w of 0.05. A closer inspection around this area is required to verify this result.
- The most favourable region of m differs along the problems.

Now the results of the random sampling will be discussed, before selecting a performance region for the next tuning phase.

10.2.2 Random Search

Also for the random search, not all results are shown here. The results can be found in Appendix B. An impression of the results that were typically obtained is given in Figure 10.3. It is noted explicitly that only the best 100 samples of a total of 500 samples are depicted in this plot.
The EVEJSmga and Cassini2 First Leg VF problems were too difficult for GA and are hence not considered in this phase of the tuning process.
From the results of all five other problems, two clear conclusions can immediately be drawn:

- Best 20 selection ($sel_{type} = 0$) significantly outperforms roulette wheel selection.
- Gaussian mutation ($m_{type} = 0$) significantly outperforms random mutation.

To analyze the other results, again Algorithm 9.1 is used to score all the different combinations of tuning parameters. The maximal score of 1 is appointed to the best sample encountered, and a score of 0 is given to a sample that requires 3 times the number of evaluations or more. A continuous scoring scheme from 0 to 1 is used for all values in between. These scores are subsequently grouped based on the regime of the tuning parameter to which they belong. This gives scores to the different tuning parameter regimes. The fraction of weighed best solutions that is present in each regime is finally obtained by dividing by the total score.

This was done for the five problems in which sufficiently reliable results were obtained. The results are given in Figures 10.4 to 10.9. It is noted here that the performance on the Cassini2 First Leg DSM PF problem differed so strongly that only 11 points out of the total of 100 were given a score. The results on this problem are therefore considered less reliable. On the other hand the results are considered important as well: GA is relatively insensitive to its tuning parameters for the other problems, but is

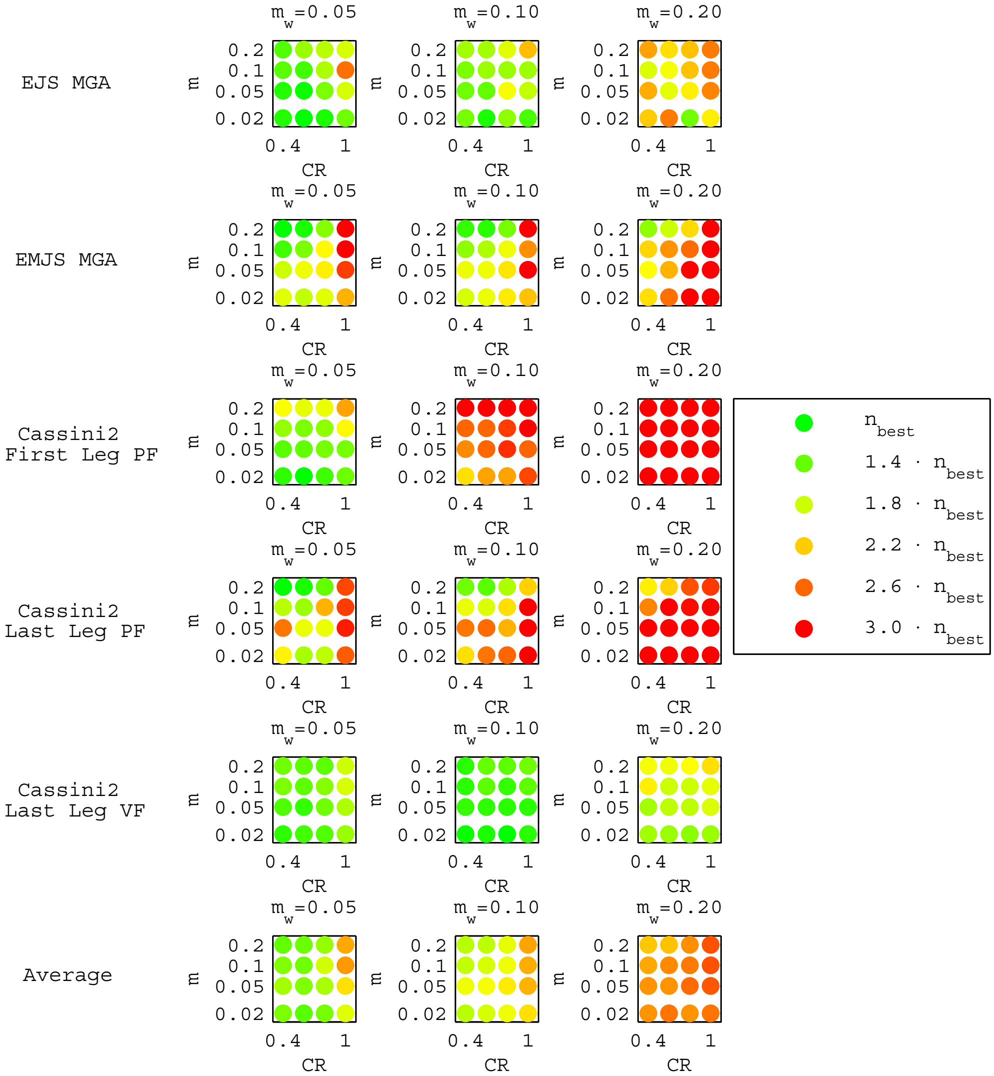


Figure 10.2: The performance of GA for various problems, with $sel_{type} = 0$, $m_{type} = 0$ and for the best population size tested (out of 25, 50, 100 and 200). The performance is scaled to the best result for that specific problem. Also an average is computed. See text for more explanation.

highly sensitive to its tuning parameters for the Cassini2 First Leg PF problem.

Figure 10.4 does not give a conclusive view on the ideal population size. The Cassini2 Last Leg VF problem clearly favors a small population size. Given that it is the easiest problem for GA, this is not unexpected. The other problems show a relatively stable performance over a large part of the population size spectrum and are deemed much less sensitive to the population size.

The results are not very sensitive to CR either, as can be seen in Figure 10.5. Typically a CR of 0.2 seems to be best, but up until 0.6 or 0.8 the performance is still good and for some problems better than lower values of CR.

Figure 10.6 shows that the results are more sensitive to the mutation rate. For most problems, the performance peaks around 0.2 or 0.25. A very different result is found for the Cassini2 First Leg PF problem though. This may be due to the low amount of samples, or because it really peaks between 0.05 and 0.1.

Figure 10.7 clearly shows the preference for Gaussian mutation.

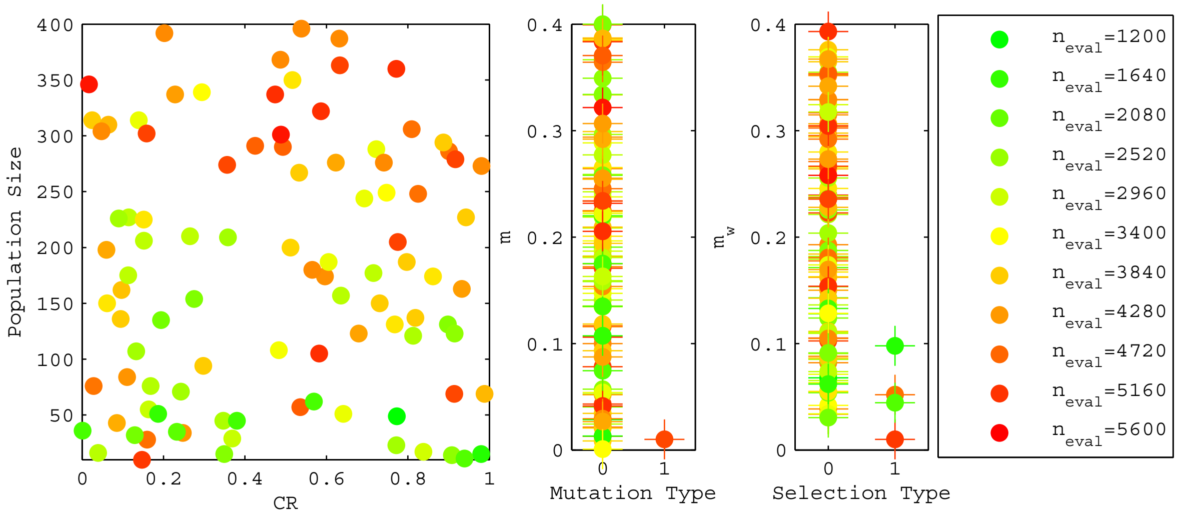


Figure 10.3: The results of the first phase tuning process of GA on the Cassini2 Last leg VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

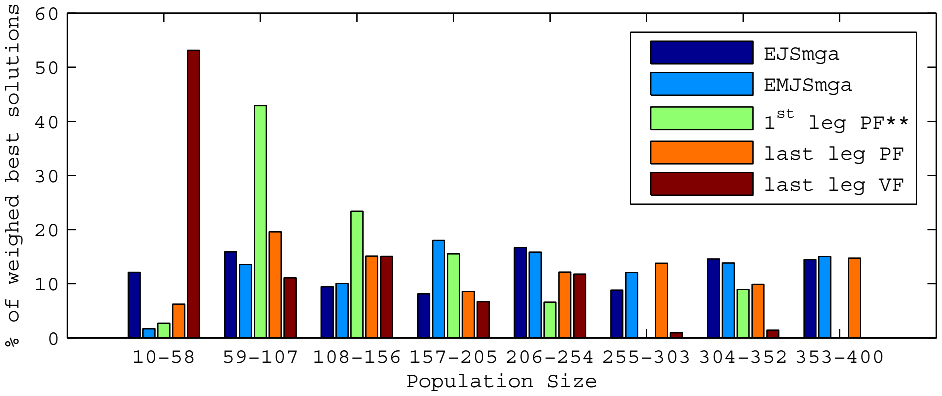


Figure 10.4: The distribution of good results over different regimes of population sizes. See text for explanation and notes.

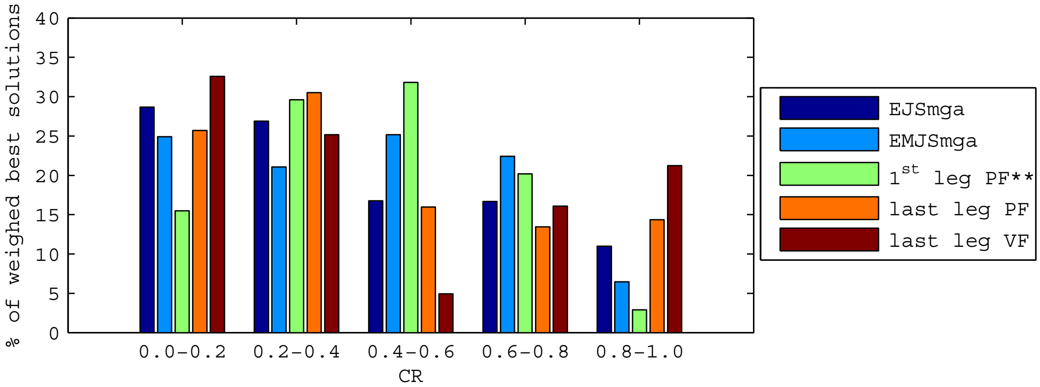


Figure 10.5: The distribution of good results over different regimes of CR. See text for explanation and notes.

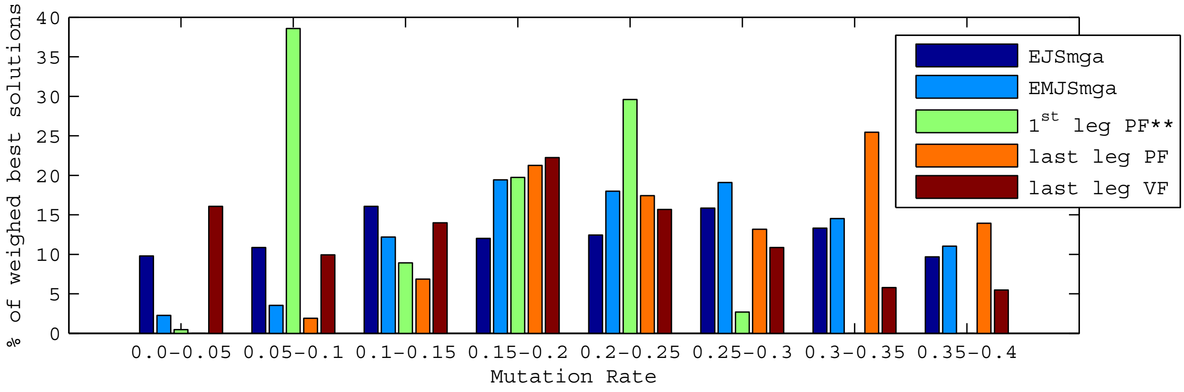


Figure 10.6: The distribution of good results over different regimes of the mutation rate. See text for explanation and notes.

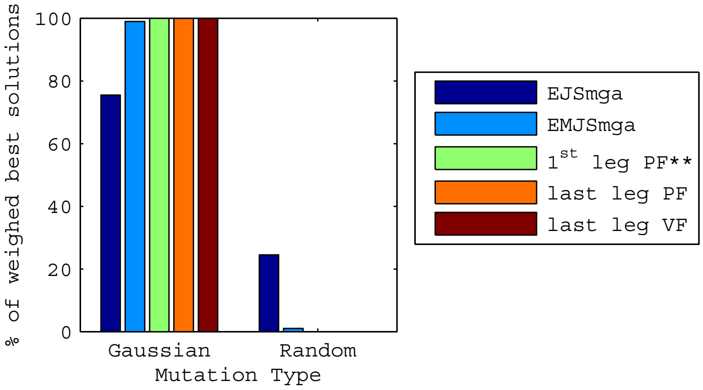


Figure 10.7: The distribution of good results over the two mutation types. See text for explanation and notes.

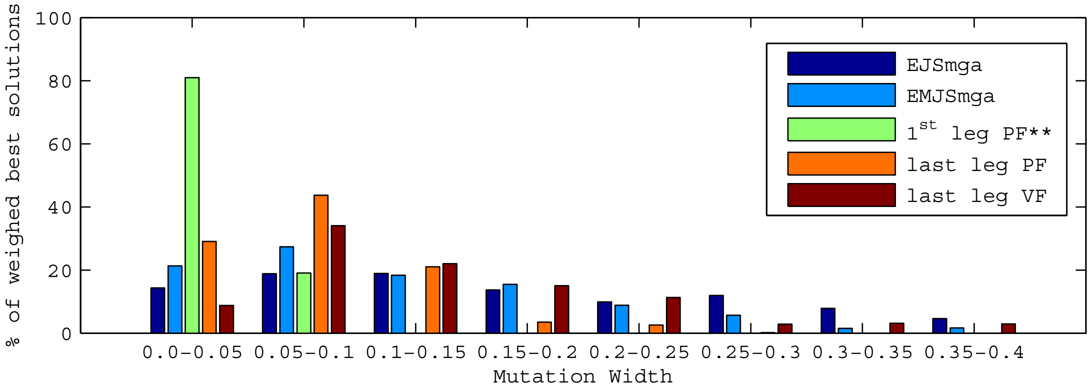


Figure 10.8: The distribution of good results over different regimes of the mutation width. See text for explanation and notes.

The results for the mutation width in Figure 10.8 clearly show that very large mutation widths are not preferred. The exact location of the best performing mutation width is unclear though. The performance peaks between 0.05 and 0.1 for most problems, but very decisively peaks between 0.0 and 0.05 for the Cassini2 First Leg PF problem. More simulations are needed to find the actual peak.

Figure 10.9 clearly shows the preference for Best-20 selection.

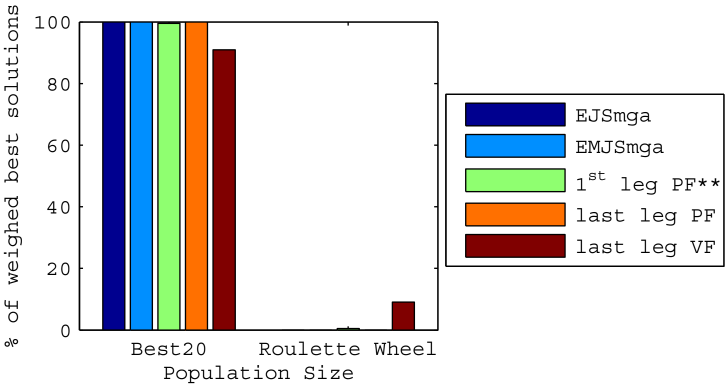


Figure 10.9: The distribution of good results over the two selection types. See text for explanation and notes.

10.2.3 Conclusions First Tuning Phase

The following conclusions are drawn based on the results of the previous sections to define the new tuning regimes:

- Gaussian mutation performs best.
- Best-20 selection performs best.
- The results are relatively insensitive to the population size and CR.
- The results indicate that a mutation rate of slightly above 0.20 is good for various problems. The EJS MGA problem and especially the Cassini2 First Leg PF problem indicate a much lower mutation rate though.
- The results appear to be highly sensitive to the mutation width for some problems. Also the values of m_w which perform well appear to be dependent on the problem. A mutation width slightly above 0.05 seems to be a fair initial guess, but a more thorough analysis is required.

Based on these conclusions a decision was made to investigate the influence of the population size, CR and mutation rate on more difficult problems first. Subsequently a separate analysis will be done to determine the optimal value of the mutation rate and mutation width. Tuning all four parameters at the same time would have required too much computational time. Compared to DE, GA namely performs much worse and hence requires more function evaluations per simulation.

The tuning parameters that are included for the next phase are given in Table 10.3.

Table 10.3: The set of values that will be used in the second GA tuning phase.

n_{pop}	CR	m	m_{type}	m_w	sel_{type}
80, 200, 400	0.2, 0.6	0.08, 0.24	Gaussian	0.06	Best-20

10.3 Second Phase

This second phase is performed to get a better understanding of the performance of GA on more difficult problems. The main issue with selecting the problems for the second-phase GA tuning is that GA performs significantly worse than DE. Hence only the problems that were relatively easy to optimize by DE were selected for tuning GA. Hence the following six problems were selected for the second-phase tuning of GA:

- EVEJS MGA,
- Cassini2 First Leg DSM VF,
- Cassini2 First And Last Leg PF,

- Cassini2 Last Two Legs VF,
- Cassini2 PF, reduction to box of 0.01 around the best putative solution in this thesis,
- Cassini2 VF, reduction to box of 0.01 around the best putative solution in this thesis.

All problems were run for 10^6 evaluations, except for the Cassini2 First Leg DSM VF problem, which was run for $2 \cdot 10^6$ evaluations. Although this number was still not enough to provide fully accurate results, the results are presented here nevertheless. It is noted that if success was achieved, it was typically before $2 \cdot 10^6$ evaluations considered here. It is furthermore noted that the required number of evaluations is already roughly 200 times that of DE on the same problem. The other problems required $7 \cdot 10^4$ to $3 \cdot 10^5$ evaluations and results can hence be considered reliable.
The simulations were performed using the settings presented in Table 10.3. The results are plotted in Figure 10.10. Note that all results on a particular problem were scaled to the best performing sample.

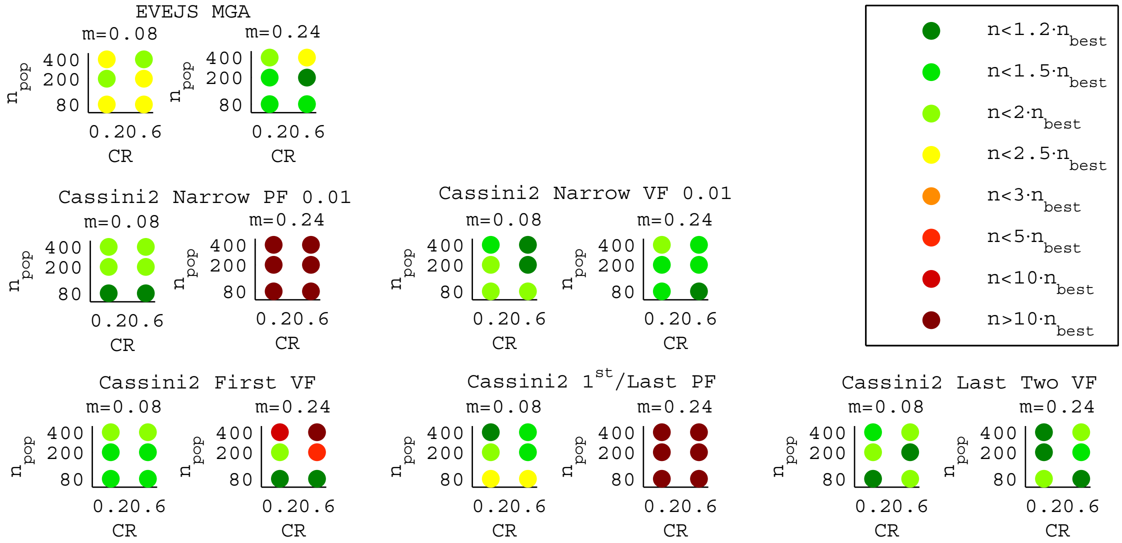


Figure 10.10: The required number of evaluations required by different settings of GA. Six different problems were optimized and each time the results are scaled to the best performing setting of GA on that particular problem.

The Cassini2 DSM PF Narrowed 0.01 and Cassini2 First And Last Leg DSM PF problems show very bad performance for m=0.24, as can be seen in Figure 10.10. This bad performance is also present in part of the plot in Cassini2 First Leg DSM VF problem. For the latter this may partially also be caused by the fact that not enough function evaluations were performed for that problem though.
Apart from the mutation rate, the other variables do not appear to influence the results significantly. The difference in CR is quite large, just as the difference in population size. The actual results are typically very close together though. Furthermore, each time a different setting is the best in each problem.

The results were also grouped, using a similar process as was done earlier. A 1 is given to the best problem and a 0 is given to settings requiring 3 times the number of evaluations of the best. Subsequently the average performance was calculated for the three different means of simplification types. Finally the average over the results of the three different simplification types was taken. The results can be seen in Figure 10.11.

It appears from this plot that m=0.08 and CR=0.6 are the best bet. The population size for these problems should be roughly around 200 to 400. It is noted that the population size and the CR are not very influential, but that m is very influential for some problems. Given the high sensitivity for m_w in the first phase, some more combinations of m and m_w will be tested in Section 10.4 to get a better understanding of this phenomenon.

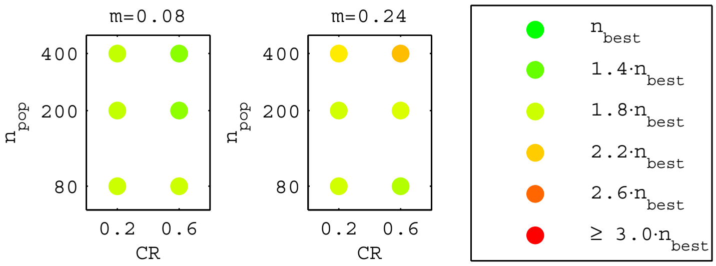


Figure 10.11: The average performance of different settings of GA on the problems in the second tuning phase of DE. See text for explanation.

10.4 Third Phase: Tuning Mutation Rate and Width

This section will discuss the tuning of both the mutation rate and the mutation width. In doing so, all problems that were solvable in a reasonable amount of time were used. These problems are:

- Trajectory model simplification problems:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA.
- Specific leg problems:
  - Cassini2 First Leg PF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF,
  - Cassini2 First And Last Leg DSM PF,
  - Cassini2 Last Two Legs VF.
- Narrowed bounds problems:
  - Cassini2 DSM PF Narrowed 0.01,
  - Cassini2 DSM VF Narrowed 0.01.

First the mutation width is analyzed. The following five values for m_w were taken: 0.02, 0.04, 0.06, 0.10, 0.15. The other settings were taken as follows: m=0.08, CR=0.6 and n_{pop}=200. The same problems were optimized for this set of values for m_w. The results can be seen in Figure 10.12.

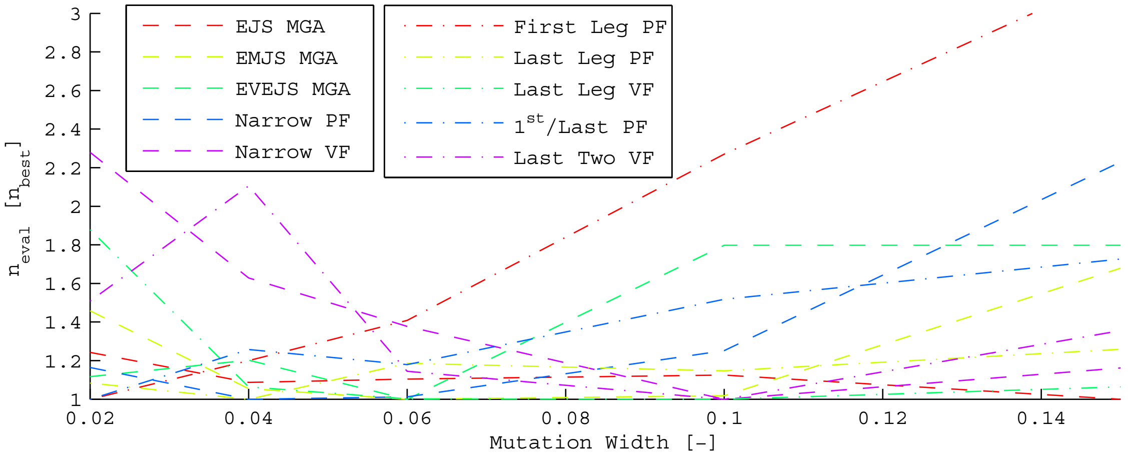


Figure 10.12: The performance of GA on different problems for different settings of m_w. See text for explanation.

It can be seen in Figure 10.12 that the effect of m_w is not extremely large. A mutation width of 0.06 seems to be the best compromise.

A similar test was done on the value for m. The following five values for m were taken: 0.04, 0.08, 0.12, 0.16, 0.20. A value of 0.06 was used for m_w, 0.6 for CR and 200 for n_{pop}. Again the same problems were solved, resulting in Figure 10.13.

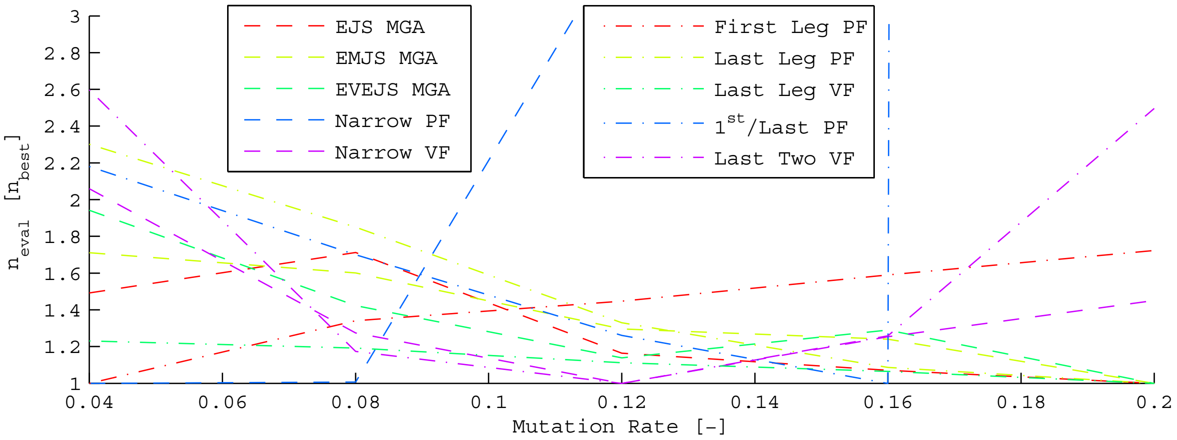


Figure 10.13: The performance of GA on different problems for different settings of m. See text for explanation.

It can be seen in Figure 10.13 that the influence of m is slightly larger than that of m_w. A mutation rate of 0.12 is a good value for all problems except for the Cassini2 DSM PF Narrowed 0.01 problem. Because there are various problems for which it is considerably better than a lower mutation rate of 0.08, a mutation rate of 0.12 is selected.

10.5 Fourth Phase: Determining the Population Size

This section investigates the effects of the population size on the performance of GA. Ten different population sizes were used to optimize the same problems that were considered in the third phase. The following population sizes were used: 15, 30, 50, 80, 120, 180, 250, 350, 480, and 600. The results of this analysis can be seen in Figures 10.14 and 10.15. It is noted that for some population sizes, the EMJS MGA problem and the Cassini2 Last Two Legs VF problem were solved using a very low success rate. This causes the very wobbly behavior of both problems.
It is noted here that the results of Figures 10.14 and 10.15 are very difficult to interpret. It appears as if GA is very insensitive to the population size. Furthermore the results are not consistent enough to calculate a trend. It seems that some problems favour a very small population size, where others favour a very large population size. The performance drops are not very significant though. Still it may be noted that a good strategy may be to try both a comparatively small population size of about 50 and also a population size that is about 500. Chances are that either one of them will perform well.
This conclusion is not very useful in practice though, as the ‘small’ and ‘large’ population sizes likely still depend on the actual problem dimension.

10.6 Conclusions

The previous sections have described the process that was used to tune the GA. This tuning process turned out to be much more difficult than the tuning process used for DE. The main problem is that the performance of GA deteriorates rapidly as the problems become more difficult. Because the tuning parameters had little effect on the performance of GA on the easy problems, it was difficult to apply a good tuning strategy. Still also on the more difficult problems, GA was much less sensitive to its tuning

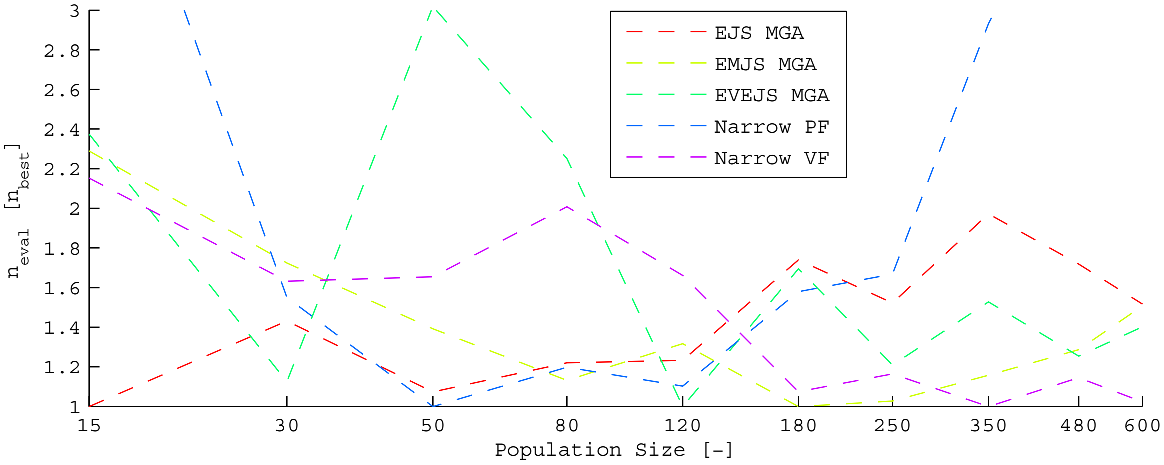


Figure 10.14: The performance of GA on different problems for different population sizes. Note that a logarithmic x-axis is used. See text for explanation.

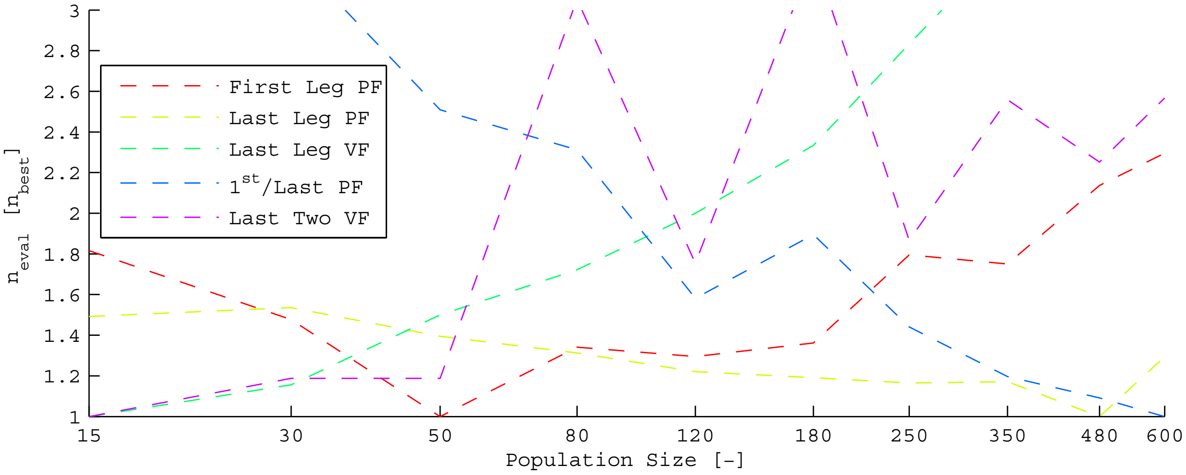


Figure 10.15: The performance of GA on different problems for different population sizes. Note that a logarithmic x-axis is used. See text for explanation.

parameters than DE was. Although this can be regarded as an advantage in general, it also makes it much more difficult to identify the best performance regime. In particular the ideal population size was difficult to determine.

Table 10.4 gives an overview of the best tuning settings according to the analysis in this chapter.

Table 10.4: The best settings of the tuning parameters of GA, as determined in this chapter.

n_{pop}	CR	m	m_{type}	m_w	sel_{type}
indecisive, try 50 or 500	0.6	0.12	Gaussian	0.06	Best-20

Chapter 11

Tuning Particle Swarm Optimization

This chapter discusses the tuning process that was used for PSO. Section 11.1 describes the version of PSO that is implemented in PaGMO. The various tuning parameters allow for a very wide range of options. The tuning process is hence split up in five phases. The first three phases intend to prune settings that perform badly from the total set of options. The fourth phase is used to determine the settings that perform best. Finally the fifth phase determines the ideal population size. Section 11.2 discusses the first phase, Section 11.3 discusses the second phase, Section 11.4 discusses the third phase, Section 11.5 discusses the fourth phase and Section 11.6 discusses the fifth phase. Finally conclusions are given in Section 11.7.

11.1 PSO in PaGMO

The PSO algorithm that is implemented in PaGMO offers a wide variety of optimization strategies. In total eight parameters are used to tune the algorithm. An overview of those parameters is given in Table 11.1.

Table 11.1: Overview of the tuning parameters present in the PSO algorithm implemented in PaGMO.

Parameter	Domain, if applicable	Parameter Name
n_p	At least 1	Population size
ω	In domain [0,1]	Particle's inertia (constriction coefficient if $var = 6$)
η_1	In domain [0,4]	Cognitive component coefficient
η_2	In domain [0,4]	Social component coefficient
v_{coeff}	In domain [0,1]	Velocity coefficient (maximum velocity)
var	1,2,3,4,5,6	Variant
$neigh_{type}$	1,2,3,4	Neighbour type
$neigh_{param}$	Positive	Particle's indegree (# of neighbours), only if $neigh_{type} = 2$

No further explanation is required for n_p. Also ω, η_1 and η_2 are commonly used in PSO, although the variable symbols and names may differ. Together they determine the velocity of a particle for the next generation. ω causes the particle to move in the direction it had been moving, η_1 causes the particle to move into the direction of the best location it has encountered so far, η_2 finally causes it to move towards the best location found by another particle in its neighborhood. The velocity is subsequently bounded by v_{coeff}.

A total of six variants is considered in the PSO algorithm implemented in PaGMO. The first four are commonly used and differ in two aspects: they may employ the same or different random numbers in determining the size of the social and cognitive components; they may use the same random number for all variables or a different one for each variable in the decision vector. An overview of the first four variants is given in Table 11.2.

Table 11.2: The difference between the first four variants in the PSO algorithm implemented in PaGMO.

Variant	Different/same random number for social and cognitive component	Different/same random number for each variable
1	different	different
2	same	different
3	different	same
4	same	same

The fifth variant is a more exotic one in which the inertia weight is replaced with a constriction coefficient that controls the convergence of a particle. More information on this variant can be found in [Clerc and Kennedy, 2002]. The sixth variant is also exotic. It is a fully-informed particle swarm algorithm, in which the particle is affected by all its neighbors. For this variant η_1 and η_2 are not used. More on this variant can be found in [Mendez et al., 2004].

Furthermore PaGMO has implemented four neighborhood types. $neigh_{type} = 4$ caused the algorithm to crash though, both on the author's computer as well as on those of fellow students. Hence it is not further considered in this thesis.
The first neighborhood type is a fully connected graph, in which all other particles are known to each particle. This type is similar to the traditional version. The second type is one in which the swarm is modelled as a ring in which all particles are influenced by a number of particles to either side. The number of particles is determined by $neigh_{param}$. Finally a third type is the Von Neumann neighborhood. It arranges the particles in a lattice and each particle interacts with its immediate neighbors to the North, South, East and West, wrapping the edges.

All eight tuning parameters are used in the analysis in this thesis. It is noted though that the eighth parameter, $neigh_{param}$, is only applicable if $neigh_{type} = 2$.

11.2 First Phase

Similar to the other algorithm test suites, the following seven problems were selected for the initial tuning phase:

- Trajectory model simplification problems:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA.
- Specific leg problems:
  - Cassini2 First Leg PF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF.

Given the large number of variables, a grid search is not an option if some accuracy is required in the results. Hence a random suite was used to get an estimate of the settings that typically perform well. An overview of the bounds that were used can be seen in Table 11.3
A typical result can be seen in Figure 11.1. Note that not all of these 'raw' results are given here because of the required size, but they can be accessed in Appendix C.
To get a better idea of which areas are performing well, the results are processed. Algorithm 9.1, which was described in Section 9.2, is used to do so with one adaptation: the scoring range is extended to $4 \cdot n_{best}$ because the results differ very much for PSO. Subsequently these scores are grouped according to the regime of the tuning parameter to which they belong. In this way the fraction of weighed best solutions is obtained for each regime in the tuning parameters. It is noted explicitly that some parameters are not used by all variants or neigbor types. These are hence only counted when they were used.

Table 11.3: The set/range of values that was used in the first PSO tuning phase.

Parameter	Range for random samples
n_p	6 - 200
ω	0.0 - 1.0
η_1	0.0 - 4.0
η_2	0.0 - 4.0
v_{coeff}	0.0 - 1.0
var	1 - 6
$neigh_{type}$	1 - 3
$neigh_{param}$	1 - 20

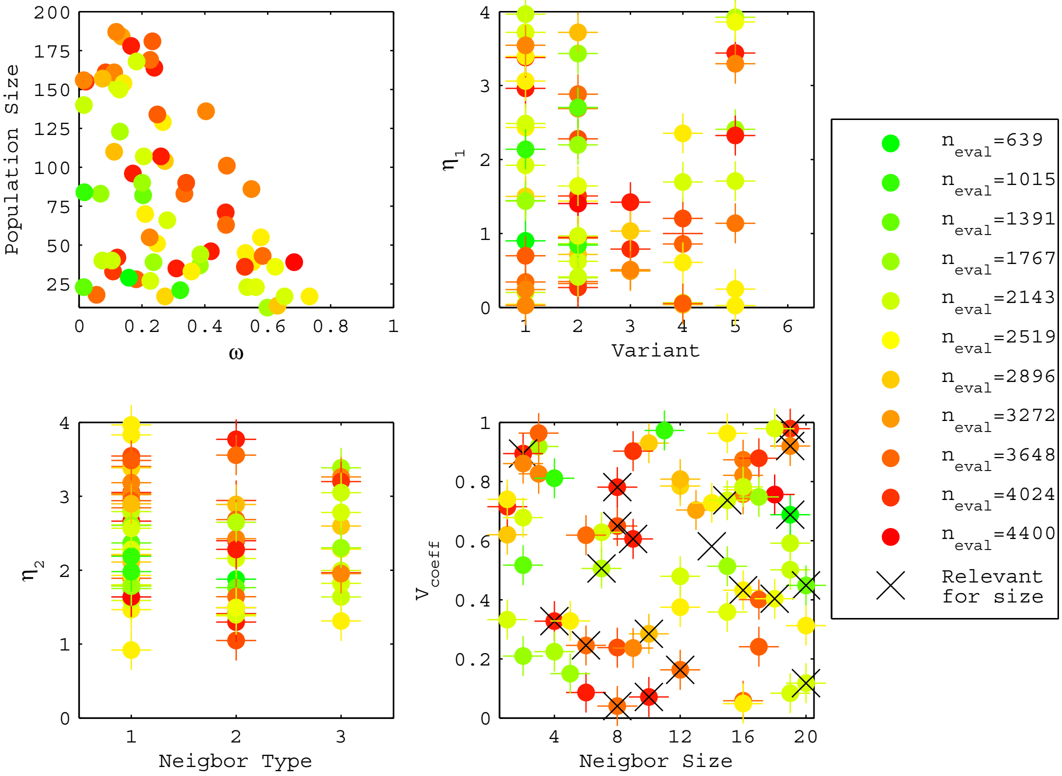


Figure 11.1: The results of the first phase tuning process of PSO on the Cassini2 Last Leg VF problem with random samplnig. The results show the number of function evaluations required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale, where green coresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

The results are given in Figures 11.2 to 11.9. It is noted that the EVEJSmga problem and the Cassini2 First Leg VF problem were too difficult for PSO to allow those problems to be used in this tuning phase.

From Figure 11.2 it is concluded that typically smaller population sizes are preferred. However, given that good results are obtained up to a population size of 150, only the region from 150-200 will be pruned.

Figure 11.3 shows that a small ω is to be preferred. The region 0.6 - 1.0 will be pruned in the next phase.

The good solutions are much more spread over different regions for η_1 as can be seen in Figure 11.4. This distribution seems to peak between 0.5 and 1.0. The region from 3.0 to 4.0 will be pruned for the

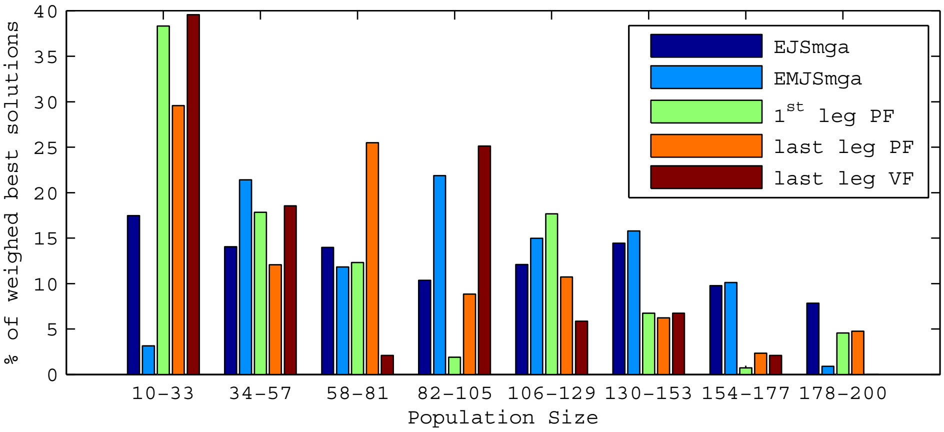


Figure 11.2: The distribution of good results over different regimes of the population size. See text for explanation.

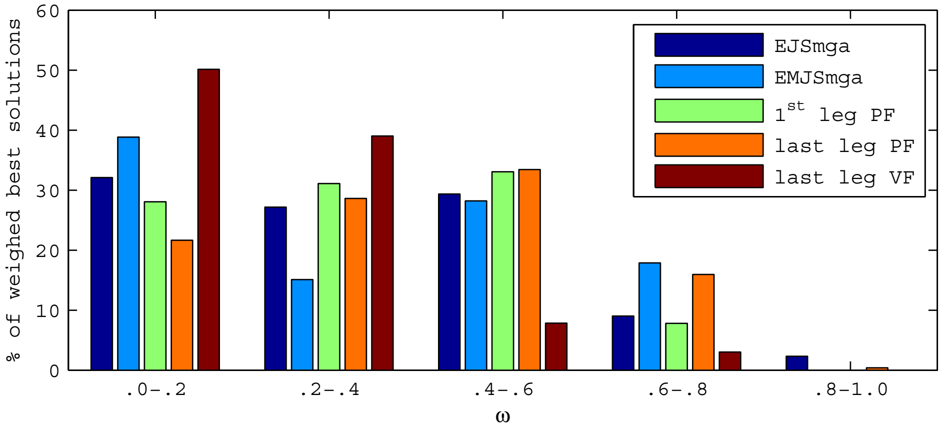


Figure 11.3: The distribution of good results over different regimes of ω. See text for explanation.

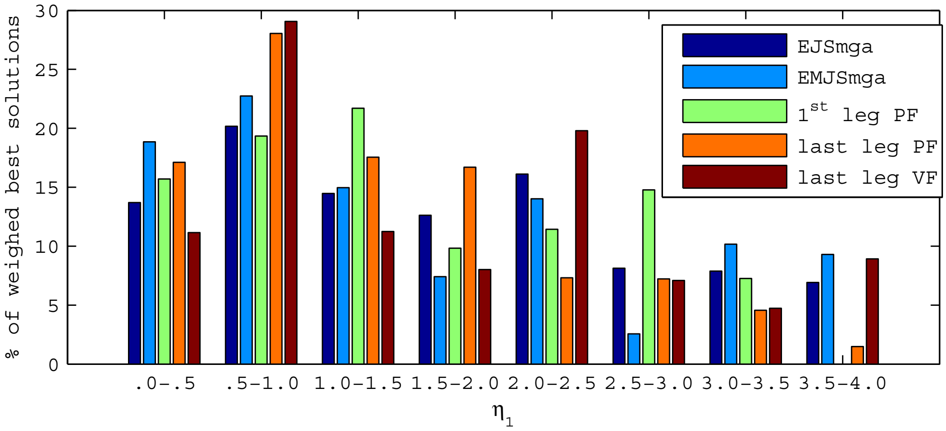


Figure 11.4: The distribution of good results over different regimes of η_1. See text for explanation.

next phase.

η_2 has a strong peak around a value of 2.0, as can be seen in Figure 11.5. Given the comparatively low percentages near 0.0 and 4.0, only the region from 1.0 to 3.0 will be used in the next phase.

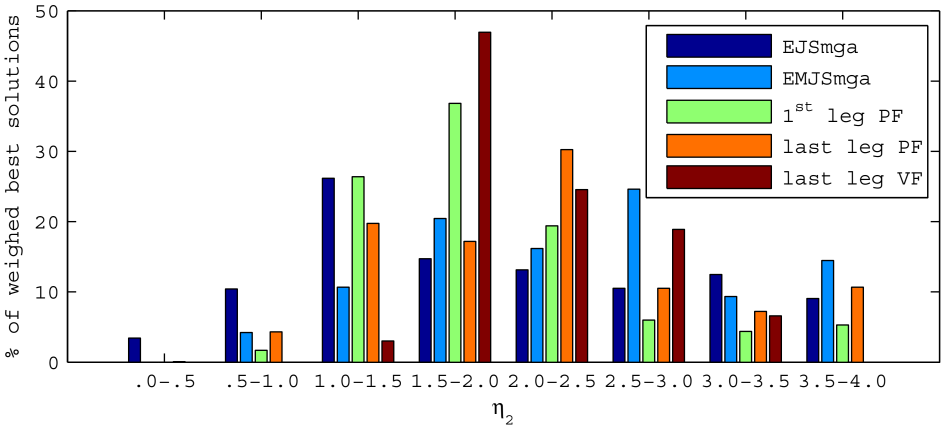


Figure 11.5: The distribution of good results over different regimes of η_2. See text for explanation.

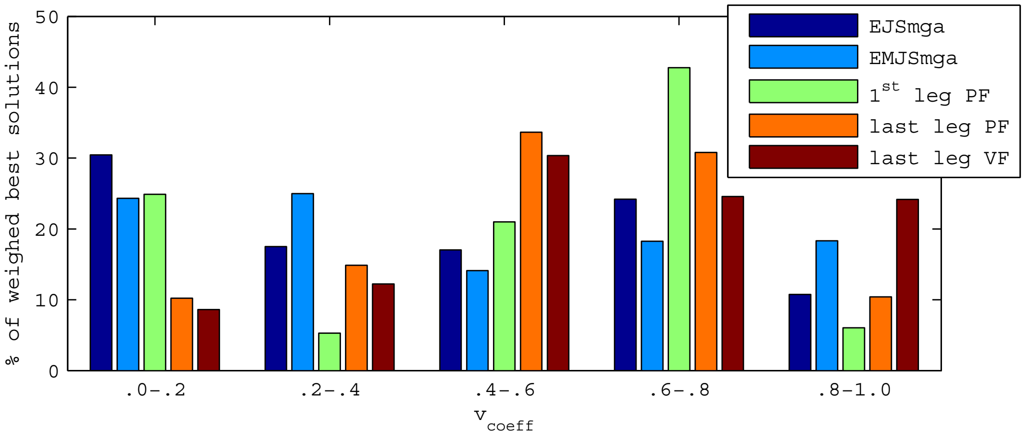


Figure 11.6: The distribution of good results over different regimes of v_{coeff}. See text for explanation.

From Figure 11.6 it is concluded that the spread in v_{coeff} is too large to prune certain parts of the regime already.

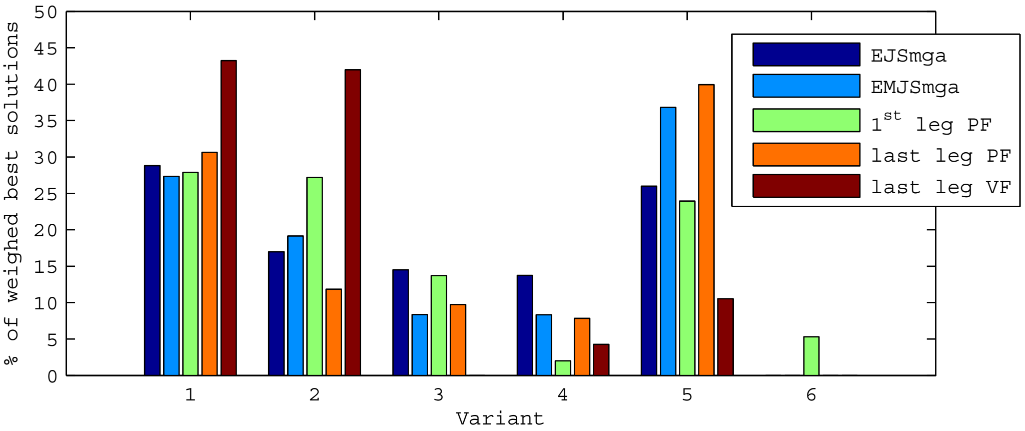


Figure 11.7: The distribution of good results over the different variants. See text for explanation.

Variants 1, 2 and 5 seem to clearly outperform the other three variants, as can be seen in Figure 11.7. Hence variant 3, 4 and 6 are pruned from the next tuning phase.

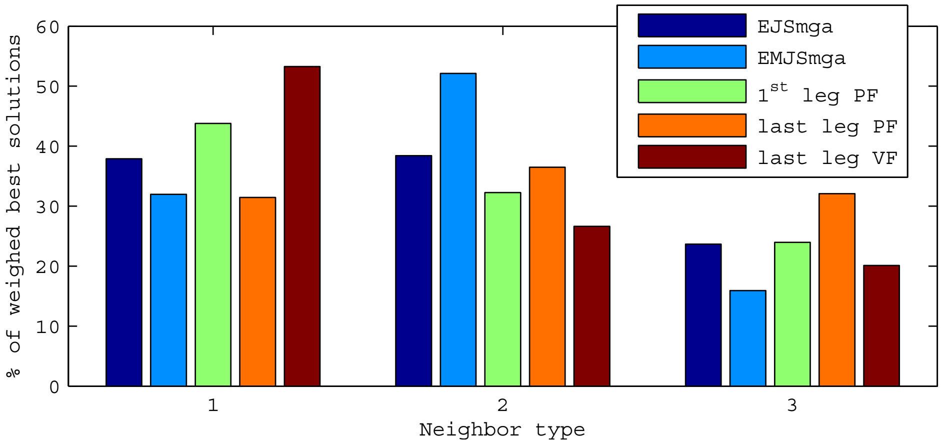


Figure 11.8: The distribution of good results over the different neighbor types. See text for explanation.

Figure 11.8 shows that the neighbor type 1 and 2 deliver better solutions than neighbor type 3 across all problems. Hence neighbor type 3 is pruned. It is noted here that neighbor type 2 is expected to perform better if the neighbor size is chosen better, as discussed in the next paragraph.

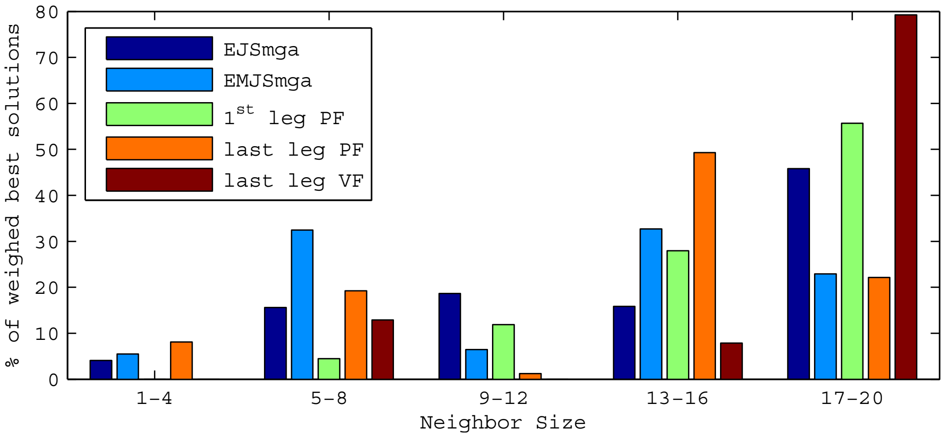


Figure 11.9: The distribution of good results over different regimes of the neighbor size. See text for explanation.

Figure 11.9 indicates that the neighbor size was not chosen correctly. A strong preference for a neighbor size of close to 20 seems to be present. Given that larger sizes are also possible, this also hints at the fact that the regime was not chosen correctly. Hence the new regime will be set to range from 11 to 30 in the next phase. Note that only neighbor type 2 uses this neighbor size. Hence this wrong regime definition may have caused neighbor type 2 to have underperformed in this tuning phase.

Although this first pruning phase has not pruned a significant part in a single parameter, the fact that the regime for most parameters was reduced slightly did reduce the total parameter space by a factor 20. Hence another pruning phase is performed using the stricter regime definition. These regimes can be seen in Table 11.4.

11.3 Second Phase

Again 500 random samples were taken to estimate the performance in different regimes of these tuning parameters. Only the five problems that were solved reliably in the previous section are considered in this section. This time the original scoring mechanism defined in Algorithm 9.1 is applied. This was

Table 11.4: The set/range of values that was used in the second PSO tuning phase.

Parameter	Range for random samples
n_p	6 - 150
ω	0.0 - 0.6
η_1	0.0 - 3.0
η_2	1.0 - 3.0
v_{coeff}	0.0 - 1.0
var	1, 2, 5
$neigh_{type}$	1 - 2
$neigh_{param}$	11 - 30

possible because of the much better average performance of PSO in this second tuning phase. The processed results are given in Figures 11.10 to 11.17. The 'raw' results are again available in Appendix C.

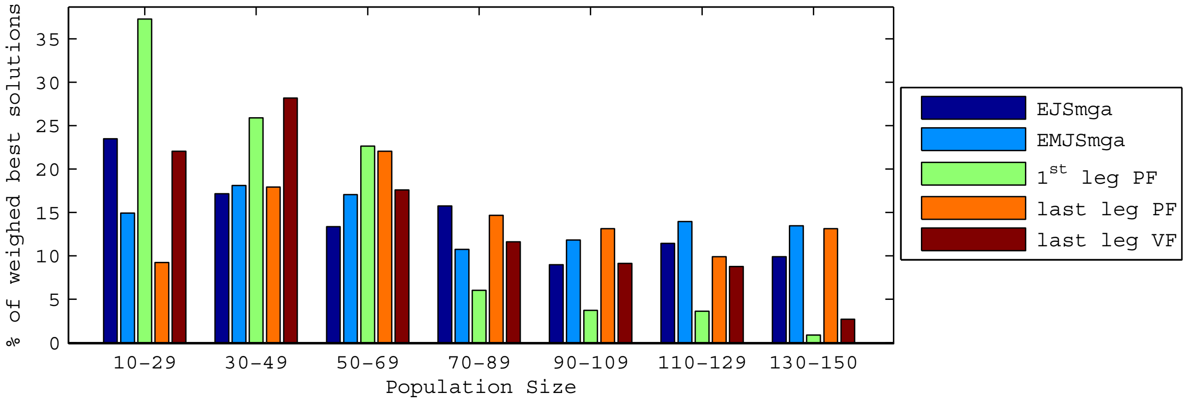


Figure 11.10: The distribution of good results over different regimes of the population size. See text for explanation.

From Figure 11.10 it appears that the best region in terms of population size is between 10 and 70 individuals.

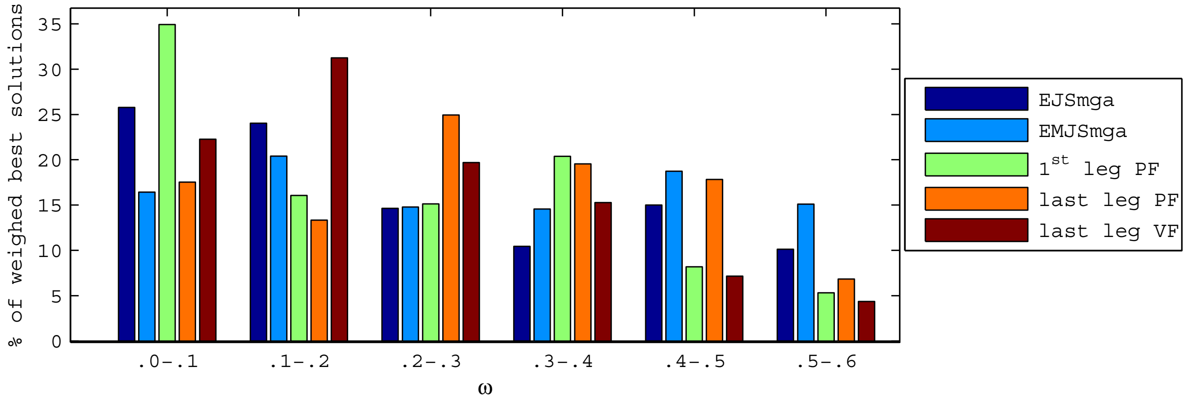


Figure 11.11: The distribution of good results over different regimes of ω. See text for explanation.

It appears from Figure 11.11 that the best region for ω is relatively low. This effect is not very clear for all problems though. For instance the EMJSmga problem is fairly constant over the entire spectrum from 0.0 to 0.6.

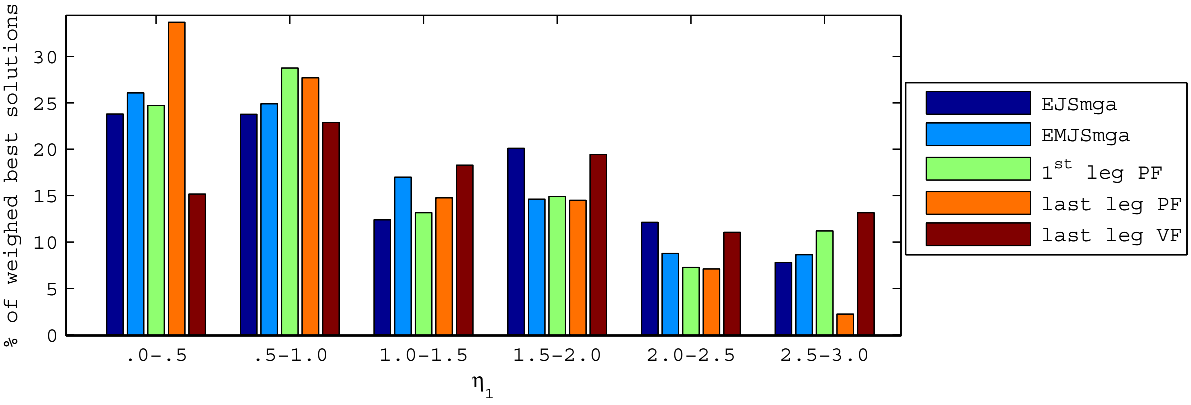


Figure 11.12: The distribution of good results over different regimes of η_1. See text for explanation.

Figure 11.12 seems to indicate that the good region for η_1 is below 1.0.

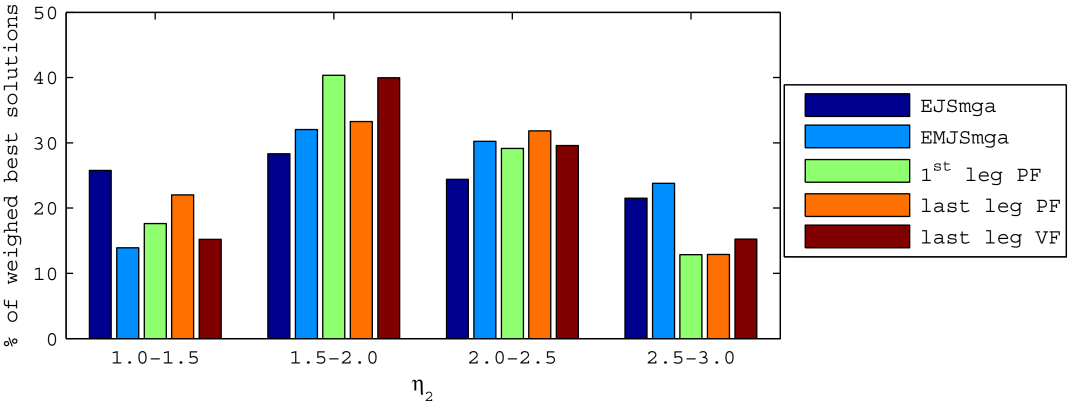


Figure 11.13: The distribution of good results over different regimes of η_2. See text for explanation.

Figure 11.13 indicates that η_2 should be around 2.0.

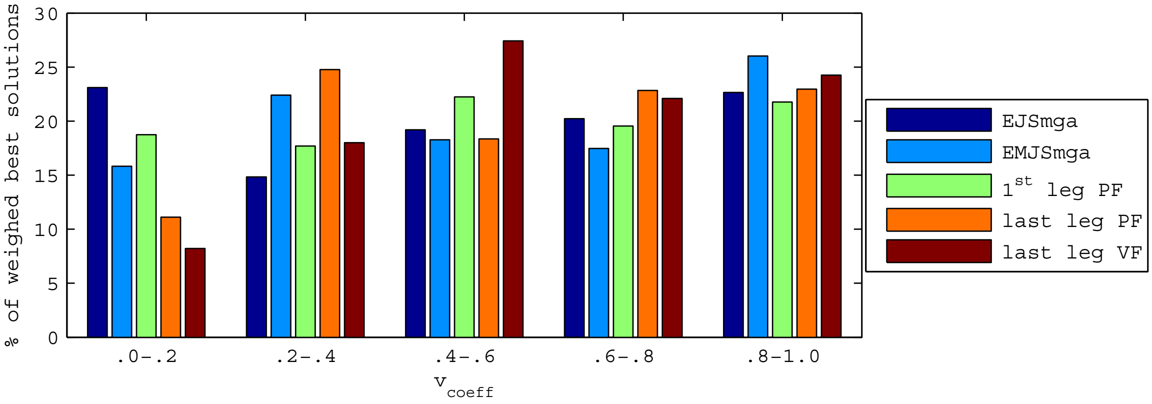


Figure 11.14: The distribution of good results over different regimes of v_{coeff}. See text for explanation.

Given the fairly equal spread in Figure 11.14, the results do not appear to be very sensitive to v_{coeff}.

Figure 11.15 clearly indicates that variant 1 and 2 are better than variant 5.

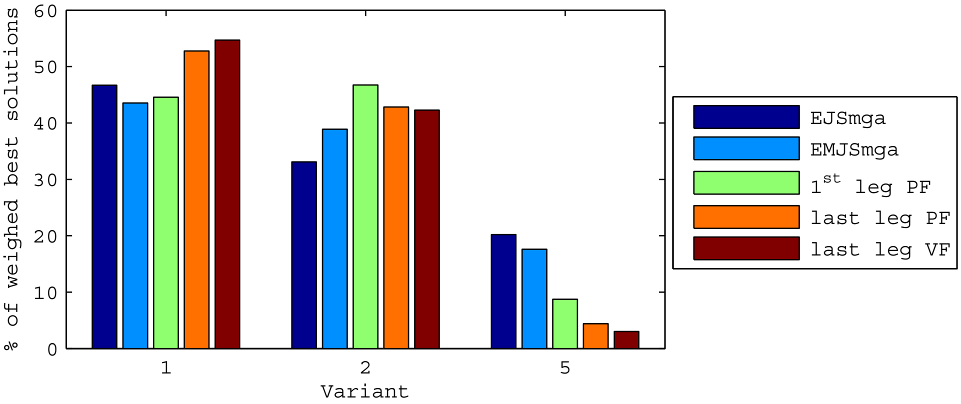


Figure 11.15: The distribution of good results over the different variants. See text for explanation.

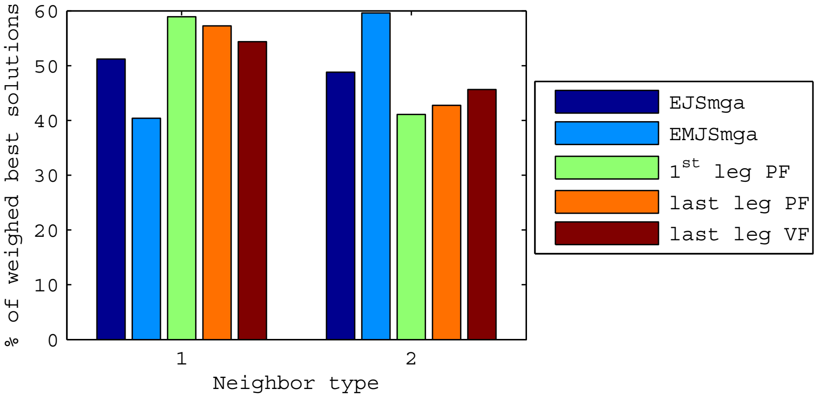


Figure 11.16: The distribution of good results over the different neighbor types. See text for explanation.

Figure 11.16 shows that both neighbor types appear to perform equally well.

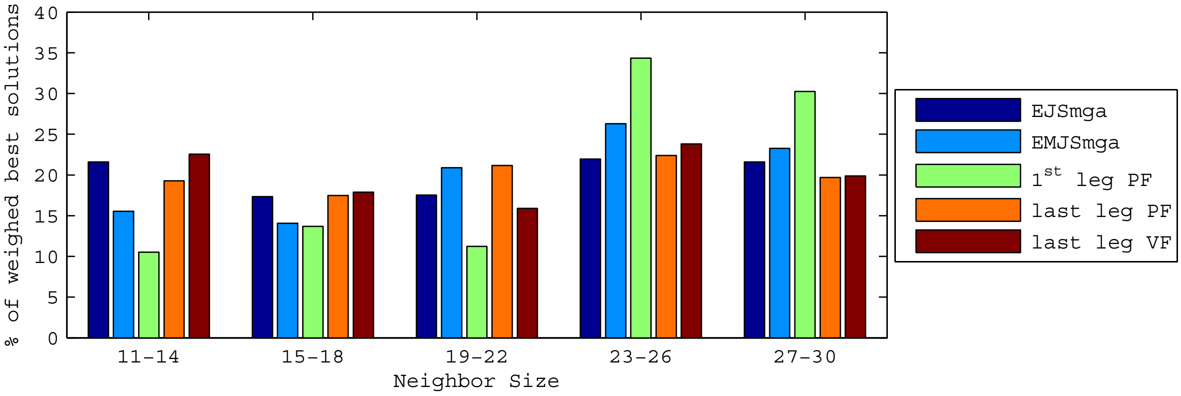


Figure 11.17: The distribution of good results over different regimes of the neighbor size. See text for explanation.

Figure 11.17 seems to indicate that the neighbor size should be about 25, but also that it is fairly well spread over the entire range of values.

In general it is noted that the results are not very clear regarding the performance of different parameter regimes. Apparently PSO is relatively insensitive to these tuning parameters when considering these problems. Because all problems considered are still very simple, some more difficult problems are considered in the third phase to avoid premature judgement.

11.4 Third Phase

The following problems are considered in this third phase:

- EVEJS MGA,
- Cassini2 First Leg VF,
- Cassini2 First And Last Leg PF,
- Cassini2 Last Two Legs Dsm VF,
- Cassini2 VF, reduction to box of 0.01 around the best putative solution in this thesis.

Given the large computational effort required for these problems, the number of random samples was reduced from 500 to 200. This reduces the statistical certainty in the random samples, but was the only way to obtain results in a reasonable amount of time. The results of this analysis have to be considered slightly less accurate and stable. The overall trend does help significantly in determining the most promising regions of PSO for the fourth phase.

The processed results are given in Figures 11.18 to 11.25. The 'raw' results are again available in Appendix C.

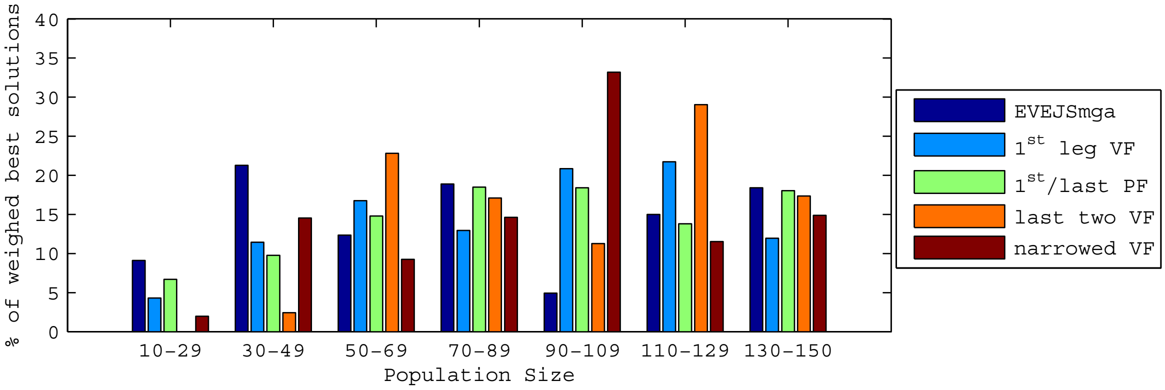


Figure 11.18: The distribution of good results over different regimes of the population size. See text for explanation.

From Figure 11.18 it appears that the best region in terms of population size for these more difficult problems is between 50 and 150 individuals. On the easier problems the best region was between 10 and 70 individuals. Hence a decision was made to include 60 and 140 individuals for the grid search in the next phase. Similar to the DE tuning, the optimal number of individuals will be determined once the other settings have been determined.

Figure 11.19 seems to indicate that the best value for ω is somewhere around 0.3-0.6, which is the opposite of what Figure 11.11 indicated. Hence both a value of 0.2 and 0.5 will be used in the next phase.

Figure 11.20 seems to indicate that the good regime is between 0.5 and 1.0. Since this was also the conclusion in Figure 11.12, a value of 0.8 will be used in the next phase.

Figure 11.21 indicates that η_2 should be around 2.0. This was also the conclusion in Figure 11.13 and hence a value of 2.0 is used for η_2 in the next phase.

The large difference in spread in Figure 11.22 seems to indicate that v_{coeff} is highly problem-dependent. Hence a grid with 0.1, 0.4 and 0.7 will be used in the next phase to get a better understanding of this effect.

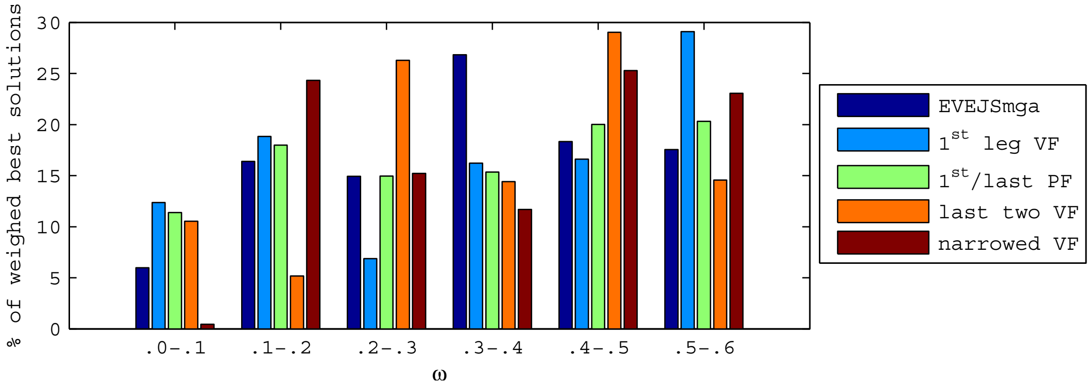


Figure 11.19: The distribution of good results over different regimes of ω. See text for explanation.

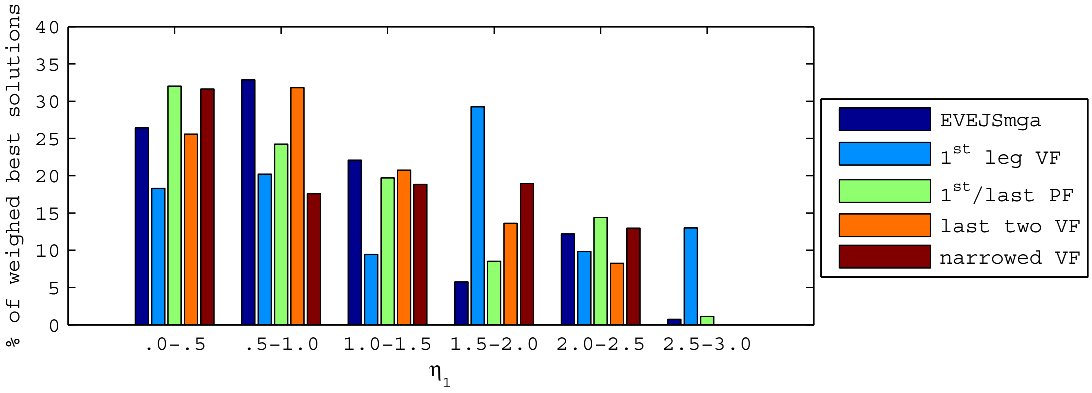


Figure 11.20: The distribution of good results over different regimes of η_1. See text for explanation.

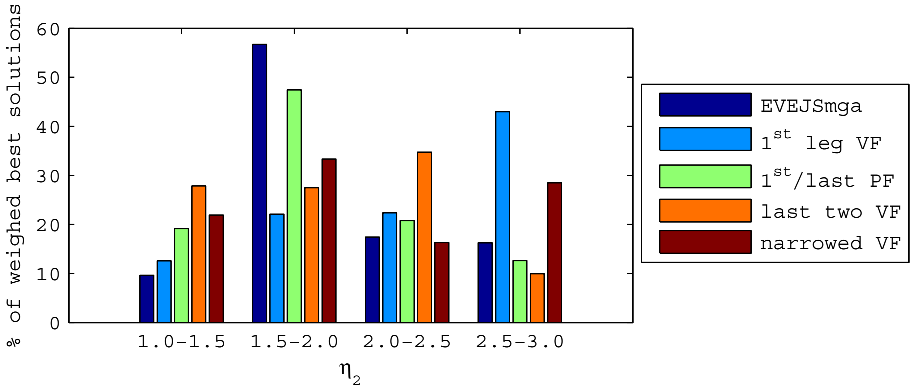


Figure 11.21: The distribution of good results over different regimes of η_2. See text for explanation.

Figure 11.23 shows that variant 1 is typically better than variant 2, although the difference is not very large. Because this result was also found in all previous simulations, variant 1 is selected in favor of variant 2.

Figure 11.24 shows that neighbor type 2 performs slightly better, although the results do not appear to be very sensitive to the neighbor type. Neighbor type 2 is hence selected for the next phase.

The results of the neighbor size in Figure 11.25 are difficult to interpret, because only 100 samples were performed in which neighbor type 2 was selected and hence this neighbor size was active. Taken this into account there does not appear to be a strong correlation between the results and the neighbor size. The slightly larger neighbor sizes do appear to be dominant though, which is in accordance with the results

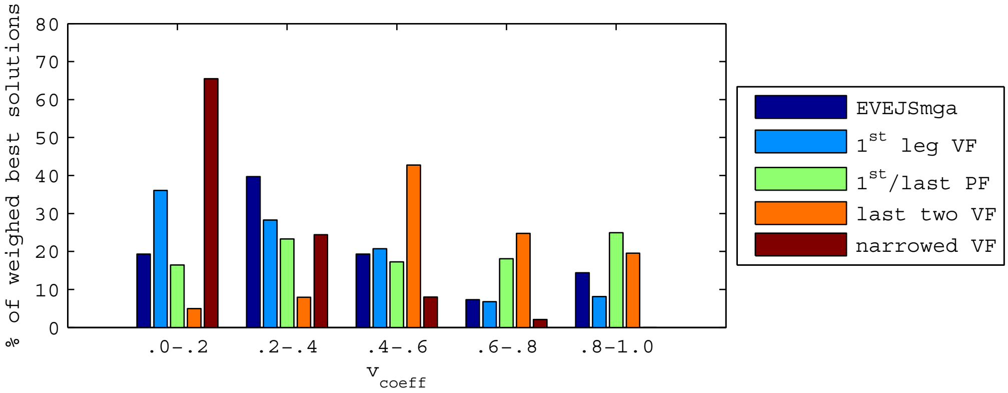


Figure 11.22: The distribution of good results over different regimes of v_{coeff}. See text for explanation.

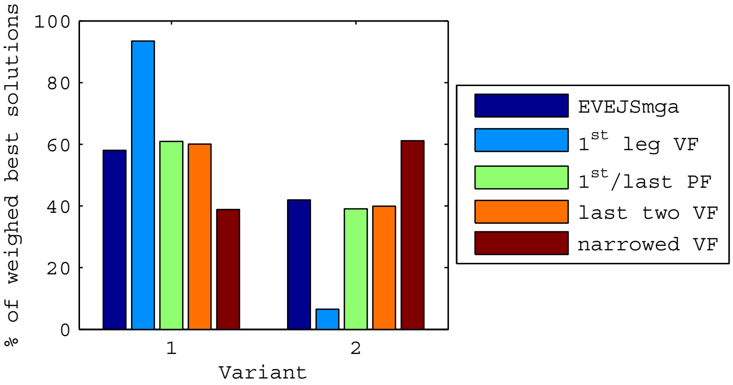


Figure 11.23: The distribution of good results over the different variants. See text for explanation.

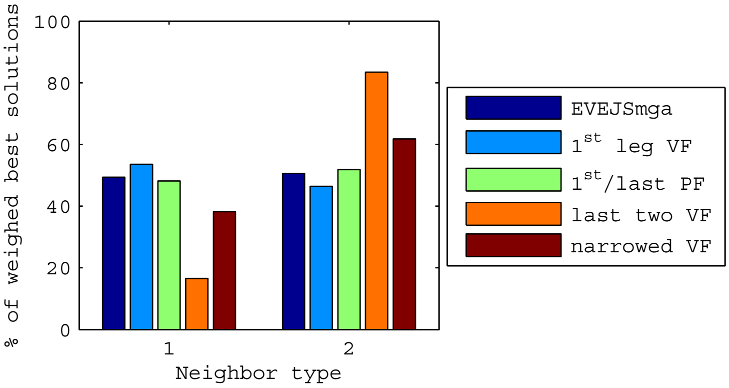


Figure 11.24: The distribution of good results over the different neighbor types. See text for explanation.

in previous simulations. Hence a neighbor size of 25 is selected for the next phase.

The resulting regimes for the fourth phase can now be seen in Table 11.5.

Table 11.5: The set of values that will be used in the fourth PSO tuning phase.

n_{pop}	ω	η_1	η_2	v_{coeff}	var	$neigh_{type}$	$neigh_{size}$
60, 140	0.2, 0.5	0.8	2.0	0.1, 0.4, 0.7	1	2	25

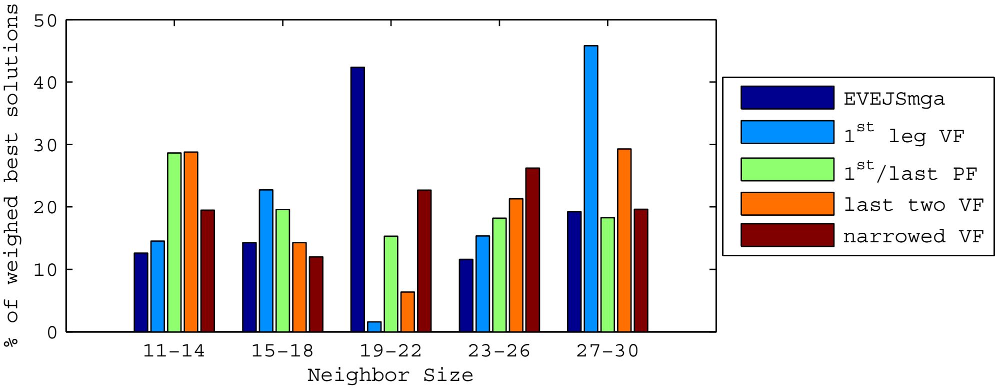


Figure 11.25: The distribution of good results over different regimes of the neighbor size. See text for explanation.

11.5 Fourth Phase

The following problems are considered for the fourth phase:

- Problems reducing the complexity of the trajectory model:
  - EVEJS MGA,
  - Cassini2 MGA.
- Problems reducing the number of parameters:
  - Cassini2 First And Last Leg PF,
  - Cassini2 First And Last Leg VF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Two Legs PF,
  - Cassini2 Last Two Legs VF.
- Problems narrowing the bounds of the problem:
  - Cassini2 PF Narrowed 0.01,
  - Cassini2 VF Narrowed 0.01,
  - Cassini2 PF Narrowed 0.1,
  - Cassini2 VF Narrowed 0.1.

It is noted here that the Cassini2 DSM In Second Leg Only problems were too difficult for the tuning phase. Also both the Cassini2 and Cassini1 MGA problems required many function evaluations. Hence only the Cassini2 MGA problem is included here.

These problems were solved using the settings in Table 11.5. The results can be seen in Figure 11.26.

The results of this analysis are very difficult to interpret. Especially the bad performance of the higher values for v_{coeff} stand out for the problems with narrowed bounds.

To get a better understanding of the other trends, the various problems are combined depending on their problem category. In doing so the earlier described scoring scheme is used, which gives a 1 for the best performing setting on a problem and a 0 for three times the best number of evaluations or more. The results of this process can be seen in Figure 11.27.

Various trends are visible in Figure 11.27. In general the low value of v_{coeff} is ideal for most problems. Also for the model simplification problems and for the problems optimizing less legs the low value of v_{coeff} typically performs well. Also it appears that ω=0.5 typically performs best. The results are less clear for the population size. This will be investigated in Section 11.5. The settings for the other tuning parameters are given in Table 11.6.

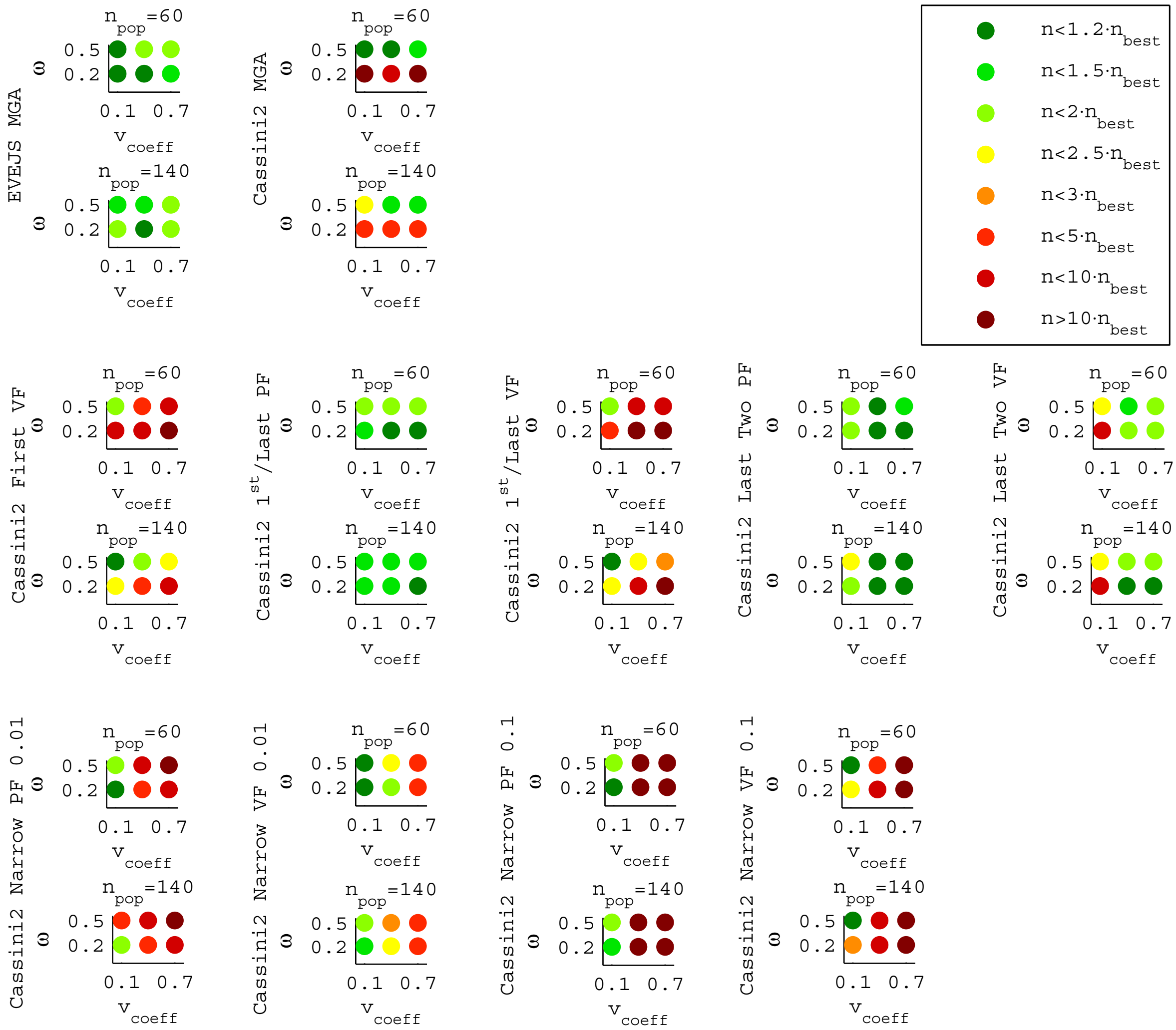


Figure 11.26: The performance of different settings of ω, v_{coeff} and n_{pop} of PSO for different problems. See text for explanation.

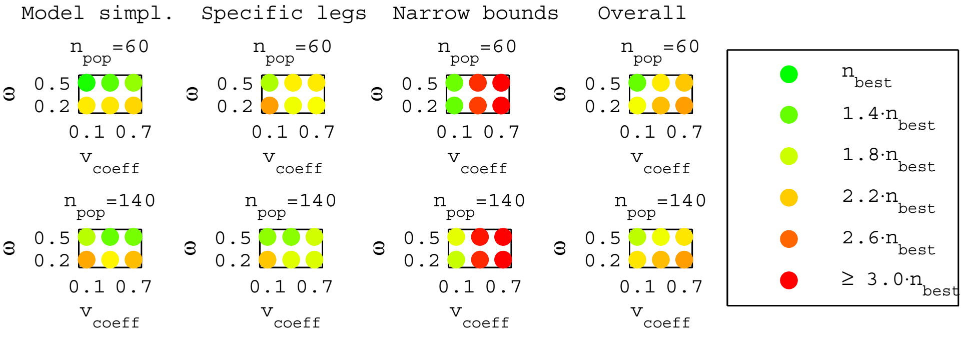


Figure 11.27: The performance of different settings of ω, v_{coeff} and n_{pop} of PSO on different problem types. See text for explanation. Note that n in the legend is identical to $n_{95\%}$.

Table 11.6: The set of values that will be used in the fifth PSO tuning phase.

ω	η_1	η_2	v_{coeff}	var	$neigh_{type}$	$neigh_{size}$
0.5	0.8	2.0	0.1	1	2	25

11.6 Fifth Phase: Determining the Population Size

In this final tuning phase, the ideal population size will be determined depending on the problem considered. The following grid of 10 population sizes will be used to determine the ideal population size: 15, 24, 36, 50, 70, 90, 115, 140, 170, and 210. All problems that can be optimized in reasonable time are considered for this analysis. These are:

- Problems reducing the complexity of the trajectory model:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA,
  - Cassini2 MGA.
- Problems reducing the number of parameters:
  - Cassini2 First Leg PF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF,
  - Cassini2 First And Last Leg PF,
  - Cassini2 First And Last Leg VF,
  - Cassini2 Last Two Legs PF,
  - Cassini2 Last Two Legs VF.
- Problems narrowing the bounds of the problem:
  - Cassini2 PF Narrowed 0.01,
  - Cassini2 VF Narrowed 0.01,
  - Cassini2 PF Narrowed 0.1,
  - Cassini2 VF Narrowed 0.1.

It is noted that the Cassini1 MGA and Cassini2 MGA problems drive the computation time required. Hence only the Cassini2 MGA problem is considered. Again the Cassini2 DSM In Second Leg Only problems are too difficult to be used in the tuning phase.

The results of the trajectory model simplification problems can be seen in Figure 11.28. The results of the specific leg problems are given in Figure 11.29. Finally the results of the narrowed bounds problems are given in Figure 11.30.

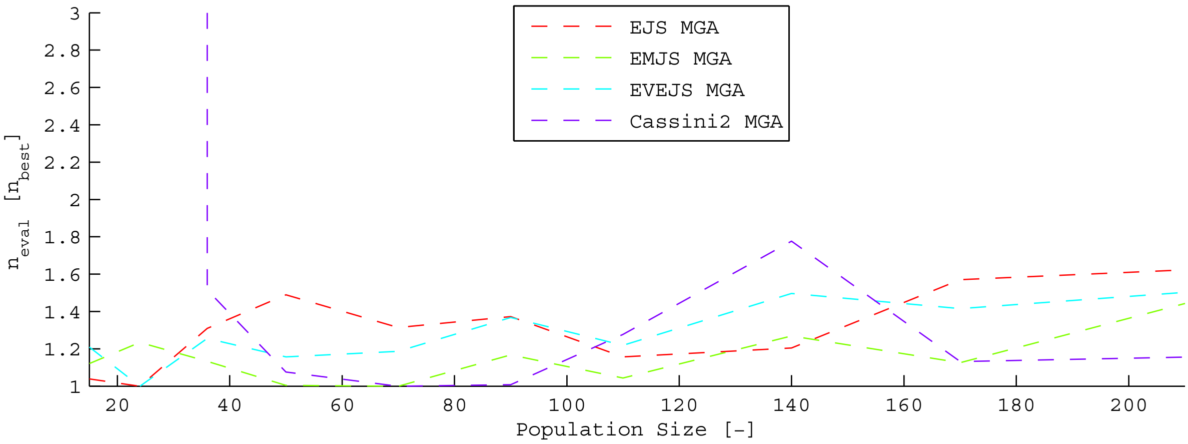


Figure 11.28: The performance of different population sizes of PSO on different trajectory model simplification problems.

A remarkable trend is visible in Figure 11.28. The performance is namely practically constant over a very large range of population sizes. Only for the Cassini2 MGA problem, a clear minimum population size of about 40 or 50 is required. The performance on these problems barely drops over the entire range up to a population size of 210.

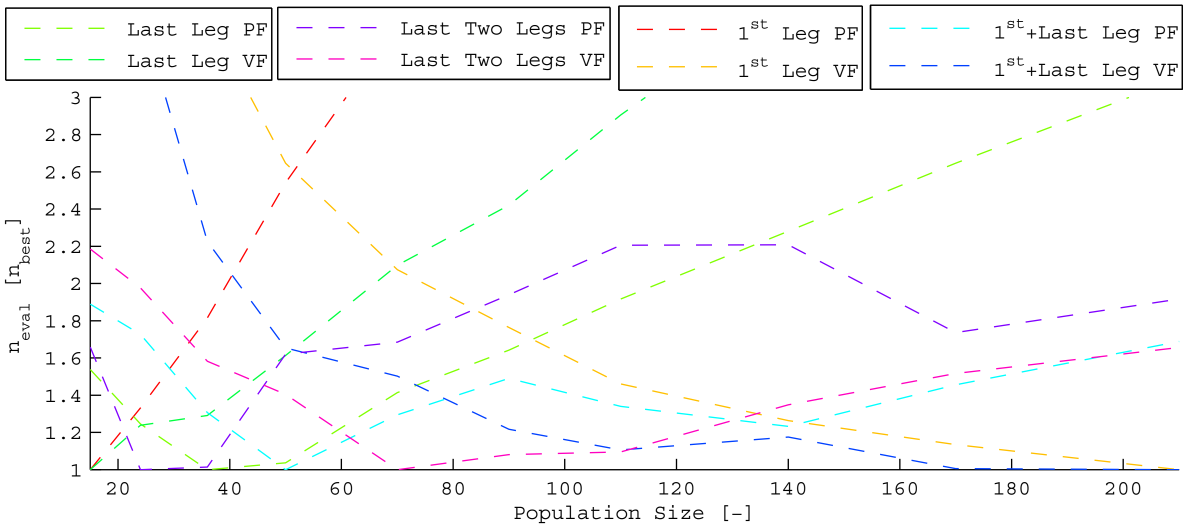


Figure 11.29: The performance of different population sizes of PSO on different specific leg problems.

It can be seen in Figure 11.29 that the population size has more influence on the specific leg problems. Still in general a larger population size is better for many problems. A significant performance penalty is only present for the problems that can be solved easily, the Cassini2 Last Leg PF, Last Leg VF and First Leg PF problems. These problems are namely solved in only 1000-5000 evaluations, whereas some other problems require up to 500,000 evaluations. Hence the performance penalty is deemed less important for these 'easy' problems.

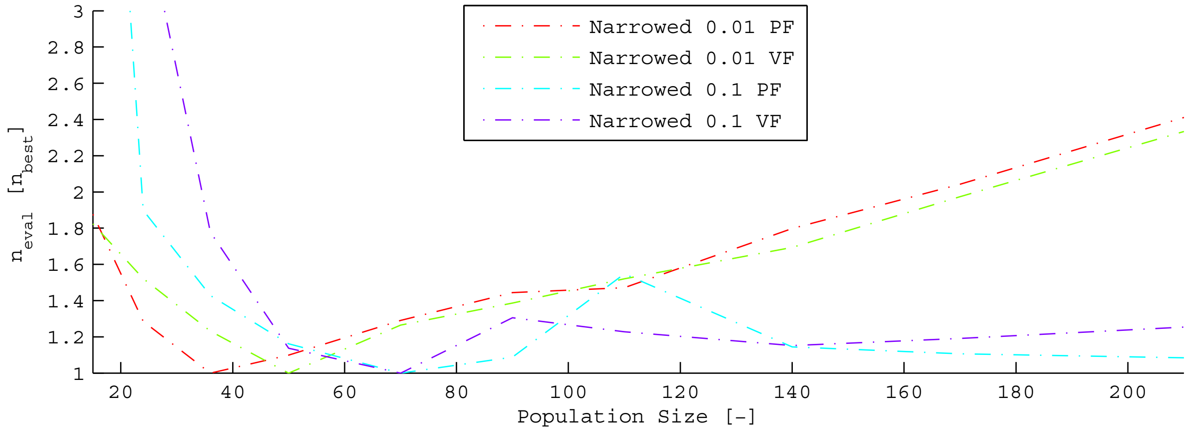


Figure 11.30: The performance of different population sizes of PSO on different problems with narrowed bounds.

Figure 11.30 again shows that the performance penalty of using a larger population size is small. This performance penalty is present for the problems with very narrowed bounds, but not for the problems with wider bounds. Since these are likely to resemble the full difficulty problems the most, this result is deemed more important.

In general it is noted that it is very difficult to determine the ideal population size. It was noted though that the performance penalty is very small if a population size is selected that is larger than the ideal population size. This performance penalty is largest for easy problems. On the other hand, if the population size is chosen too small it appears that no solution will be found for some problems. Hence it makes sense to define a minimum population size that is significantly larger than the threshold at which a population becomes too small.

In case the population size is set to $10 \cdot n_{var}$, all difficult tuning problems are roughly in the region in which they perform well. Hence this empiric relation will be used. An overview of the accompanying performance penalties caused by this employing this relation is given in Table 11.7.

Table 11.7: Performance penalty caused by using $10 \cdot n_{var}$ as a population size instead of the ideal population size. Note that problems flagged with * are very easy to solve and hence considered less important for determining the population size.

Problem	Penalty [%]		Problem	Penalty [%]	
EJS MGA	15		Cassini2 First And Last Leg PF	41	
EMJS MGA	10		Cassini2 First And Last Leg VF	16	
EVEJS MGA	16		Cassini2 Last Leg PF	4	*
Cassini2 MGA	0		Cassini2 Last Leg VF	61	*
Cassini2 First Leg PF	154	*	Cassini2 Last Two Legs PF	99	
Cassini2 First Leg VF	165		Cassini2 Last Two Legs VF	9	

11.7 Conclusion

The performance of PSO was found to be sensitive to various of its tuning parameters. This sensitivity was however not present in the easy problems. Because PSO has a large number of tuning parameters, and because it performed significantly worse than DE, the tuning process was more difficult. It is also noted that the apparent best settings may differ from the actual best settings, as bad decisions are much more likely in this tuning process than in the tuning process of DE.

That said, Table 11.8 gives an overview of the apparent best settings for PSO.

Table 11.8: The apparent best settings for PSO, as determined in this chapter.

n_p	ω	η_1	η_2	v_{coeff}	var	$neigh_{type}$	$neigh_{size}$
$n_p = 10 \cdot n_{var}$	0.5	0.8	2.0	0.1	1	2	25

Chapter 12

Repeated Local Searches

This chapter discusses the selection of local algorithms for performing repeated local searches. To this end Section 12.1 gives a description of the local optimization algorithms implemented in PaGMO. Subsequently Section 12.2 discusses the first phase of the comparison between various local optimization algorithms. The best two algorithms are subsequently applied to more difficult problems in Section 12.3. Finally conclusions are given in Section 12.4.

12.1 Local Optimization Algorithms in PaGMO

This section describes the local optimization algorithms that were used in this thesis. As discussed in Section 6.3.4, not all local optimization algorithms that were available in PaGMO could be linked successfully. Also some methods imported via the GSL library turned out to behave very unstable. Among them were all three variants of the Nelder-Mead algorithm and also the second variant of the Broyden-Fletcher-Goldfarb-Shanno algorithm.
In total this left the author a set of six algorithms that were relatively stable, which are described very briefly below.

BFGS: Broyden-Fletcher-Goldfarb-Shanno is a quasi-Newton method. By employing the difference between successive gradient vectors, BFGS builds up an approximation of the second-order derivatives of the objective function. It subsequently combines the derivatives to make steps towards the minimum while assuming quadratic behavior in that region. [GNU, 2012]

BOBYQA: Bound Optimization BY Quadratic Approximation also approximates the function by iteratively building a quadratic approximation for the objective function. [Powell, 2009]

COBYLA: Constrained Optimization BY Linear Approximation was developed by the same author as BOBYQA. As the name suggests, the main difference is that it constructs a linear approximation of the search space while searching for the minimum. Note that some small changes have been made in the implementation with respect to the orginal paper. [Johnson, 2012]

FR: Fletcher-Reeves is a conjugate gradient algorithm that proceeds as a succession of line minimizations. [GNU, 2012]

PR: Polak Ribiere is another conjugate gradient algorithm that is very similar to FR. [GNU, 2012]

SBPLX: SBPLX, the NLOPT implementation of subplex, is a special variant of Nelder-Mead. It uses a simplex to ‘run’ over the search space looking for the minimum. [Johnson, 2012].

12.2 First Phase

This section will compare the performance of the six local optimization algorithms, which were discussed in Section 12.1, when applied to various problems. A simple repeated local search algorithm is employed, which is also referred to as a Multi-Start (MS) scheme. In this thesis the simplest MS scheme is considered, in which a large number of random initial points are used as starting points for the local optimization algorithms. It is noted that the performance may be improved slightly by employing a smarter distribution, such as a Latin Hypercube distribution.
Two different versions are employed. One in which the local optimization runs are run just once on a certain point. In the second version the search is followed by a second search starting from the final point of the first search. It is sometimes reported that local optimization algorithms may benefit for a second restart. This restart is performed until the gain of using the restart was less than 10 m/s.

The following nine problems were selected for the initial tuning phase:

- Trajectory model simplification problems:
  - EJS MGA,
  - EMJS MGA,
  - EVEJS MGA.
- Specific leg problems:
  - Cassini2 First Leg PF,
  - Cassini2 First Leg VF,
  - Cassini2 Last Leg PF,
  - Cassini2 Last Leg VF.
- Narrowed bound problems:
  - Cassini2 DSM PF Narrowed 0.01,
  - Cassini2 DSM VF Narrowed 0.01.

The different local optimization algorithms were started a large number of times at a random location and subsequently asked to optimize the problem. The number of successful runs was subsequently determined and also the number of evaluations required for that on average. From these figures, $n_{95\%}$ could be calculated in a similar fashion as was done for the global optimization algorithms. Similarly a threshold of 50 m/s is used to define success. 10000 samples are performed for most problems, but the narrowed bounds problems and the Cassini2 First Leg VF problem were only optimized 1000 times because of the required computational effort.
The same random locations were used for all algorithms to allow for a fair comparison. The resulting values for $n_{95\%}$ are given in Tables 12.1 and 12.2 for the algorithms without a restart and with a restart at the best location respectively. Note that the success rates and the average number of evaluations that led to these values for $n_{95\%}$ are not included here for the sake of preserving space. They can be accessed in Appendix D.

Various conclusions are drawn from Tables 12.2 and 12.2. First of all it is noted that PR, FR, COBYLA and BFGS heavily oversample the search space belonging to the simple low-dimensional problems. This results in the large $n_{95\%}$ reported therein. Also it is noted that both SBPLX and BOBYQA perform well on the easy problems, but do not manage to find good solutions for the narrowed bounds problems. It is noted that the performance of all algorithms considered is very bad on the multi-dimensional problems. BOBYQA and SBPLX do not manage to find good solutions frequently, whereas PR, FR, COBYLA and BFGS oversample the search space thereby leading to a large number of required evaluations.
Finally it is noted that typically an MS scheme without restart has a better $n_{95\%}$ although it is also noted that various algorithm-problem combinations showed improved performance by the additional restarts.

SBPLX will be subjected to various more difficult problems in the next phase, given its high performance on the simple problems and the fact that it had a non-zero chance of optimizing both narrowed bound problems. Also BFGS will be subjected to these more difficult problems as the best performing algorithm

Table 12.1: The values for $n_{95\%}$ $[\cdot 10^3]$ for various problems using various local algorithms in an MS scheme without restarting at the best location. All problems were solved 10000 times, except for the Cassini2 First Leg VF problem, which was solved 1000 times. [1]: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	3.0	108	65	230	3.0	103
EMJS	6.6	98	94	100	4.2	117
EVEJS	31	-	-	-	-	47630
First Leg PF	2.6	-[1]	-[1]	7580	17.8	-[1]
First Leg VF	76	2630	2423	2630	-	3166
Last Leg PF	4.0	3522	3274	143	5.1	3651
Last Leg VF	2.6	7287	7296	82	1.7	7600
PF Narrowed 0.01	1793	-	-	-	-	-
VF Narrowed 0.01	5715	773	512	760	-	929

Table 12.2: The values for $n_{95\%}$ $[\cdot 10^3]$ for various problems using various local algorithms in an MS scheme that restarts at the best location until no significant improvement is found anymore. [1]: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	3.3	188	93	176	3.3	150
EMJS	7.9	165	160	123	6.3	231
EVEJS	38	256032	-	5130	316	14666
First Leg PF	4.0	-[1]	-[1]	1659	4.5	-[1]
First Leg VF	43	3830	3326	-	4493	4240
Last Leg PF	5.6	852	990	119	4.2	625
Last Leg VF	3.4	1131	1498	110	1.8	1141
PF Narrowed 0.01	659	-	-	9760	997	-
VF Narrowed 0.01	6268	702	643	563	-	779

that assumes quadratic behavior. It is closely tied with PR and FR, but performed better on average.

Finally it is noted that all algorithms except SBPLX and BOBYQA showed unstable behavior for various of the more difficult problems. The exact reason is not clear. Possibly the algorithms try to evaluate points outside the search space on special occasions, which may subsequently result in infeasible trajectory calculations. This instability caused the algorithms to be less useful in optimizing various trajectories and became worse on the problems in the second phase, as will be further discussed in Section 12.3.

12.3 Second Phase

The following problems were selected for the second phase:

- Trajectory model simplification problems:
  - Cassini1 MGA,
  - Cassini2 MGA.
- Specific leg problems:
  - Cassini2 First And Last Leg PF,
  - Cassini2 First And Last Leg VF,
  - Cassini2 Last Two Legs PF,
  - Cassini2 Last Two Legs VF.
- Narrowed bound problems:
  - Cassini2 DSM PF Narrowed 0.1

  - Cassini2 DSM VF Narrowed 0.1

It is noted that BFGS was very unstable on all problems, except for the Cassini2 Last Two Legs VF problem and the First And Last Leg VF Problem. Still this instability may significantly influence those results. This instability issue of various algorithms will need to be fixed before they can be used in practice.

Table 12.3 gives an overview of the results obtained using SBPLX for both the MS version with and without restart.

Table 12.3: The values for $n_{95\%}$ $[\cdot 10^3]$ for various problems using SBPLX in an MS scheme with and without restart.

Problem	without restart	with restart
Cassini1 MGA	2702	2645
Cassini2 MGA	6303	6653
Cassini2 First And Last PF	61	89
Cassini2 First And Last VF	595	278
Cassini2 Last Two Legs PF	9350	14217
Cassini2 Last Two Legs VF	107	141
Cassini2 DSM PF Narrowed 0.1	>100,000	>100,000
Cassini2 DSM VF Narrowed 0.1	>100,000	>100,000

Given that the performance of SBPLX is orders of magnitude higher than that of DE, it is concluded that this SBPLX is not ideally suited to solve more complex problems. It is noted that more powerful local optimization algorithms will be required to solve these difficult trajectory optimization problems.
Given that the performance is also bad when sampling in a small region around the best optimum, a MBH scheme will not increase the performance substantially. MBH namely relies on the fact that the actual solution can be found reliably once a point close to the actual solution has been found already. The bad performance on the narrowed bound problems indicates that a MBH scheme with SBPLX will not be effective.

12.4 Conclusions

The GSL algorithms turned out to be unstable when subjected to more difficult optimization problems. This made them unsuited for optimizing difficult problems. It is unclear what causes this instability.
BOBYQA and SBPLX turned out to be fast algorithms in optimizing simple problems. Both algorithms however suffered from a significant loss in performance as the problems became more difficult. Given the bad performance on high-dimensional problems, these algorithms are also unsuited for optimizing difficult trajectory problems.
Future research should include more powerful local optimization algorithms, such as SNOPT. These do have a proven performance on trajectory optimization problems, as can be seen for instance in [Addis et al., 2008] or [Vasile et al., 2008].

Part IV

Results

Chapter 13

Search Space Analysis

This chapter will investigate the search space belonging to typical problems considered in this thesis report. First of all an introduction is given in Section 13.1. Subsequently the MGA problems are investigated in Section 13.2. Section 13.3 discusses various MGA-1DSM problems. Finally an overview of the conclusions is given in Section 13.4.

13.1 Introduction

A proper understanding of the problem and its accompanying search space may help significantly in determining the optimization strategy that will be adopted. This thesis compares the application and use of different trajectory models. Hence it is very interesting to see how the search space changes when more legs are added, or a different model is employed. Comparing the 'difficulty' of the search space across different trajectory models may also help in the choice for the trajectory model that is preferred for a certain problem.

There are many aspects that determine the difficulty of a certain optimization problem. These aspects include: [Myatt et al., 2004]

- the affordable number of function evaluations,
- the size of the basin of attraction of the global optimum,
- number of local minima (and their basin sizes),
- embedded or isolated global minima.

Each of these aspects will be discussed in the section below, before the structure of this chapter will be further outlined.

Affordable number of function evaluations

The problem difficulty increases if the evaluation time of one decision vector increases. A larger evaluation time means that less evaluations can be performed within a reasonable amount of time. This reasonable time is of course relative to the requirements of the engineer and the computational resources available.

Since the evaluation time is of the same order of magnitude across the different problems and different trajectory models, this is not a main differentiating element when comparing different problems. Of course when comparing between different trajectory models, the computational time required by the different models needs to be taken into account.

Size of Basin of Attraction

Of course an optimization technique has a larger chance on finding the global optimum if its basin of attraction is larger. The basin of attraction is defined as the region that will cause an optimization technique to eventually converge on the minimum. Many publications, among which [Myatt et al., 2004], define the size of the basin of attraction as the fraction of the search space from which a strictly descending method will converge to the global minimum. This is indeed a good measure to define the difficulty of finding the global minimum using a repeated local search. It does however not necessarily give a good indication of the difficulty of finding the global minimum using evolutionary algorithms. In this case basically the fraction of the global basin that is lower than that of other minima is more interesting. This difference is illustrated in Figure 13.1.

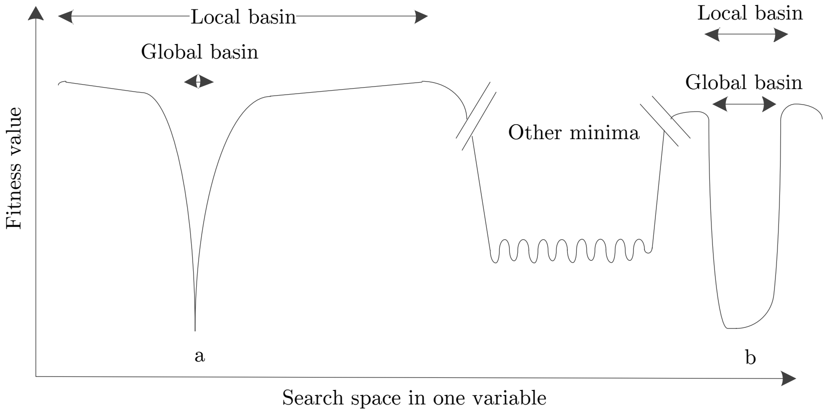


Figure 13.1: Difference between the relevant basin size for local and global optimization techniques illustrated by two example minima.

It can be seen in Figure 13.1 that the basin of attraction from which a strictly descending technique will find the minimum is comparatively larger for minimum a than for minimum b. On the other hand a global evolutionary optimization algorithm will struggle heavily to find minimum a, because the chance of obtaining a low fitness value within the basin of attraction is low. The basin has very steep sides and only a very small region has low fitness values. Hence the global algorithm will more likely find that its lowest fitness is located near another minimum and start converging towards the wrong minimum.

On the other hand minimum b has a very wide basin in which low fitness values are located. Hence a global evolutionary optimization algorithm will much more likely find a point located in the lower parts of the basin and subsequently start converging towards the global minimum.

This discussion leads to two different definitions of the basin. From now on the ‘common’, local optimization oriented, definition of the basin of attraction will be referred to as the local basin. The global evolutionary optimization algorithm oriented basin will be referred to as the global basin. Both have very different means of determining their size, as will be discussed in the next paragraphs.

Local Basin

In theory the local basin size can be found by sampling the domain uniformly with infinitely small steps and using a strictly descending local minimization technique with infinitely small step size. The fraction of successes obtained with such a method represents the fraction of the search space that is part of the basin of attraction of the global optimum.

In practice this can become very difficult if the problem consists of a large number of decision variables, if it contains a large number of minima, or if the search space is very sensitive. In those cases, one may choose to reduce the size of the domain in which the samples are taken. To be able to do so, one must have a good idea of the size of the basin of attraction in all parameters. If one prunes too much of the total search space, the basin of attraction will be underestimated; parts of the search space will be pruned that do belong to the basin of attraction.

Another problems in determining the size of the basin of attraction is that local optimization algorithms are never strictly descending along the steepest gradient of the search space contours because an infinitely small step size is not possible in practice. Also many local optimization algorithms may converge prematurely. Depending on the frequency and consequence of both events, the results may be distorted significantly.

Global Basin
Unlike the local basin, it is much more difficult to come up with a good definition of size of the global basin. One option could be to define the global basin as the fraction of the search space that is fitter than the fitness at which the total basin of equal fitness of other minima becomes equally large. This idea is schematically explained in Figure 13.2.

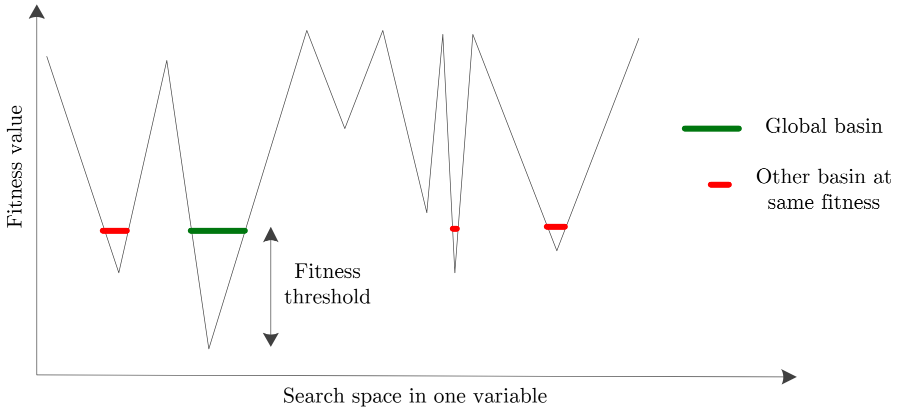


Figure 13.2: Schematic illustration of a definition of the global basin. Note that at the given threshold the total size of the other minima is equally large as that of the global basin. If one moves further away from the global minimum, the other minima will become larger attractors of global optimization methods.

The threshold at which this happens is difficult to determine in practice. It will differ from problem to problem and knowledge of the entire search space would be required, which is impossible in practice. Hence it can only be determined using an educated guess. For most problems suboptima have been encountered at ΔV's that were about 100-200 m/s worse. Some problems have suboptima that are even closer, some have suboptima at a larger difference in fitness. Hence a more practical definition of the global basin size is used here. It is defined as the fraction of the search space at which fitness values within 200 m/s of the optimum can be found.

Still this definition is very difficult to determine for high-dimensional problems. Hence for the first-order estimate in this thesis work, the size of the suboptima will be determined for each parameter separately. This is done by sampling each parameter at many different distances from the optimum and measure the deviations in fitness values from the optimum. By calculating the intersection with the fitness value line of 200 m/s above the global optimum, the size of the global basin can be determined for that parameter. Note that each variable is perturbed in two directions (positive and negative) to do so.
Subsequently the total size is then estimated by assuming that the global basin is a hypercube with side lengths equal to the basin sizes determined for each parameter.

Apart from determining the size of the global basin, this analysis also offers good insight into the sensitivity of each parameter and how the deviation in fitness value increases when the perturbation from the optimum grows. Optimum in this sense may refer to the global or a local optimum. Typically the global optimum will be analyzed in the following, but sometimes also the size of a suboptimum is analyzed to explain consistent misconvergence that occurs for some problems.

It is noted explicitly that this method does not take into account the interaction between different parameters. This is a large assumption that is not validated. It is however not straight-forward to determine the size of the basin otherwisely. One option would be to use a large number of random directions in which a similar analysis will be performed as described above. By processing the sizes in each direction in a proper way, a better estimate of the size may be determined.
Another idea would be to perform a sensitivity test, such as Fourier Amplitude Sensitivity Testing (FAST). Another option would be to use Taguchi methods to determine the effects of the relations between parameters. [Stracquadanio et al., 2011] already performed such a sensitivity analysis using

FAST. It provides a good overview of the parameters to which the problem is most sensitive and to what extent the influence is caused by interaction with other parameters. By itself it does however not help either in determining the size of the basin of attraction. Instead of determining the sensitivity, the perturbation at which a certain fitness penalty is reached should be determined.
As a final idea it may be interesting to perform a sensitivity analysis on the resulting global basin size, rather than the fitness value. This may solve the issues raised in the previous paragraph, but severely complicates the analysis.

Number of Local Minima

Typically the difficulty of the problem increases if more minima are present in a problem. If more minima are present, the chance on misconvergence will typically increase. Also the number of local minima determines the applicability of various optimization techniques. Local optimization techniques benefit from problems with a small number of local minima, whereas nature-inspired metaheuristics such as DE, GA and PSO typically perform comparatively better if the search space includes a large number of minima.

The total number of minima may be difficult to determine. Consider for instance a valley with an oscillatory surface. This may yield many minima adjacent to each other. This situation is difficult to distinguish from many point/single funnel minima, whereas the implications may be very different. Also a large number of minima may exist at very high fitness values, which is irrelevant for global optimization techniques. Finally it is noted that one is typically interested in the size of the basins of attraction of all these minima, rather than just the number of minima.
A further practical issue in determining a minimum is that there exists a large number of equally optimal solutions to most MGA-1DSM problems. In case a leg exists in which no DSM takes place, η has no contribution to the fitness value for a VF model. Similarly a large number of combinations of η, r, ϕ and θ exist with exactly the same fitness for a PF model. Hence a minimum can not simply be determined by comparing the decision vectors of different minima.
Another difficulty in obtaining a representative set of minima is that a very strong local optimizer is required for that. This poses a practical issue because the local optimizers used in this thesis tend to frequently converge prematurely and show unstable behavior.

Although not further investigated in this thesis, an analysis of the total number of local minima across different problems may lead to interesting conclusions which may help in drafting better optimization strategies for optimizing trajectories.

Embedded or Isolated Minima

In case the minima of a problem are close to each other, finding one minimum may aid in finding a better minimum and eventually the global minimum. In case all minima are isolated, the optimization techniques will not be able to find other optima if it starts converging towards one minimum. If combined with a large number of minima, this will severely increase the problem complexity.

The embeddedness of these minima may be determined by calculating the distance to other minima. To do so first the search space is converted to a hypercube with sides from 0 to 1 in all dimensions. Subsequently the average distance to other local minima may be determined for each variable. An interesting approach was followed by [Vasile et al., 2009], in which different levels were defined based on the fitness value of the minimum. Subsequently the average distance with respect to other minima in the same level was calculated, and the average distance to minima of one level deeper was calculated. This gives a measure of the difficulty to 'hop' from a certain minimum to other minima of the same level and of better levels. In practice this is of course dependent on more aspects.
Again also the imperfections of local optimizers and the very large size of the search space pose practical issues in determining the number of local minima. Specifically for MGA-1DSM problems the same practical issue is present when differentiating between different optima remains.

No further investigation is performed in determining the embeddedness of minima in this thesis, but it is noted that it may be an interesting topic for future research. A better understanding of the spread of minima may lead to improved optimization strategies.

Outline

The following sections will investigate the search space around the best solutions to different problems. Also occasionally suboptima will be investigated to verify the previous discussion on the global basin.
Main goals are to estimate the size of the local basin and the global basin and to discuss the effects of these basin sizes. The local basin will be investigated only for a part of the MGA problems, given the large computational cost at increasing dimensions. The global basin, and the related sensitivity of individual parameters, are determined for all problems.
First the MGA problems will be discussed in Section 13.2. Subsequently the DSM problems will be discussed in Section 13.3. Finally conclusions are given in Section 13.4.

13.2 MGA Problems

This section discusses the analysis of the search space corresponding to the MGA problems used in this thesis. To this end first the EJS MGA problem will be analyzed in Section 13.2.1. Subsequently the EMJS MGA problem will be analyzed in Section 13.2.2. Then the EVEJS MGA problem is analyzed in Section 13.2.3. Section 13.2.4 discusses the Cassini1 MGA problem and Section 13.2.5 discusses the Cassini2 MGA problem. Finally an overview is given in Section 13.2.6.

13.2.1 EJS MGA Problem

Local Basin

To get an estimate of the local basin, a large grid was set up with 50 steps in each of the three variables. Subsequently for each combination a local optimizer was started. In case the optimizer reached the best putative solution to within 50 m/s, it was counted as a success. In this way the average success rate can be calculated, as well as the average success rate for a certain parameter combination. This allows to make a plot of the success rate depending on different combinations of t_0, T_1 and T_2.
The simulations were performed with the two different local optimization techniques. First of all the SBPLX method of the NLOPT library, which is an implementation of the Nelder-Mead simplex method. This algorithm is not strictly descending, allowing it to possibly 'hop' over dents in the search space. Also the BFGS algorithm as implemented in the GSL library was used, which can be regarded more as strictly descending. It is noted that the local optimization algorithms were not restarted after they had converged.
The results of both simulations are visualized in Figures 13.3 and 13.4.

Various conclusions can be drawn from Figures 13.3 and 13.4. First of all it can be seen that all variables have a certain range from which convergence to the best putative solution is possible, especially for the BFGS algorithm. The BFGS algorithm has a zero chance of finding the solution when starting in a large part of the search space. Especially the plot between T_2 and t_0 shows a clear region from which the best putative solution can be found. This gives an interesting overview of the 3D search space of this problem, and especially the funnel that causes local optimization algorithms to converge to the best solution. If started from outside this region, the gradient of the search space apparently causes the local optimization algorithm to converge to a suboptimal solution.
Also a clear difference is visible between both local optimization methods. SBPLX manages to find the best putative solution from a much larger part of the search space than BFGS, and also from farther away from the best solution. As explained earlier, this may partially be caused by the fact that SBPLX is descending less strictly than BFGS. Sometimes this may cause SBPLX to find the solution in spite of the wrong starting point. Especially the narrow band of success at a t_0 of -700 MJD2000 is very remarkable in this respect, because it is located very far away from other areas of success and seems to

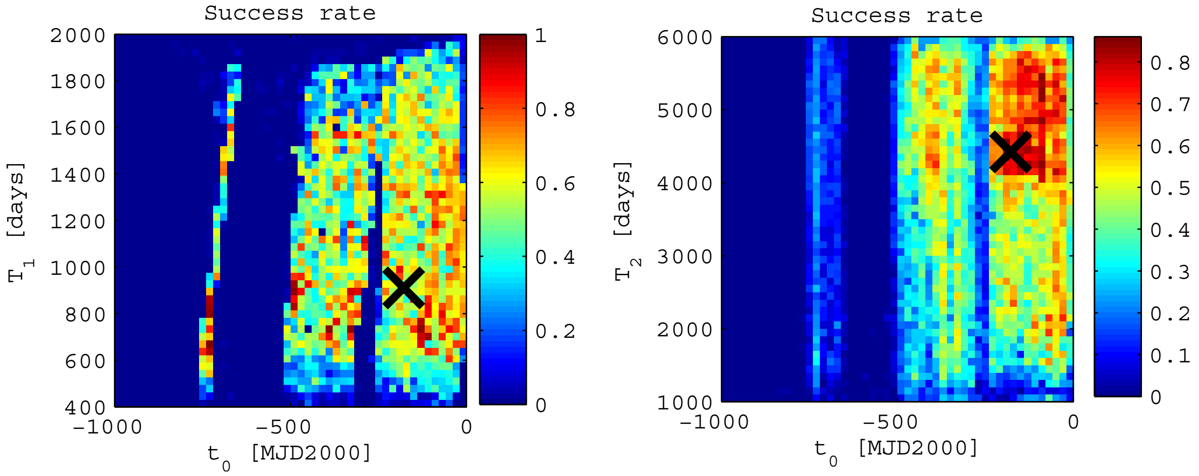


Figure 13.3: The success rate of SBPLX on the EJS MGA problem depending on the decision variables t_0, T_1 and T_2. The black cross indicates the location of the best putative solution.

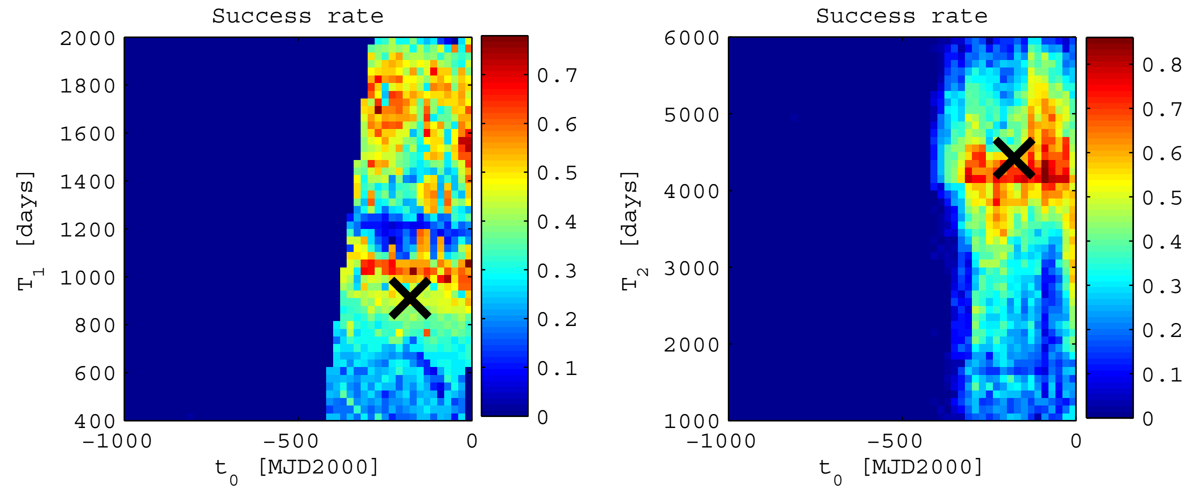


Figure 13.4: The success rate of BFGS on the EJS MGA problem depending on the decision variables t_0, T_1 and T_2. The black cross indicates the location of the best putative solution.

be unconnected.

The size of the local basin was also determined by calculating the average success rate in Figures 13.3 and 13.4. The results can be seen in Table 13.1. Note that also the results of the Monte Carlo analysis of 10^5 samples are shown. Note that for this problem both methods come up with exactly the same sizes.

Table 13.1: The sizes of the local basin in % of the total search space, calculated using various methods. Note that the numbers between brackets refer to the number of trials.

	BFGS	SBPLX
Grid Search (50×50×50)	12.9	22.7
Monte Carlo (10^5)	12.9	22.7

From the results in Table 13.1 it is concluded that the size of the local basin is very large still for the EJS MGA problem.

Global Basin

To get a first impression of the search space around the best putative minimum, the sensitivity to each design parameter was analyzed. Each variable was perturbed from the best putative minimum by varying amounts from 10^{-7} to 10^{-1} of the total search space of that variable. A total of 100 regular perturbation samples are taken in this way on a logarithmic scale. The change in objective value because of this perturbation is then stored. Subsequently this change in objective value is plotted against the size of the perturbation. This process was repeated for all three variables and performed in both the positive and negative direction. The results can be seen in Figure 13.5.

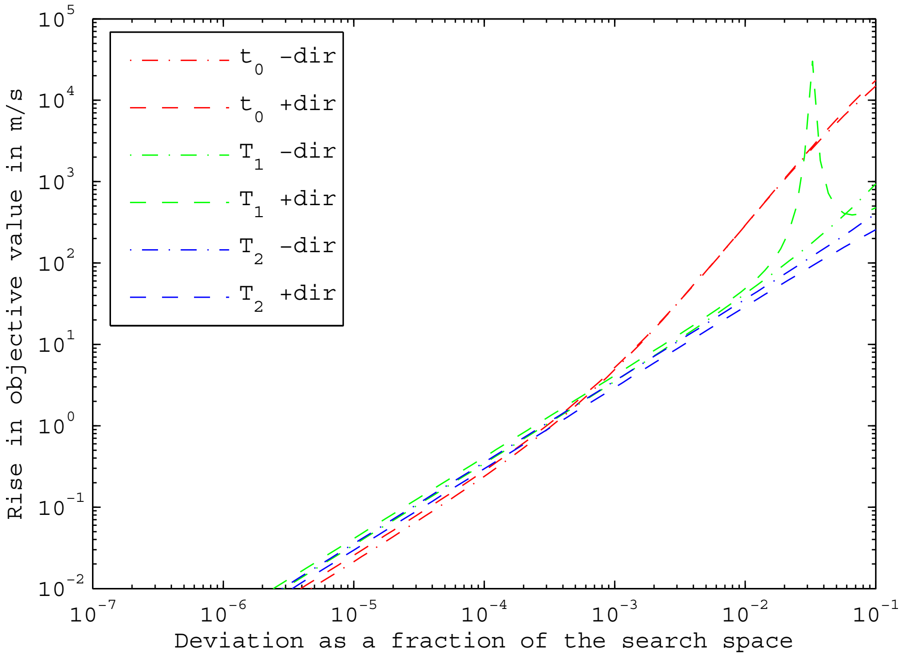


Figure 13.5: The results of the sensitivity analysis of the various variables of the EJS MGA problem. See text for explanation. Note that '-dir' and '+dir' represent the negative and positive perturbation direction respectively.

From Figure 13.5 it appears that all three parameters are almost equally sensitive very close to the best putative solution. Slightly further away t_0 becomes comparatively more influential. The deviation in objective value seems to increase about linearly with the perturbation size in T_1 and T_2. This is also the case for the first part of the perturbations in t_0 until a perturbation size of 10^{-3}, after which the effect increases at a higher rate than linear. Also the sudden peak in T_1 at a perturbation of 0.03 and the accompanying decrease afterwards is interesting. Apparently a shock event of some sort occurs there.

The global basin can be estimated from the previously discussed perturbation analysis. The maximum perturbation size at which the fitness penalty does not exceed 200 m/s is determined for each of the three parameters in both directions. This results in the sizes given in Table 13.2. Note t_0 is the most important driver in the size of the global basin.

13.2.2 EMJS MGA Problem

Local Basin

Similar to the EJS MGA problem, a grid was set up in all decision variables to determine the size of the local basin and the influence of all parameters. Because four variables are present in this problem, only 25 steps were considered in all variables to keep the computation time within acceptable limits.
Again both the SBPLX and BFGS method were used. The results can be seen in Figure 13.6 and 13.7

Table 13.2: The sizes of the global basin in % of the total search space, calculated for the different parameters in both directions. It is noted explicitly that all values are given as a percentage (not as a fraction) and one has to take care of that when calculating the overall basin size.

Variable	t_0	T_1	T_2	Overall
Positive Direction	0.82	3.6	5.3	
Negative Direction	0.82	2.0	7.5	
Total Size	1.64	5.6	12.8	0.012

and in Table 13.3.

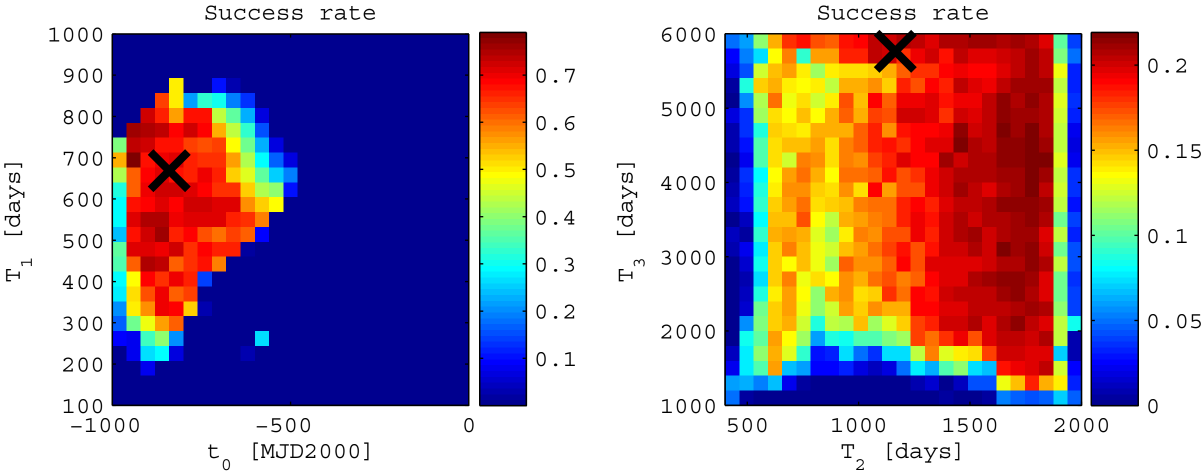


Figure 13.6: The success rate of SBPLX on the EMJS MGA problem depending on the decision variables t_0, T_1, T_2 and T_3. The black cross indicates the location of the best putative solution.

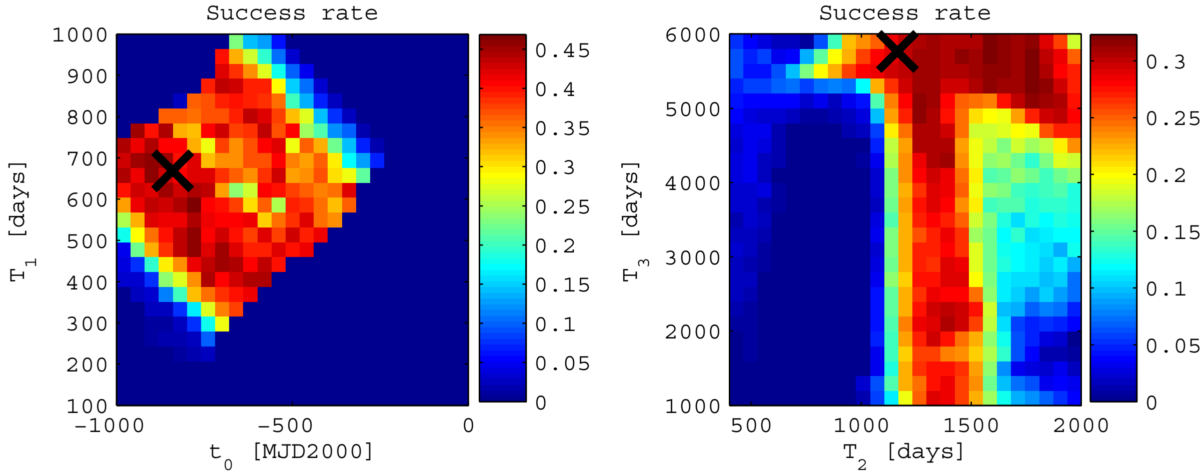


Figure 13.7: The success rate of BFGS on the EMJS MGA problem depending on the decision variables t_0, T_1, T_2 and T_3. The black cross indicates the location of the best putative solution.

It is interesting to note that although the size of the local basin is almost the same for both algorithms, the shape of the basin is considerably different. The BFGS algorithm shows a very clear regions from which convergence may be achieved for both plots, whereas the convergence to the global optimum is mostly driven by t_0 and T_1 for the SBPLX algorithm. For both algorithms T_3 is the least important

Table 13.3: The sizes of the local basin in % of the total search space, calculated using various methods.

	BFGS	SBPLX
Grid Search (25×25×25×25)	12.8	13.9
Monte Carlo (10^5)	12.8	14.0

parameter because convergence to the best putative solution may be achieved from any value for T_3, given that a good combination of the other parameters is used. Finally it is interesting to note that the size of the local basin for BFGS is equally large for the EMJS MGA problem as it is for the EJS MGA problem, despite the increase in dimension.

Global Basin

Again the sensitivity of each parameter was analyzed in a similar way as was done in Section 13.2.1. This results in the plot shown in Figure 13.8.

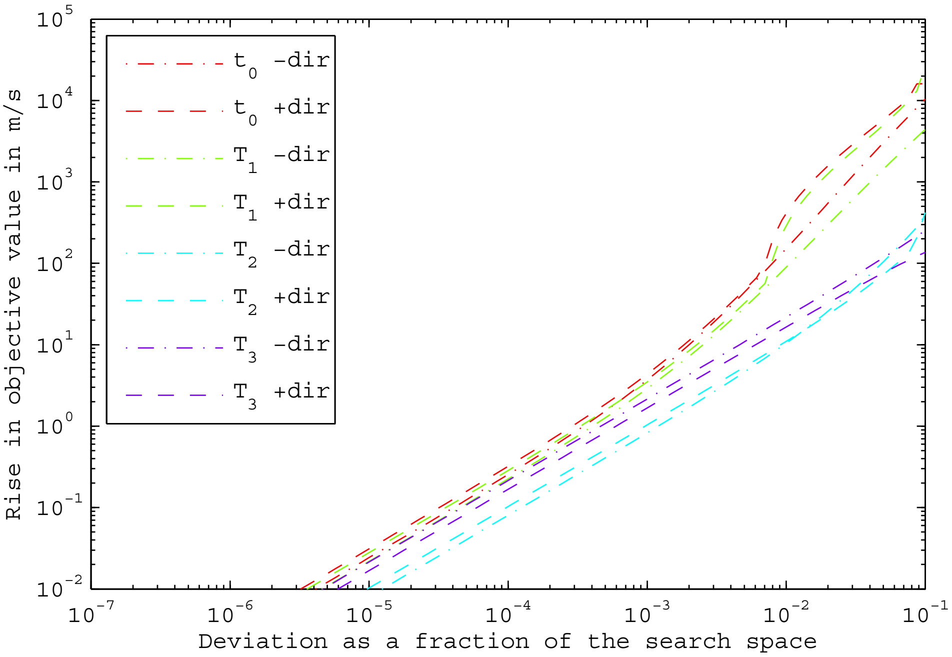


Figure 13.8: The results of the sensitivity analysis of the various variables of the EMJS MGA problem. See text for explanation.

Comparing Figure 13.8 with Figure 13.5 it is interesting to note that the average sensitivity per parameter is similar, and for some parameters actually lower. This may help to explain why the EMJS MGA problem was not more difficult to optimize than the EJS MGA problem. The resulting basin sizes are given in Table 13.4.

Table 13.4: The sizes of the global basin in % of the total search space, calculated for the different parameters.

Variable	t_0	T_1	T_2	T_3	Overall
Total Size	2.0	2.5	16	21	0.0016

13.2.3 EVEJS MGA Problem

Local Basin

Again a grid was set up in all decision variables to determine the size of the local basin and the influence of all parameters. Because five variables are present in this problem, only 15 steps were considered in all variables. The computation time required for the analysis was very large nevertheless.

Again both the SBPLX and BFGS method were used. The results can be seen in Figures 13.9 and 13.10 and in Table 13.5.

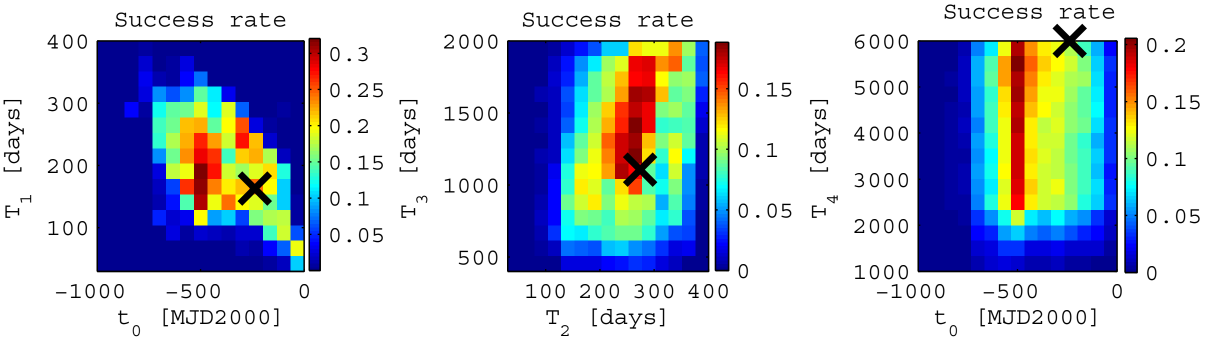


Figure 13.9: The success rate of SBPLX on the EVEJS MGA problem depending on the decision variables t_0, T_1, T_2, T_3 and T_4. The black cross indicates the location of the best putative solution.

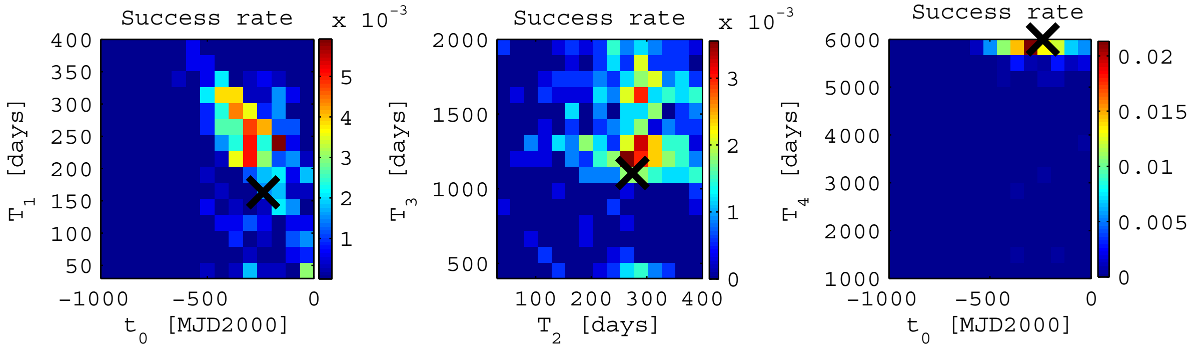


Figure 13.10: The success rate of BFGS on the EVEJS MGA problem depending on the decision variables t_0, T_1, T_2, T_3 and T_4. The black cross indicates the location of the best putative solution.

Table 13.5: The sizes of the local basin in % of the total search space, calculated using various methods.

	BFGS	SBPLX
Grid Search (25×25×25×25)	0.018	2.2
Monte Carlo (10^5)	0.019	2.3

A very big difference is observed between both local optimization methods this time. The basin corresponding to the BFGS algorithm is 100 times smaller than that belonging to the SBPLX algorithm. It is unclear what causes this behavior. Possibly the search space contains many small suboptimal minima, causing the more strictly descending BFGS to get stuck, but over which the SBPLX may hop. Also it may be related to the unstable behavior of BFGS that was observed in Chapter 12.

Apart from that it can be observed that the each variable only has a small region from which convergence may be achieved. It is noted that apparently convergence with SBPLX is more likely from a value of t_0 that is well below the optimal value. Finally it is noted that the grid is becoming less adequate because of the small number of steps that were included for this analysis. This demonstrates the practical issues of this analysis for problems with a larger number of variables. Still the number of variables of this problem was only 5, whereas the most difficult problems are governed by 26 variables.

Global Basin

Again the sensitivity of each parameter was analyzed in a similar way as was done in the previous sections. This results in the plot shown in Figure 13.11.

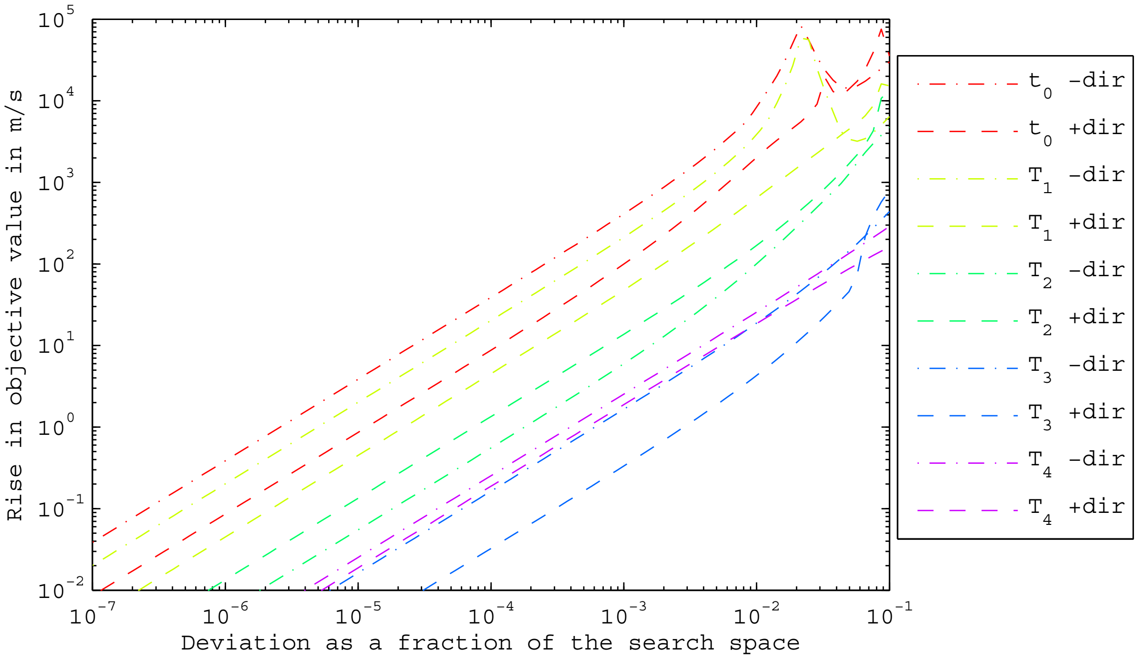


Figure 13.11: The results of the sensitivity analysis of the various variables of the EVEJS MGA problem. See text for explanation.

It is immediately clear from Figure 13.11 that the sensitivity of some variables has increased by more than an order of magnitude with respect to the EJS MGA problem by adding the Earth and Venus as swing-by planets. Hence the problem complexity has increased significantly faster than one would expect from just the addition of one or two variables. The global basin size estimates can be seen in Table 13.6.

Table 13.6: The sizes of the global basin in % of the total search space, calculated for the different parameters.

Variable	t_0	T_1	T_2	T_3	T_4	Overall
Total Size	0.23	0.46	2.7	13	18	$6.7\cdot10^{-7}$

These results also explain the fact that the EVEJS MGA problem was much more difficult to optimize than the EJS and EMJS MGA problems.

13.2.4 Cassini1 MGA Problem

The Cassini1 MGA problem was found to be too difficult for a proper local basin test. This test is hence not performed. This is the case for all problems from now on, because the dimensionality only increases further.

The sensitivity for each parameter was analyzed in a similar way as was done in the previous sections. This results in the plot shown in Figure 13.12.

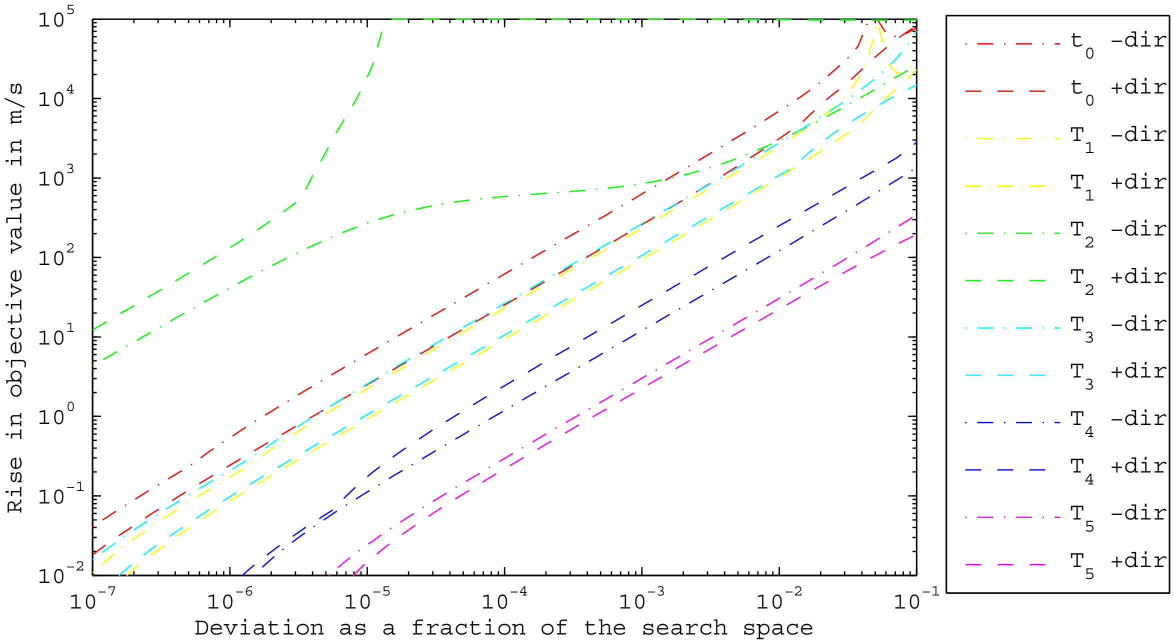


Figure 13.12: The results of the sensitivity analysis of the various variables of the Cassini1 MGA problem. See text for explanation.

The high sensitivity to T_2 immediately becomes clear from Figure 13.12. Compared to the EVEJS MGA trajectory, most of the other parameters have roughly equal sensitivity. The second leg corresponds to a Venus-Venus transfer and causes this very high sensitivity. The transfer is namely a 1-1 resonance orbit with Venus. If the time of flight is slightly too short, the solution to the Lambert problem is significantly inclined, similar to the effect shown in the comparison between GTOP and Tudat ephemeris in Section 5.2.4. Similar if the time of flight is too long, the orbit is inclined as well. However if it is taken even longer, the trajectory changes to an orbit with very high eccentricity directed away from the Sun. This is the only solution to the single-revolution Lambert problem considered in the MGA trajectory model. This effect is visualized in Figure 13.13.

During the optimization of the Cassini1 MGA problem, many optimization techniques got stuck on a different minimum. The sensitivity of its parameters can be seen in Figure 13.14.

The high sensitivity to T_2 is not present around this minimum. The Venus-Venus transfer belonging to this suboptimum is not a strict 1-1 resonance orbit. This suboptimum has a fitness value of 5303 m/s, which is only 370 m/s worse than the best putative minimum.

The global basin sizes are given for both minima in Table 13.7. Note that also the size of the basin of the actual optimum is given at the same fitness threshold as the suboptimum. The results confirm that the actual optimum is very sensitive. It is noted though that at this fitness threshold, the actual optimum appears to still have a larger basin size.

Table 13.7: The sizes of the global basin in % of the total search space, calculated for the different parameters and for different optima of the Cassini1 MGA problem.

Variable	t_0	T_1	T_2	T_3	T_4	T_5	Overall
Optimum	0.11	0.29	0.00078	0.26	2.4	15	$2.4 \cdot 10^{-14}$
Optimum at suboptimum threshold	0.31	0.82	0.0088	0.74	6.7	45	$5.0 \cdot 10^{-11}$
Suboptimum	0.11	0.30	0.31	0.21	2.4	15	$8.3 \cdot 10^{-12}$

The results of the search space analysis help to explain the fact that various optimization techniques

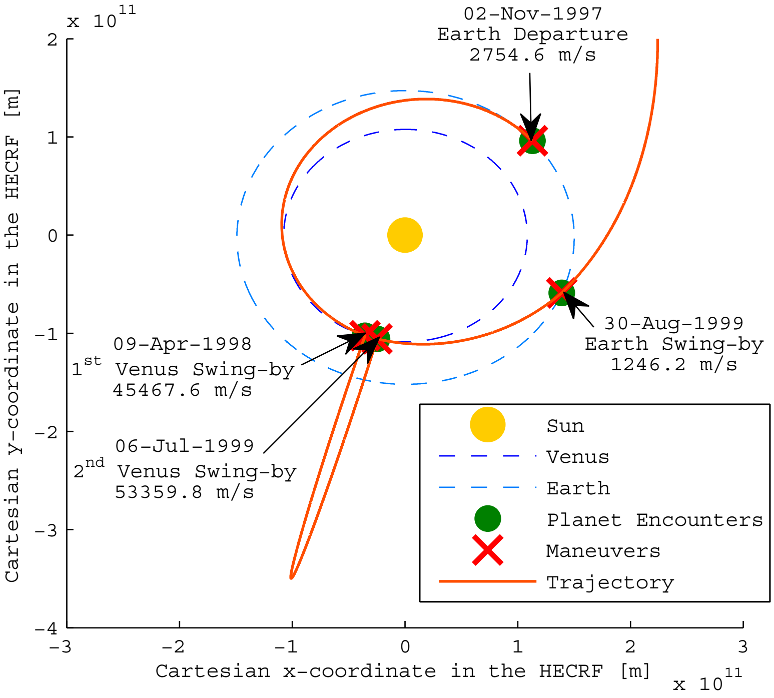


Figure 13.13: Visualization of the best putative solution to the Cassini1 problem, with T_2 perturbed with 0.01 of the domain belonging to T_2 in the positive direction.

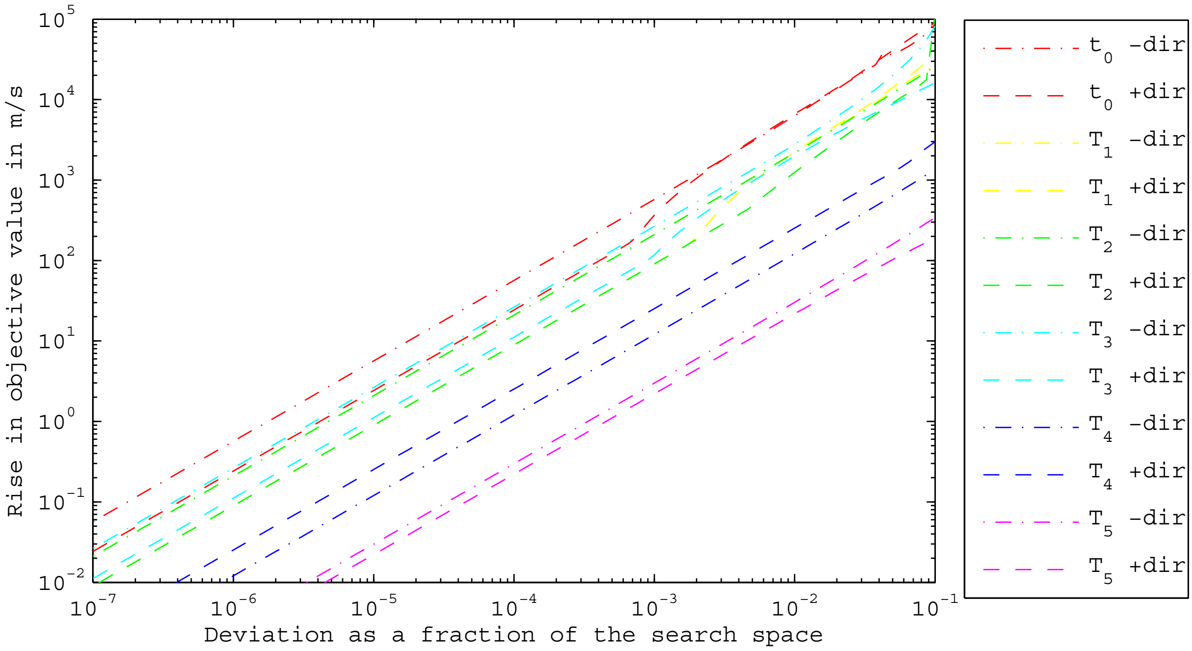


Figure 13.14: The results of the sensitivity analysis of the various variables of the Cassini1 MGA problem. This figure shows the sensitivity of the suboptimum of 5303 m/s that frequently caused misconvergence during the optimization of the problem. See text for further explanation.

in this thesis got stuck on the suboptimal solution repeatedly and also the best optimization technique only finds the best solution at a low success rate. The global basin size of the suboptimum is much smaller than that of the optimum. However in case the size of optimum basin is calculated with the same threshold as the suboptimum, it is still larger than the basin of the suboptimum. Hence in that respect it has to be noted that the global basin size did not fully predict the fact that only about 5% of the optimization runs found the best solution and others found the suboptimal solution. Possibly other differences exist regarding the minima. It may be that the suboptimum is surrounded by various other minima and that the actual optimum is not, possibly because of the highly sensitive T_2. Other explanations include that the global basin size definition is not adequate or that interdependencies between variables cause a different actual global basin size.

The high sensitivity in T_2 of the actual optimum is also likely to be the reason that various early publications did not identify the best putative solution, among which [Myatt et al., 2004] and [Vinko et al., 2007a].

13.2.5 Cassini2 MGA Problem

A similar test was done on the Cassini2 problem to investigate the sensitivity of each parameter. The results can be seen in Figure 13.15.

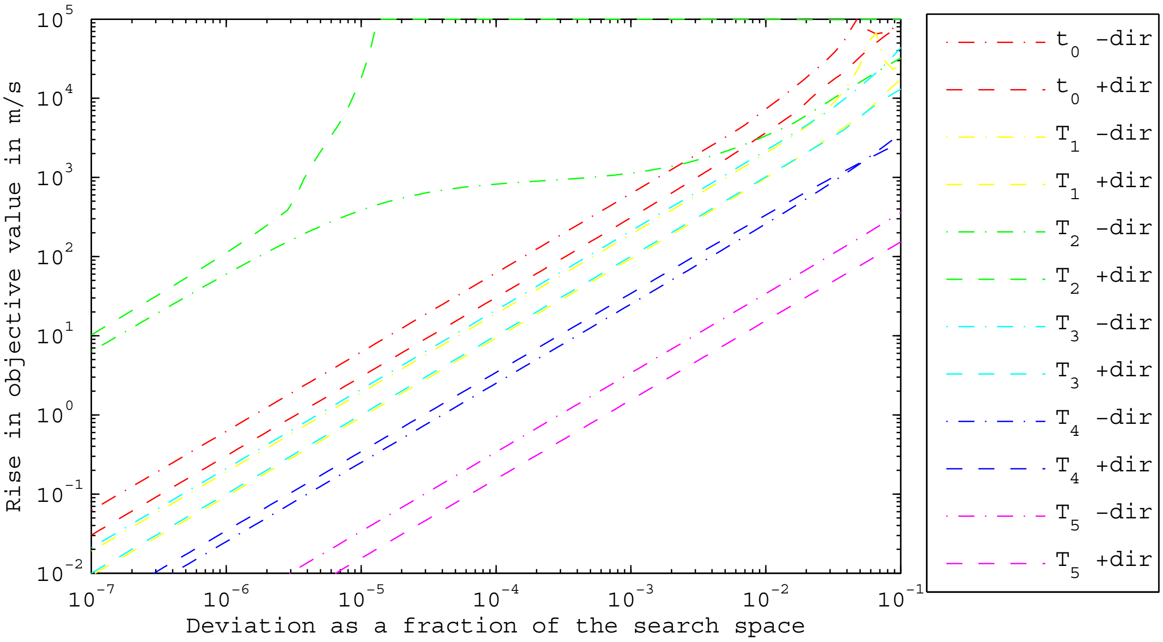


Figure 13.15: The results of the sensitivity analysis of the various variables of the Cassini2 MGA problem. See text for explanation.

The trend in Figure 13.15 is very similar to that of the best putative Cassini1 solution. This is interesting because although the swing-by sequence is the same, the bounds and the actual trajectory is different for the Cassini2 MGA problem. The high sensitivity to T_2 is again caused by the Venus-Venus transfer.

Similar to the Cassini1 problem, the Cassini2 problem also has a strong suboptimal attractor close to the best putative solution. This attractor also has similar characteristics as the Cassini1 suboptimal attractor and a picture is therefore not included here. The accompanying global basin sizes can be seen in Table 13.8.

13.2.6 Overview MGA Problems

Various conclusions can be drawn based on the previous sections:

- Typically the sensitivity of the decision variables increases significantly when the problem consists of more legs, complicating the optimization severely. This is however not always the case, as was

Table 13.8: The sizes of the global basin in % of the total search space, calculated for the different parameters and for different optima of the Cassini2 MGA problem.

Variable	t_0	T_1	T_2	T_3	T_4	T_5	Overall
Optimum	0.097	0.32	0.00056	0.29	1.3	19	$1.3 \cdot 10^{-14}$
Optimum at suboptimum threshold	0.27	0.89	0.0025	0.83	3.8	54	$1.0 \cdot 10^{-11}$
Suboptimum	0.071	0.24	0.22	0.27	1.4	19	$2.6 \cdot 10^{-12}$

pointed out by the fact that the variables of the EMJS problem were not more sensitive than those of the EJS problem.

- The low success rates for the Cassini1 and Cassini2 MGA problems can (partly) be explained by investigating the search space. The best putative optimum was namely found to be highly sensitive.
- The resulting local basins of attraction are influenced significantly by the performance of the local optimization techniques.
- The method for visualizing the local basins of attraction becomes too difficult for problems with more than 5 variables.

13.3 DSM Problems

This section discusses the search space analysis that was performed on the problems with DSMs. To this end the Cassini2 DSM In Second Leg Only PF problem is discussed in Section 13.3.1. The Cassini2 DSM In Second Leg Only VF problem is discussed in Section 13.3.2. Section 13.3.3 discusses the Cassini2 DSM PF problem. Section 13.3.4 discusses the Cassini2 DSM VF problem. Hybrids between the PF and VF trajectory models are discussed in Section 13.3.5. Finally an overview is presented in Section 13.3.6.

13.3.1 Cassini2 DSM In Second Leg Only PF Problem

The sensitivity of each parameter was analyzed in a similar way as was done in Section 13.2. This results in the plot shown in Figure 13.16. Note that the positive and negative direction are combined in one graph. The *least* sensitive direction was plotted in the graph, because it has the largest contribution to the global basin size. The graphs hence give a slightly optimistic representation of the sensitivity, which is however more representative than its more pessimistic counterparts.

Comparing the results in Figure 13.16 with the results of the Cassini2 MGA trajectory in Figure 13.15, it can be noted that the sensitivity of t_0, T_1, T_3, T_4 and T_5 is almost the same for both problems. The addition of a DSM has however reduced the sensitivity in T_2 by three orders of magnitude. The overall problem complexity has increased of course, because four new parameters were added. An overview of the global basin size is given in Table 13.9 in the next section.

The absence of an extremely sensitive T_2 does however likely mean that other minima are not attractors of much larger size, which was the case in the Cassini2 MGA problem. Hence this explains why this problem can be optimized at a much higher success rate.

13.3.2 Cassini2 DSM In Second Leg Only VF Problem

Again a parameter sensitivity test was done on the Cassini2 DSM In Second Leg Only VF problem. The results are shown in Figure 13.17. Both directions were combined in one line in a similar fashion as was done for the PF variant of the problem.

Comparing the VF version of the problem in Figure 13.17 with the PF version in Figure 13.16 shows that both version are equally sensitive to the timing variables. The VF version is however less sensitive to the four variables that define the DSM, on average. This also explains the fact that the VF version was easier to optimize. A comparison between the global basin size of the Cassini2 DSM In Second Leg Only VF problem with the PF and MGA variants can be seen in Table 13.17.

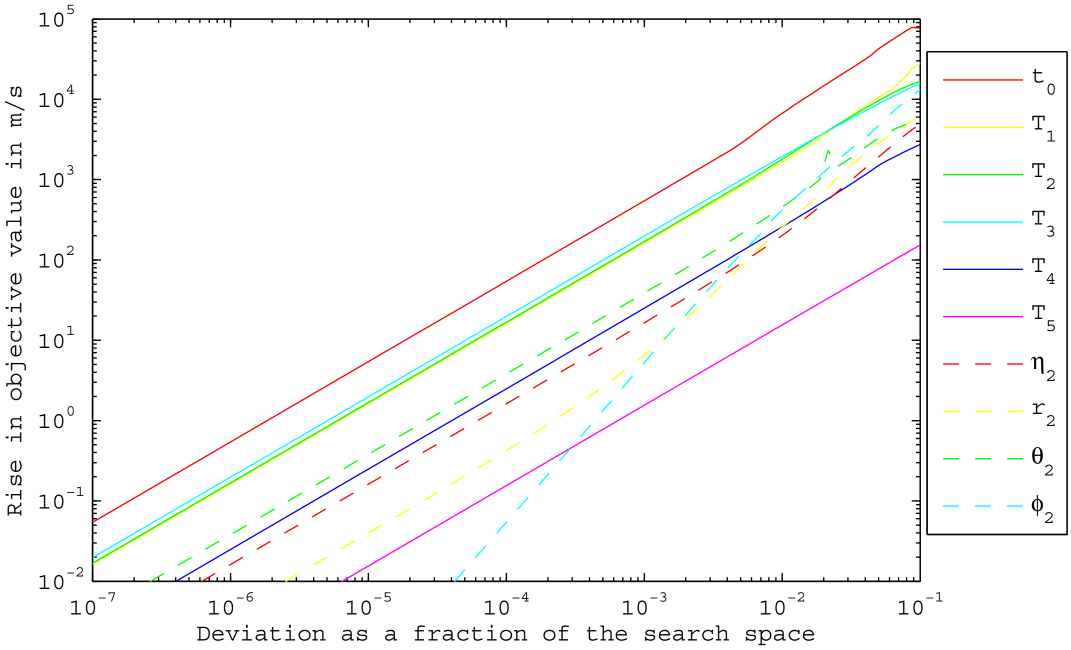


Figure 13.16: The results of the sensitivity analysis of the various variables of the Cassini2 DSM In Second Leg Only PF problem. See text for explanation.

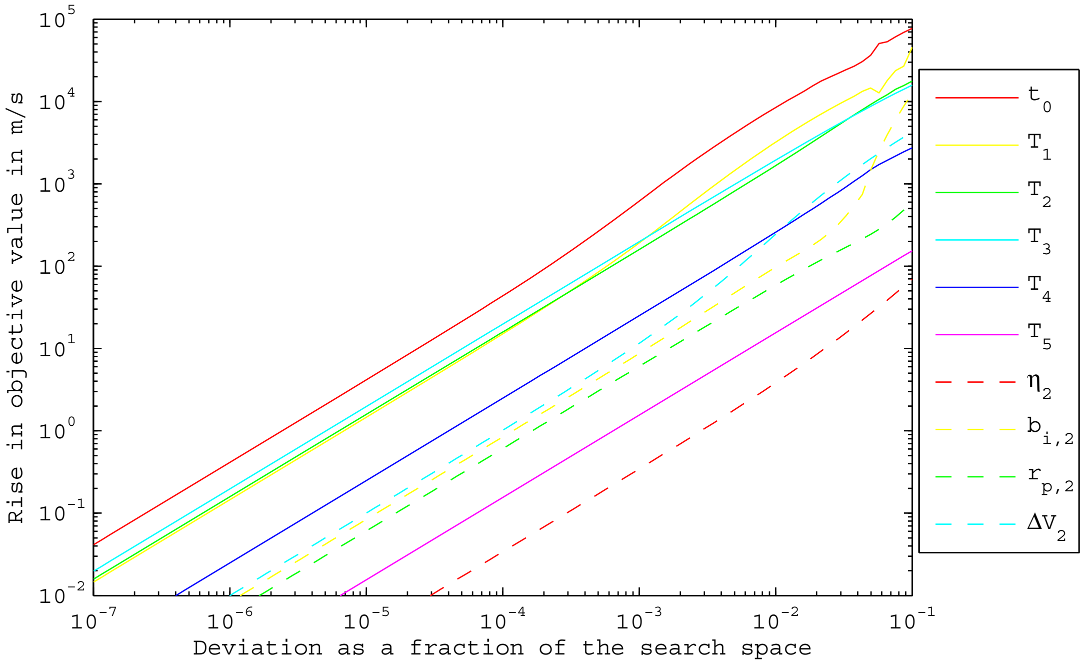


Figure 13.17: The results of the sensitivity analysis of the various variables of the Cassini2 DSM In Second Leg Only PF problem. See text for explanation.

Apart from the fact that the VF variant is less sensitive than the PF variant, Table 13.9 also shows that both the PF as well as the VF variant are very sensitive, given that only $1.8 \cdot 10^{-19}$ % of the search space has a fitness within 200 m/s of the best putative solution. Although DE managed to cope well with this sensitive search space, this sensitivity is likely to be the reason that GA and PSO struggled to optimize these problems.

Table 13.9: The sizes of the global basin in % of the total search space, calculated for the different parameters. The table gives sizes for three different variants of the Cassini2 problem for comparison.

Var. Model	t_0	T_1	T_2	T_3	T_4	T_5	η_2	$r_2/r_{p,2}$	$\theta_2/b_{i,2}$	$\phi_2/\Delta V_2$	Overall
MGA	0.097	0.32	0.00056	0.29	1.3	19					$1.3 \cdot 10^{-14}$
DSM 2^{nd} Leg PF	0.073	0.24	0.22	0.20	1.4	19	1.6	1.7	0.77	1.3	$5.7 \cdot 10^{-22}$
DSM 2^{nd} Leg VF	0.075	0.19	0.23	0.20	1.4	19	39	2.9	5.5	1.6	$1.8 \cdot 10^{-19}$

13.3.3 Cassini2 DSM PF Problem

A parameter sensitivity test was done on the Cassini2 DSM PF problem. The results are shown in Figure 13.18. Again both directions were combined in one line, as was done in the previous DSM problems.

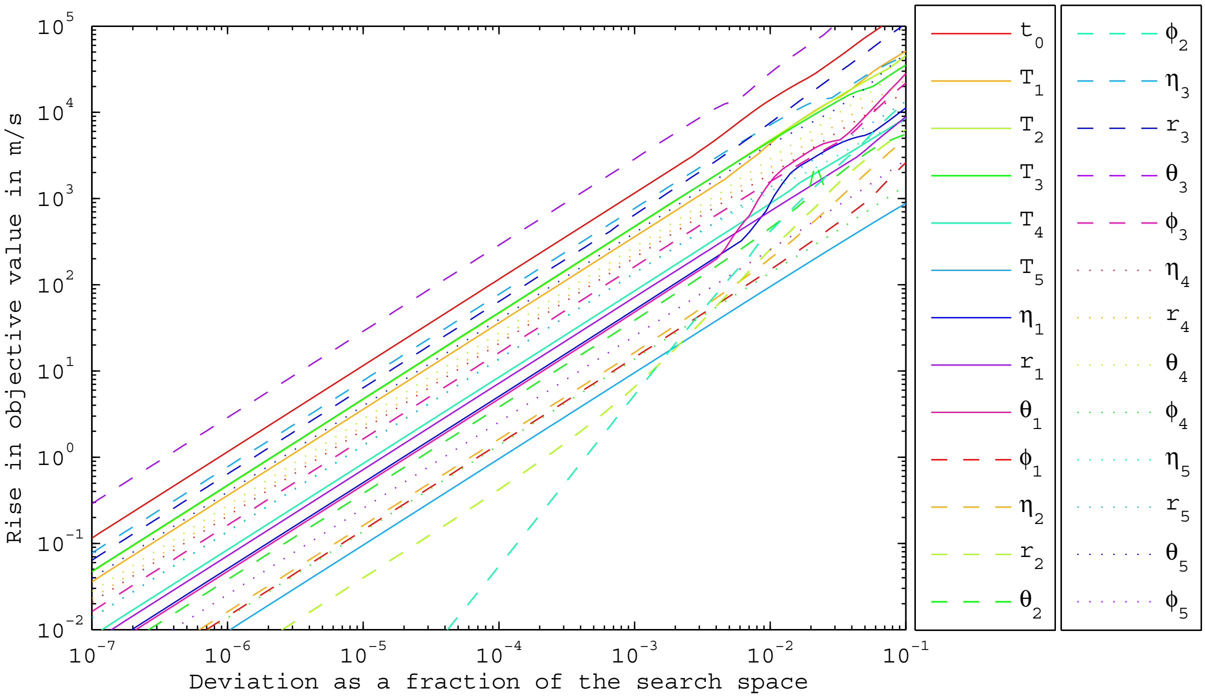


Figure 13.18: The results of the sensitivity analysis of the various variables of the Cassini2 DSM PF problem. See text for explanation.

Various conclusions can be drawn comparing Figure 13.18 with Figure 13.16. First of all the sensitivity in the time variables has increased. Also a large number of very sensitive parameters was added. On the other hand it can also be noticed that the variables in the second leg (i.e. η_2, r_2, θ_2 and ϕ_2) have not increased in sensitivity at all. For an overview of the global basin size, the reader is referred to Table 13.11 in Section 13.3.4. In this table the other variants of the Cassini2 problem is also added for comparison.

One of the important conclusions from Table 13.11 is that the PF variant is several orders of magnitude more very sensitive to η than the VF variant (i.e. compare 0.19 to 70). This variable has little effect in the VF variant if no DSM is present in the specific leg. Also if a DSM is present it is much less sensitive, as can be seen in the comparison of the second leg in Table E.1 in Appendix E. This high sensitivity can be explained by the fact that the DSM location also changes in case the DSM is applied later. The location does not change accordingly for the PF trajectory model, but simply changes the time of flight distribution between the first and second part of the leg. This causes both parts of the leg to change significantly, quickly causing a large increase in ΔV.

Apart from the large increase in sensitivity in η, it is also noted that θ and r are also very sensitive variables. Also it is noted that all these variables add up to a very sensitive search space in general.

Only $4.0 \cdot 10^{-64}$ % of the search space has a fitness within 200 m/s of the best putative solution. Hence it is no wonder that this solution could not be found by any optimization technique without prior knowledge.

13.3.4 Cassini2 DSM VF Problem

The results of the parameter sensitivity test on the Cassini2 DSM VF problem are shown in Figure 13.19. Again both directions were combined in one line, as was done in the previous DSM problems.

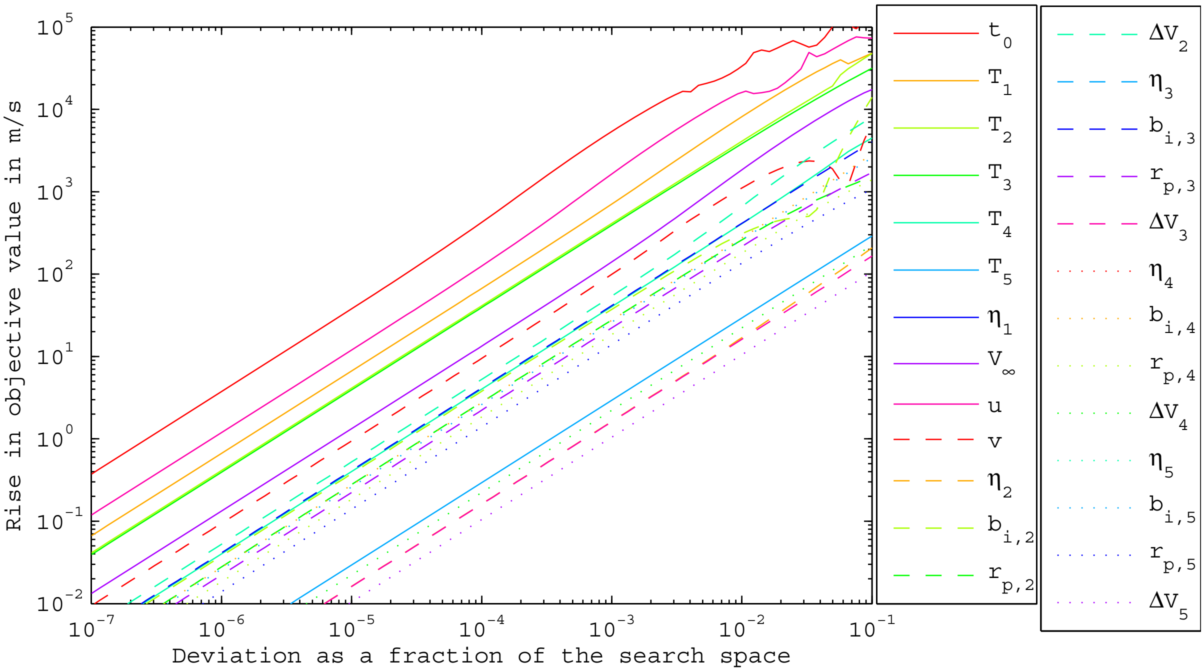


Figure 13.19: The results of the sensitivity analysis of the various variables of the Cassini2 DSM VF problem. See text for explanation.

Comparing Figure 13.19 with Figure 13.17 shows that the variables corresponding to the second leg (i.e. η_2, $r_{p,2}$, $b_{i,2}$ and ΔV_2) have increased by adding DSMs in the other legs. This is a disadvantage of the DSM VF model. On the other hand the overall sensitivity is still much lower than that of the DSM PF model, as can be seen in Table 13.11. Especially for the last legs the sensitivity of the VF model is comparatively low. For the first two legs the PF is equally sensitive or better, as can be seen in Table 13.11. This is caused by the dependency of the last legs on the first legs in the VF model. This led to the assumption that it may be possible to reduce the overall sensitivity by hybridizing both models, which will be further investigated in Section 13.3.5.

Table 13.10: The average contribution to the global basin size of the different type of parameters used by both the PF and VF variant of the trajectory model applied to the Cassini2 problem.

Variable / Model	t_0	T_n	η_n	$r_n/V_\infty/r_{p,n}$	$\theta_n/\theta/b_{i,n}$	$\phi_n/\phi/\Delta V_n$
DSM PF	0.032	0.25	0.19	0.30	0.15	1.2
DSM VF	0.0077	0.29	70	0.76	0.68	2.7

It is noted here that Table 13.11 also helps to explain some observations during the optimization of the Cassini2 specific leg problems. The PF variant of the first leg problems was namely easier to optimize than the VF variant, whereas the last leg problems were easier to optimize with the VF variant. Both of these results may be explained by the basin sizes in Table 13.11. The PF variant namely has a larger

Table 13.11: The sizes of the global basin in % of the total search space, calculated for each leg individually. The table gives sizes for five different variants of the Cassini2 problem for comparison. The reader is referred to Table E.1 in Appendix E for the basin sizes for each individual parameter. This table shows the resulting total basin size per leg.

Model	Leg 1	Leg 2	Leg 3	Leg 4	Leg 5	Overall
MGA	$3.1 \cdot 10^{-4}$	0.00056	0.29	1.3	19	$1.3 \cdot 10^{-14}$
DSM 2^{nd} Leg PF	$1.8 \cdot 10^{-4}$	$5.9 \cdot 10^{-9}$	0.20	1.4	19	$5.7 \cdot 10^{-22}$
DSM 2^{nd} Leg VF	$1.4 \cdot 10^{-4}$	$2.3 \cdot 10^{-6}$	0.20	1.4	19	$1.8 \cdot 10^{-19}$
DSM PF	$7.2 \cdot 10^{-14}$	$2.4 \cdot 10^{-9}$	$9.2 \cdot 10^{-15}$	$6.2 \cdot 10^{-11}$	$4.1 \cdot 10^{-10}$	$4.0 \cdot 10^{-64}$
DSM VF	$2.2 \cdot 10^{-15}$	$6.7 \cdot 10^{-9}$	$1.3 \cdot 10^{-6}$	$1.4 \cdot 10^{-5}$	$5.3 \cdot 10^{-4}$	$1.5 \cdot 10^{-45}$

basin size for the first leg, whereas the VF variant has a larger basin size for the last legs.

13.3.5 Hybrids of VF and PF Trajectory Models

It was mentioned in Sections 13.3.3 and 13.3.4 that the overall sensitivity may possibly be reduced by employing a hybrid model consisting of PF and VF legs. To verify this assumption, a hybrid trajectory model was built consisting of both PF and VF legs. All 32 possible combinations were subjected to the sensitivity test. The resulting sensitivities of the best 8 combinations are given in Table 13.12.

Table 13.12: The sizes of the global basin in % of the total search space for the 8 least sensitive hybrids between PF and VF trajectory models for the Cassini2 trajectory. They are ordened by sensitivity level.

Model type per leg					
1	2	3	4	5	Sensitivity
VF	PF	VF	VF	VF	$1.3 \cdot 10^{-43}$
VF	VF	VF	VF	VF	$1.5 \cdot 10^{-45}$
PF	PF	VF	VF	VF	$1.9 \cdot 10^{-46}$
VF	PF	VF	PF	VF	$1.5 \cdot 10^{-48}$
VF	PF	VF	VF	PF	$3.0 \cdot 10^{-49}$
PF	VF	VF	VF	VF	$4.7 \cdot 10^{-50}$
VF	VF	PF	VF	VF	$2.1 \cdot 10^{-50}$
VF	VF	VF	PF	VF	$1.9 \cdot 10^{-50}$

It is concluded from 13.12 that the least sensitive combination appears to be one in which the PF model is used for the second leg and the VF model for the other legs. It would be very interesting to see if a similar conclusion will be drawn for other problems as well. If a similar pattern is observed for more problems than just the Cassini2 problem, this may lead to a general strategy regarding the modeling approach in general. This result does show that it is sometimes possible to decrease the sensitivity of a problem by employing a hybrid trajectory model.

13.3.6 Overview

Section 13.3 revealed the advantages and disadvantages of both the VF and the PF trajectory model. The VF model is in general a very effective means of defining a trajectory and allows for the easiest optimization. On the other hand the VF model becomes very sensitive to the first legs if many legs are present. Adding a PF leg in between may reduce this sensitivity, as it does not propagate the 'errors' in the first legs to the last legs. The PF is more sensitive in general though. This is primarily caused by the fact that the time at which a DSM is applied is not coupled to the position at which that occurs. Furthermore specifying the DSM position using θ and r allows for a much wider search space, which is much more sensitive.

13.4 Conclusions

In this chapter a new definition for the basin of attraction was proposed which intends to represent the difficulty of optimizing a problem using a global optimization algorithm. The sensitivity of all parameters was analyzed. The results revealed why many optimization runs got stuck on suboptimal solutions in the Cassini1 and Cassini2 problems. Also the analysis revealed the high sensitivity of a VF model on its first legs. The PF model was found to be highly sensitive to all of its legs, mainly caused by a much higher sensitivity in η, but also in θ and r. The sensitivity of hybrids was also analyzed, revealing that a hybrid between a PF and a VF model is less sensitive than a trajectory purely made up of VF legs.
It is noted here explicitly that the definition of the global basin size may be improved upon. Especially the fact that correlation between variables was not taken into account may have a strong influence on the results obtained this way. The analysis did yield very interesting results though.

Chapter 14

Optimization Algorithm Comparison

This chapter aims at determining which optimization algorithm is best suited to optimize the trajectories considered in this thesis. Furthermore this chapter compares the results obtained in this thesis with those obtained by others on the Cassini1 and Cassini2 problems.

Section 14.1 compares the performance of the algorithms considered in this thesis on solving the problems defined in Chapter 8. Subsequently Section 14.2 compares the performance of the algorithms considered in this thesis with those encountered in literature. Furthermore some settings for optimization algorithms that were encountered in literature are compared to those obtained in this thesis. Finally an overview of the conclusions is given in Section 14.3.

14.1 Comparison between Algorithms

This chapter will discuss the performance of the algorithms considered in this thesis. To this end, first a comparison is made between the performances of the various algorithms on the problems used in the tuning process. Table 14.1 gives an overview of the performance of the different algorithms used in this thesis. This table presents an overview of the best performance encountered on a certain problem during the tuning process of GA, PSO and DE, which was described in Chapters 9, 10 and 11. Furthermore the table gives an overview of the performance of the determined tuning parameter setting(s) of GA, PSO and DE on each of these problems. This result has been calculated to a greater level of accuracy, using 1000 samples for all problems except the Cassini1 and Cassini2 MGA problems for which 5000 samples were used, and the Cassini2 DSM VF problem for which 400 samples were used because of the required computation time. Also it gives the best performance achieved by the different MS schemes discussed in Chapter 12.

It is noted that the performance was not determined for all algorithms on all problems. Some algorithms had a performance which made it practically impossible to determine the actual performance. Especially the performance of GA was very bad, meaning the performance could not be determined for the more difficult problems. It can be noted that the performance is at least 20-100 times as bad as the performance of other algorithms though.

Table 14.1: Overview of the performance ($n_{95\%}$) of various algorithms on the problems described in Chapter 8. It is noted that for the narrowed bounds problems not the actual derived relation for the population size was used in determining the performance of DE1, DE2, DE3, DE4 and PSO. This was done because this relation was not determined for problems with such narrow bounds. Instead the best population size encountered during tuning was used. See text for further explanation and discussion..

	Best $n_{95\%}$ found during tuning [$\cdot 10^3$]			$n_{95\%}$ of tuned algorithms [$\cdot 10^3$]							
Problem	DE	GA	PSO	DE1	DE2	DE3	DE4	GA(50)	GA(500)	PSO	MS
Model simplification problems											
EJS MGA	1.4	2.7	6.2	3.1	2.9	4.1	2.2	5.0	6.4	12.5	3.0
EMJS MGA	1.5	2.0	3.8	2.0	1.6	3.4	1.8	4.9	4.1	6.3	6.6
EVEJS MGA	7.0	87	34	9.3	11.1	21	9.3	246	167	38	31
Cassini1 MGA	358	n.a.	n.a.	393	3907	1854	412	>10000	>10000	>10000	2645
Cassini2 MGA	140	n.a.	7980	178	2592	465	155	9860	-	7980	6653
Cassini2 DSM In Second Leg PF	72	n.a.	n.a.	187	106	427	142	8942	-	-	n.a.
Cassini2 DSM In Second Leg VF	21	n.a.	n.a.	61	25	117	52	>10000	>10000	4582	n.a.
Specific leg problems											
Cassini2 First Leg PF	0.8	4.2	1.2	1.2	2.3	1.3	1.2	11	19	4.4	2.6
Cassini2 First Leg VF	8.2	3800	152	17	113	24	24	8208	3922	671	43
Cassini2 Last Leg PF	1.0	7.0	2.9	1.2	2.4	1.7	1.2	19	17	4.8	4.0
Cassini2 Last Leg VF	0.7	1.0	0.6	1.0	1.6	1.0	0.9	1.7	5.2	2.0	2.6
Cassini2 First/Last Leg PF	3.4	114	21	6.1	12.0	5.6	5.0	333	133	31	61
Cassini2 First/Last Leg VF	20	n.a.	623	24	97	33	22	n.a.	n.a.	820	278
Cassini2 Last Two Legs PF	7.7	n.a.	137	12	12	21	8.5	n.a.	n.a.	269	9350
Cassini2 Last Two Legs VF	3.2	187	8.6	5.0	9.2	3.8	4.4	285	507	16	107
Narrowed bound problems											
Cassini2 Narrowed 0.01 PF	7.6	71	8.8	11.2	9.2	9.2	8.3	223	805	15	659
Cassini2 Narrowed 0.01 VF	3.8	79	15	5.8	3.8	4.4	3.8	131	82	16	563
Cassini2 Narrowed 0.1 PF	93	n.a.	203	243	108	110	112	n.a.	n.a.	372	n.a.
Cassini2 Narrowed 0.1 VF	44	n.a.	674	117	53	72	68	n.a.	n.a.	694	n.a.
Full difficulty problem											
Cassini2 VF	4032	n.a.	n.a.	4966	5879	-	6195	-	-	-	

Various conclusions can be drawn from Table 14.1. The following paragraphs will discuss the most important ones.

First the best results encountered by DE, GA and PSO will be compared, which can be seen in the first three entries. It is noted that the performance of these algorithms is quite comparable on the easiest five problems: EJS MGA, EMJS MGA, Cassini2 First Leg PF, Cassini2 Last Leg PF and Cassini2 Last Leg VF. However as soon as the problems start to become more complicated, the performance of GA drops very rapidly. Especially the performance on the Cassini2 First Leg VF problem is interesting because it only consists of 5 parameters. As discussed in Chapter 13, the search space of this problem is highly sensitive. Apparently GA is very bad at coping with this. This extreme performance drop is however present on the difficult tuning problem of all three simplification types. Also GA was not able to find the solution for any of the actual DSM problems reliably.

Also the performance of PSO drops significantly faster than that of DE as the problem becomes more difficult. Although PSO is significantly better than GA on more difficult problems, the performance of PSO on actual problems such as the Cassini2 DSM In Second Leg Only is incomparable to the performance of DE. Furthermore PSO did not manage to find the solution to the full-difficulty problem even once. Hence it is concluded that this version of PSO is also not suited for optimizing complex DSM problems. One note needs to be made with respect to that conclusion though. The tuning process of PSO was not as successful as the tuning process of DE. Hence it is possible that the performance of PSO may be improved upon using a more powerful tuning scheme. It is however unlikely that it will result in an increase in performance that will make the performance comparable to that of DE. Furthermore, the fact that an even more careful tuning process is required for PSO than for DE may be regarded as a disadvantage in itself.

A further note that is made is that the performances of DE, PSO and GA do not take into account the effectiveness of the convergence criteria. The author only looked into the convergence of DE as it is the best performing algorithm. The fact that DE collapses very rapidly towards the best solution allows the user to define a very effective stopping criterion, as was already discussed in Section 7.3.2. In case PSO or GA do not allow for such an effective stopping criterion, their actual performance may be worse than listed here.

Finally it is noted here that this thesis only looked into the DE, GA and PSO variants that were present in PaGMO. Although these algorithms are very representative, different implementations may be thought of that outperform their PaGMO variants. It can hence not be ruled out that other variants of DE, GA or PSO may be thought of that change the trend sketched in Table 14.1. It is however noted that DE outperforms GA and PSO to such an extent that this is unlikely.

Also comparing MS with DE reveals that DE is clearly performing better on more difficult problems. On simple problems MS performs similar to the other metaheuristics. It was already noted in Chapter 12 that this bad performance may be significantly improved upon by selecting better local optimization algorithms.

Subsequently it is time to compare the performance of different versions of DE. First of all it is noted that the performance of DE1 and DE4 is always very close to the best performance encountered during the tuning phase. Given that these algorithms are trade-offs between the performances on all problems, it is to be expected that better settings can be found for each individual problem. Still the performance is within double the number of evaluations for most problems, and typically closer than that.

DE2 also performs well on most problems, except for four problems. Reasons can be identified for the comparatively bad performances though. The fact that it performs badly on the Cassini2 First Leg VF problem and Cassini2 First And Last Leg VF problem may be explained by the fact that these problems are highly sensitive. Because DE2 adds the mutant vectors to random individuals, it will explore the search space compared to the other DE strategies that focus on convergence by adding the mutant vector to the best individual. This may cause the difference in performance.

A very interesting phenomenon was discovered while identifying the bad performance on the Cassini1 MGA and Cassini2 MGA problems. As was already noted in Chapter 13, both problems contain a very strong suboptimal basin of attraction. An analysis was performed on the frequency at which DE2 with different population sizes converged to this suboptimum and the actual optimum. The results can be seen in Table 14.2.

Table 14.2: Percentage of runs that converged to the suboptimum, optimum and other minima for the Cassini2 MGA problem. Optimizer used was DE2 for variable population sizes and 1000 runs for each setting.

n_{pop} / Minimum	10	14	18	24	30	36	42	50	58	66
Optimum	0.3	2.4	3.7	0.4	0	0	0	0	0	0
Suboptimum	1.8	46.3	79.2	93.6	97.5	99.1	99.9	100	100	100
Other minimum	97.9	48.7	17.1	6.0	2.5	0.9	0.1	0	0	0

Table 14.2 shows that the increase in population size causes the optimization algorithm to converge to the suboptimum rather than the actual optimum. In practice, a typical guideline is to increase the population size if the optimum is not found reliably. This case illustrates that a larger population size may have very bad effects on the success rate of an optimizer, apart from the additional cost of evolving a larger population for a larger number of generations.

14.2 Comparison with Performance in Literature

The effectiveness of the tuning process of DE is evaluated here by comparing the results with those from literature. This comparison is difficult to make because many publications did not find the best solution to the difficult problems, tested on different problems or did not use robust test criteria. However all publications did clearly state the tuning parameters used for DE. Hence a first comparison is made by subjecting many settings for DE that were encountered in literature to a similar testing suite as the settings for DE that were determined by the author. Secondly the success rate of DE is compared to the success rate of various other algorithms that were described in [Vasile et al., 2011]. Vasile et al. [2011], and the previous publications by the same authors, did namely adopt a robust test criteria.

Section 14.2.1 discusses the comparison using the performance criterion used in this thesis. Subsequently Section 14.2.2 describes two comparisons with respect to a publication by Vasile et al. [2011].

14.2.1 Comparison on Cassini2 DSM VF problem

The performance of the settings used by various publications is compared to that of the various algorithms used in this thesis. The settings of DE used in [Vinko et al., 2007a], [Vinko and Izzo, 2008], [Myatt et al., 2004] are used for comparison. Also the settings that performed best on the Cassini problem in Olds et al. [2007] are tested, upscaling the population size to the more difficult Cassini2 DSM VF problem. Furthermore the three suggestions that are given in PaGMO are tested at various population sizes ($n_{pop} = 2 \cdot n_{var}$, $n_{pop} = 5 \cdot n_{var}$ and $n_{pop} = 10 \cdot n_{var}$).

Each setting was tested both using a run of 250 samples with a maximum number of iterations of 2M and 25 samples with a maximum number of iterations of 20M. The best results are given in Table 14.3. It can be concluded from Table 14.3 that a large number of algorithms did not manage to provide a realistic chance of optimizing the Cassini2 DSM VF problem. Almost all settings from literature and from PaGMO did not come up with solutions reliably. The two settings that did come up with a good result only managed to do so with a very low success rate and a fast convergence to the result.

Also it can be concluded that the best settings found in this thesis are very sensitive to changes in both F as well as CR. Only an increase in F for DE1 or DE2 does not lead to a large decrease in performance. It was noted in Chapter 9 that the sensitivity to the population size was not as high as the sensitivity to

Table 14.3: Comparison between the DE settings determined in this thesis with settings that are slightly different and with settings obtained from literature and from PaGMO. The algorithms were subjected to the Cassini2 DSM VF problem and the $n_{95\%}$ was calculated. It is noted that performances above 10^8 were all given a score of $>100 \cdot 10^6$ because giving the actual score would not be accurate enough. Note that all high results should be considered inaccurate. ** means that the result was based on only 1 or 2 samples and is especially inaccurate.

DE strategy	F	CR	n_{pop}	Strategy	$n_{95\%}$ [10^6]
DE1	0.7	0.9	117	best/1/exp	5.0
DE2	0.7	0.97	67	rand/1/exp	5.9
DE4	0.5	0.94	99	best/2/exp	6.2
Close to DE1	0.5	0.9	117	best/1/exp	>100
Close to DE1	0.9	0.9	117	best/1/exp	6.6
Close to DE1	0.7	0.7	117	best/1/exp	>100
Close to DE2	0.5	0.97	67	rand/1/exp	52
Close to DE2	0.9	0.97	67	rand/1/exp	12
Close to DE2	0.7	0.77	67	rand/1/exp	>100
Close to DE4	0.3	0.94	99	best/2/exp	>100
Close to DE4	0.7	0.94	99	best/2/exp	>100
Close to DE4	0.5	0.74	99	best/2/exp	>100
[Myatt et al., 2004]	0.8	0.5	260	rand/1/bin	>100
[Olds et al., 2007]	0.6	0.8	73	rand/1/exp	>100
[Vinko et al., 2007a]	0.95	0.9	20	rand/1/bin	>100
[Vinko and Izzo, 2008]	0.8	0.9	20	best/2/bin	32 **
PaGMO default $n_{pop} = 2 \cdot n_{var}$	0.8	0.9	52	rand/1/exp	>100
PaGMO default $n_{pop} = 5 \cdot n_{var}$	0.8	0.9	130	rand/1/exp	>100
PaGMO default $n_{pop} = 10 \cdot n_{var}$	0.8	0.9	260	rand/1/exp	>100
PaGMO suggestion 2 $n_{pop} = 2 \cdot n_{var}$	0.7	0.5	52	rand/1/exp	>100
PaGMO suggestion 2 $n_{pop} = 5 \cdot n_{var}$	0.7	0.5	130	rand/1/exp	>100
PaGMO suggestion 2 $n_{pop} = 10 \cdot n_{var}$	0.7	0.5	260	rand/1/exp	>100
PaGMO suggestion 3 $n_{pop} = 2 \cdot n_{var}$	0.85	1.0	52	rand/1/exp	>100
PaGMO suggestion 3 $n_{pop} = 5 \cdot n_{var}$	0.85	1.0	130	rand/1/exp	>100
PaGMO suggestion 3 $n_{pop} = 10 \cdot n_{var}$	0.85	1.0	260	rand/1/exp	16 **

F and especially CR that was investigated here. Chapter 9 showed that a population size of 30% more or 30% less only resulted in a performance decrease of about 50%.

14.2.2 Comparison with Test Suite Defined by Vasile

[Vasile et al., 2008], [Vasile et al., 2009], [Vasile et al., 2010] and [Vasile et al., 2011] are the only publications that have adopted a robust test criterion for measuring algorithm performances on problems in the GTOP database. They also employed a success-rate representation, the only difference being that they did not calculate $n_{95\%}$ from the success-rates. As discussed in Section 7.3.3, the 'ideal' performances calculated in this thesis can not be used for comparing with other publications because they neglect the influence of stopping criteria. To allow for a fair comparison, stopping criteria were implemented based on the average distance between individuals in the population. A comparatively 'safe' criterion of 0.001 was used in these calculations. It is noted that the performance may be further improved by selecting a higher criterion. This was not done though, because in general the stopping criterion can not be defined using knowledge of the problem in question. Similarly the considered algorithms were tuned for the total variety of problems considered in this thesis. This may hence be improved upon by tuning specifically for this problem.

A comparison is made with respect to [Vasile et al., 2011] as the best performance is presented therein. The difference with respect to earlier publications is very small though. Two similar problems are present in [Vasile et al., 2011], namely the Cassini1 and the Cassini2 problem. First a comparison will be made based on the Cassini1 problem, subsequently a comparison will be made on the Cassini2 problem.

Cassini1

This problem was optimized using the four DE algorithms identified in Chapter 9 and a stopping criterion of 0.001 average distance. A safety stopping criterion of 20000 iterations was used also, which is significantly more than the average number of evaluations per run $(4000 - 6000)$. Similar to [Vasile et al., 2011], 200 runs were performed with a maximum number of iterations of $4 \cdot 10^5$ and a success threshold of 68.8 m/s was used. The resulting success rates can be seen in Figure 14.1. Note that the performance of the algorithms considered by Vasile et al. [2011] were extracted with highest possible accuracy from their paper. Note that they only used four read-out points, and the author used many more to improve the accuracy within these points. This has no influence on the actual performance measured.

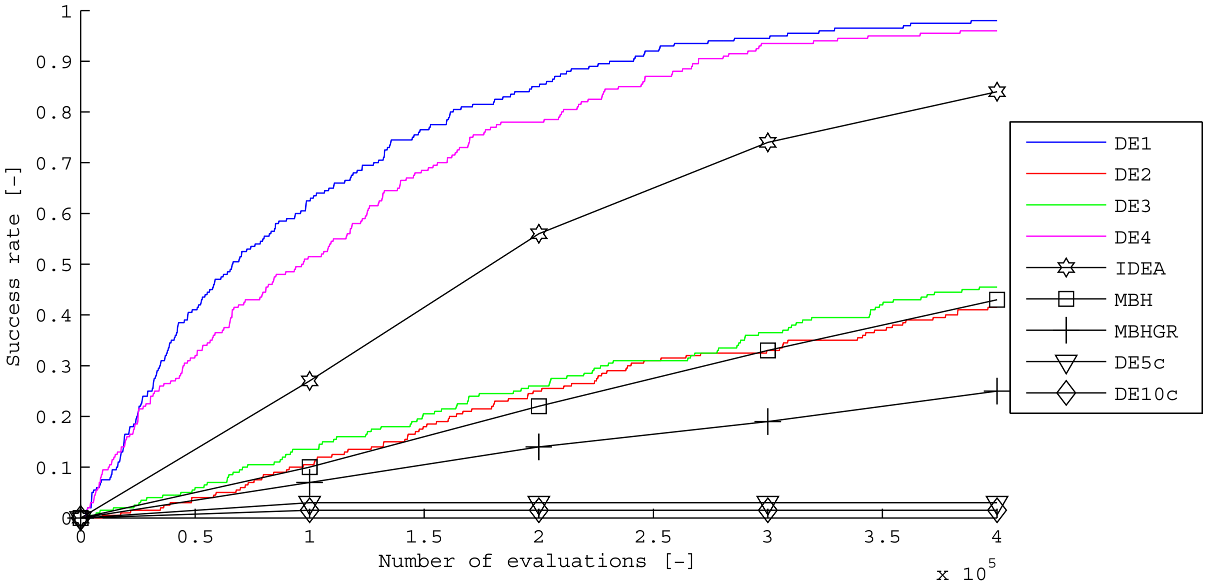


Figure 14.1: Comparison between performance of the tuned versions of DE in this thesis with the results of various algorithms considered in [Vasile et al., 2011] (black). The results show the success rate on the Cassini1 problem.

Figure 14.1 shows that algorithms DE1 and DE4 consistently outperform all algorithms considered in [Vasile et al., 2011], especially when compared with the DE algorithms considered in [Vasile et al., 2011]. This may be explained by the fact that DE was not consequently restarted in [Vasile et al., 2011].
It is especially interesting that DE1 actually reaches each performance threshold about 2.5 times faster than the IDEA algorithm considered in [Vasile et al., 2011]. Note that especially the performance of DE2 can be improved upon by selecting a slightly smaller population size due to the misconvergence discussed in Section 14.1 and illustrated in Table 14.2.

Cassini2

A new problem definition was written for the Cassini2 problem, because Vasile et al. [2011] used the Cassini2 problem in GTOP, which assumes unpowered swing-bys instead of powered swing-bys. Hence for comparison this problem was also defined in this thesis' software. The implementation has one minor difference, which is the order in which the variables are ordened in the decision vector. This is irrelevant for the algorithms considered in Vasile et al. [2011], but possibly important for this thesis' DE algorithm. It namely uses exponential cross-over, meaning that a series of subsequent variables are mutated. Hence it makes sense to group the additional variables by leg instead of by variable type as is done by default in GTOP. The variable ordening is the same as for the Cassini2 DSM VF problem, except that all ΔV

variables are taken out of the decision vector.

The results of this analysis are presented in Figure 14.2.

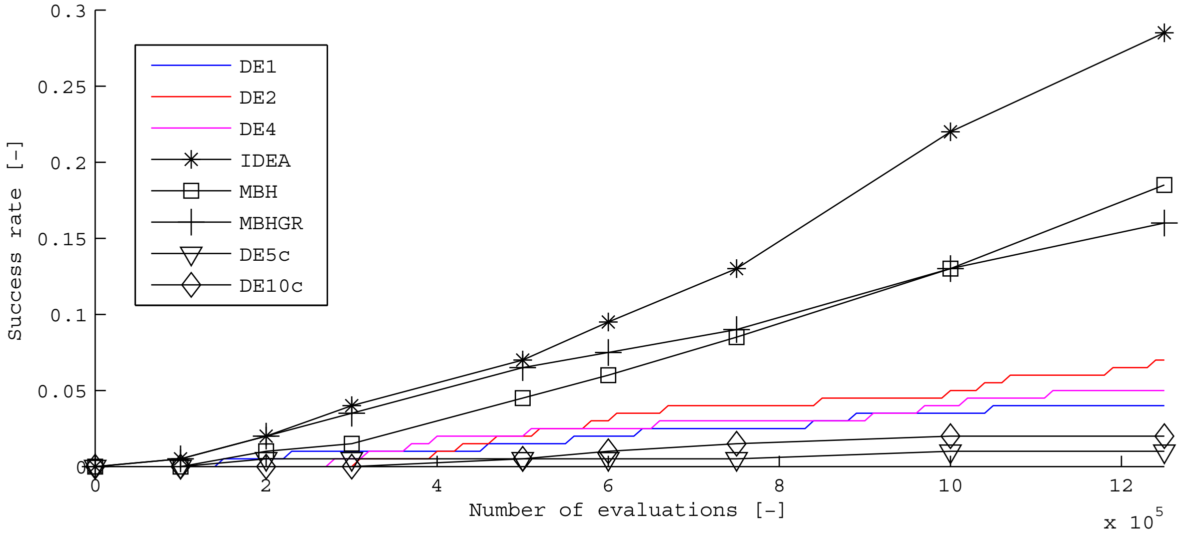


Figure 14.2: Comparison between performance of the tuned versions of DE in this thesis with the results of various algorithms considered in [Vasile et al., 2011] (black). The results show the success rate on the Cassini2 problem without powered swing-bys.

Figure 14.2 clearly shows that IDEA outperforms the DE algorithms considered in this thesis. Also the performance of MBH is better than that of DE. Still it is noted that the DE algorithms considered in this thesis outperform those used by Vasile et al. [2011].
By analyzing the optimization results, the reason for the relatively bad performance of the DE algorithms could be identified. The problem without DSMs namely contains a very strong suboptimal basin of attraction at 8608 m/s. Actually 53% of the DE2 runs managed to identify that minimum. This minimum is comparatively close to the actual optimum: all time variables are within 10% of the best solution, measured relative to the search space bounds. This may explain the good results of IDEA and MBH, which search repeatedly in the neighborhood of the current best solution. Still the good results of IDEA are noteworthy and it is likely that the local optimization runs, used to speed up the convergence of DE in IDEA, are very effective. A similar strategy will be discussed while optimizing the TandEM trajectories in Chapter 15.

Also it is interesting to note that the unpowered version of the Cassini2 trajectory is apparently more difficult to optimize than the powered version of the Cassini2 trajectory. A success rate of 52% is namely obtained after $1.25 \cdot 10^6$ evaluations using DE2 on the Cassini2 trajectory including powered swing-bys.

14.3 Conclusions

The following summation gives an overview of the conclusions drawn in this chapter:

- DE significantly outperformed PSO and GA in optimizing difficult trajectories including DSMs. GA was found unfit for finding solutions to these trajectories. PSO is able to find good solutions to the easier DSM problems, but fails to find the actual minimum in the full difficulty problems.
- The three best settings of DE manage to reliably find the best solution to the complex Cassini2 DSM VF problem within a very reasonable number of evaluations. $5 \cdot 10^6$ evaluations are required to have a 95% chance of finding the best solution. This corresponds to about 8-10 minutes on a single core of a modern CPU.

- It appears wise to include both a convergence strategy as well as an exploration strategy DE algorithm, given differences in problems. On some problems DE2, which employs an exploration strategy, outperforms DE1 and DE4, which employ a convergence strategy, by a factor of 2.5. On other problems DE1 and DE4 outperform DE2 by a factor of 10.
- The settings for DE1, DE2 and DE4 were found to be very sensitive to its tuning parameters. CR was especially sensitive but also a change in F resulted in a quick degradation of performance typically. Chapter 9 already showed that the population size was less influential.
- DE1, DE2, and DE4 consistently outperform other settings for DE encountered in literature and in PaGMO by various factors.
- DE1 and DE4 outperform all algorithms considered by Vasile et al. [2011] on the Cassini1 problem. Especially the fact that they outperform IDEA by a factor of 2.5 is noteworthy.
- The Cassini2 DSM VF Unpowered problem was found to be significantly more difficult to optimize than the Cassini2 DSM VF problem. The main reason is that a strong suboptimal basin of attraction is present in the unpowered variant of the problem.
- DE1, DE2 and DE4 outperform other versions of DE used by Vasile et al. [2011] on the Cassini2 DSM VF Unpowered problem. The suboptimal basin of attraction however did cause them to converge to a suboptimum frequently though. This is likely the main reason that IDEA outperforms DE1, DE2 and DE4 significantly.

Chapter 15

Optimization of the TandEM Problems

This chapter discusses the optimization of the TandEM problems of the GTOP database. To this end Section 15.1 describes the TandEM problems and the reasons for choosing these problems. Section 15.2 discusses the process used to verify the correct implementation of these problems. Subsequently Section 15.3 discusses the optimization process. Finally the conclusions are summarized in Section 15.4.

15.1 Problem Description

The TandEM problems are inspired by the Titan and Enceladus Mission (TandEM). The mission aims at reaching two moons in Saturn orbit: Titan and Enceladus. The launch window for the TandEM problem is specified to be between 2015 and 2025. The goal of the problems is to maximize the mass delivered into a high-eccentric final orbit around Saturn. The departure mass is calculated by interpolating the actual Atlas-501 rocket performance, dependent on the declination and V_∞ of the escape orbit. All problems consist of four legs and three gravity assists. Also a DSM is present in each of the four legs. The problems differ in the swing-by sequence to get to Saturn. Different planets are available depending on the swing-by number, as can be seen in the overview in Table 15.1.

Table 15.1: Swing-by planets used in the different TandEM sequences. [Izzo and Vinko, 2012]

Swing-by Number	Available Planets
1	Venus, Earth
2	Venus, Earth, Mars
3	Venus, Earth, Mars, Jupiter

The swing-by options give a total of 24 different combinations. The TandEM problems are further divided into unconstrained and constrained variants. The 24 unconstrained variants simply have a maximum transfer time of 2500 days for each leg, but no specific maximum for the total transfer time. The 24 constrained variants have a constraint of 10 years for the total transfer time from Earth to Saturn.

The TandEM problems were selected for the final optimization round because of various reasons. First of all it consists of many optimization problems, meaning that the optimization techniques and trajectory models can be subjected to a large number of test cases. The other test cases are namely single problems. Furthermore the trajectories belonging to these problems will likely have the most resemblance to the tuning problems, which were based on the Cassini trajectory.
Finally, and more importantly, the constrained TandEM problems are deemed to be the most realistic test cases among the trajectory problems. In practice the total time is also limited, but no specific time limit will be given for individual legs. Furthermore also the fact that the total mass is optimized, instead of the minimization of the ΔV of the interplanetary part of the mission, also makes the problem

more realistic. In practice, the launcher performance is dependent on the required $\boldsymbol{V}_\infty$ and may have significant impact on the trajectory.

The GTOP/PaGMO version of the TandEM constrained problems employs 18 decision variables to describe the problem, whose boundaries can be seen in Table 15.2.

Table 15.2: The state vector bounds for the TandEM constrained problems. [Izzo and Vinko, 2012]

State	Variable	Lower Bound	Upper Bound	Units
$\boldsymbol{x}(1)$	t_0	5475	9132	MJD2000
$\boldsymbol{x}(2)$	V_∞	2.5	4.9	km/s
$\boldsymbol{x}(3)$	u	0	1	-
$\boldsymbol{x}(4)$	v	0	1	-
$\boldsymbol{x}(5)$	T_1	0.01	0.99	t_{rem}
$\boldsymbol{x}(6)$	T_2	0.01	0.99	t_{rem}
$\boldsymbol{x}(7)$	T_3	0.01	0.99	t_{rem}
$\boldsymbol{x}(8)$	T_4	0.01	1.00	t_{rem}
$\boldsymbol{x}(9)$	η_1	0.01	0.99	-
$\boldsymbol{x}(10)$	η_2	0.01	0.99	-
$\boldsymbol{x}(11)$	η_3	0.01	0.99	-
$\boldsymbol{x}(12)$	η_4	0.01	0.99	-
$\boldsymbol{x}(13)$	$r_{p,1}$	1.05	10	R_{pl}
$\boldsymbol{x}(14)$	$r_{p,2}$	1.05	10	R_{pl}
$\boldsymbol{x}(15)$	$r_{p,3}$	1.05	10	R_{pl}
$\boldsymbol{x}(16)$	$b_{incl,1}$	$-\pi$	π	rad
$\boldsymbol{x}(17)$	$b_{incl,2}$	$-\pi$	π	rad
$\boldsymbol{x}(18)$	$b_{incl,3}$	$-\pi$	π	rad

Note that the time of flight variables (T_1 to T_4) are now scaled to the remaining time t_{rem}, which is calculated using Equation 15.1.

$$t_{rem,i} = 3652.5[days] - \sum_{1}^{i-1} T_i \tag{15.1}$$

Also it is noted that the variables u and v are used instead of θ and ϕ to determine the 3D orientation of the departure velocity. The variables relate to each other as follows in Equations 15.2 and 15.3.

$$\theta = 2\pi u \tag{15.2}$$

$$\phi = \arccos(2v - 1) - \frac{\pi}{2} \tag{15.3}$$

Finally it is noted that the pericenter radius of the swing-bys are not realistic for trajectories including Jupiter as a swing-by planet. This is also mentioned by the authors of the trajectory problems. They chose not to further complicate the set-up of the problems by making the bounds dependable on the swing-by sequence as the goal of the problems is purely academical [Izzo and Vinko, 2012].

The decision vectors belonging to the best putative solutions to the TandEM problems are not listed on the GTOP website, meaning that no knowledge is available on the best solutions except their fitness values.

15.2 Verification of the Implementation of the TandEM Problems

To allow the TandEM problems to be solved using the trajectory models developed in this thesis, the problems were implemented in this thesis' modeling software. The functions that calculate the launch mass were taken directly from GTOP. For all other calculations this thesis' software is used. The software needed small modifications, because the launch conditions had to be extractable for all trajectory models.
The problem was set up such that the trajectory model can be varied from leg to leg and the swing-by planet sequence can be changed. Both the unconstrained as well as the constrained variants were implemented.

To verify the correct implementation the trajectory, a verification was done for both the unconstrained and constrained variants. Note that the only difference between these variants is the computation of the time-of-flight of the legs. Both test cases were based on the example trajectory available in GTOP. Because the time of flight is smaller than 10 years, it allows for verifying both the implementation of the constrained and the unconstrained case.
The trajectory uses Venus, Earth and Earth to perform swing-bys before continuing to Saturn. A visualization of the trajectory is given in Figure 15.1.

The calculated final mass using various trajectory models described in this thesis is subsequently compared to the values calculated using GTOP. The result is given in Table 15.3.

Table 15.3: Comparison of the final mass of the TandEM trajectory problem using various trajectory models and both unconstrained and constrained variants with the mass calculated using GTOP (1437.57937 kg). Note that both the constrained and unconstrained trajectories achieved the same accuracy. This was expected because the only difference between both implementations is the way in which the times of flight are calculated, which is a very precise calculation compared to the Lambert targeting involved.

Variant	Model	Resulting mass [kg]	Absolute difference [kg]	Relative difference [-]
Unconstrained	MGA-1DSM-VF-UNP	1437.57909	$2.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-7}$
	MGA-1DSM-VF	1437.57909	$2.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-7}$
	MGA-1DSM-PF	1437.57925	$1.1 \cdot 10^{-4}$	$7.8 \cdot 10^{-8}$
Constrained	MGA-1DSM-VF-UNP	1437.57909	$2.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-7}$
	MGA-1DSM-VF	1437.57909	$2.8 \cdot 10^{-4}$	$1.9 \cdot 10^{-7}$
	MGA-1DSM-PF	1437.57925	$1.1 \cdot 10^{-4}$	$7.8 \cdot 10^{-8}$

The deviations are of the same order of magnitude as the earlier tests in Chapter 5. They are small enough to be caused by the various root-finder calls and the propagation of errors that are present therein.

15.3 Optimization Using Own Trajectory Model

The 24 different TandEM constrained problems were optimized using the best-performing settings of DE. It was quickly found out that this did not result in very good trajectories. The number of function evaluations that was required before the algorithms converged was orders of magnitude larger than what was found for the Cassini2 problem. This is especially peculiar because the Cassini2 problem has more design variables. Hence it was concluded that something strange was going on.

One of the main differences with respect to the Cassini2 trajectory is that the bounds on the time variables are much larger for the TandEM trajectories. The larger bounds on the departure time allows for much more repeat options than were present in the Cassini2 trajectory. Furthermore the options in the subsequent legs are also much larger. The transfers are not limited to single-revolution orbits and hence

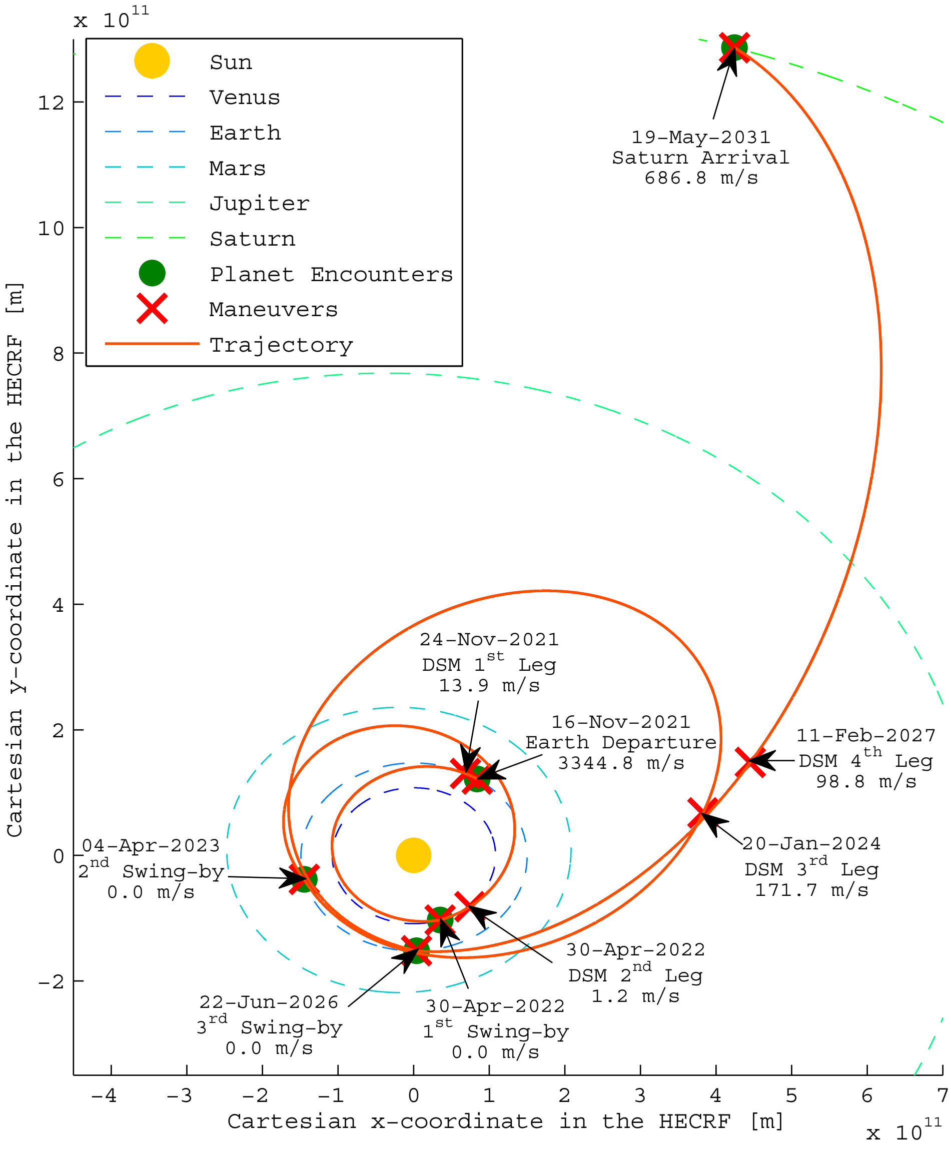


Figure 15.1: The trajectory belonging to the test case that was set up to verify the TandEM implementation in this thesis. Note that the trajectory is not the optimal trajectory for this swing-by sequence.

this also increases the number of options significantly. The number of local minima was already very large in the Cassini2 trajectory and now a large number of *isolated* clusters of minima are added. This significantly complicates the problem.
A hypothesis was formed that the individuals would all converge to different minima that are far apart. In case that would happen, the chance on finding a decision vector with a good fitness is very small. This is especially so given the high sensitivity around minima, as concluded in Chapter 13.

Hence to overcome the problem of isolated minima, the author thought about reducing the search space. This process had helped Vasile et al. [2008] to obtain good results before. The search space was reduced by equal amounts for all variables in [Vasile et al., 2008], [Vasile et al., 2009], [Vasile et al., 2010] and [Vasile et al., 2011].
It was however concluded in Chapter 14 that a properly tuned DE algorithm should manage to find the minimum for very complex and sensitive problems such as the Cassini2 trajectory. Hence the author noted that the main reason for reducing the search space should only be to get rid of all the isolated minima of comparable fitness. DE should be able to find the best solution to a problem with many minima that are part of the same group of solutions. The departure time and the times of flight were

deemed to be the main driver for isolated minima. Hence the author decided to test the performance if the search space of only t_0, T_1, T_2 and T_3 was reduced. Reducing the search space in more variables comes with an increased risk of pruning the best solution from the search space.

The results were very interesting. It turned out that reducing the search space to 6% of the total search domain for t_0, T_1, T_2 and T_3 resulted in convergence after 10^5-10^6 evaluations instead of 10^6-10^8 evaluations that were required before. The reduction to 6% was determined experimentally. The search space could have been reduced by less for some problems.

New Algorithm: Time Inflationary Algorithm

This result led to developing a new optimization algorithm for the TandEM problems. The algorithm is given in the pseudo-code in Algorithm 15.1. The algorithm is also discussed in words in the next paragraph.

Algorithm 15.1: Time Inflationary Algorithm (TIA)

```
 1: Set values for $m_g$, $\rho_g$, $n_g$, $m_l$, $\rho_l$, $n_l$, $m_{noImp}$, $\Delta_{fit}$
 2: Prepare global and local algorithm $A_g$ and $A_l$
 3: for $i = 1 : m_g$ do
 4:    while $n < n_g \&\& \rho_{av} < \rho_g$ do
 5:       Optimize problem using $A_g$
 6:    end while
 7:    Store champion of optimization
 8: end for
 9: Select best $m_l$ champions from step 7
10: for $i = 1 : m_l$ do
11:    Set $j$ to 0, set best champion to champion $i$
12:    while $j < m_{noImp}$ do
13:       Narrow bounds around $t_0$, $T_1$, $T_2$, $T_3$ to 6 % around best champion
14:       while $n < n_l$ && $\rho_{av} < \rho_l$ do
15:          Optimize problem using $A_l$
16:       end while
17:       if champion > best champion
18:          if champion > best champion + $\Delta_{fit}$
19:             $j = 0$
20:          end if
21:          best champion = champion
22:       end if
23:    end while
24: end for
25: if $n_{tot} < n_{max}$
26:    goto step 3
27: end if
```

First m_g global searches are done using global algorithm A_g. The global runs are stopped after the average distance between individuals, ρ_{av}, drops below ρ_g or the number of evaluations reaches the maximum of n_g evaluations. The m_l best decision vectors are then used as initial guess for the local searches. Local searches are performed using local algorithm A_l, reducing the bounds of t_0, T_1, T_2 and T_3 to x_{loc} % of the search space around the best decision vector found in the global round. The search continued until the average distance between individuals, ρ_{av}, drops below ρ_l, or until the maximum number of evaluations for the local phase, n_l is reached. To take into account that the success rate of the local searches is less than 100%, m_{noImp} runs are done for each setting. In case the local search finds a result that is better than the previous best of the particular local run, the location is updated and the next search is centered on the newly found point. In case the local search finds a result that is more than Δ_{fit} better than the previous solution, m_{noImp} is reset to allow for enough local searches around the new optimum. After all local searches have been performed, a new global search will be done to obtain new starting points.

Settings of the Algorithm

Some preliminary runs were performed to determine the large number of variables that is present in Algorithm 1. First of all it turned out that algorithm DE2 significantly outperformed the other algorithms. Where the other algorithms struggled to converge, DE2 converged quite rapidly and also to better solutions. Both in the local as well as the global optimization rounds this rapid convergence is very useful, as it allows to take much more samples. The empirical relation determined in Section 9.4 would indicate a population size of 56 for a problem of 21 variables. The population size was rounded to 60 for both the local and the global rounds.
Both rounds use a ρ of 10^{-5} as a stopping criterium for the optimization. This should mean that the results are accurate to roughly 0.1 to 1 kg if the problem is equally sensitive as the typical trajectories encountered in Section 13. One important note here is that the average distance should *not* be taken on all decision variables. A large number of different decision vectors namely result in the same physical optimal trajectory if one or more legs do not contain a DSM. To cope with this issue, the average distance was taken over all variables except the η's.
As an additional stopping criterium $1.5 \cdot 10^6$ and $4 \cdot 10^6$ were taken as maximum number of evaluations for the global and local rounds respectively. It is noted that the global round is typically stopped because of this criterium. For the local rounds, the distance stopping criterium is much more important and the maximum number of evaluations is just a measure to make sure the algorithm does not 'get stuck'.
In total 40 global samples were taken each global round. The best 10 samples would then be used as initial guesses for the local rounds. The local rounds would continue until 8 optimization runs were done in which no significant improvement was found. An improvement of more than 10 kg was deemed significant for resetting the no-improvement counter.

An overview of the settings is given in Table 15.4.

Table 15.4: Settings used for TIA in the optimization of the TandEM trajectories.

Variable	$A_g(n_{pop})$	m_g	ρ_g	n_g	$A_l(n_{pop})$	m_l	ρ_l	n_l	m_{noImp}	Δ_{fit} [kg]
Value	DE2(32)	40	10^{-5}	$1.5{\cdot}10^6$	DE2(60)	10	10^{-5}	$4{\cdot}10^6$	8	10

Optimization Results

This section gives the results of the actual optimization runs. The 24 constrained problems were optimized both using an MGA-1DSM-VF (Multiple Gravity Assist with 1 Deep Space Maneuver per leg adopting a Velocity Formulation) trajectory model as well as using an MGA-1DSM-VF-UNP (Multiple Gravity Assist with 1 Deep Space Maneuver per leg adopting a Velocity Formulation and assuming UNPowered swing-bys) trajectory model. A total limit of 10^9 function evaluations was used in each optimization run. Table 15.5 gives an overview of the results obtained in this way. It is noted that the best two solutions to both the powered and unpowered model were subsequently refined by an additional local search round with a ρ_l of 10^{-6}.
The table also lists the best solutions on the GTOP site. These solutions were found by Addis et al. [2008] using MBH, except for those belonging to sequence 6 which was found by Izzo. It is noted that the results found by the unpowered trajectory model give the best estimation of the performance of the optimization algorithm. The results of the powered model namely also include the option to perform a powered swing-by, which is not possible in GTOP.

Before discussing the actual results, it is noted that the solutions of all sequences were implemented in GTOP and subsequently run to verify the correctness of the improvements found in this thesis. All tests were positive. The results will be refined locally in the GTOP problem definition and will subsequently be mailed to ESA such that they can be listed on the GTOP site.

Table 15.5: Best solutions found for the various constrained TandEM problems, compared to those listed on the GTOP website [Izzo and Vinko, 2012]. Note that a coloring scheme is used to visualize good and bad results. Dark green means an improvement of more than 10% was made, light green means that the improvement was less than 10%, yellow/orange means that the solution was worse than GTOP by less than 10% and red means that it was worse than GTOP by more than 10%.

Sequence #	Swing-by Planets	GTOP Solution	MGA-1DSM-VF-UNP	MGA-1DSM-VF
		(kg)	Result (kg)	Result (kg)
1	Venus-Venus-Venus	1082.03	1087.66	1087.66
2	Venus-Venus-Earth	1265.43	1149.87	1338.72
3	Venus-Venus-Mars	544.61	664.59	686.31
4	Venus- Venus -Jupiter	650.86	651.55	683.08
5	Venus- Earth -Venus	938.06	930.92	930.92
6	Venus- Earth -Earth	1476.01	1410.41	1410.41
7	Venus- Earth -Mars	712.70	814.54	815.34
8	Venus- Earth -Jupiter	736.50	740.60	790.38
9	Venus- Mars -Venus	589.03	524.70	606.90
10	Venus- Mars -Earth	794.81	859.21	891.88
11	Venus- Mars -Mars	334.88	336.62	336.62
12	Venus- Mars -Jupiter	261.40	264.45	341.39
13	Earth-Venus-Venus	554.33	539.83	578.31
14	Earth -Venus-Earth	806.48	738.13	844.34
15	Earth -Venus-Mars	293.82	460.00	532.25
16	Earth - Venus -Jupiter	428.06	418.27	469.67
17	Earth - Earth -Venus	551.94	558.51	588.42
18	Earth - Earth -Earth	956.39	969.96	1012.85
19	Earth - Earth -Mars	498.85	506.27	543.79
20	Earth - Earth -Jupiter	700.00	663.49	710.07
21	Earth - Mars -Venus	454.94	502.32	512.78
22	Earth - Mars -Earth	839.75	928.49	927.95
23	Earth - Mars -Mars	454.51	459.33	483.65
24	Earth - Mars -Jupiter	280.49	281.36	360.03

It can be seen in Table 15.5 that improvements have been found for 16 of the 24 TandEM constrained cases with the MGA-1DSM-VF-UNP trajectory model. In general it may be noted that the algorithm described in this chapter performs very well, given that it managed to find very good solutions to these difficult problems frequently.

On the other hand, it is also noted that there are various cases for which the best solution has not been found using the strategy described in this chapter. It is possible that they will be found in case more optimization runs would be performed. Also it may be caused by the fact that the strategy described in this chapter is less suited for these problems than the MBH algorithm employed by Addis et al. [2008]. It appears that there is no significant correlation between the swing-by planets and the result with respect to [Addis et al., 2008].

Also it may be possible that the performance on some problems may be improved by using other settings of TIA. Possibly the 6% in each variable is too restrictive, or not restrictive enough to produce good results. Also the other settings may influence the results obtained.

Table 15.5 also shows that improvements have been found for 22 of the 24 sequences using the MGA-1DSM-VF trajectory model. It is however noted that these improvements are a mix of the different optimization strategy and the different trajectory model, causing for an unfair comparison. Chapter

16 will discuss the differences between the powered and unpowered swing-by trajectory models in more detail.

15.4 Conclusions

This chapter developed a new algorithm specifically for optimizing the TandEM constrained trajectories present in GTOP. Preliminary runs indicated that a large number of isolated minima were present in these trajectory problems. This caused DE to converge very slowly. A Time Inflationary Algorithm (TIA) was developed to overcome this issue. Basically it relies on narrowing the search space of the time related variables, which are likely to be the most important causes for the isolated minima. This allowed DE to converge significantly faster.

Using TIA, 16 of the 24 constrained TandEM trajectories was improved upon, demonstrating the proper performance of TIA. On the other hand, TIA was unable to identify the best solutions for some other trajectories. In general it can be noted that this algorithm was very effective, as a majority of cases resulted in equal or better solutions than those listed on the GTOP website.

Chapter 16

Trajectory Model Comparison

This chapter deals with the comparison between different trajectory models. Some of the analyses that are performed here have been partially discussed before, but in this chapter the main focus is on comparing the different trajectory models. In particular, the effects of adding DSMs and powered swing-bys to the trajectory will be evaluated and quantified. This will be done both on the basis of the Cassini2 trajectory, as well as on the TandEM trajectories. Section 16.1 discusses the effects of adding DSMs to the trajectory. Section 16.2 subsequently discusses the effects of powered swing-bys with respect to unpowered swing-bys.

Also the advantages and disadvantages of the PF and VF trajectory models, and hybrids between them, are further analyzed in Section 16.3. A proposition for a new trajectory model is made in Section 16.4. Finally the conclusions of this chapter are summarized in Section 16.5.

16.1 Improvement of Trajectories Including DSMs

This section analyses the improvements that can be made by including DSMs to the trajectory. This analysis will be performed by comparing results obtained using the MGA trajectory model with those obtained using the MGA-1DSM trajectory models.

It needs to be noted that in general, the trajectory models that include DSMs implicitly also allow for legs of more than one revolution around the Sun. This issue is important when comparing between MGA models and DSM models, because the MGA model does not allow for multiple revolutions. In practice these multiple revolutions are not encountered in many of the trajectories considered here, because of the time constraints when going to Saturn. A fairer comparison would however be to compare the MGA-1DSM models with an MGA model in which multiple revolutions are possible as well.

A first comparison between trajectories with and without DSMs is made based on the Cassini2 trajectory. Because the optimal DSM trajectory does not include multiple revolutions, the comparison can be considered fair. Both the Cassini2 MGA problem as well as the Cassini2 DSM problems have been optimized very often during the tuning phase that the results are likely to be the actual solution, although one can never be sure of course. The resulting ΔV for both cases is given in Table 16.1.

Table 16.1: Analysis of the required ΔV for the Cassini2 Model with and without DSMs. Note that the best solution for the MGA-1DSM-PF trajectory model is given here. No constraint on the departure velocity was namely present in the MGA model either. This issue was discussed in more detail in Section 8.2.

	With DSMs	Without DSMs
ΔV-cost [m/s]	7958	9674

It is concluded from Table 16.1 that the addition of DSMs to the Cassini2 trajectory results in a very significant improvement of 1.7 km/s.

A second comparison is made based on the TandEM trajectories. MGA variants of the TandEM constrained trajectories were optimized 1000 times using both DE1 and DE2. Subsequently they are compared against the solutions found using the MGA-1DSM-VF trajectory model in Chapter 15. The results can be seen in Table 16.2. It is noted explicitly that the probability that the solutions listed in Table 16.2 are also the actual solutions is much lower than for the results given in Table 16.1.

Table 16.2: Comparison between the best solutions found for the constrained TandEM trajectories using an MGA and an MGA-1DSM-VF trajectory model.

	Result MGA-1DSM-VF	Result MGA	Improvement by DSM model	
Sequence #	model (kg)	model (kg)	Absolute (kg)	Relative (%)
1	1087.66	423.36	664.3	157
2	1338.72	935.22	403.50	43
3	686.31	319.51	366.80	115
4	683.08	460.00	223.08	48
5	930.92	403.89	527.03	130
6	1410.41	649.90	760.52	117
7	815.34	476.44	338.90	71
8	790.38	744.86	45.52	6
9	606.90	484.40	120.78	25
10	891.88	608.88	283.00	46
11	336.62	123.50	213.12	173
12	341.39	417.04	-75.65	-18
13	578.31	195.38	382.93	196
14	844.34	754.70	79.22	10
15	532.25	77.38	454.87	588
16	469.67	278.30	191.37	69
17	588.42	155.94	431.25	277
18	1012.85	506.27	506.58	100
19	543.79	222.05	321.74	145
20	710.07	645.60	64.47	10
21	512.78	249.50	263.28	106
22	927.95	754.33	173.62	23
23	483.65	354.56	129.09	36
24	360.03	217.37	142.66	66
			Average improvement	105

Before discussing the results in detail, the result of sequence number 12 needs some further explanation. The MGA trajectory model namely managed to find a better solution than the MGA-1DSM-VF trajectory model. This is caused by the fact that the ΔV for the swing-bys is not limited for the MGA trajectory model, whereas a maximum of 2000 m/s was used for the MGA-1DSM-VF trajectory model. The solution of the MGA trajectory contains a swing-by with 3768 m/s in that trajectory, thereby outperforming the MGA-1DSM-VF model. In hindsight, the bounds for the MGA-1DSM-VF problem should have been set differently of course.

Table 15.5 clearly shows that the MGA trajectory model results in trajectories that are suboptimal. For the cases listed in Table 16.2, the MGA-1DSM-VF trajectory model is able to find solutions with 105% more mass than those calculated with the MGA trajectory model. It is noted that the difference is not that large on all instances. Especially on trajectories with suboptimal swing-by sequences this difference is very large. Also a comparison between the best solutions found by the MGA-1DSM-VF model (1410 kg) and the best solution found by the MGA model (935 kg) clearly shows that the MGA trajectory model performs much worse than the MGA-1DSM-VF model.

16.2 Improvement of Trajectories Including Powered Swing-bys

This section analyzes the improvements that can be made by including the option to perform powered swing-bys to the trajectory. This analysis is performed by comparing the results obtained using the MGA-1DSM-VF trajectory model with those obtained using the MGA-1DSM-VF-UNP trajectory model. It is noted explicitly that the only difference between the MGA-1DSM-VF-UNP and MGA-1DSM-VF trajectory models is the inclusion of an option for a powered swing-by in the latter model. Hence fair comparisons can be made with respect to the effects of adding a powered swing-by.
Two comparisons are made in this section. First the results on the Cassini2 trajectory will be discussed. Subsequently the results on the TandEM trajectories will be discussed.

Both the unpowered and powered variants of the Cassini2 trajectory have been optimized very frequently in this thesis. Hence it is very likely that the best putative solution to these trajectories is equal to the actual solution. An overview of the ΔV required for both trajectories is given in Table 16.3.

Table 16.3: Analysis of the required ΔV for the Cassini2 Model with and without powered swing-bys.

	With powered swing-bys	Without powered swing-bys
ΔV-cost [m/s]	8237	8383

Although the improvement obtained using a powered swing-by is much less than that of adding DSMs to the trajectory, a difference in ΔV of 147 m/s is definitely significant in the optimization of high-thrust trajectories.

A similar analysis is performed based on the best results found during the optimization of the TandEM trajectories. Again it is noted that these results are much less likely to be the actual solutions to the trajectory problems than those found for Cassini2. Still the results allow for an interesting comparison. The results are given in Table 16.4.

Table 16.4 shows that a reasonable gain can be made by employing powered gravity assists. The gain differs significantly across the considered problems. For five of the considered trajectories no significant improvement was found using the powered swing-by model. Actually the solution to one of the instances (22) was better using the unpowered swing-by model. This is likely caused by premature convergence of the powered swing-by model. Although much less significant than the gain of employing DSMs, the average improvement of 7.8% can be regarded as significant.

16.3 Comparison Between Optimizing PF and VF Trajectory Models

The search space analysis in Chapter 13 already revealed that there was a big difference between the PF and the VF trajectory models. The PF offers the advantage that the last legs are much less dependent on the first legs. On the other hand it is noted that the PF model is much more sensitive in general. This sensitivity is caused by the large range of possible DSM locations and also by the fact that the DSM location is fixed independent of the moment of application. The VF model prunes a large number of bad trajectories because the DSM location is bound by the range of feasible outgoing velocities after planetary encounters. Furthermore the fact that the trajectory is propagated until the moment at which the DSM is applied significantly reduces the dependency on η. Also it was noted in Chapter 13 that a hybrid between a VF and PF trajectory was less sensitive than a full VF trajectory model.

This chapter analyzes the effects of this reduced sensitivity on the optimization of the trajectories. All 32 possible hybrids between PF and VF trajectory models were optimized to calculate the $n_{95\%}$ of these hybrids. The trajectories were optimized 200 times using both the DE1 and DE2 algorithm. The best $n_{95\%}$ is then used as measure for the difficulty of optimizing a certain hybrid model. The runs lasted for

Table 16.4: Comparison between the best solutions found for the constrained TandEM trajectories using an MGA-1DSM-VF and an MGA-1DSM-VF-UNP trajectory model.

	MGA-1DSM-VF	MGA-1DSM-VF-UNP	Improvement by powered swing-by	
Sequence #	model (kg)	model (kg)	Absolute (kg)	Relative (%)
1	1087.66	1087.66	0.00	0
2	1338.72	1149.87	188.85	16
3	686.31	664.59	21.72	3.3
4	683.08	651.55	31.53	4.8
5	930.92	930.92	0.00	0
6	1410.41	1410.41	0.00	0
7	815.34	814.54	0.80	0.1
8	790.38	740.60	49.77	6.7
9	606.90	526.51	78.68	15
10	891.88	859.21	32.67	3.8
11	336.62	336.62	0.00	0
12	341.39	264.45	76.94	29
13	578.31	539.83	38.49	7.1
14	844.34	738.13	95.79	13
15	532.25	460.00	72.25	16
16	469.67	417.27	51.39	12
17	588.42	557.45	29.74	5.3
18	1012.85	969.96	42.89	4.4
19	543.79	506.27	37.52	7.4
20	710.07	663.49	46.58	7.0
21	512.78	502.03	10.75	2.1
22	927.95	928.49	-0.54	-0.1
23	483.65	459.33	24.31	5.3
24	360.03	281.36	78.66	28
			Average improvement	7.8

$2\cdot10^6$ evaluations, which is significantly more than the average number of evaluations at which success is obtained ($5\cdot10^5$ for DE1 and $1\cdot10^6$ for DE2). Still it is possible that some runs were stopped prematurely as a consequence.

An overview of the best 8 combinations can be seen in Table 16.5 and the full results can be accessed in Appendix E. One note must be made with respect to these results though. Because the departure velocity was not bound for the PF variant of the departure leg, the optimum for sequences starting with a PF leg is lower than that of a VF departure leg. It is noted that this optimum may be easier or more difficult to find. If that is the case, it will influence the results in Table 16.5.

Table 16.5: The best $n_{95\%}$ encountered by either DE1 or DE2 after 200 runs for various hybrid models consisting of VF and PF legs for the Cassini2 DSM problem. This table shows the best 8 hybrids.

Model type per leg					
1	2	3	4	5	$n_{95\%}$ $[\cdot10^6]$
VF	VF	VF	VF	VF	6
PF	VF	VF	VF	VF	9
VF	PF	VF	VF	VF	13
VF	VF	VF	VF	PF	13
PF	VF	VF	VF	PF	19
VF	PF	VF	VF	PF	23
PF	PF	VF	VF	VF	24
VF	VF	VF	PF	VF	29

Table 16.5 shows that the DSM VF trajectory model is still easier to optimize than any hybrid with a

PF model.
Something else that can be seen in Table 16.5 is that it shows a strong similarity with Table 13.12. Although the results are not exactly what would be expected from the global basin sizes, many of the same sequences are present in both tables. Hence it appears that, although not very accurate, the sensitivity analysis does reveal which sequences are more likely to be optimized successfully.

Finally it is noted that 15 of the hybrid trajectories could not be optimized reliably at all. Again it is noted that a large similarity is observed with the models that were revealed to be very sensitive in Chapter 13.

16.4 Proposition for New Trajectory Model

Following the observations in Section 16.3, the author noted that the PF is not the only way to remove the dependency of the last legs on the first legs, which is present in the VF trajectory model.
An alternative way of doing so is to use three variables to specify the *outgoing* velocity after a swing-by planet. Similar to a departure leg, this may be done using the following three variables:

- V_∞: the magnitude of the relative velocity with the swing-by planet,
- θ: the in-plane angle of the relative velocity with the swing-by planet,
- ϕ: the out-of-plane angle of the relative velocity with the swing-by planet.

Subsequently a gravity assist calculator may be used to patch the incoming and outgoing velocities at the swing-by planet, thereby removing the dependency on the previous legs. Subsequently the trajectory can be propagated using a Kepler propagator up to the DSM location.
The range of possible DSM locations is much more limited this way than in the PF. Furthermore the use of a Kepler propagator means that the trajectory will be much less sensitive to η. Finally the Kepler propagator will speed up the computation of the leg when compared to the Lambert targeter required for the PF.
An overview of this calculation scheme can be seen in Figure 16.1.

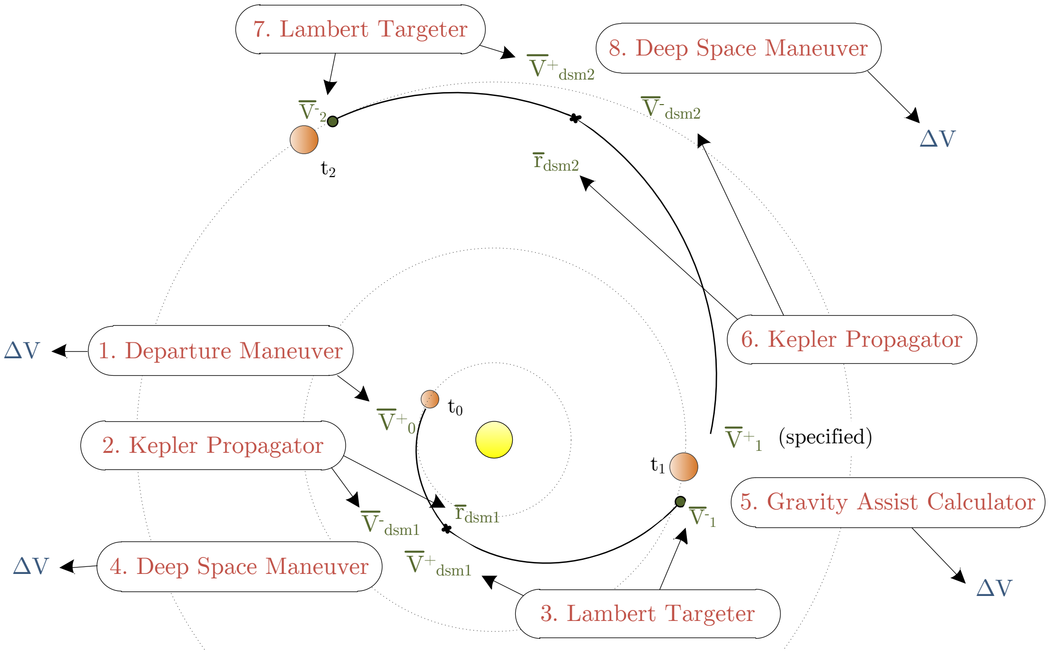


Figure 16.1: A schematic representation of the proposed trajectory model.

The effectiveness of this trajectory model was not investigated by the author, given the time constraints for an MSc thesis. It is noted that it may be a very interesting lead for a future student to investigate. In general it is noted that a normal VF leg is likely less sensitive. However as more legs are added, replacing one or multiple legs by the new trajectory model may become very attractive as the new model will significantly reduce the interdependency of different legs. Hence hybrids between the VF trajectory model and the newly proposed model may reduce the sensitivity of a long trajectory significantly, thereby allowing for easier optimization.

16.5 Conclusions

The following summation gives an overview of the conclusions drawn in this chapter:

- The MGA-1DSM-VF model performs significantly better than the MGA trajectory model. The MGA-1DSM-VF trajectory model found a solution of 1.7 km/s better than the best solution for the MGA model on the Cassini2 trajectory. Also for the constrained TandEM problems a very considerable average mass increase of 105% is reported. These results show the importance of using a model that includes an option to perform DSMs.
- The addition of powered swing-bys may increase the efficiency of a trajectory. An improvement of 147 m/s was found for Cassini2. Also an average mass increase of 7.8% is reported for the constrained TandEM problems.
- A trajectory modeled with the MGA-1DSM-VF trajectory model is easier to optimize using DE than a trajectory modeled with an MGA-1DSM-PF trajectory model, or hybrids of both.
- The sensitivities calculated in Chapter 13 provide a reasonable guess of the effort required to optimize a trajectory.
- A proposition of a new trajectory model was explained, which may reduce the sensitivity of the MGA-1DSM-VF trajectory model for long trajectories.

Chapter 17

Conclusions and Recommendations

This chapter gives an overview of the conclusions in this chapter and the recommendations for future research. The conclusions are dealt with in Section 17.1 and the recommendations are given in Section 17.2

17.1 Conclusions

This section discusses the most important conclusions drawn in this thesis. The conclusions are divided into the six separate thesis goals defined in Chapter 3.

Astrodynamic Tool Development

As a part of this thesis project, various astrodynamic functions were developed within the Tudat context. The Kepler propagator was rewritten with better starter values for the mean to eccentric anomaly conversions. Also an unpowered and a powered gravity assist propagator were written. Finally the gravity assist calculator was completely rewritten. These functions were properly tested and are now present in Tudat.
Also trajectory modelling software was developed with a very general set-up. The software allows the user to build up trajectories consisting of different trajectory models. The software was verified against GTOP and successfully used in this thesis to compare different trajectory models, and hybrid trajectory models.

Search Space Investigation

A new definition for the basin of attraction was proposed which intends to represent the difficulty of optimizing a problem using a global optimization algorithm. The sensitivity of all parameters was analyzed for different problems. This revealed why many optimization runs got stuck on suboptimal solutions for various problems. Also the analysis revealed the high sensitivity of a VF model on its first legs. The PF model was found to be highly sensitive to all of its legs, mainly caused by a much higher sensitivity in η, but also in θ and r. The sensitivity of hybrids was also analyzed, revealing that a hybrid between a PF and a VF model is less sensitive than a trajectory that is purely made up of VF legs. Although this hybrid could not be optimized faster than the full VF variant, the global basins were found to be a reasonably accurate measure for the difficulty of optimizing a certain problem in general.
It is noted here explicitly that the definition of the global basin size may be improved upon. Especially the fact that correlation between variables was not taken into account may have a strong influence on the results obtained this way. The analysis did yield very interesting results though.

Importance of Tuning

A very rigorous tuning scheme was applied to DE, GA and PSO. A large number of tuning problems were set up that simplify the trajectory in three different methods. First of all the trajectory model was simplified for all or some legs. The second way was to optimize only a subset of legs, fixing the variables of the others. Thirdly the bounds on decision variables were reduced. The number of evaluations to obtain a 95% chance on finding the actual solution within 50 m/s was used as a performance measure to compare between algorithms.

A large number of different settings were tested for DE, GA and PSO. The tuning process was split up into phases in which each phase pruned part of the parameter settings, eventually ending up with the apparent best performing settings. The tuning process was very successful for DE, but more complicated for GA and PSO. GA and PSO were namely insensitive to their tuning parameters on easy problems, but also not powerful enough to include a large number of settings on the more difficult problems.

The tuned versions of DE were finally compared with settings from literature. The tuned settings of DE significantly outperformed all DE settings encountered in literature very. Most settings in literature were unable to identify the best solution to the Cassini2 DSM VF problem reliably. It is hence concluded that proper tuning of DE is critical.

The tuned settings of DE were found to be highly sensitive to CR and also very sensitive to F. Although the population size was determined as comparatively less sensitive, choosing a wrong population size will also result in a significant reduction of performance.

Comparison Between Different Optimization Algorithms

The tuned versions of DE, GA and PSO were compared revealing that DE is the only algorithm that is capable of reliably optimizing difficult trajectory optimization problems. Also a MS algorithm was tested, but the local optimization algorithms used in this thesis were not powerful enough for MS to be effective. Various local algorithms furthermore proved to be unstable.

The properly tuned versions of DE were compared against performances mentioned in literature. They significantly outperformed other versions of DE and also managed a better performance than IDEA when applied to the Cassini1 problem. This result was obtained both by the proper tuning scheme as well as the implementation of a good restart criterion. When applied to the Cassini2 problem with unpowered swing-bys, IDEA significantly outperformed the DE algorithms in this thesis. This appeared to be caused by a strong suboptimal basin of attraction near the best minimum, which IDEA copes with much better because of the local search rounds performed therein.

Comparison Between Different Trajectory Models

The MGA-1DSM trajectory models consequently found solutions that were significantly better than those obtained using an MGA trajectory model. The difference was 1.7 km/s for the Cassini2 problem and for the TandEM problems, the MGA-1DSM models came up with masses that were double those of the MGA models. These results show the importance of using a model that includes the option to perform DSMs.

Also the addition of a powered swing-by may increase the efficiency of a trajectory significantly. An improvement of 147 m/s was found for Cassini2 and an average mass increase of 7.8% was found by comparing the results on the TandEM problems. Also it is noted that the Cassini2 trajectory was easier to optimize using a powered swing-by model, despite the increase in decision vector.

Finally it was shown that the VF model is easier to optimize than a PF model, or hybrids between them. An analysis of the sensitivities of both models led to a proposal for a new trajectory model, which may help to reduce the sensitivity of the VF model to its first legs.

Strategy to Optimize Complex MGA-1DSM Trajectories

A new algorithm was developed to optimize the constrained TandEM cases in GTOP. Because these problems have of the large number of isolated clusters of minima, DE failed to converge. The proposed algorithm narrows the bounds on the time variables, thereby reducing the search space. This allowed DE to converge significantly faster and more reliably. Global and local searches are performed in a smart manner to search for the global minimum.

Time Inflationary Algorithm (TIA) was able to find better solutions to 16 of the 24 TandEM trajectories. Hence it can be noted that TIA is a very effective algorithm to find solutions to difficult trajectory optimization problems. Still it is also noted that the exact optimization approach is problem-dependent. A proper analysis of the problem and its difficulties is indispensable for drafting a good optimization algorithm.

17.2 Recommendations

Many aspects related to high-thrust interplanetary trajectory optimization have still been left untreated. The following summation intends to give an overview of possible future work in this area. This future work is divided into two parts: aspects related to trajectory models and aspects related to optimization.

Trajectory models

- The proposal for a new trajectory model in Chapter 16 may be worked out in detail. The model has significant advantages over the VF model with respect to the propagation of the sensitivity across different legs. A hybrid between both models will likely be less sensitive and may therefore be easier to optimize.
- Multiple revolution Lambert targeting has been neglected in this thesis. Future work could look into the inclusion of multiple revolution Lambert targeters. Especially for trajectories going to the inner planets this may yield significant improvements.
- The search space analysis may be conducted more thoroughly. Although interesting results were obtained, the effects of correlation between variables was neglected. This is likely to have a strong effect on the resulting sensitivities. Note especially that the global basin definition may also be improved upon. Also it is noted that the application of the search space analysis on a larger set of problems may reveal interesting trends.
- The options to include multiple DSMs per leg may be analyzed. Olympio [2009] already showed that the addition of a second DSM in some legs may yield a considerable improvement for some trajectories. This analysis may either be conducted by employing the primer vector theory to improve the solutions found using current methods or by optimizing a semi-analytic model with options to perform more than 1 DSM per leg.

Optimization

- No powerful local optimization techniques were included in this thesis. Given the good performance of MBH, it may be very worthwile to investigate the performance of SNOPT or other powerful local optimization techniques. Both MS as well as MBH schemes can be tested.
- The exact way in which the rapid convergence is achieved by IDEA may be further analyzed. A powerful local optimization technique should be used for this. IDEA showed significantly better performance than DE on the Cassini2 problem with unpowered swing-bys. Given that the DE considered in this thesis outperformed the DE algorithms considered in [Vasile et al., 2008], improvements may be made with respect to IDEA as used in [Vasile et al., 2008].
- Branching and separate evolution strategies were not investigated in this thesis. These may further improve the efficiency.
- The effects of the problem type on the tuning settings of DE may be further analyzed. It may be interesting to investigate if the ideal tuning regime changes when going to Mercury for instance.
- Primer vector theory could be used to improve the solutions found using optimization runs performed with global optimization algorithms.

Part V

Appendices

Appendix A

Additional Tuning Results of DE

This appendix shows additional tuning results of DE. First all the relevant ‘raw’ results for the tuning process of DE are given in Appendix A.1. Subsequently the additional pictures for the population size determination are given in Appendix A.2.

A.1 Raw Results

An overview of the figures where the results can be found is given in Table A.1 below.

Table A.1: Overview of the results that were obtained for DE and where they can be found.

Problem	Phase	Sampling	Figure	Page
EJS MGA	1	Grid	A.1	200
EMJS MGA	1	Grid	A.2	201
EVEJS MGA	1	Grid	A.3	202
Cassini2 First Leg PF	1	Grid	A.4	203
Cassini2 First Leg VF	1	Grid	A.5	204
Cassini2 Last Leg PF	1	Grid	A.6	205
Cassini2 Last Leg VF	1	Grid	A.7	206
EJS MGA	1	Random	A.8	207
EMJS MGA	1	Random	A.9	207
EVEJS MGA	1	Random	A.10	208
Cassini2 First Leg PF	1	Random	A.11	208
Cassini2 First Leg VF	1	Random	A.12	209
Cassini2 Last Leg PF	1	Random	A.13	209
Cassini2 Last Leg VF	1	Random	A.14	210

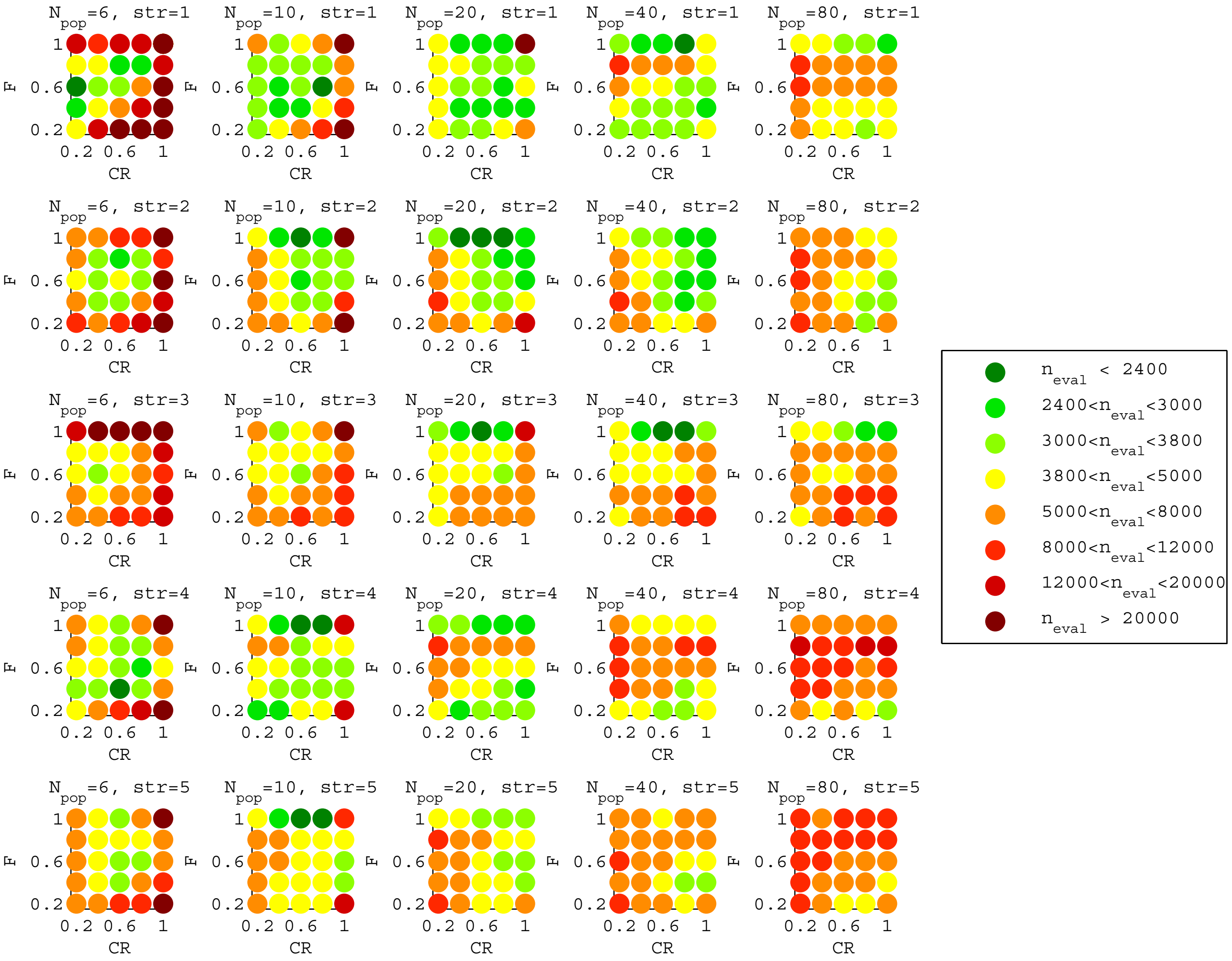


Figure A.1: The results of the first phase tuning process of DE on the EJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The resuls are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

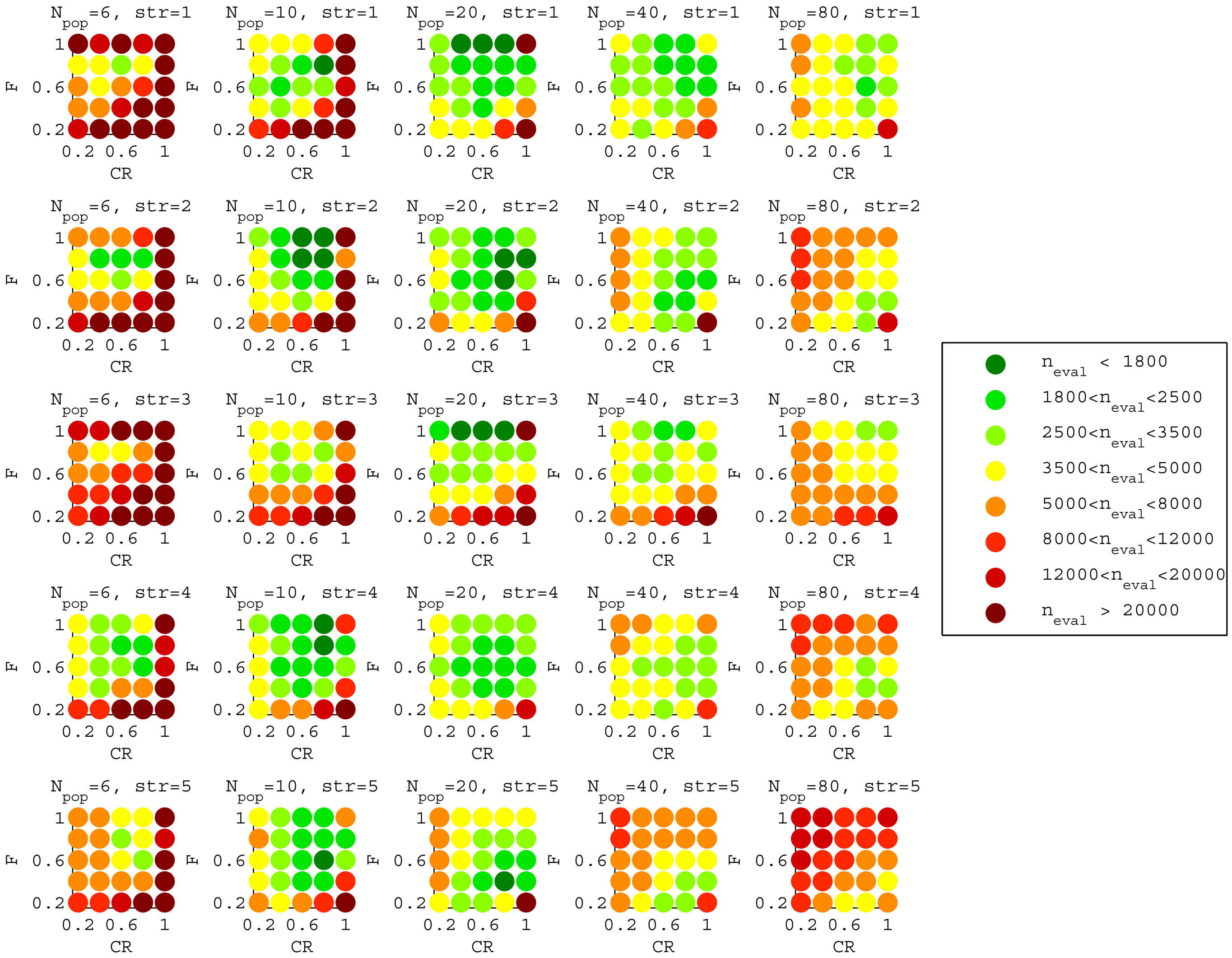


Figure A.2: The results of the first phase tuning process of DE on the EMJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The resuls are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

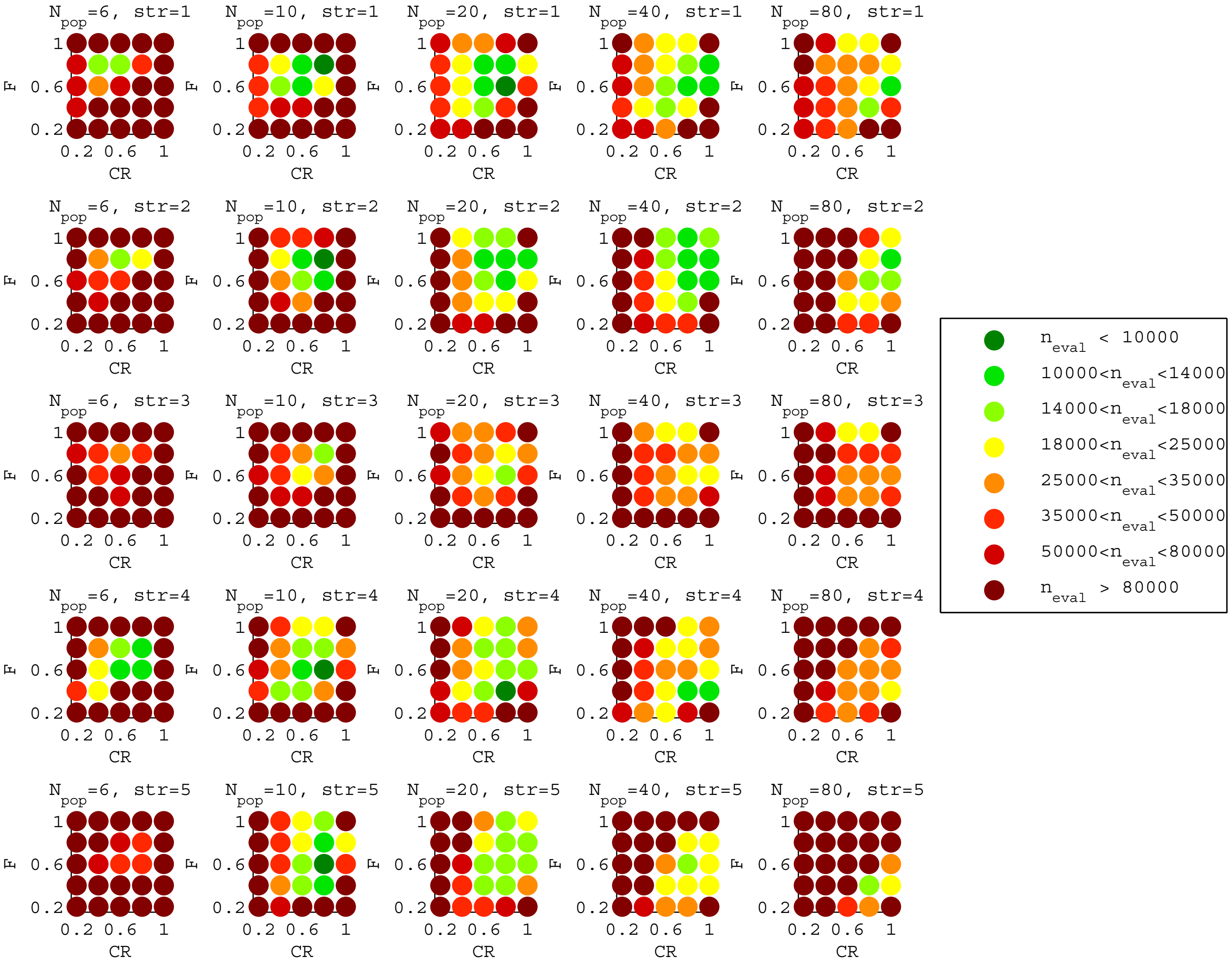


Figure A.3: The results of the first phase tuning process of DE on the EVEJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

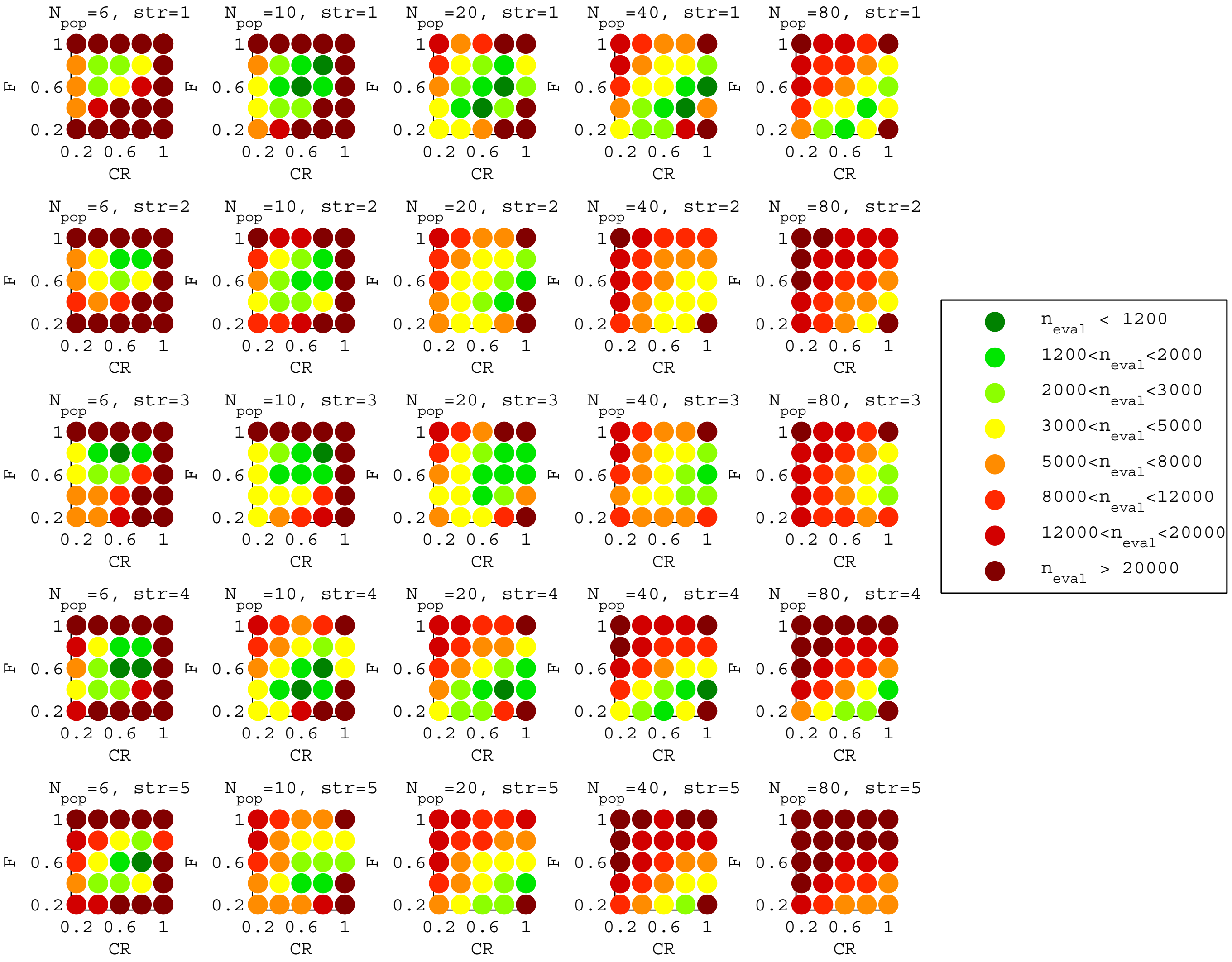


Figure A.4: The results of the first phase tuning process of DE on the Cassini2 First Leg PF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

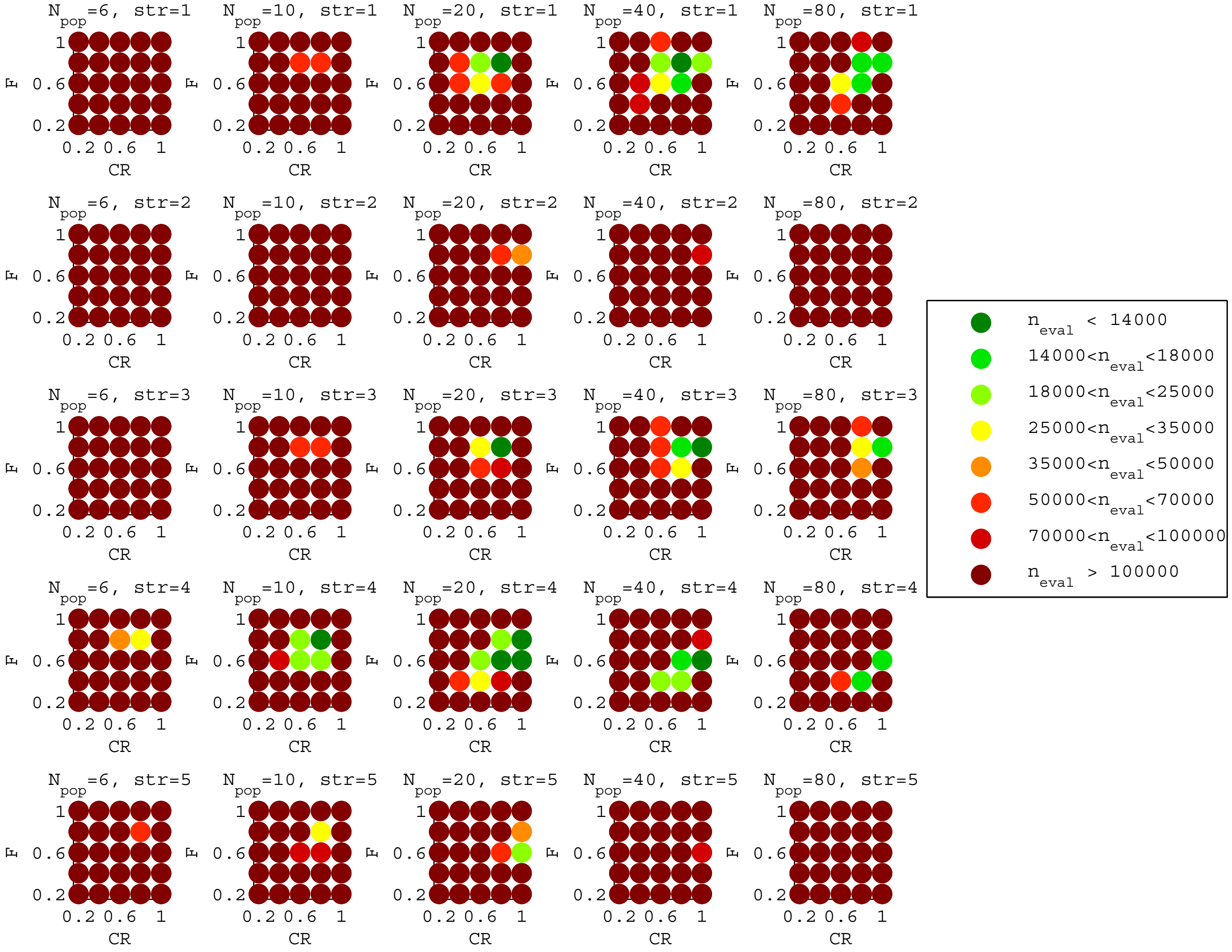


Figure A.5: The results of the first phase tuning process of DE on the Cassini2 First Leg VF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

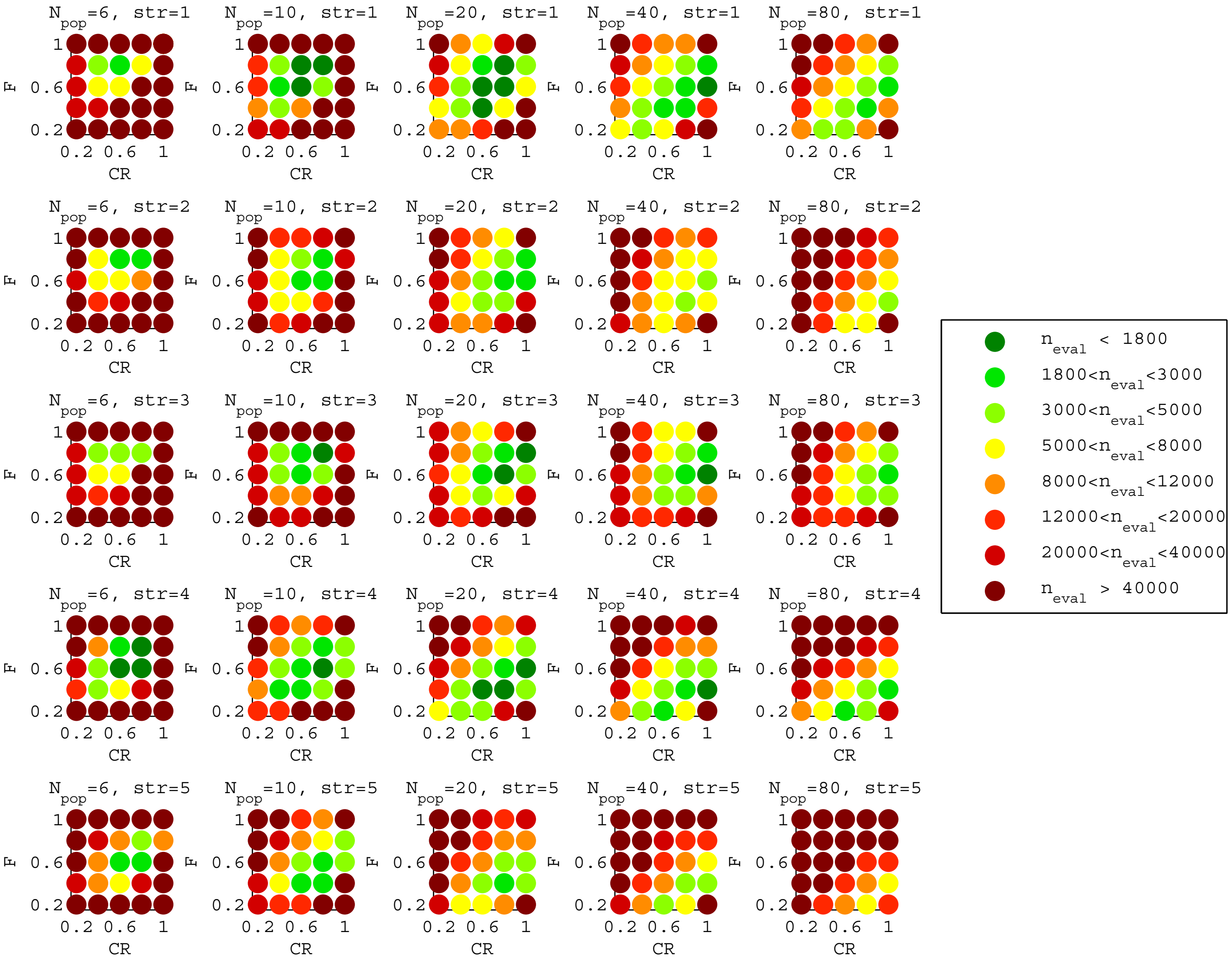


Figure A.6: The results of the first phase tuning process of DE on the Cassini2 Last leg PF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

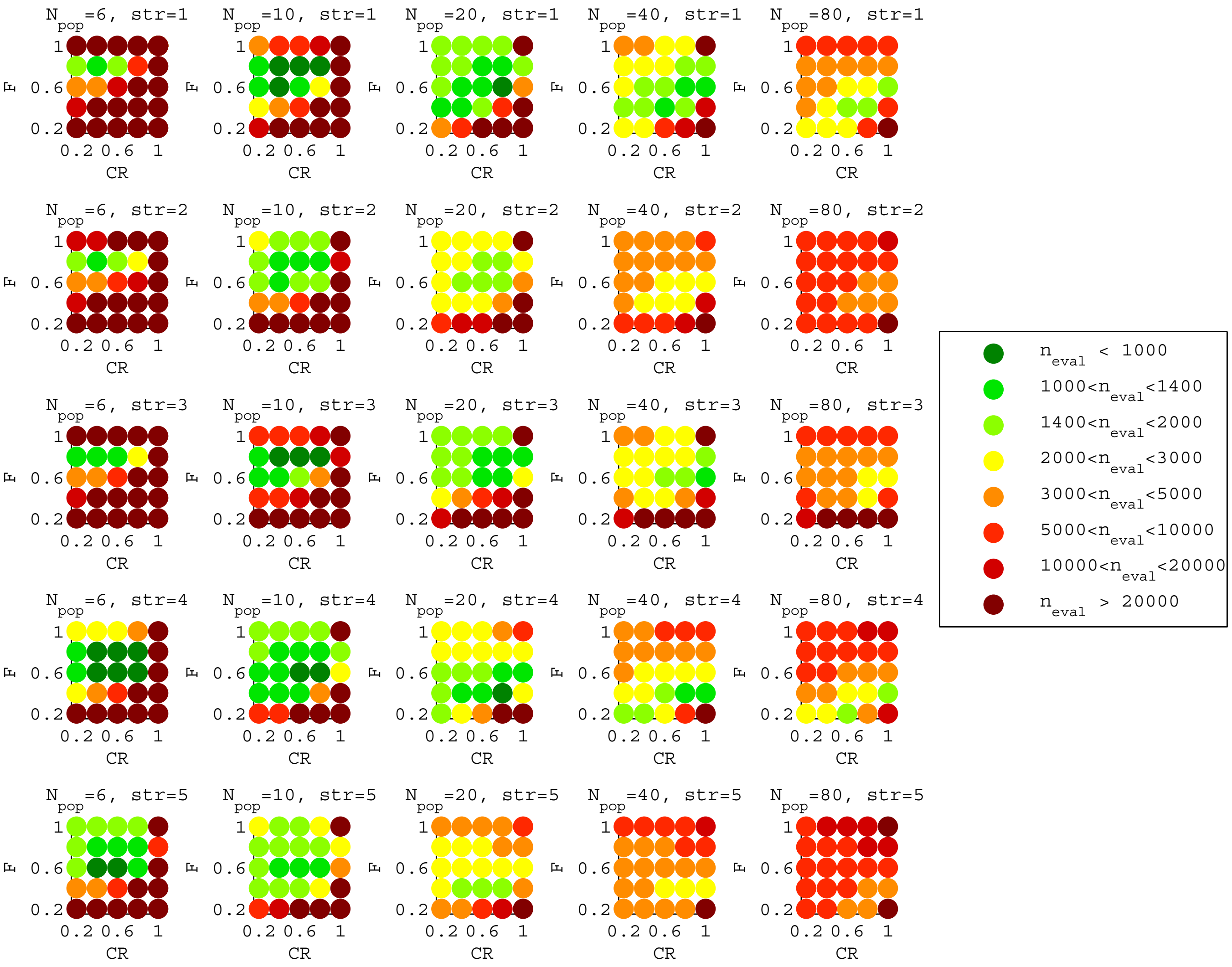


Figure A.7: The results of the first phase tuning process of DE on the Cassini2 Last leg VF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. Note that n_{eval} in the legend is identical to $n_{95\%}$.

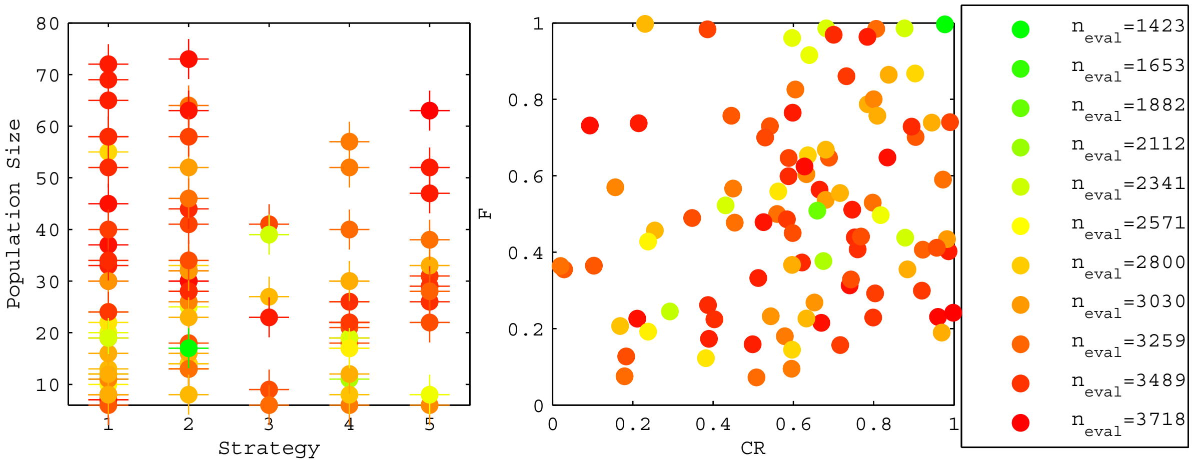


Figure A.8: The results of the first phase tuning process of DE on the EJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

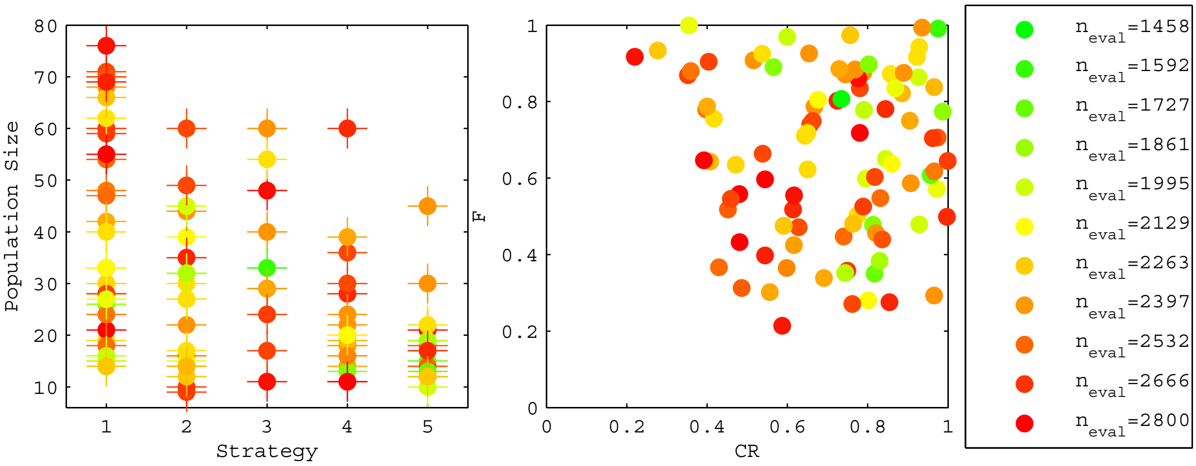


Figure A.9: The results of the first phase tuning process of DE on the EMJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

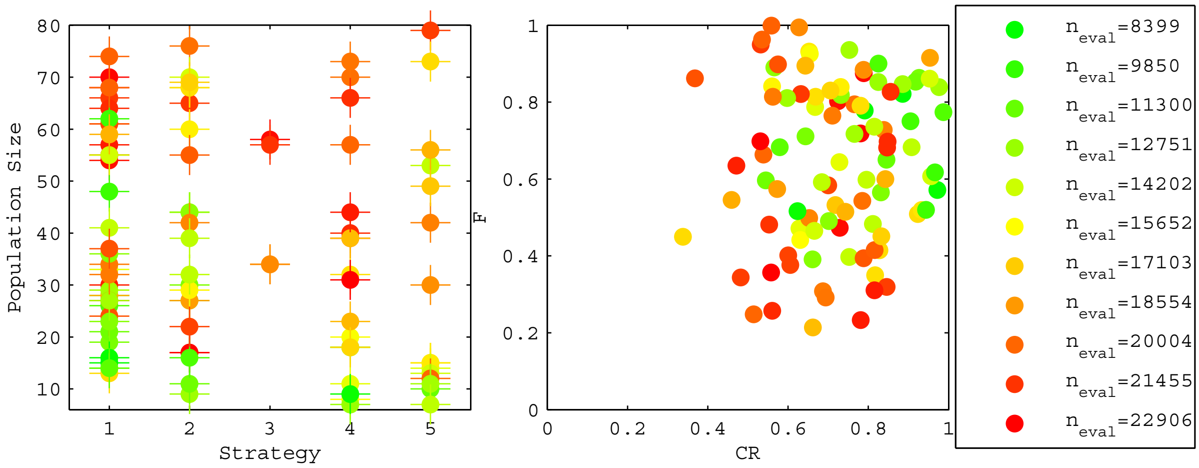


Figure A.10: The results of the first phase tuning process of DE on the EVEJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

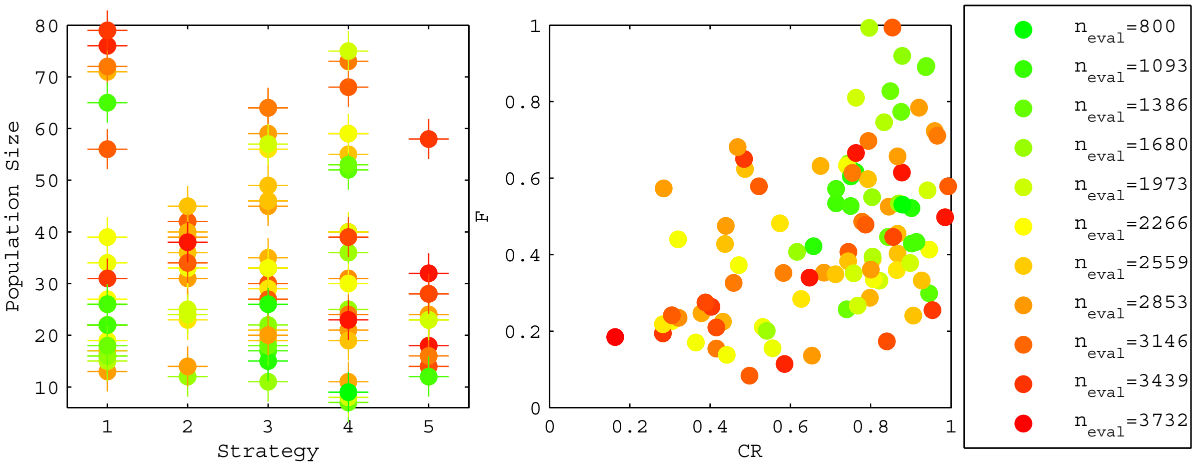


Figure A.11: The results of the first phase tuning process of DE on the Cassini2 First Leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

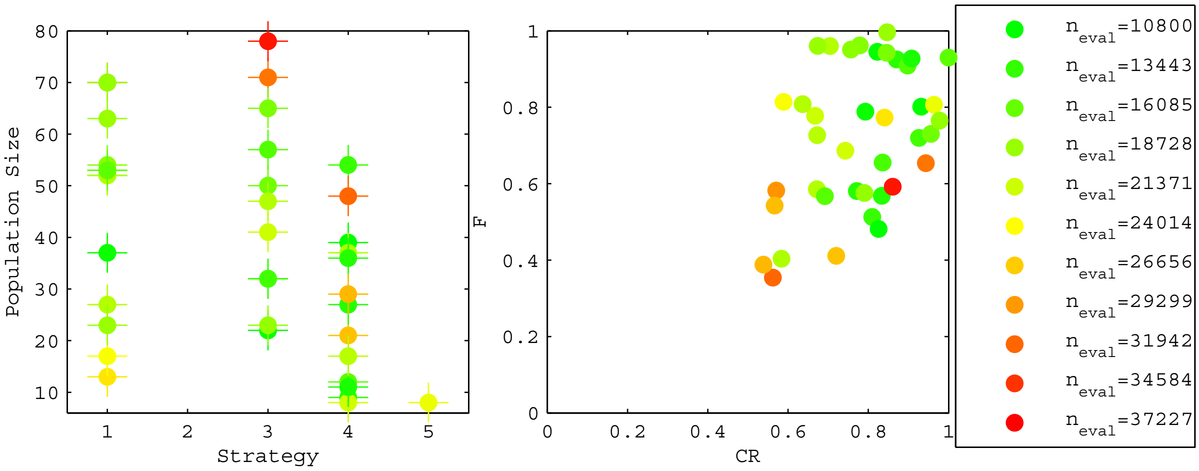


Figure A.12: The results of the first phase tuning process of DE on the Cassini2 First Leg VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 35 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

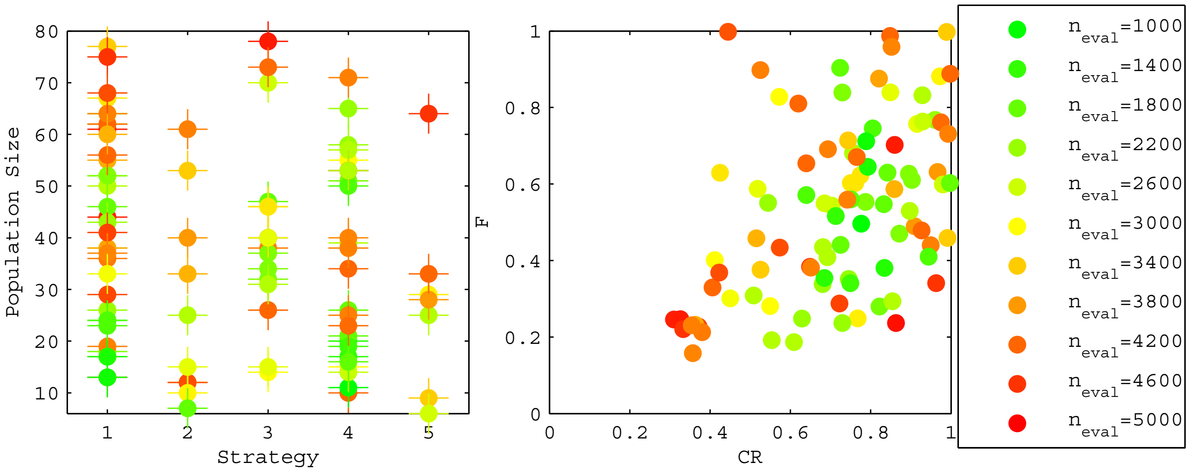


Figure A.13: The results of the first phase tuning process of DE on the Cassini2 Last leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

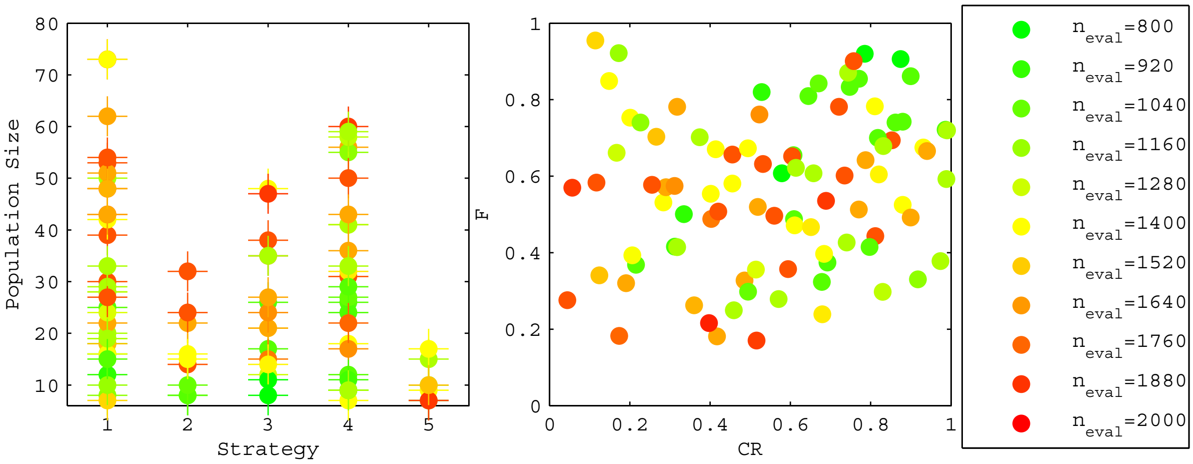


Figure A.14: The results of the first phase tuning process of DE on the Cassini2 Last leg VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

A.2 Population Size Graphs

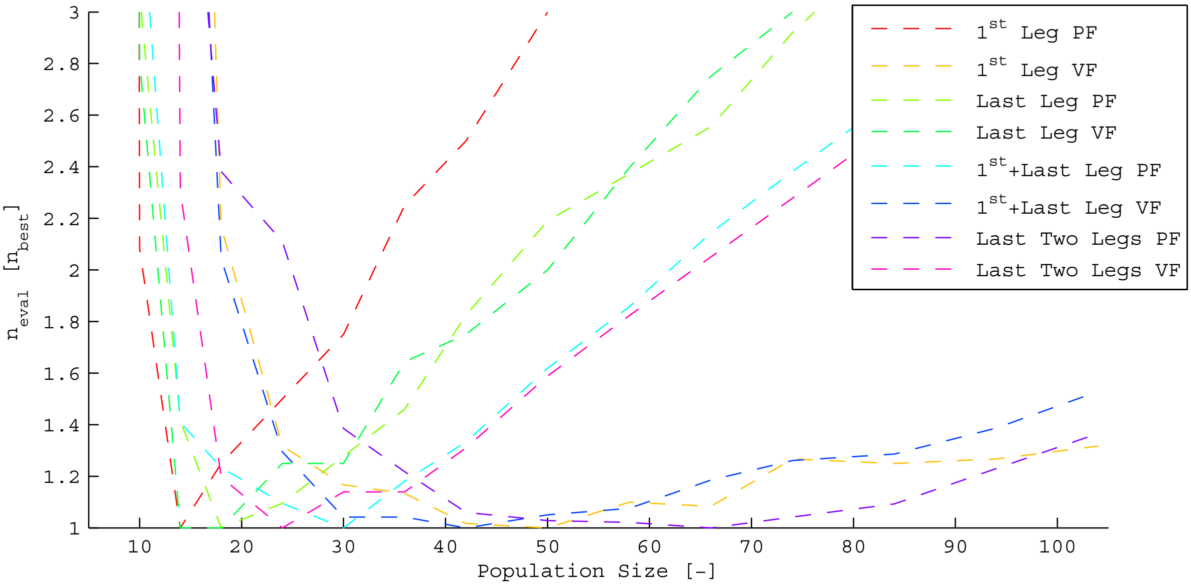


Figure A.15: The performance of different population sizes of DE1 on the problems of optimizing certain legs of the entire problem. The performance is scaled to the best performing population size on a particular problem.

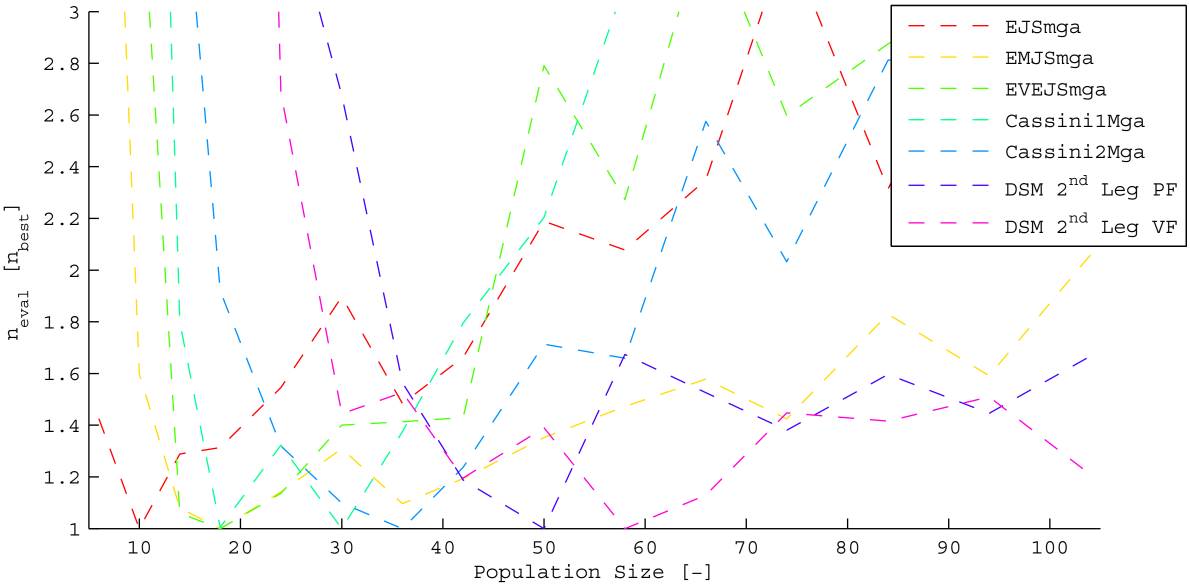


Figure A.16: The performance of different population sizes of DE1 on the trajectory model simplification problems. The performance is scaled to the best performing population size on a particular problem.

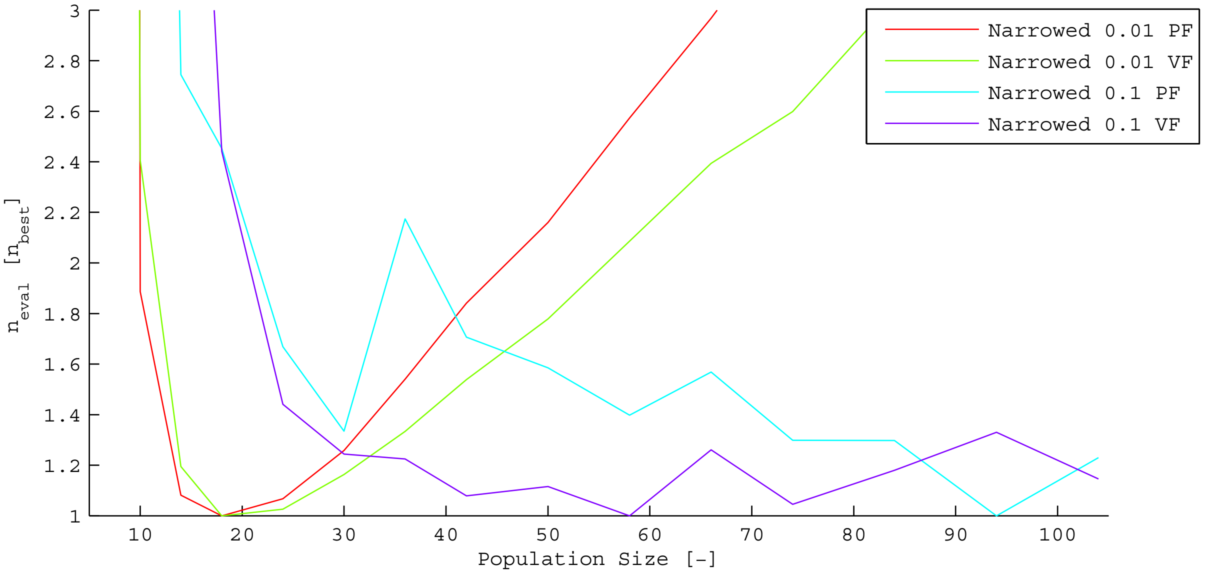


Figure A.17: The performance of different population sizes of DE1 on the problems with narrowed search spaces. The performance is scaled to the best performing population size on a particular problem.

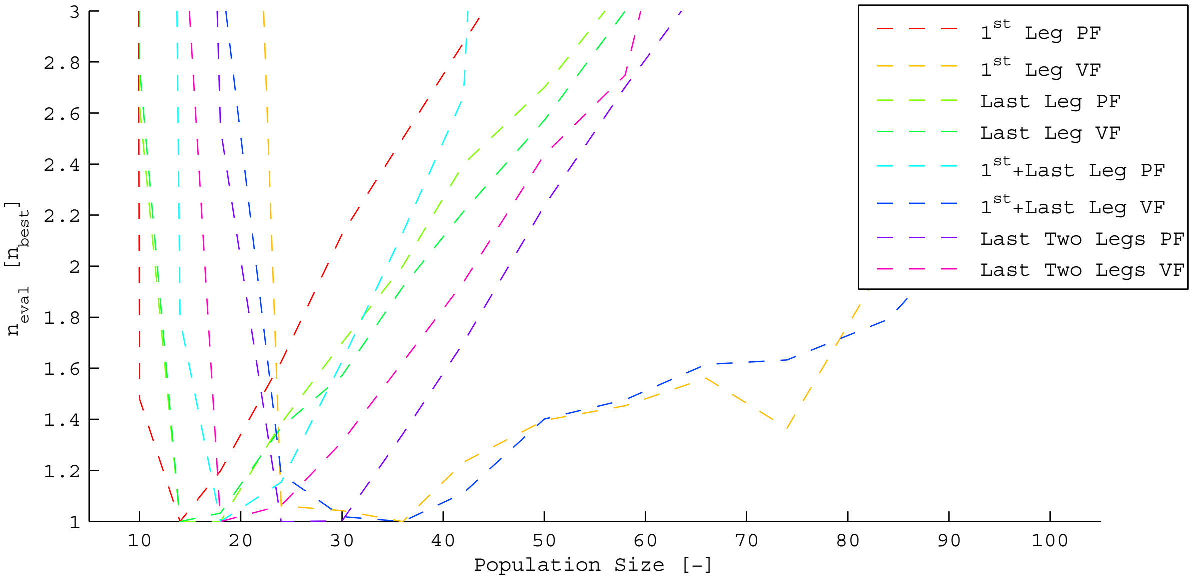


Figure A.18: The performance of different population sizes of DE2 on the problems of optimizing certain legs of the entire problem. The performance is scaled to the best performing population size on a particular problem.

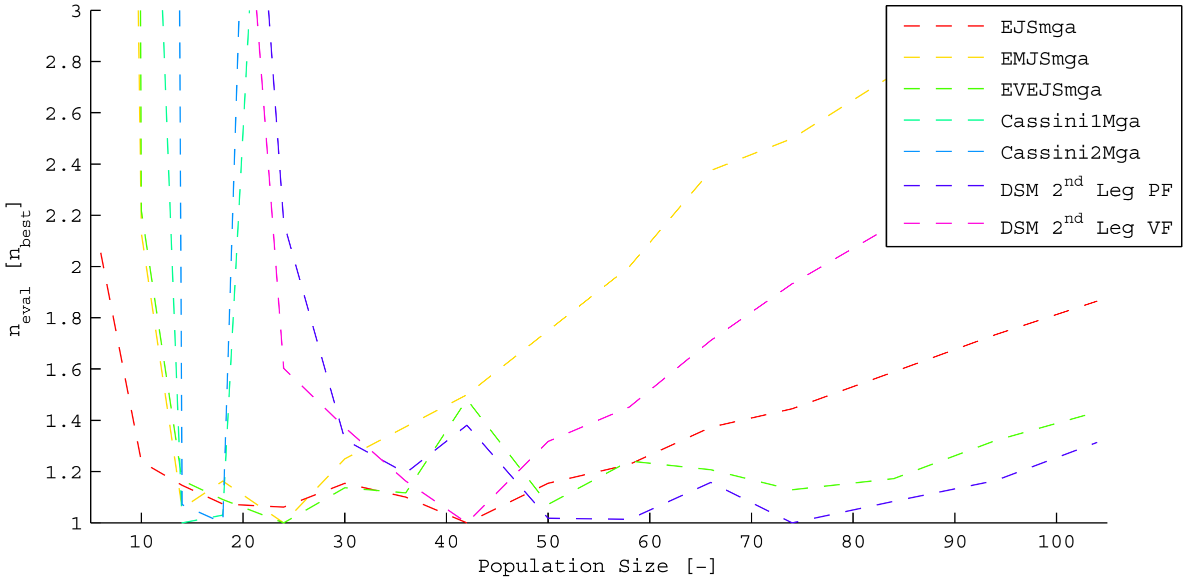


Figure A.19: The performance of different population sizes of DE2 on the trajectory model simplification problems. The performance is scaled to the best performing population size on a particular problem.

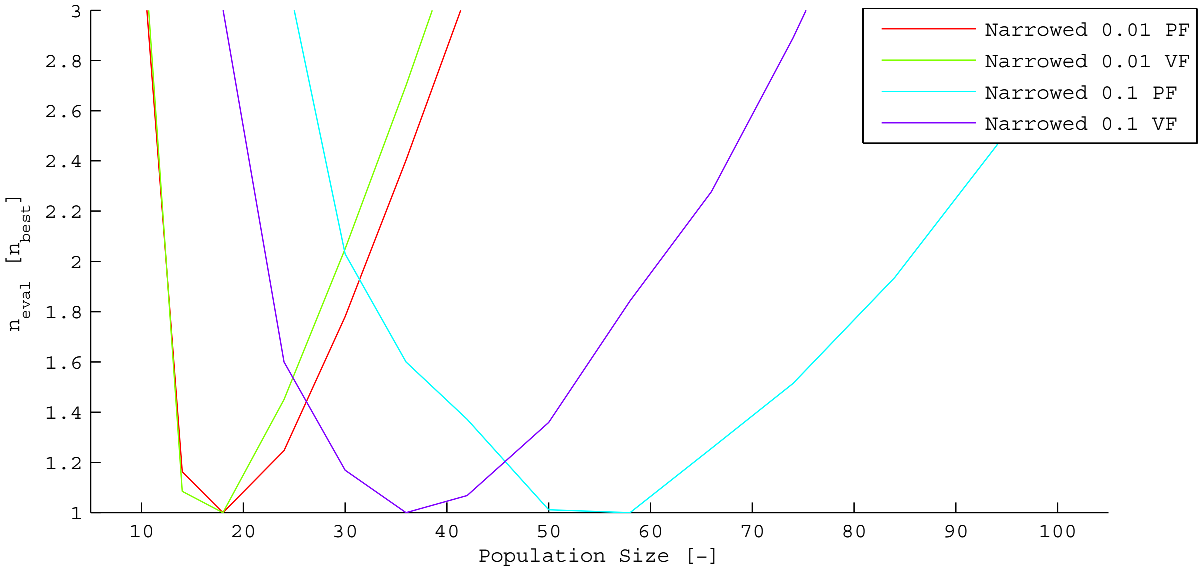


Figure A.20: The performance of different population sizes of DE2 on the problems with narrowed search spaces. The performance is scaled to the best performing population size on a particular problem.

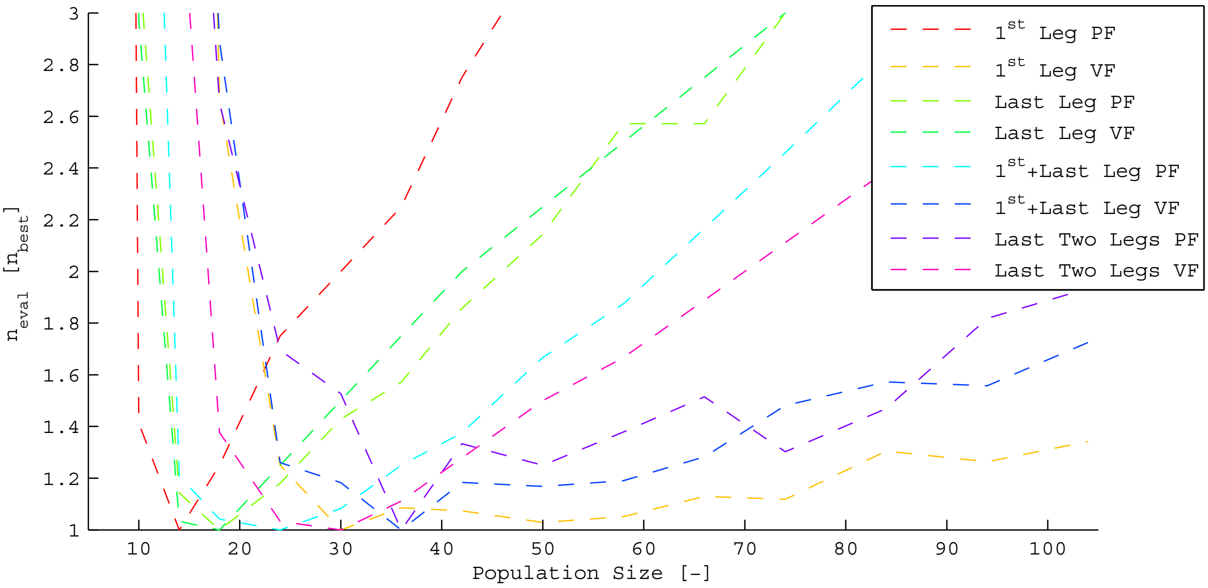


Figure A.21: The performance of different population sizes of DE3 on the problems of optimizing certain legs of the entire problem. The performance is scaled to the best performing population size on a particular problem.

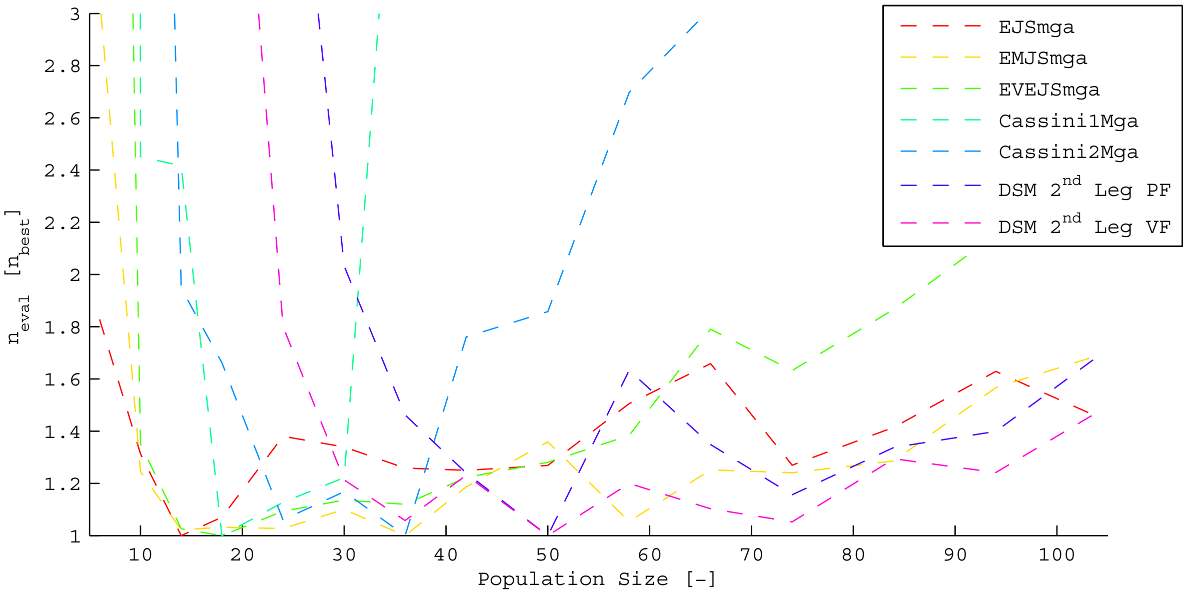


Figure A.22: The performance of different population sizes of DE3 on the trajectory model simplification problems. The performance is scaled to the best performing population size on a particular problem.

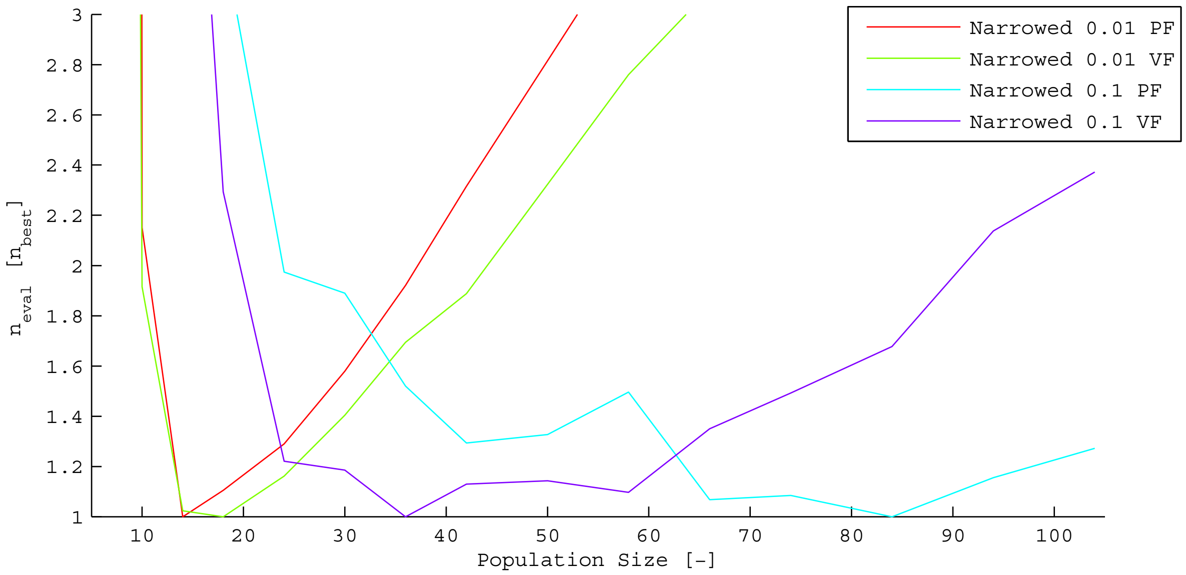


Figure A.23: The performance of different population sizes of DE3 on the problems with narrowed search spaces. The performance is scaled to the best performing population size on a particular problem.

Appendix B

Additional Tuning Results of GA

This appendix shows all the relevant ‘raw’ results for the tuning process of GA.

An overview of the figures where the results can be found is given in Table B.1 below.

Table B.1: Overview of the results that were obtained for GA and where they can be found.

Problem	Phase	Sampling	Figure	Page
EJS MGA	1	Grid	B.1	218
EMJS MGA	1	Grid	B.2	219
EVEJS MGA	1	Grid	B.3	220
Cassini2 First Leg PF	1	Grid	B.4	221
Cassini2 First Leg VF	1	Grid	B.5	222
Cassini2 Last Leg PF	1	Grid	B.6	223
Cassini2 Last Leg VF	1	Grid	B.7	224
EJS MGA	1	Random	B.8	225
EMJS MGA	1	Random	B.9	225
Cassini2 First Leg PF	1	Random	B.10	226
Cassini2 Last Leg PF	1	Random	B.11	226
Cassini2 Last Leg VF	1	Random	B.12	227

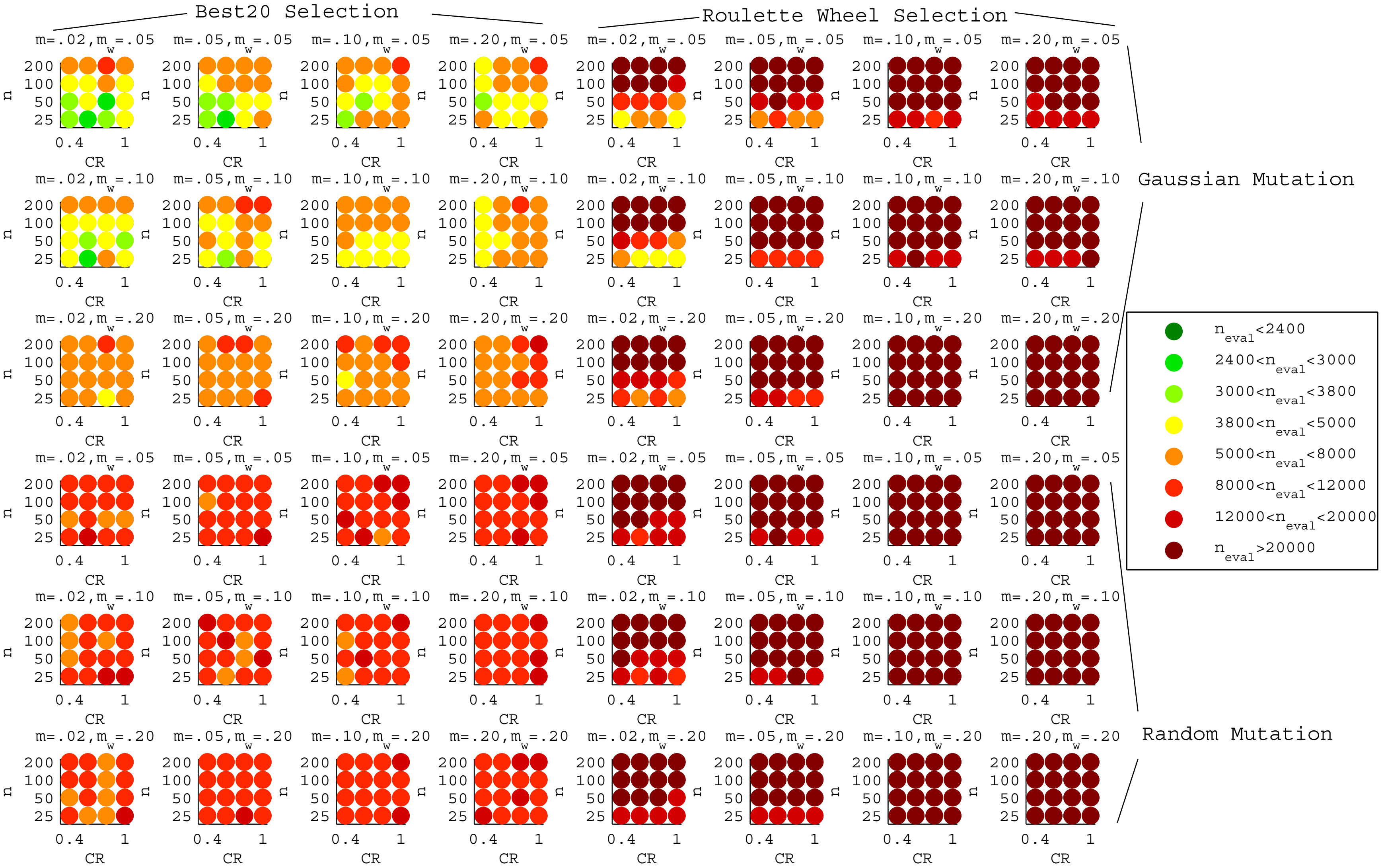


Figure B.1: The results of the first phase tuning process of GA on the EJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. Note that n_{eval} in the legend is identical to $n_{95\%}$.

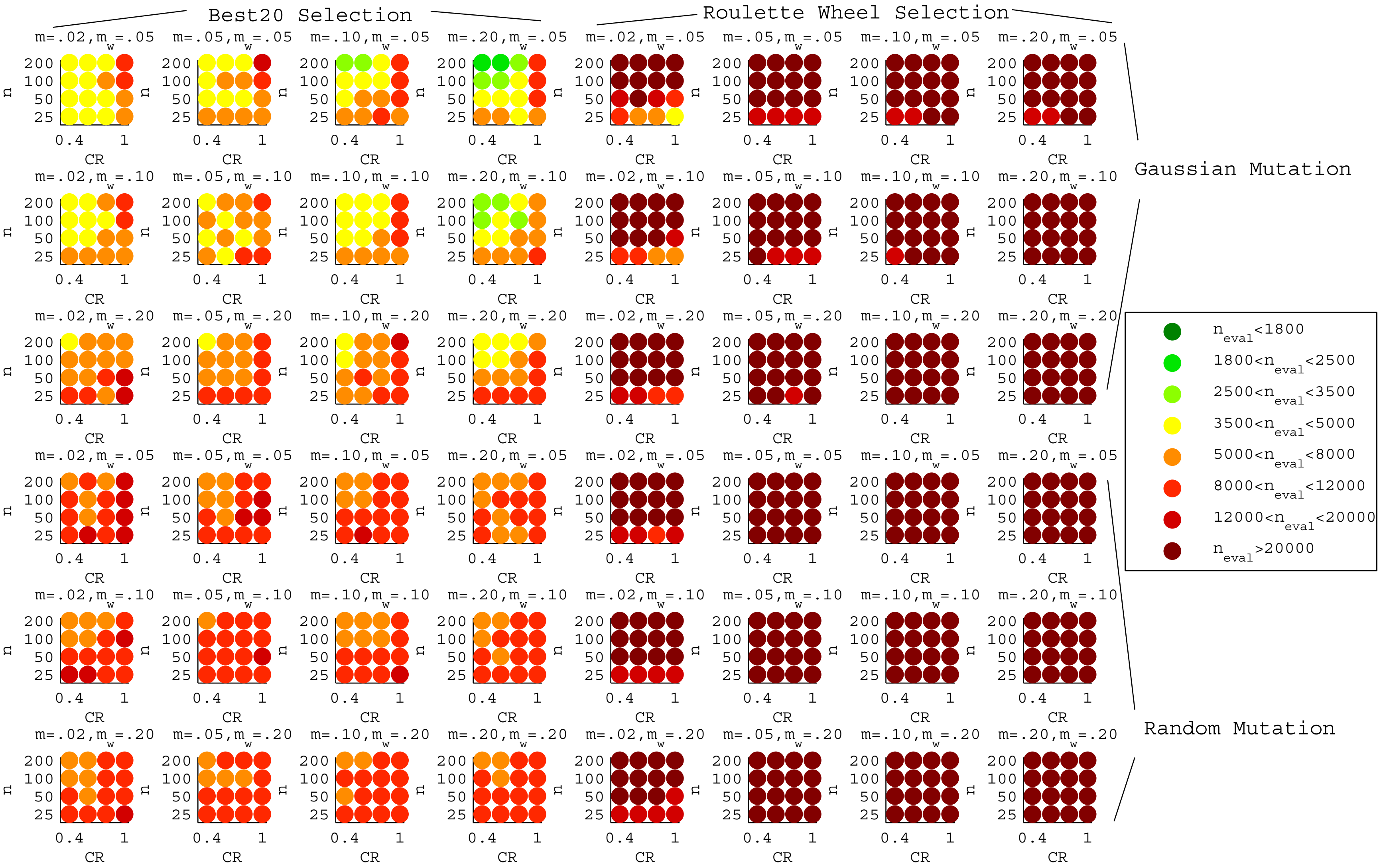


Figure B.2: The results of the first phase tuning process of GA on the EMJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. Note that n_{eval} in the legend is identical to $n_{95\%}$.

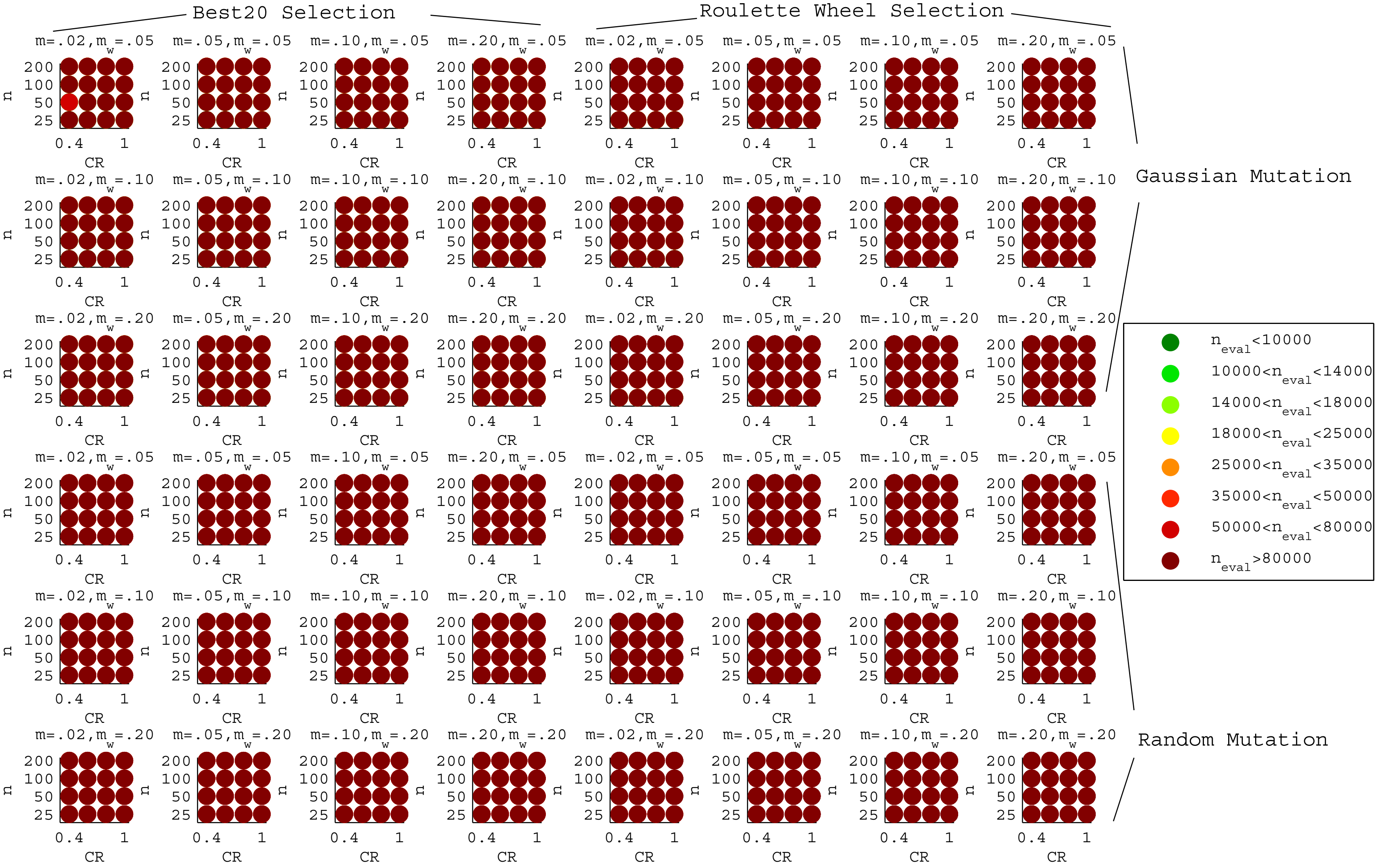


Figure B.3: The results of the first phase tuning process of GA on the EVEJS MGA problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

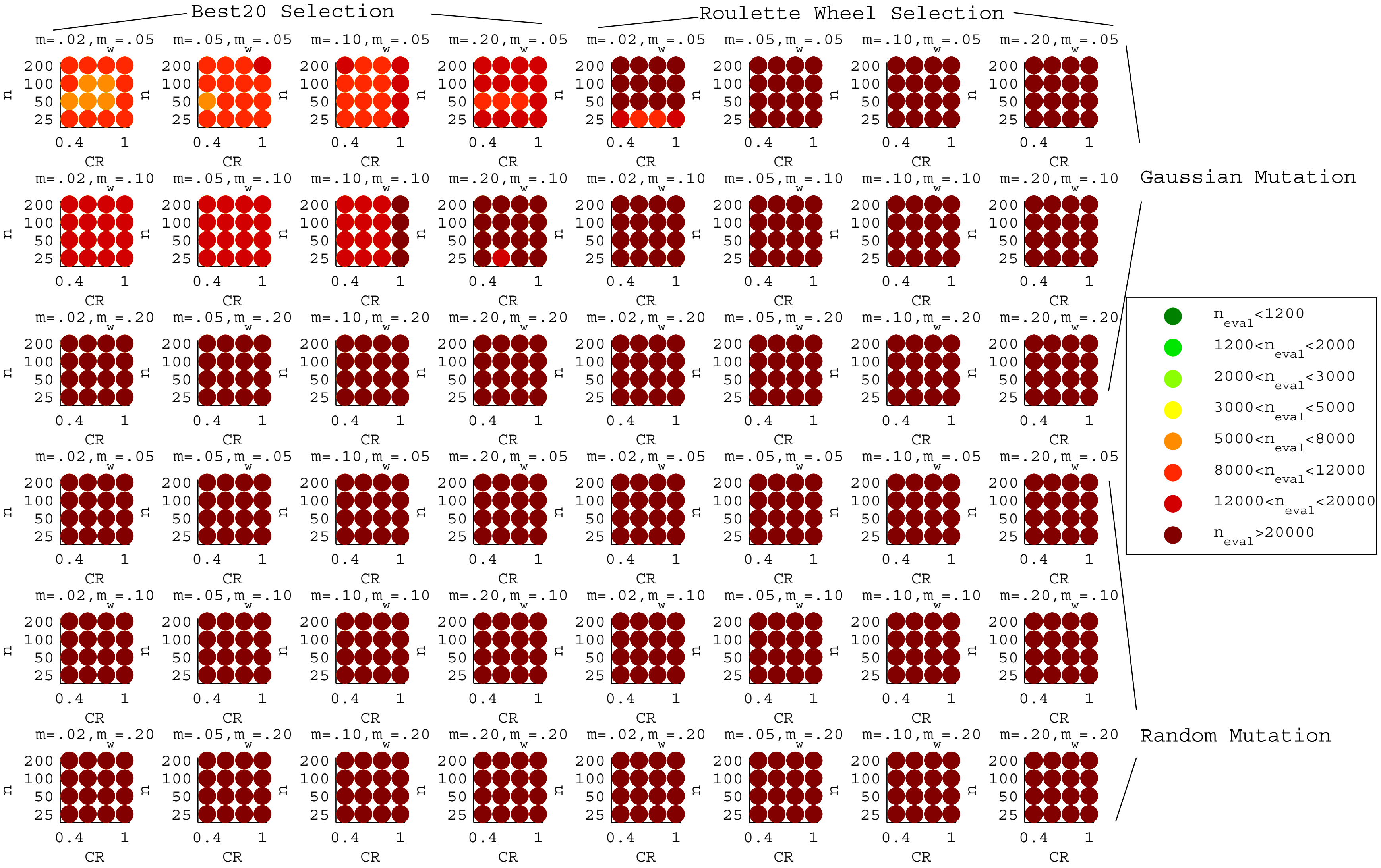


Figure B.4: The results of the first phase tuning process of GA on the Cassini2 First Leg PF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. Note that n_{eval} in the legend is identical to $n_{95\%}$.

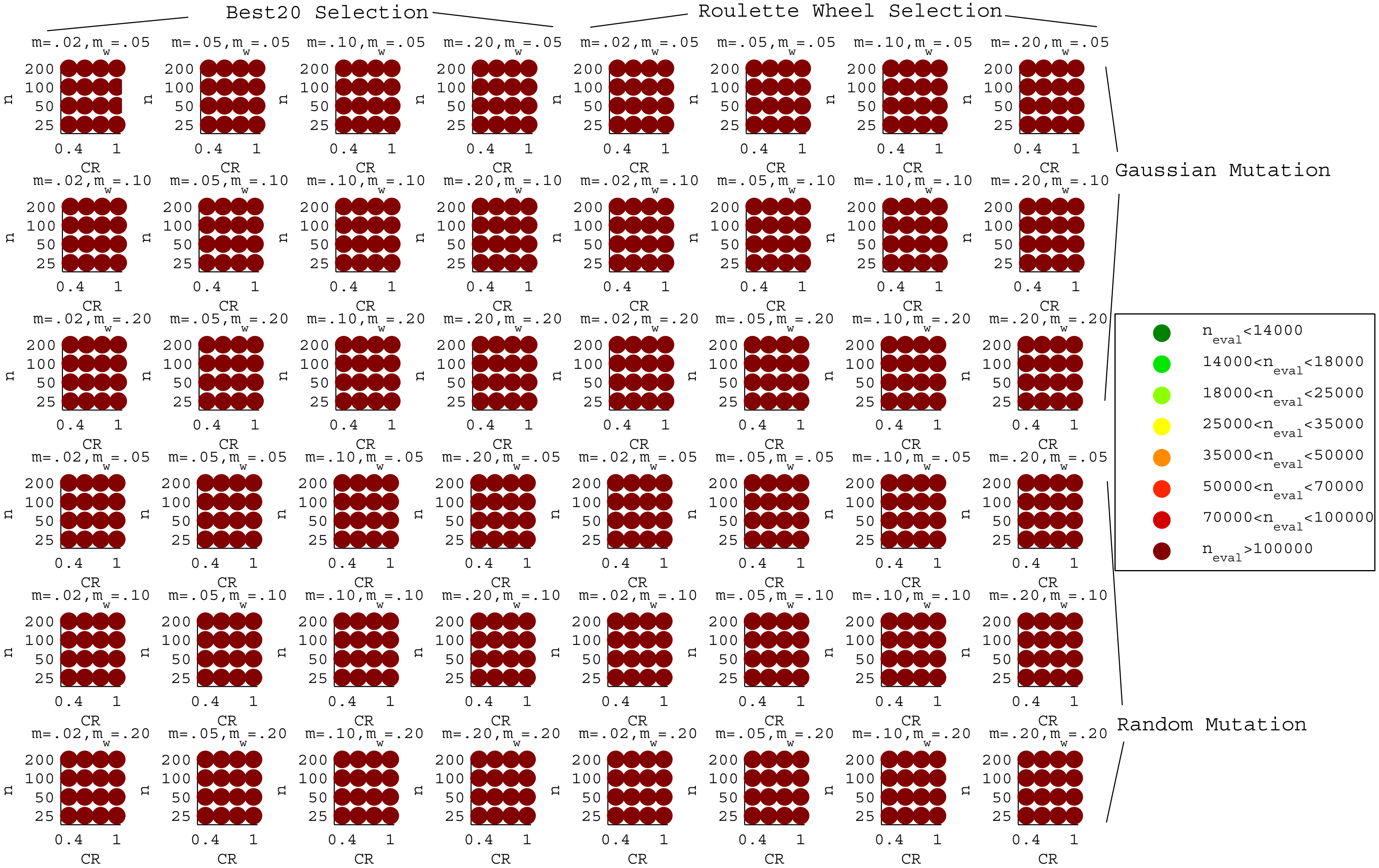


Figure B.5: The results of the first phase tuning process of GA on the Cassini2 First Leg VF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. It is emphasized here that the categorization above 20000 evaluations is not necessarily correct: only 20000 evaluations were performed in each run, and algorithms that reliably converge after 20000 evaluations are underestimated in this plot. Note that n_{eval} in the legend is identical to $n_{95\%}$.

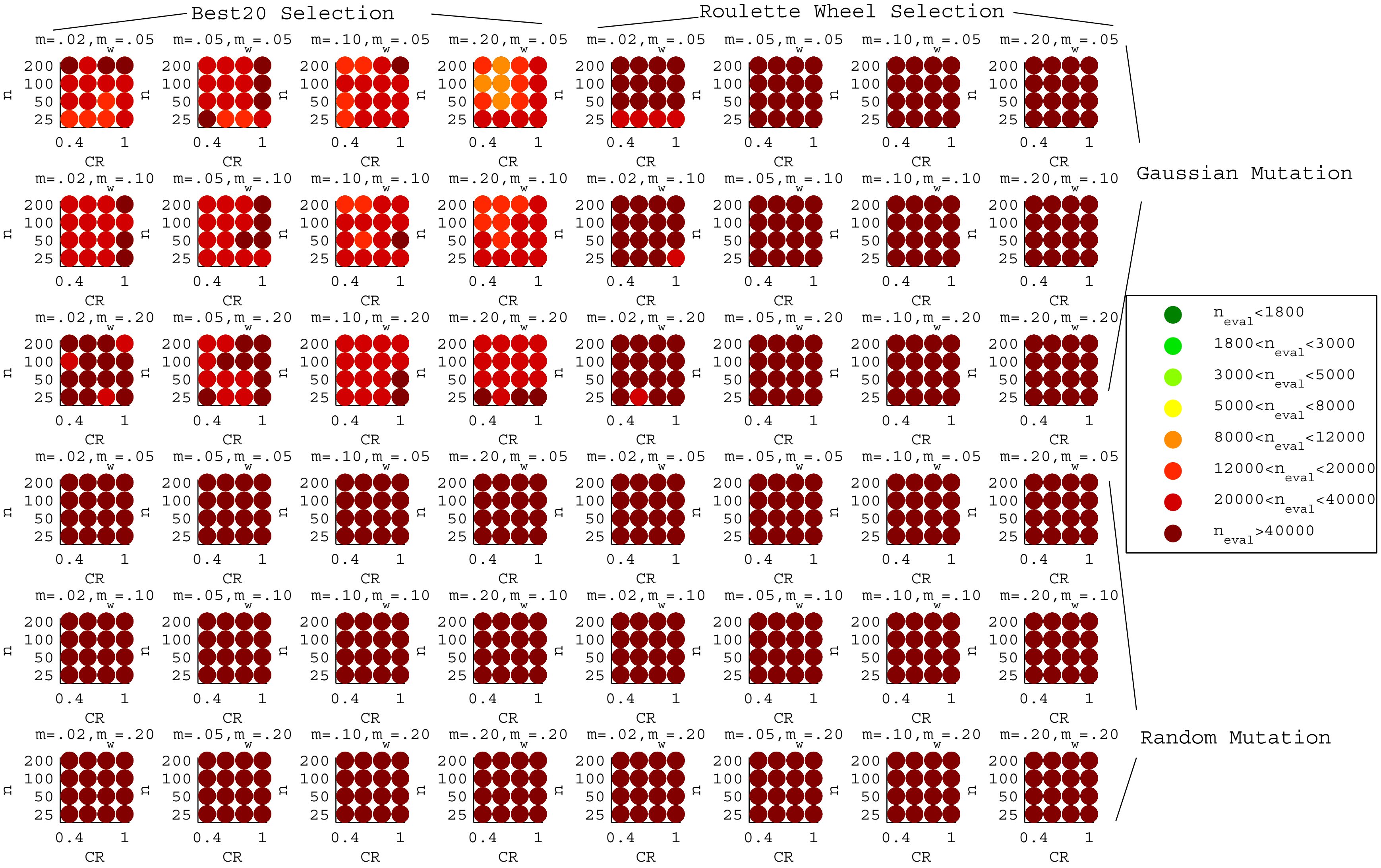


Figure B.6: The results of the first phase tuning process of GA on the Cassini2 Last leg PF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. Note that n_{eval} in the legend is identical to $n_{95\%}$.

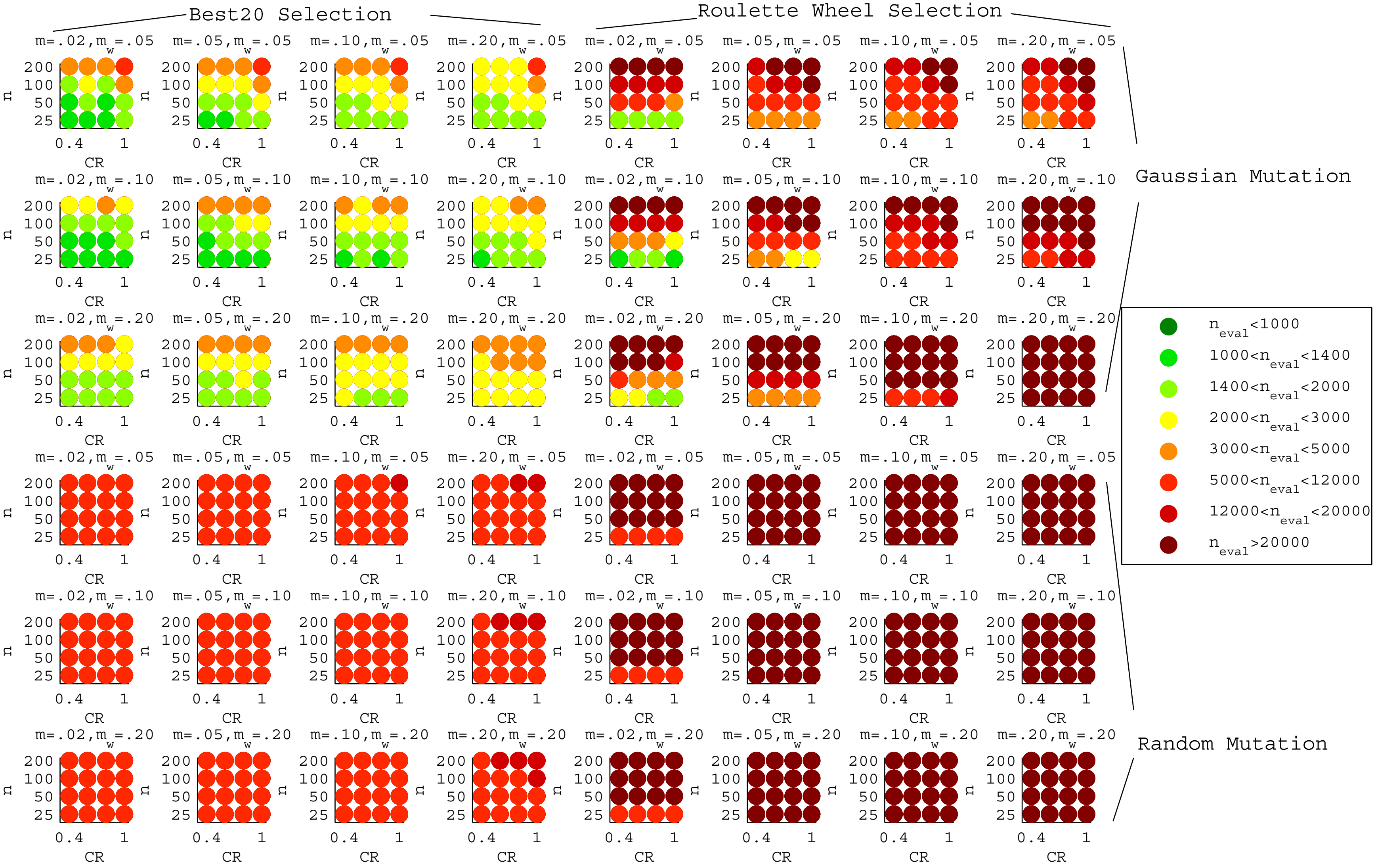


Figure B.7: The results of the first phase tuning process of GA on the Cassini2 Last leg VF problem. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The results are categorized based on the color scheme visible on the right. The categorization is similar to that of the DE tuning, to allow for comparison. Note that n_{eval} in the legend is identical to $n_{95\%}$.

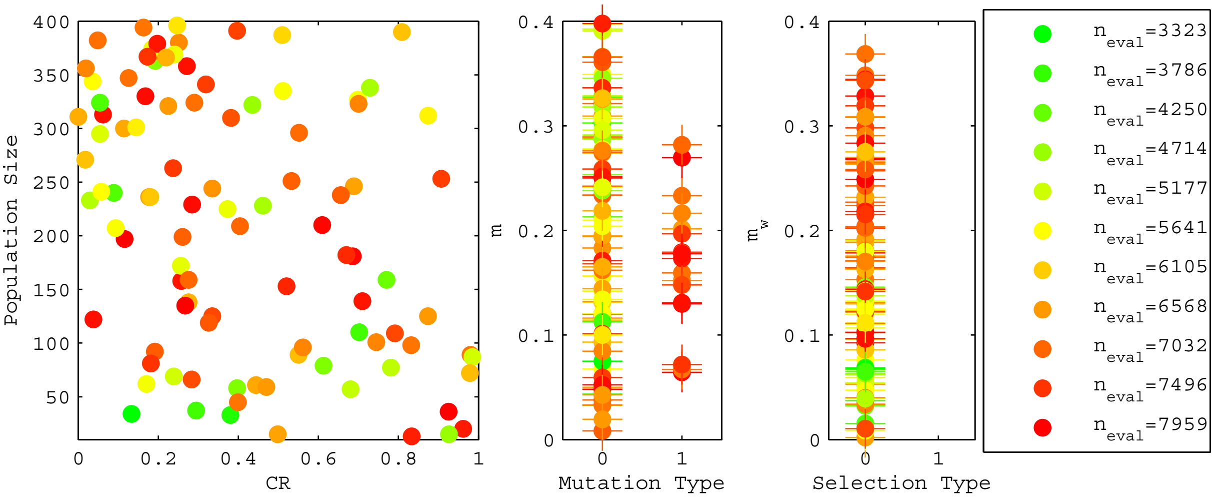


Figure B.8: The results of the first phase tuning process of GA on the EJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

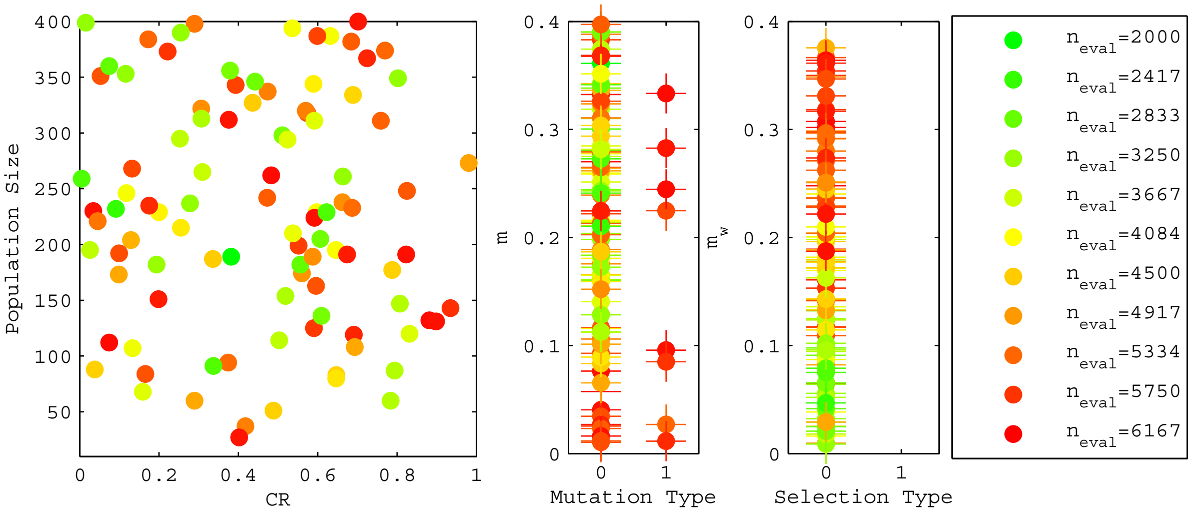


Figure B.9: The results of the first phase tuning process of GA on the EMJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

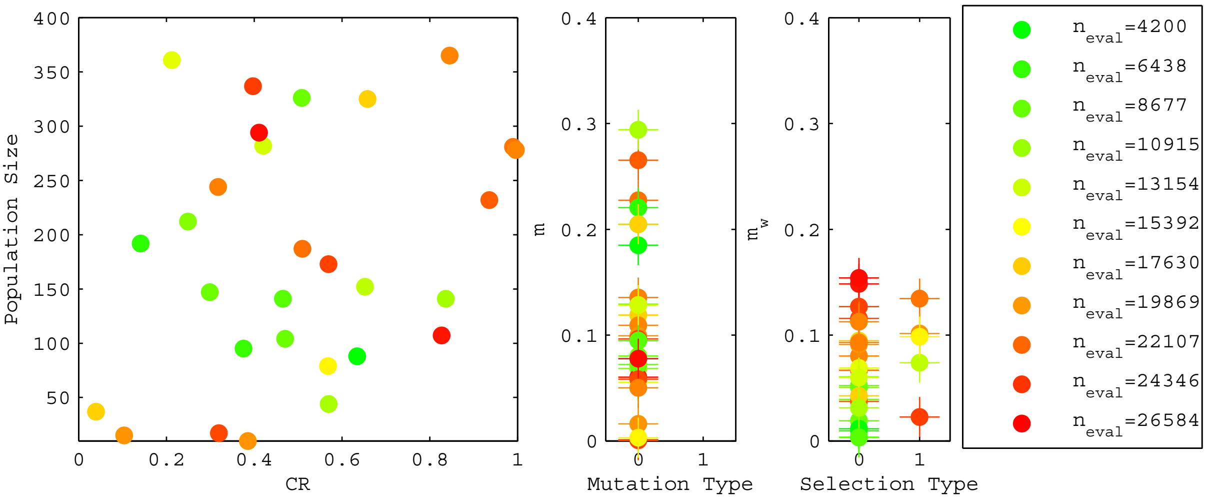


Figure B.10: The results of the first phase tuning process of GA on the Cassini2 First Leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 30 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

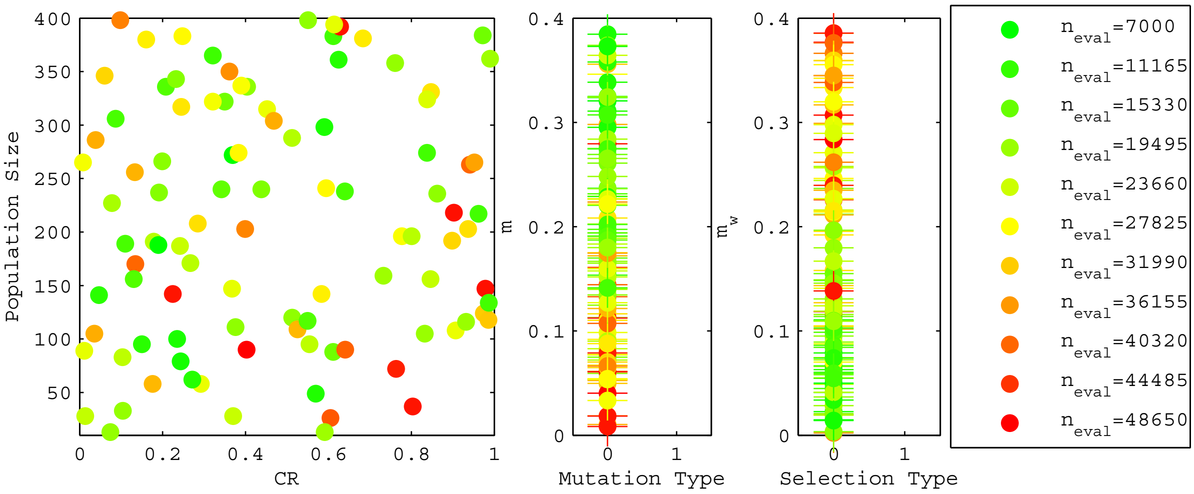


Figure B.11: The results of the first phase tuning process of GA on the Cassini2 Last leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

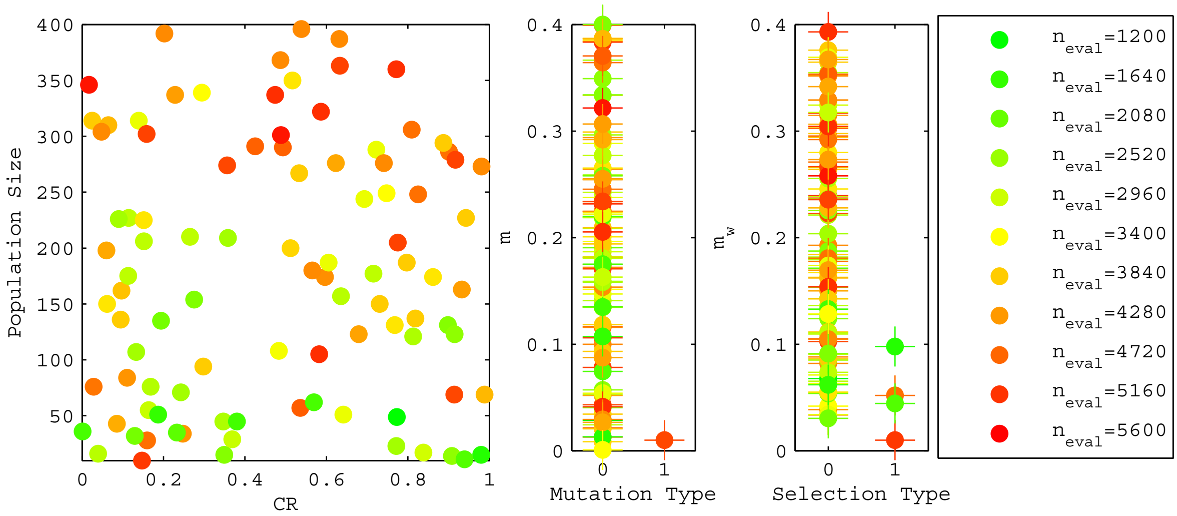


Figure B.12: The results of the first phase tuning process of GA on the Cassini2 Last leg VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that n_{eval} in the legend is identical to $n_{95\%}$.

Appendix C

Additional Tuning Results of PSO

This section shows all the relevant 'raw' results for the tuning process of PSO.

An overview of the figures where the results can be found is given in Table C.1 below.

Table C.1: Overview of the results that were obtained for PSO and where they can be found.

Problem	Phase	Sampling	Figure	Page
EJS MGA	1	Random	C.1	230
EMJS MGA	1	Random	C.2	231
Cassini2 First Leg PF	1	Random	C.3	232
Cassini2 Last Leg PF	1	Random	C.4	233
Cassini2 Last Leg VF	1	Random	C.5	234
EJS MGA	2	Random	C.6	235
EMJS MGA	2	Random	C.7	236
Cassini2 First Leg PF	2	Random	C.8	237
Cassini2 Last Leg PF	2	Random	C.9	238
Cassini2 Last Leg VF	2	Random	C.10	239
EVEJS MGA	3	Random	C.11	240
Cassini2 First Leg VF	3	Random	C.12	241
Cassini2 First And Last Leg PF	3	Random	C.13	242
Cassini2 Last Two Legs VF	3	Random	C.14	243
Cassini2 VF Narrowed 0.01	3	Random	C.15	244

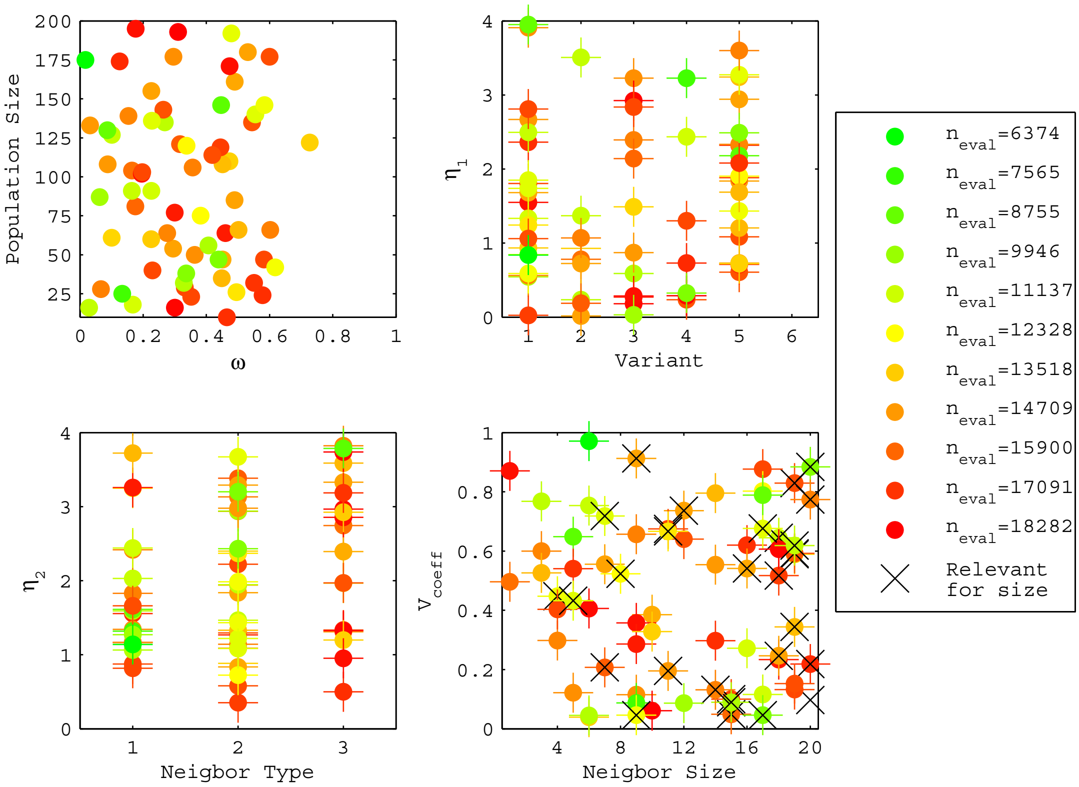


Figure C.1: The results of the first phase tuning process of PSO on the EJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

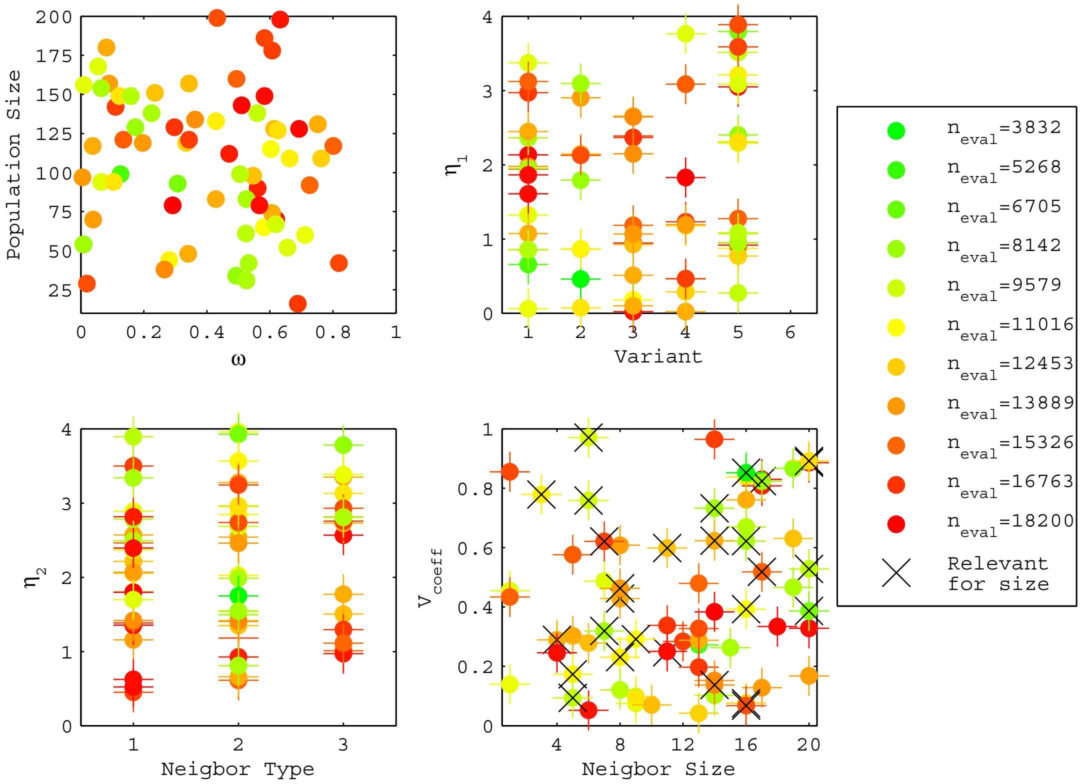


Figure C.2: The results of the first phase tuning process of PSO on the EMJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

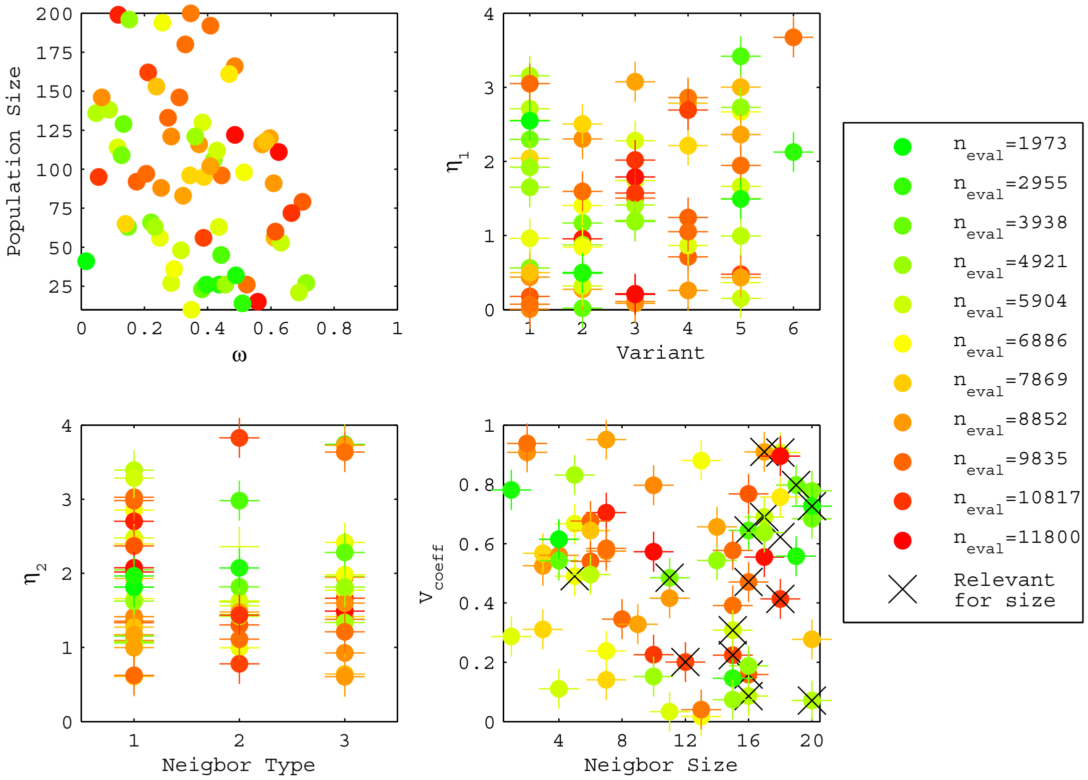


Figure C.3: The results of the first phase tuning process of PSO on the Cassini2 First Leg DSM PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

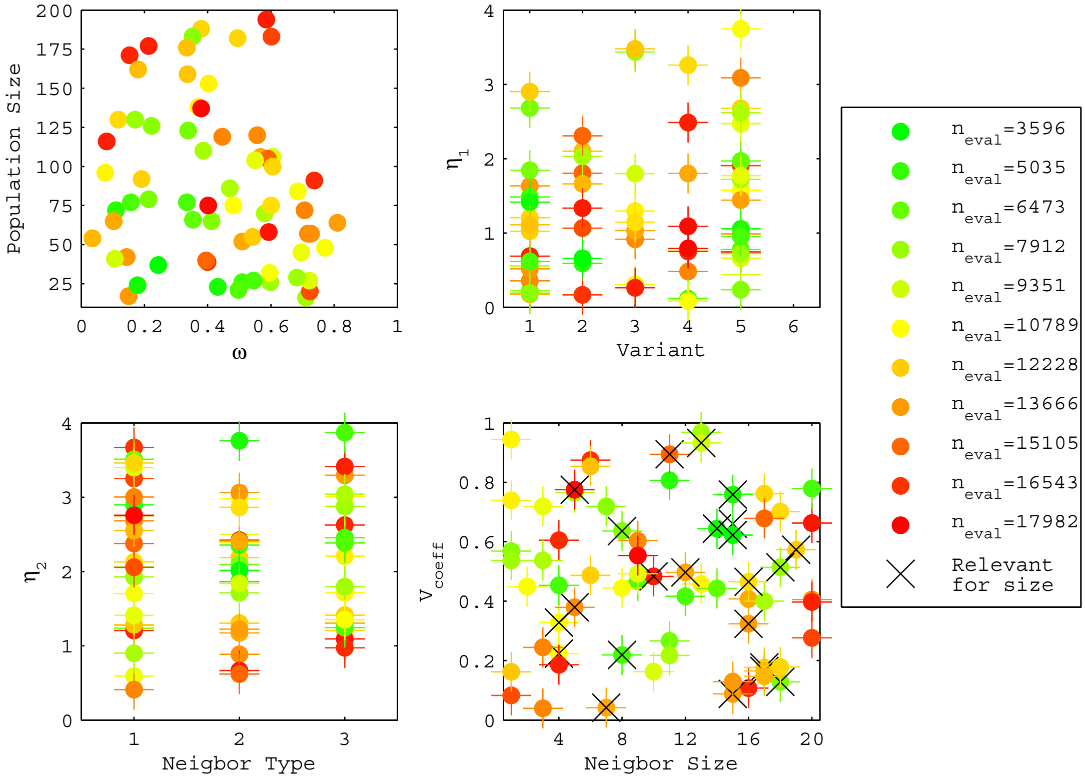


Figure C.4: The results of the first phase tuning process of PSO on the Cassini2 Last Leg DSM PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

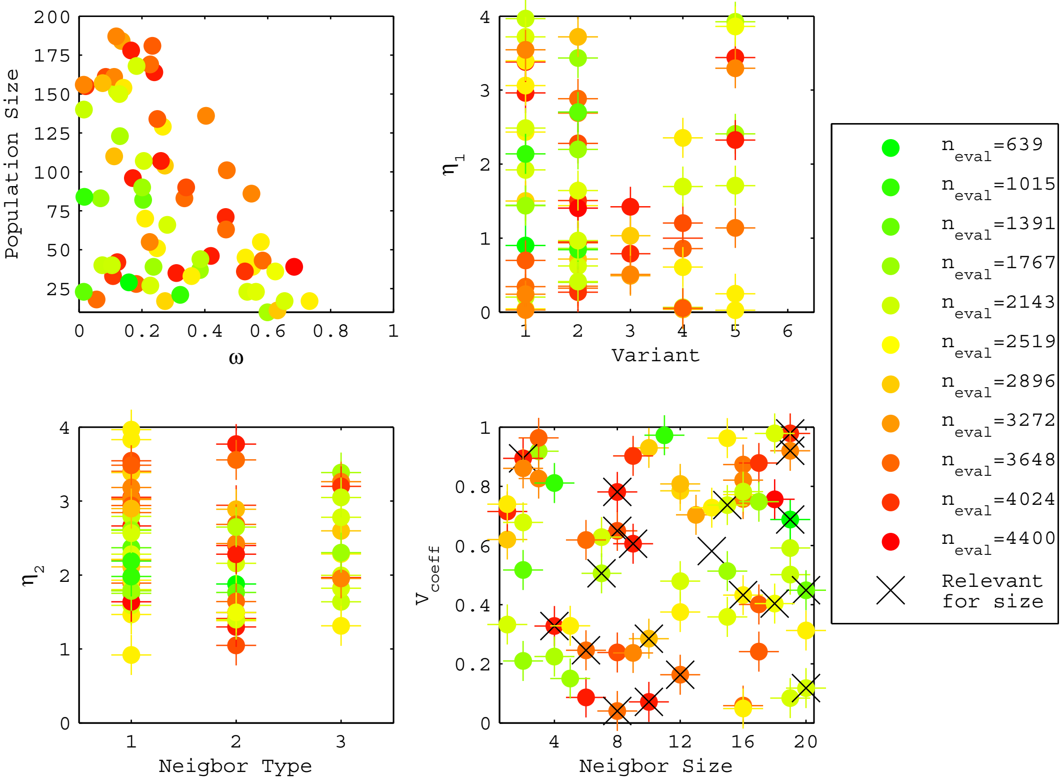


Figure C.5: The results of the first phase tuning process of PSO on the Cassini2 Last Leg DSM VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 70 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

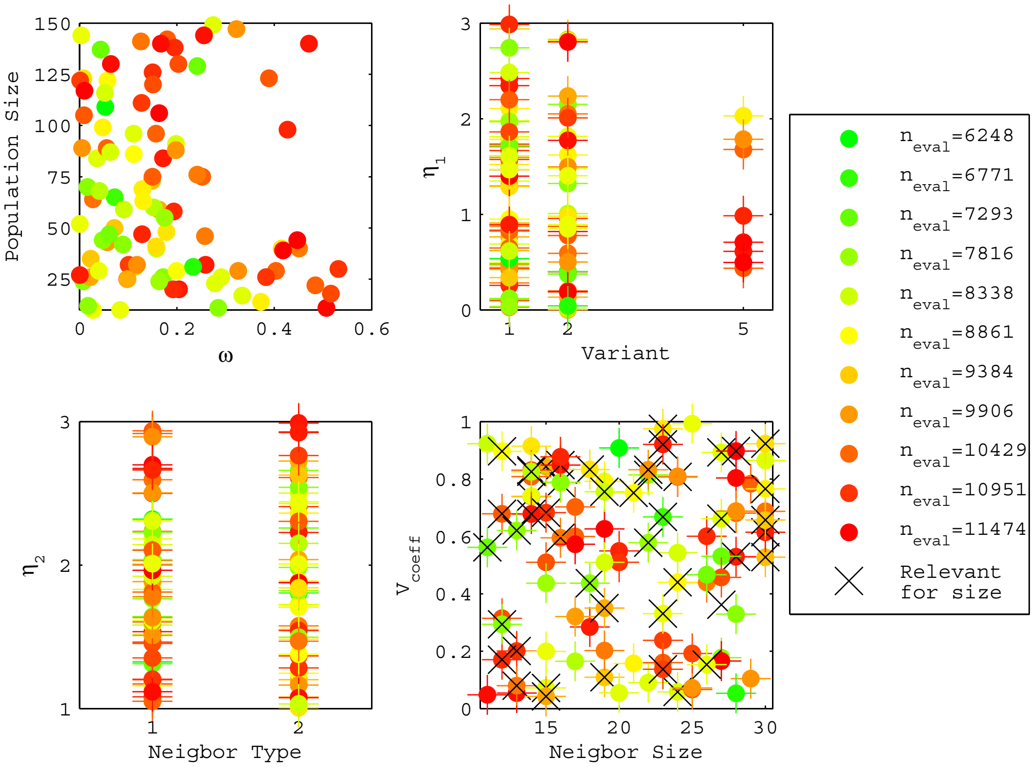


Figure C.6: The results of the second phase tuning process of PSO on the EJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

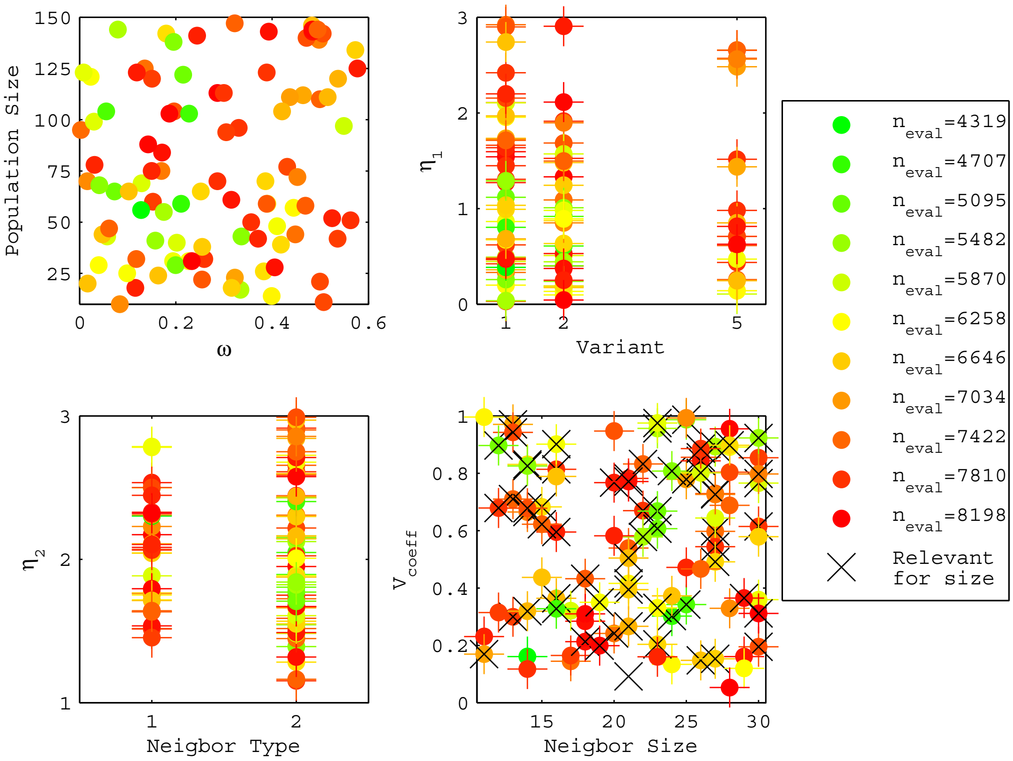


Figure C.7: The results of the second phase tuning process of PSO on the EMJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

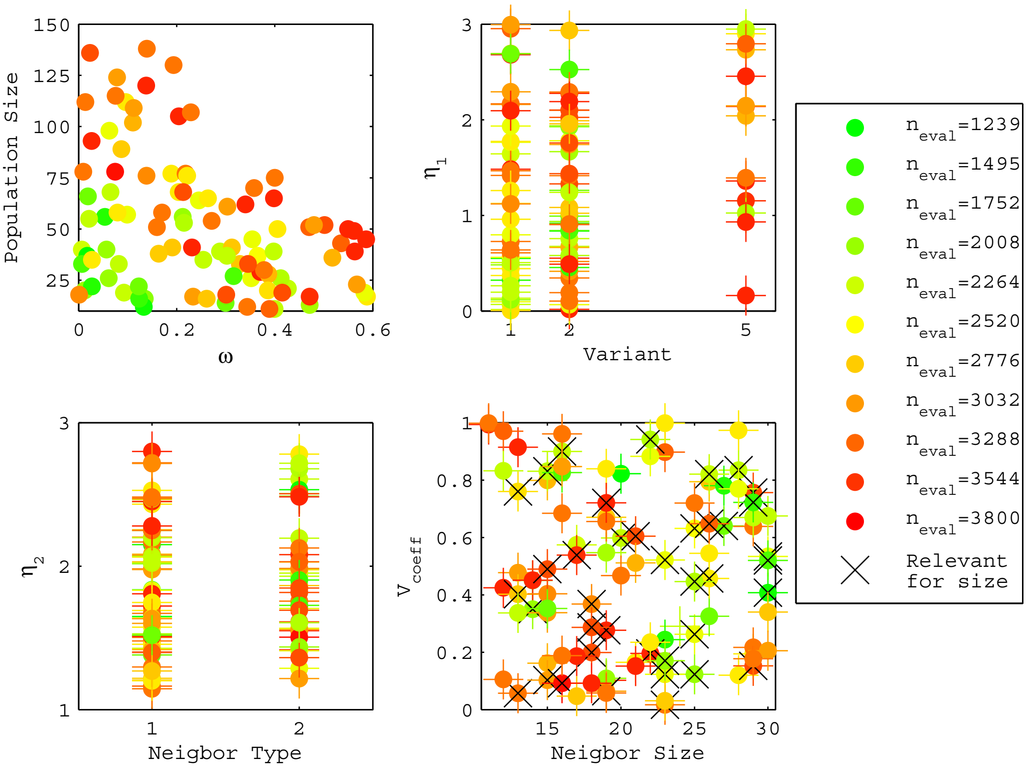


Figure C.8: The results of the second phase tuning process of PSO on the Cassini2 First Leg DSM PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

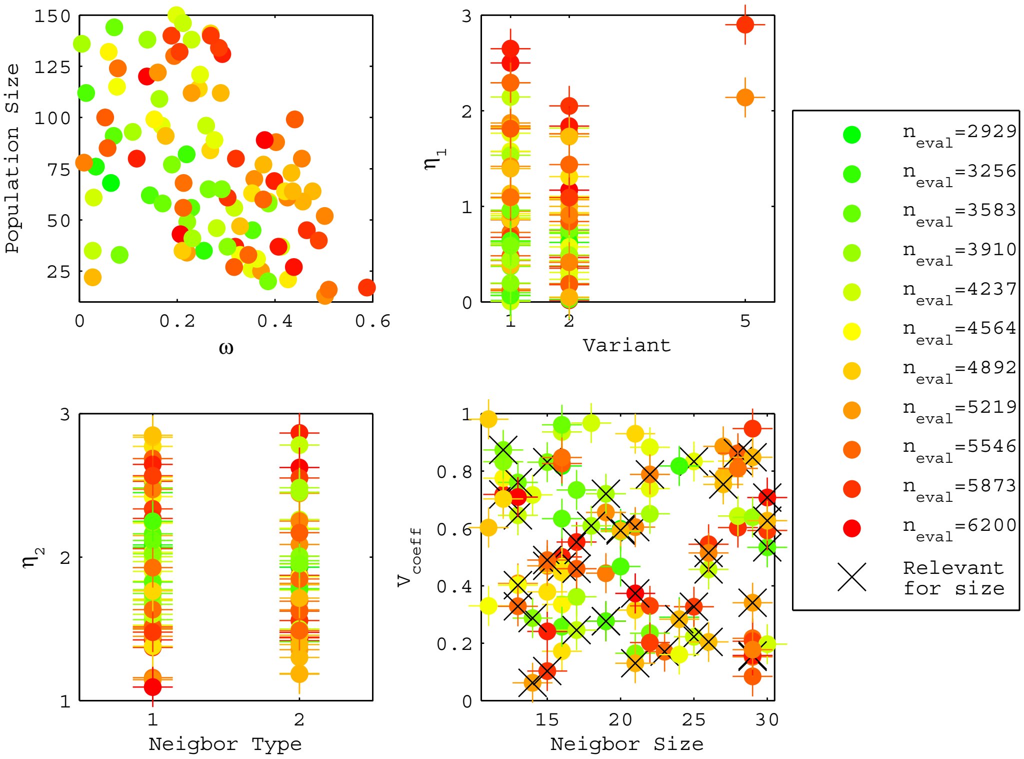


Figure C.9: The results of the second phase tuning process of PSO on the Cassini2 Last Leg DSM PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

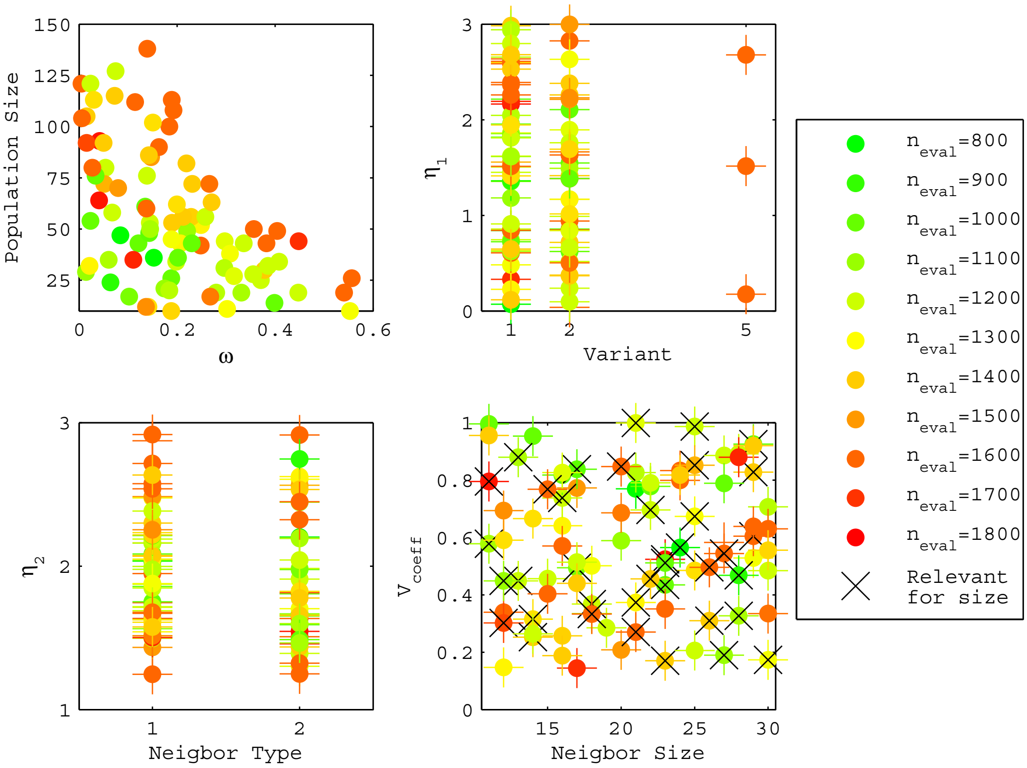


Figure C.10: The results of the second phase tuning process of PSO on the Cassini2 Last Leg DSM VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 100 out of 500 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

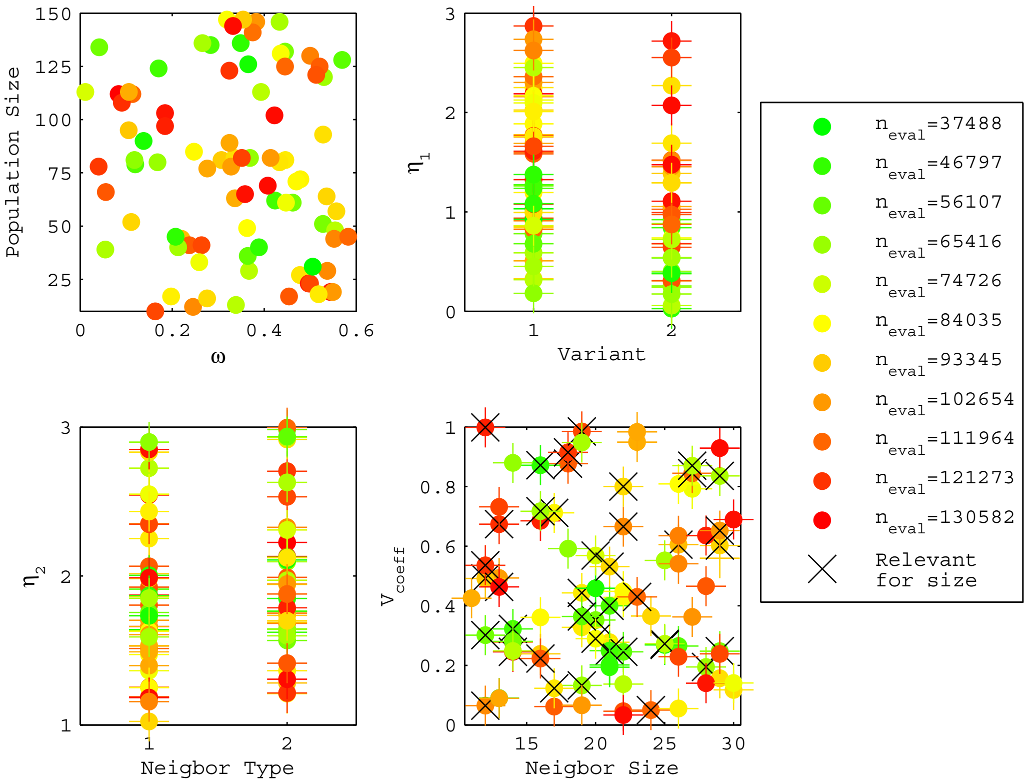


Figure C.11: The results of the third phase tuning process of PSO on the EVEJS MGA problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 80 out of 200 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

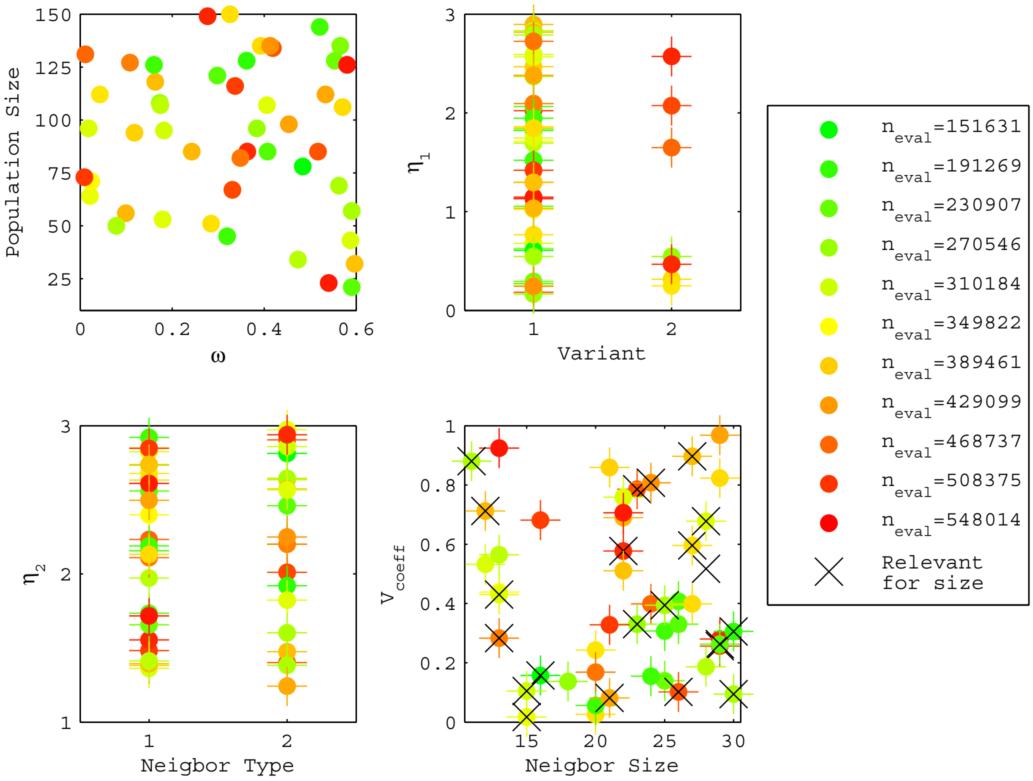


Figure C.12: The results of the third phase tuning process of PSO on the Cassini2 First Leg VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 50 out of 200 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

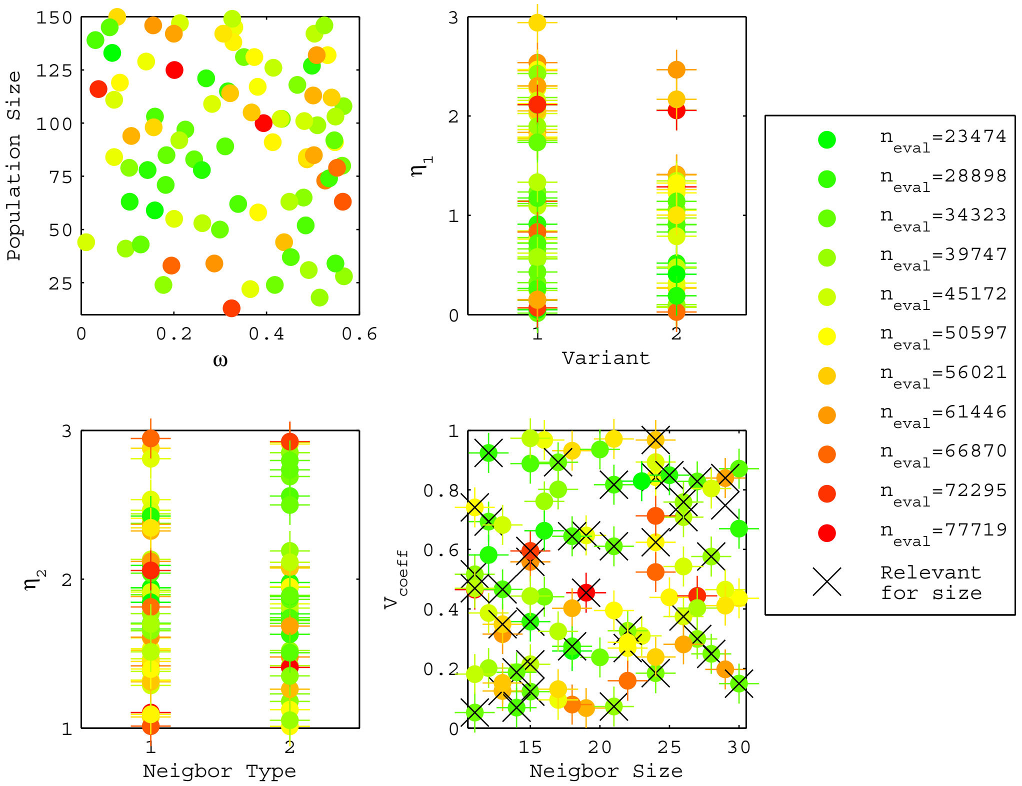


Figure C.13: The results of the third phase tuning process of PSO on the Cassini2 First And Last Leg PF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 80 out of 200 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

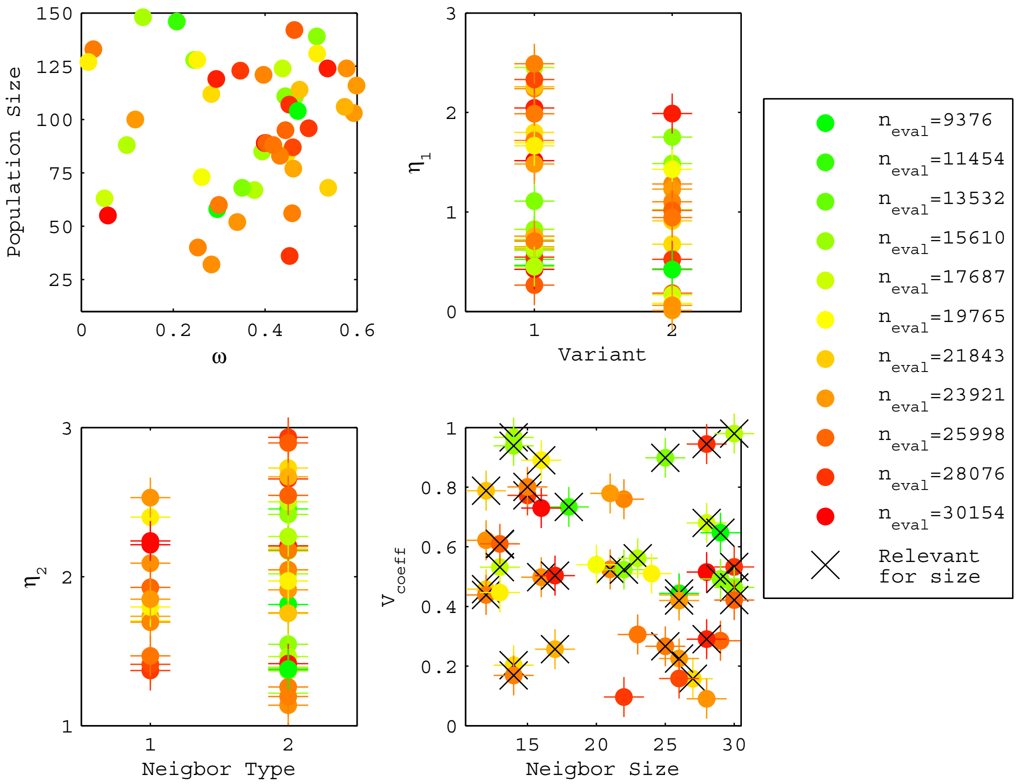


Figure C.14: The results of the third phase tuning process of PSO on the Cassini2 Last Two Legs VF problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 50 out of 200 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

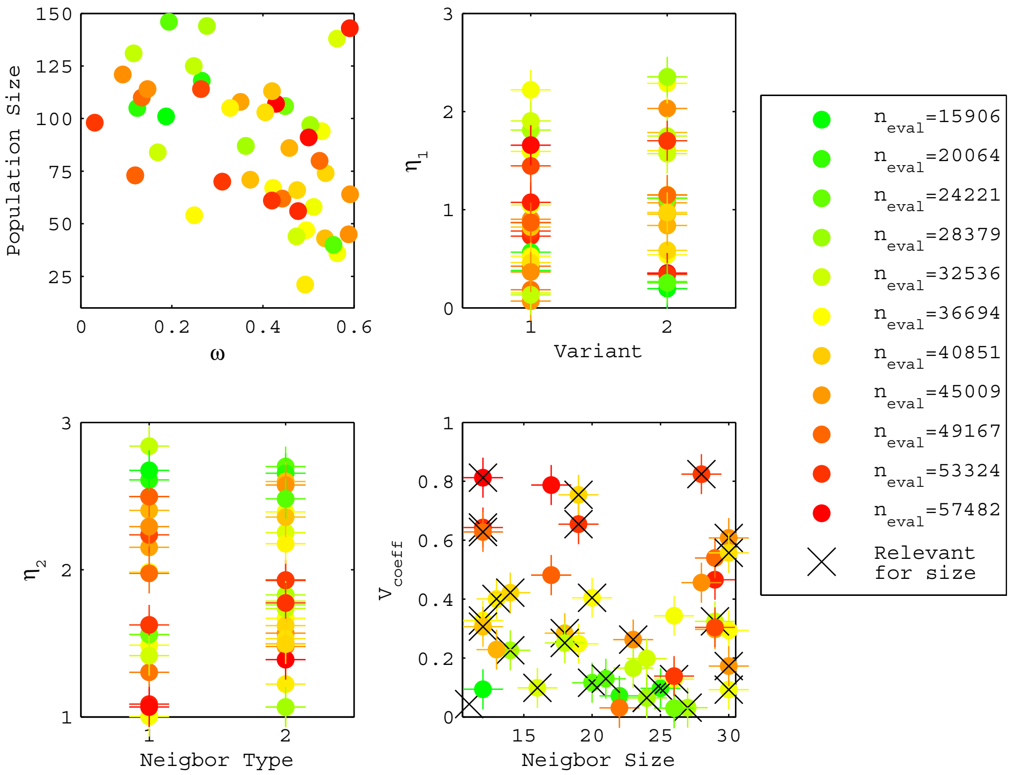


Figure C.15: The results of the third phase tuning process of PSO on the Cassini2 VF Narrowed 0.01 problem with random sampling. The results show the number of function evaluations that are required to obtain a 95% chance on finding a solution within 50 m/s of the best putative minimum. The best 50 out of 200 random points are plotted with a continuous green-red color scale where green corresponds to the best sample, and red to the worst sample among those selected. Note that not all samples of the neighbor size are relevant, as this parameter is only meaningful if the neighbor type is 2. Hence the black cross is present to indicate for which samples this applies. Note that n_{eval} in the legend is identical to $n_{95\%}$.

Appendix D

Detailed Results of Repeated Local Search

This appendix gives an overview of the success rates of the various local optimization algorithms discussed in Chapter 12. Also the required number of evaluations for them is given. Tables D.1 and D.2 give the success rates and the required evaluations for MS scheme without restarts and Tables D.3 and D.4 give the success rates and the required evaluations for MS scheme with restarts.

Table D.1: The success rates for various problems using various local algorithms in an MS scheme without restarting at the best location. All problems were solved 10000 times, except for the Cassini2 First Leg VF problem, which was solved 1000 times. [1]: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	0.2276	0.1235	0.1204	0.0621	0.092	0.1305
EMJS	0.1421	0.1073	0.1027	0.0524	0.077	0.1282
EVEJS	0.0601	0	0	0	0	0.0004
First Leg PF	0.4702	-[1]	-[1]	0.0024	0.0275	-[1]
First Leg VF	0.0407	0.124	0.137	0.124	0	0.125
Last Leg PF	0.2484	0.0067	0.0071	0.1558	0.1003	0.0084
Last Leg VF	0.6674	0.0034	0.0042	0.1452	0.24	0.0032
PF Narrowed 0.01	0.021	-	-	-	-	-
VF Narrowed 0.01	0.022	0.505	0.785	-	-	0.445

Table D.2: The average number of evaluations for various problems using various local algorithms in an MS scheme without restarting at the best location. All problems were solved 10000 times, except for the Cassini2 First Leg VF problem, which was solved 1000 times. [1]: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	257	4759	2766	4923	97.28	4817
EMJS	337	3720	3395	1792	112.6	5366
EVEJS	640	4108	4324	2679	155	6361
First Leg PF	557	-[1]	-[1]	6080	166	-[1]
First Leg VF	1051	116233	119160	116233	201	141141
Last Leg PF	377	7903	7787	8071	180	10281
Last Leg VF	970	8284	10250	4281	155	8131
PF Narrowed 0.01	12705	-	-	-	-	-
VF Narrowed 0.01	42442	181421	262755	-	-	182637

Table D.3: The success rates for various problems using various local algorithms in an MS scheme that restarts at the best location until no significant improvement is found anymore. ^{1}: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	0.3298	0.1308	0.1254	0.0941	0.1361	0.1483
EMJS	0.2133	0.1329	0.1224	0.0749	0.1084	0.165
EVEJS	0.0997	0.0001	-	0.0042	0.0048	0.0029
First Leg PF	0.5693	-	-	0.0767	0.2933	-
First Leg VF	0.1805	0.201	0.21	-	0.0003	0.211
Last Leg PF	0.287	0.15	0.128	0.242	0.1996	0.215
Last Leg VF	0.745	0.0507	0.049	0.1673	0.3355	0.0497
PF Narrowed 0.01	0.081	-	-	0.005	0.007	-
VF Narrowed 0.01	0.04	0.81	0.885	0.95	-	0.765

Table D.4: The required number of evaluations for various problems using various local algorithms in an MS scheme that restarts at the best location until no significant improvement is found anymore. ^{1}: the First Leg PF problem resulted in too unstable behavior for some algorithms.

Problem	SBPLX	PR	FR	COBYLA	BOBYQA	BFGS
EJS	436	8774	4158	5803	161	8058
EMJS	633	7843	6960	3205	242	13893
EVEJS	1343	8547	-	7207	508	14218
First Leg PF	1118	-	-	44199	519	-
First Leg VF	2830	286916	261697	230821	450	335423
Last Leg PF	632	46212	45271	11026	314	50513
Last Leg VF	1544	19639	25118	6708	240	19409
PF Narrowed 0.01	18576	189157	-	16331	2337	-
VF Narrowed 0.01	85409	388909	464036	563220	-	376408

Appendix E

Additional Tables for Cassini2 Analysis

This appendix gives more detailed tables related to the analysis on the Cassini2 problem. Some of them were namely not included therein to save space and make the report better readable. These tables are included here for completeness.

Table E.1 gives an overview of the global basin size of each individual parameter of five different variants of the problem. Table E.2 gives an overview of the sensitivity of each leg for all possible hybrids between PF and VF trajectory models on the Cassini2 trajectory. Subsequently an overview of the required evaluations to optimize these hybrids are given in Table E.3.

Table E.1: The sizes of the global basin in % of the total search space, calculated for the different parameters. The table gives sizes for five different variants of the Cassini2 problem for comparison.

Variable Model	t_0	T_n	η_n	$r_n/V_\infty/r_{p,2}$	$\theta_n/\theta/b_{i,n}$	$\phi_2/\phi/\Delta V_2$	Overall
Leg 1							
MGA	0.097	0.32					$3.1{\cdot}10^{-4}$
DSM 2^{nd} Leg PF	0.073	0.24					$1.8{\cdot}10^{-4}$
DSM 2^{nd} Leg VF	0.075	0.19					$1.4{\cdot}10^{-4}$
DSM PF	0.032	0.098	0.052	0.44	0.52	2.0	$7.2{\cdot}10^{-14}$
DSM VF	0.0077	0.041	100	0.17	0.020	0.20	$2.2{\cdot}10^{-15}$
Leg 2							
MGA		0.00056					0.00056
DSM 2^{nd} Leg PF		0.22	1.6	1.7	0.77	1.3	$5.9{\cdot}10^{-9}$
DSM 2^{nd} Leg VF		0.23	39	2.9	5.5	1.6	$2.3{\cdot}10^{-6}$
DSM PF		0.082	1.7	1.7	0.77	1.3	$2.4{\cdot}10^{-9}$
DSM VF		0.077	17	0.82	1.1	0.57	$6.7{\cdot}10^{-9}$
Leg 3							
MGA		0.29					0.29
DSM 2^{nd} Leg PF		0.20					0.20
DSM 2^{nd} Leg VF		0.20					0.20
DSM PF		0.084	0.052	0.063	0.014	0.25	$9.2{\cdot}10^{-15}$
DSM VF		0.081	100	0.89	1.5	12	$1.3{\cdot}10^{-6}$
Leg 4							
MGA		1.3					1.3
DSM 2^{nd} Leg PF		1.4					1.4
DSM 2^{nd} Leg VF		1.4					1.4
DSM PF		0.44	0.19	0.17	0.14	2.9	$6.2{\cdot}10^{-11}$
DSM VF		0.75	100	1.4	1.7	8.2	$1.4{\cdot}10^{-5}$
Leg 5							
MGA		19					19
DSM 2^{nd} Leg PF		19					19
DSM 2^{nd} Leg VF		19					19
DSM PF		3.6	0.28	0.28	0.093	1.55	$4.1{\cdot}10^{-10}$
DSM VF		11	100	1.5	2.6	12	$5.3{\cdot}10^{-4}$

Table E.2: The sizes of the global basin in % of the total search space, calculated for the different parameters. The table gives sizes for three different variants of the Cassini2 problem for comparison.

Model type per leg					Leg sensitivity					Overall
1	2	3	4	5	Leg 1	Leg 2	Leg 3	Leg 4	Leg 5	Size
PF	PF	PF	PF	PF	$7.2{\cdot}10^{-14}$	$2.4{\cdot}10^{-9}$	$9.2{\cdot}10^{-15}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$4.0{\cdot}10^{-64}$
PF	PF	PF	PF	VF	$6.1{\cdot}10^{-14}$	$2.2{\cdot}10^{-9}$	$8.6{\cdot}10^{-15}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.9{\cdot}10^{-58}$
PF	PF	PF	VF	PF	$4.6{\cdot}10^{-14}$	$1.9{\cdot}10^{-9}$	$5.6{\cdot}10^{-15}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$6.9{\cdot}10^{-59}$
PF	PF	PF	VF	VF	$3.6{\cdot}10^{-14}$	$1.7{\cdot}10^{-9}$	$4.8{\cdot}10^{-15}$	$1.4{\cdot}10^{-5}$	$5.5{\cdot}10^{-4}$	$2.4{\cdot}10^{-53}$
PF	PF	VF	PF	PF	$7.7{\cdot}10^{-14}$	$9.5{\cdot}10^{-11}$	$1.2{\cdot}10^{-6}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$2.1{\cdot}10^{-57}$
PF	PF	VF	PF	VF	$6.4{\cdot}10^{-14}$	$8.4{\cdot}10^{-11}$	$2.2{\cdot}10^{-6}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$2.0{\cdot}10^{-51}$
PF	PF	VF	VF	PF	$4.8{\cdot}10^{-14}$	$5.4{\cdot}10^{-11}$	$1.4{\cdot}10^{-6}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$4.5{\cdot}10^{-52}$
PF	**PF**	**VF**	**VF**	**VF**	$\mathbf{3.9{\cdot}10}^{-14}$	$\mathbf{4.6{\cdot}10}^{-11}$	$\mathbf{1.3{\cdot}10}^{-6}$	$\mathbf{1.4{\cdot}10}^{-5}$	$\mathbf{5.5{\cdot}10}^{-4}$	$\mathbf{1.9{\cdot}10}^{-46}$
PF	VF	PF	PF	PF	$1.2{\cdot}10^{-17}$	$6.3{\cdot}10^{-7}$	$9.2{\cdot}10^{-15}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$1.8{\cdot}10^{-65}$
PF	VF	PF	PF	VF	$1.1{\cdot}10^{-17}$	$6.9{\cdot}10^{-7}$	$8.6{\cdot}10^{-15}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.1{\cdot}10^{-59}$
PF	VF	PF	VF	PF	$1.0{\cdot}10^{-17}$	$4.9{\cdot}10^{-7}$	$5.6{\cdot}10^{-15}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$3.4{\cdot}10^{-60}$
PF	VF	PF	VF	VF	$9.4{\cdot}10^{-18}$	$5.2{\cdot}10^{-7}$	$4.8{\cdot}10^{-15}$	$1.4{\cdot}10^{-5}$	$5.5{\cdot}10^{-4}$	$1.9{\cdot}10^{-54}$
PF	VF	VF	PF	PF	$1.1{\cdot}10^{-19}$	$1.6{\cdot}10^{-8}$	$1.2{\cdot}10^{-6}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$5.2{\cdot}10^{-61}$
PF	VF	VF	PF	VF	$1.1{\cdot}10^{-19}$	$1.4{\cdot}10^{-8}$	$2.2{\cdot}10^{-6}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$5.6{\cdot}10^{-55}$
PF	VF	VF	VF	PF	$7.3{\cdot}10^{-20}$	$8.1{\cdot}10^{-9}$	$1.4{\cdot}10^{-6}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$1.0{\cdot}10^{-55}$
PF	VF	VF	VF	VF	$6.7{\cdot}10^{-20}$	$6.7{\cdot}10^{-9}$	$1.3{\cdot}10^{-6}$	$1.4{\cdot}10^{-5}$	$5.5{\cdot}10^{-4}$	$4.7{\cdot}10^{-50}$
VF	PF	PF	PF	PF	$4.8{\cdot}10^{-11}$	$2.4{\cdot}10^{-9}$	$9.2{\cdot}10^{-15}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$2.6{\cdot}10^{-61}$
VF	PF	PF	PF	VF	$4.5{\cdot}10^{-11}$	$2.2{\cdot}10^{-9}$	$8.6{\cdot}10^{-15}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.4{\cdot}10^{-55}$
VF	PF	PF	VF	PF	$2.9{\cdot}10^{-11}$	$1.9{\cdot}10^{-9}$	$5.6{\cdot}10^{-15}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$3.6{\cdot}10^{-56}$
VF	PF	PF	VF	VF	$2.6{\cdot}10^{-11}$	$1.7{\cdot}10^{-9}$	$4.8{\cdot}10^{-15}$	$1.4{\cdot}10^{-5}$	$5.5{\cdot}10^{-4}$	$1.6{\cdot}10^{-50}$
VF	PF	VF	PF	PF	$5.2{\cdot}10^{-11}$	$9.5{\cdot}10^{-11}$	$1.2{\cdot}10^{-6}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$1.4{\cdot}10^{-54}$
VF	PF	VF	PF	VF	$4.7{\cdot}10^{-11}$	$8.4{\cdot}10^{-11}$	$2.2{\cdot}10^{-6}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.5{\cdot}10^{-48}$
VF	PF	VF	VF	PF	$3.1{\cdot}10^{-11}$	$5.5{\cdot}10^{-11}$	$1.4{\cdot}10^{-6}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$3.0{\cdot}10^{-49}$
VF	**PF**	**VF**	**VF**	**VF**	$\mathbf{2.8{\cdot}10}^{-11}$	$\mathbf{4.6{\cdot}10}^{-11}$	$\mathbf{1.3{\cdot}10}^{-6}$	$\mathbf{1.4{\cdot}10}^{-5}$	$\mathbf{5.5{\cdot}10}^{-4}$	$\mathbf{1.3{\cdot}10}^{-43}$
VF	VF	PF	PF	PF	$1.2{\cdot}10^{-13}$	$6.3{\cdot}10^{-7}$	$9.2{\cdot}10^{-15}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$2.1{\cdot}10^{-61}$
VF	VF	PF	PF	VF	$1.4{\cdot}10^{-13}$	$6.9{\cdot}10^{-7}$	$8.6{\cdot}10^{-15}$	$3.0{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.4{\cdot}10^{-55}$
VF	VF	PF	VF	PF	$1.1{\cdot}10^{-13}$	$4.9{\cdot}10^{-7}$	$5.6{\cdot}10^{-15}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$3.6{\cdot}10^{-56}$
VF	VF	PF	VF	VF	$1.0{\cdot}10^{-13}$	$5.2{\cdot}10^{-7}$	$4.8{\cdot}10^{-15}$	$1.4{\cdot}10^{-5}$	$5.5{\cdot}10^{-4}$	$2.1{\cdot}10^{-50}$
VF	VF	VF	PF	PF	$3.4{\cdot}10^{-15}$	$1.6{\cdot}10^{-8}$	$1.2{\cdot}10^{-6}$	$6.2{\cdot}10^{-11}$	$4.1{\cdot}10^{-10}$	$1.6{\cdot}10^{-56}$
VF	VF	VF	PF	VF	$3.6{\cdot}10^{-15}$	$1.4{\cdot}10^{-8}$	$2.2{\cdot}10^{-6}$	$2.9{\cdot}10^{-11}$	$5.5{\cdot}10^{-4}$	$1.9{\cdot}10^{-50}$
VF	VF	VF	VF	PF	$2.2{\cdot}10^{-15}$	$8.1{\cdot}10^{-9}$	$1.4{\cdot}10^{-6}$	$2.9{\cdot}10^{-5}$	$4.1{\cdot}10^{-10}$	$3.1{\cdot}10^{-51}$
VF	**VF**	**VF**	**VF**	**VF**	$\mathbf{2.2{\cdot}10}^{-15}$	$\mathbf{6.7{\cdot}10}^{-9}$	$\mathbf{1.3{\cdot}10}^{-6}$	$\mathbf{1.4{\cdot}10}^{-5}$	$\mathbf{5.5{\cdot}10}^{-4}$	$\mathbf{1.5{\cdot}10}^{-45}$

Table E.3: $n_{95\%}$ for various hybrid of the PF and VF trajectory models for the Cassini 2 problem. Note that all results above 100 are simply listed as $>$ 100. Furthermore an asterisk indicates the solution has been found only once or twice. If no solution has been found at all, it is listed as '-'. See Chapter 16 for explanation of the procedure used.

Model type per leg					$n_{95\%}$
1	2	3	4	5	[-]
PF	PF	PF	PF	PF	-
PF	PF	PF	PF	VF	-
PF	PF	PF	VF	PF	-
PF	PF	PF	VF	VF	-
PF	PF	VF	PF	PF	64*
PF	PF	VF	PF	VF	$>$ 100*
PF	PF	VF	VF	PF	30
PF	PF	VF	VF	VF	24
PF	VF	PF	PF	PF	$>$ 100*
PF	VF	PF	PF	VF	-
PF	VF	PF	VF	PF	$>$ 100*
PF	VF	PF	VF	VF	-
PF	VF	VF	PF	PF	35
PF	VF	VF	PF	VF	59
PF	VF	VF	VF	PF	30
PF	VF	VF	VF	VF	9
VF	PF	PF	PF	PF	-
VF	PF	PF	PF	VF	-
VF	PF	PF	VF	PF	-
VF	PF	PF	VF	VF	$>$ 100*
VF	PF	VF	PF	PF	$>$ 100
VF	PF	VF	PF	VF	$>$ 100*
VF	PF	VF	VF	PF	26
VF	PF	VF	VF	VF	13
VF	VF	PF	PF	PF	-
VF	VF	PF	PF	VF	$>$ 100*
VF	VF	PF	VF	PF	$>$ 100*
VF	VF	PF	VF	VF	$>$ 100*
VF	VF	VF	PF	PF	34
VF	VF	VF	PF	VF	29
VF	VF	VF	VF	PF	13
VF	VF	VF	VF	VF	6

Bibliography

Abdelkhalik, O. and Gad, A. (2012). Dynamic-Size Multiple Populations Genetic Algorithm for Multigravity-Assist Trajectories Optimization. *Journal of Guidance, Control and Dynamics*, 35(2):629–641.

Abdelkhalik, O. and Mortari, D. (2011). N-Impulse Orbit Transfer Using Genetic Algorithms. *Journal of Spacecraft and Rockets*, 44(2):456–459.

Addis, B., Cassioli, A., Locatelli, M., and Schoen, F. (2008). Global Optimization for the Design of Space Trajectories. *Optimization Online.*

Addis, B., Cassioli, A., Locatelli, M., and Schoen, F. (2011). Global Optimization for the Design of Space Trajectories. *Computational Optimization and Applications*, 48(3):635–652.

Addis, B., Locatelli, M., and Schoen, F. (2005). Local Optima Smoothing for Global Optimization. *Optimization Methods and Software*, 20(4-5):417–437.

Bate, R. R., Mueller, D., and White, J. E. (1971). *Fundamentals of Astrodynamics.* Dover Publications Inc., 31 East 2nd Street, Mineola New York.

Battin, R. H. and Vaughan, R. M. (1983). An Elegant Lambert Algorithm. *Journal of Guidance*, 7(6):662–670.

Becarra, V., Nasuto, S., Anderson, J., Ceriotti, M., and Bombardelli, C. (2007). Search Space Pruning and Global Optimization of Multiple Gravity Assist Trajectories with Deep Space Maneuvers. *2007 IEEE Congres on Evolutionary Computation, CEC 2007*, pages 957–964.

Biscani, F., Izzo, D., and Chit, H. Y. (2010). A Global Optimisation Toolbox for Massively Parallel Engineering Optimisation. Presented at *ICATT 2010: International Conference on Astrodynamics Tools and Techniques*, Advanced Concept Team, ESA.

Boggs, P. T. and Tolle, J. W. (1995). Sequential Quadratic Programming. *Acta Numerica*, 4:1–51.

Bosworth, J., Foo, N. Y., and Ziegler, B. P. (1972). Comparison of Genetic Algorithms with Conjugate Gradient Methods. NASA Contractor Report CR-2093, University of Michigan and NASA.

Burkardt, T. M. and Danby, J. M. A. (1987). The Solution to Kepler's Equation II. *Celestial Mechanics*, 40:317–328.

Cassioli, A., Di Lorenzo, D., Locatelli, M., Schoen, F., and Sciandrone, M. (2009). Machine Learning For Global Optimization. *Optimization Online.*

Ceriotti, M. and Vasile, M. (2009). MGA Trajectory Planning with an ACO-Inspired Algorithm. *60th International Astronautical Congress 2009 (IAC 2009)*, 6:4357–4370.

Ceriotti, M. and Vasile, M. (2010). Automated Multigravity Assist Trajectory Planning with a Modified Ant Colony Algorithm. *Journal of Aerospace Computing, Information and Communication*, 7:261–293.

Ceriotti, M., Vasile, M., and Bombardelli, C. (2007). An Incremental Algorithm For Fast Optimisation Of Multiple Gravity Assist Trajectories. *58th International Astronautical Congress 2007 (IAC 2007)*, 7:4211–4222.

Charles, E. D. and Tatum, J. B. (1997). The Convergence of Newton-Raphson Iteration With Kepler's Equation. *Celestial Mechanics and Dynamical Astronomy*, 69(4).

Clerc, M. and Kennedy, J. (2002). The Particle Swarm - Explosion, Stability, and Convergence in a Multidimensional Complex Space. *IEEE Transactions on Evolutionary Computation*, 6(1).

Conway, B. A. (2010). *Spacecraft Trajectory Optimization.* Cambridge University Press, 32 Avenue of the Americas, New York, USA.

Cornelisse, J. W., Schöyer, H. F. R., and Wakker, K. F. (1979). *Rocket Propulsion and Spaceflight Dynamics.* Pitman Publishing Limited, 39 Parker Street, London WC2B 5PB.

Danby, J. M. A. (1987). The Solution to Kepler's Equation III. *Celestial Mechanics*, 40:303–312.

Danby, J. M. A. and Burkardt, T. M. (1983). The Solution to Kepler's Equation I. *Celestial Mechanics*, 31:95–107.

Di Lizia, P., Radice, G., and Izzo, D. (2004). Advanced Global Optimisation Tools for Mission Analysis and Design. Final Report Ariadna Study, 03/4101b, University of Glasgow and Advanced Concept Team.

Eberhart, R. and Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory. *Proceedings of the Sixth Internatinal Symposium on Micro Machine and Human Science (MHS'95)*, pages 39–43.

Eberhart, R. and Yuhui, S. (2001). Particle Swarm Optimization: Developments, Applications and Resources. *Proceedings of the 2001 Congres on Evolutionary Computation*, 1:81–86.

ESA-ACT (2012). *PaGMO Project Homepage*, ESA Advanced Concept Team, Last Accessed on 2012-03-08. `http://pagmo.sourceforge.net/`.

Feoktistov, V. (2006). *Differential Evolution In Search of Solutions.* Springer Science + Business Media, LLC, New York, USA.

Gad, A. and Abdelkhalik, O. (2011). Hidden Genes Genetic Algorithm for Multi-Gravity-Assist Trajectories Optimization. *Journal of Spacecraft and Rockets*, 48(4):520–529.

Gill, P. E., Murray, W., and Saunders, M. A. (2002). SNOPT: An SQP Algorithm for Large-Scale Constrained Optimization. *SIAM Journal on Optimization*, 12(4):979–1006.

GNU (2012). *GSL - GNU Scientific Library*, Last Accessed on 2012-03-06,. `http://www.gnu.org/software/gsl/`.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Gooding, R. H. (1990). A Procedure for the Solution of Lambert's Orbital Boundary-Value Problem. *Celestial Mechanics and Dynamical Astronomy*, 48:145–165.

Ingber, L. (1993). Simulated Annealing: Practice versus Theory. *Mathematical and Computer Modelling*, 18(11):29–57.

Ingber, L. (1996). Adaptive Simulated Annealing: Lessons Learned. *Control and Cybernetics*, 25(1):33–54.

Iorfida, E. (2011). Preliminary Design of Interplanetary Trajectories with Deep Space Manoeuvres. MSc Thesis, Universita di Pisa, Pisa, Italy.

Izzo, D. (2012). PyGMO and PyKEP: Open Source Tools for Massively Parallel Optimization in Astrodynamics (The Case of Interplanetary Trajectory Optimization. *In Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT 2012, ESTEC/ESA, Noordwijk, The Netherlands.*

Izzo, D., Becerra, V. M., Myatt, D. R., Nasuto, S. J., and M., B. J. (2007). Search Space Pruning and Global Optimisation of Multiple Gravity Assist Spacecraft Trajectories. *Journal of Global Optimization*, 38:283–296.

Izzo, D., Rucinski, M., and Ampatzis, C. (2009). Parallel Global Optimisation Meta-Heuristics Using an Asynchronous Island-Model. *Congres on Evolutionary Computation 2009 (CEC 2009)*, pages 2301–2308.

Izzo, D. and Vinko, T. (2012). *ACT - Informatics - GTOP Database*, ESA Advanced Concept Team, Last Accessed on 2012-09-24. `http://www.esa.int/gsp/ACT/inf/op/globopt.htm`.

Jezewski, D. J. (1975). Primer Vector Theory and Applications. National Aeronautics Space Administration, Technical Report, NASA-TR-R-454, JSC-S-439.

Johnson, S. G. (2012). The NLopt Nonlinear Optimization Package, available online at `http://ab-initio.mit.edu/wiki/index.php/NLopt`, last accessed 2012-03-06.

Johnson, W. R. and Longuski, J. M. (2002). Design of Aerogravity-Assist Trajectories. *Journal of Spacecraft and Rockets*, 39(1):23–30.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of the IEEE International Conference on Neural Networks*, 4:1942–1948.

Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by Simulated Annealing. *Science*, 220(4598):671–680.

Kumar, K., Abdulkadir, Y., van Barneveld, P. W. L., Belien, F., Billemont, S., Brandon, E., Dijkstra, M., Dirkx, D., Engelen, F., Gondelach, D., van der Ham, L., Heeren, E., Iorfida, E., Leloux, J., Melman, J., Mooij, E., Musegaas, P., Noomen, R., Persson, S. M., Romgens, B., Ronse, A., Leite Pinto Secretin, T. A., Tong Minh, B., and Vandamme, J. (2012). Tudat: a Modular and Robust Astrodynamics Toolbox. *In Proceedings of the Fifth International Conference on Astrodynamics Tools and Techniques, ICATT 2012, ESTEC/ESA, Noordwijk, The Netherlands.*

Kumar, K., Abdulkadir, Y., van Barneveld, P. W. L., Dirkx, D., Engelen, F., Iorfida, E., Leloux, J., Melman, J., Mooij, E., Noomen, R., and Romgens, B. (2011). Design of a Modular and Robust Astrodynamics Toolbox. *New Trends in Astrodynamics and Applications VI, Courant Institute of Mathematical Sciences, New York University.*

Lagarias, J. C., Reeds, J. A., Wright, M. H., and Wright, P. E. (1998). Convergence Properties of the Nelder-Mead Simplex Method in Low Dimensions. *SIAM Journal on Optimization*, 9:112–147.

Lawden, D. F. (1963). *Optimal Trajectories for Space Navigation.* Butterworth & Co. (Publishers) Ltd., 88 Kingsway, W.C.2 London.

Leary, R. H. (2000). Global Optimization On Funneling Landscapes. *Journal of Global Optimization*, 18(4):367–383.

Leito Pinto Secretin, T. (2012). Design of a Combinatorial Tool for Preliminary Space Mission Analysis, MSc Thesis, Delft University of Technology, Delft, The Netherlands.

Locatelli, M. (2005). On the Multilevel Structure of Global Optimization Problems. *Computational Optimization and Applications*, 30(1):5–22.

Maischberger, M. (2011). COIN-OR METSlib a Metaheuristic Framework in Modern C++, available online at `http://www.coin-or.org/metslib/docs/stable/0.5/metslib-tr.pdf`, last accessed 2012-03-06, Università degli Studi di Firenze.

Martin, W., Lienig, J., and Cohoon, J. (1997). *Island (Migration) Models.* Institute of Physical Publishing, Bristol, UK *(ch. C6.3:1-C6.3:16).*

McAdams, J. V., Dunham, D. W., Farquhar, R. W., Taylor, A. H., and Williams, B. G. (2006). Trajectory Design and Maneuver Strategy for the MESSENGER Mission to Mercury. *Journal of Spacecraft and Rockets*, 43(5):1054–1064.

Melman, J. (2007). Trajectory Optimization for a Mission to Neptune and Triton. MSc Thesis, Delft University of Technology, Delft, The Netherlands.

Mendez, R., Kennedy, J., and Neves, J. (2004). The Fully Informed Particle Swarm: Simpler, Maybe Better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210.

Meza, J. C. (1994). OPT++: An Object-Oriented Class Library for Nonlinear Optimization, Technical Report, SAND94-8225, Sandia National Laboratories, Livermore, California 94551, USA,.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 3rd edition.

Molenaar, S. (2009). Optimization of Interplanetary Trajectories with Deep Space Maneuvers - Model Development and Application to a Uranus Orbiter Mission. MSc Thesis, Delft University of Technology, Delft, The Netherlands.

Montenbruck, O. and Gill, E. (2000). *Satellite Orbits*. Springer, Berlin, Heidelberg, New York, 3rd edition.

Murray, W. (1997). Sequential Quadratic Programming Methods for Large-Scale Problems. *Computational Optimization and Applications*, 7(1):127–140.

Musegaas, P. (2012). Optimization of Space Trajectories Including Multiple Gravity Assists and Deep Space Maneuvers. MSc Literature Study Report, Delft University of Technology, Delft, The Netherlands.

Myatt, D. R., Becerra, V. M., Nasuto, S. J., Bishop, J. M., and Izzo, D. (2004). Advanced Global Optimisation for Mission Analysis and Design. Final Report Ariadna Study, 03/4101a, University of Reading and Advanced Concept Team, ESA.

NASA (2012). *ODTBX*, NASA Goddard Space Flight Center's Orbit Determination Toolbox, Last Accessed on 2012-11-05. http://opensource.gsfc.nasa.gov/projects/ODTBX/.

Navagh, J. (1993). Optimizing Interplanetary Trajectories With Deep Space Maneuvers. Technical report, National Aeronautics Space Administration, Contractor Report 4546.

Nelder, J. A. and Mead, R. (1965). A Simplex Method for Function Minimization. *The Computer Journal*, 7(4):308–313.

Oldenhuis, R. P. S. (2010). Trajectory Optimization for a Mission to the Solar Bow Shock and Minor Planets. MSc Thesis, Delft University of Technology, Delft, The Netherlands.

Olds, A. D., Kluever, C. A., and Cupples, M. L. (2007). Interplanetary Mission Design Using Differential Evolution. *Journal of Spacecraft and Rockets*, 44(5):1060–1070.

Olympio, J. T. (2009). Designing Optimal Multi-Gravity-Assist Trajectories with Free Number of Impulses. *21st International Symposium on Space Flight Dynamics, 2009, available online at http://www.esa.int/gsp/ACT/doc/MAD/pub/ACT-RPR-MAD-2009-DesignMGADSM.pdf*.

Olympio, J. T. and Marmorat, J.-P. (2007). Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Maneuvers? Final Presentation Ariadna Study, 06/4101a, Ecole des Mines de Paris.

Olympio, J. T., Marmorat, J.-P., and Izzo, D. (2007). Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Maneuvers? Final Report Ariadna Study, 06/4101a, Ecole des Mines de Paris and ESA Advanced Concept Team.

Peralta, F. and Flanagan, S. (1995). Cassini Interplanetary Trajectory Design. *Control Engineering Practice*, 3(11):1603–1610.

Petropoulos, A. E., Longuski, J. M., and Bonfiglio, E. P. (2000). Trajectories to Jupiter via Gravity Assists from Venus, Earth, and Mars. *Journal of Spacecraft and Rockets*, 44(5):1060–1070.

Powell, M. J. D. (2009). The BOBYQA Algorithm for Bound Constrained Optimization without Derivatives. Department of Applied Mathematics and Theoretical Physics, Cambridge England. Technical Report NA2009/06.,.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1992). *Numerical Recipes in C.* Cambridge University Press, Cambridge, United Kingdom, 2nd edition.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes: The Art of Scientific Computing.* Cambridge University Press, Cambridge, United Kingdom, 3rd edition.

Price, K. V., Storn, R. M., and Lampinen, J. A. (2005). *Differential Evolution: A Practical Approach to Global Optimisation.* Springer Verlag, Heidelberg, Germany.

Schlijper, R. M. H. (2003). Genetic Algorithm Optimization of Planetary Gravity-Assist Missions to Pluto. MSc Thesis, Delft University of Technology, Delft, The Netherlands.

Schlüter, M., Gerdts, M., and Rückmann, J. (2011). MIDACO: New Global Optimization Software for MINLP. *MIDACO - Solver, available online at http://midaco-solver.com/download_files/MIDACO_Paper.pdf.*

Standish (2012). Keplerian Elements for Approximate Positions of the Major Planets, available online at http://ssd.jpl.nasa.gov/txt/aprx_pos_planets.pdf, last accessed 2012-08-20, Solar System Dynamics Group, JPL/Caltech.

Storn, R. and Price, K. (1997). Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization*, 11(4):341–359.

Stracquadanio, G., La Ferla, A., De Felice, M., and Nicosia, G. (2011). Design of Robust Space Trajectories. *Research and Development in Intelligent Systems XXVIII: Proceedings of the 31st SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 341–354.

Sun, W. and Yuan, Y. X. (2006). *Optimization Theory and Methods - Nonlinear Programming.* Springer.

Taff, L. G. and Brennan, T. A. (1989). On Solving Kepler's Equation. *Celestial Mechanics and Dynamical Astronomy*, 46(2).

Tassoulis, D., Pavlidis, N., Plagianakos, V., and Vrahatis, M. (2004). Parallel Differential Evolution. *Proceedings of the 2004 Congress on Evolutionary Computation (CEC2004)*, 2:2023–2029.

Tech-X (2012). OptSolve++ User's Guide Version 2.0, available online at http://www.txcorp.com/pdf/OptSolve++/documentation/OptSolve++_UserReference.pdf, last accessed 2012-03-06, tech-x corporation, 5621 arapahoe avenue suite a, boulder co 80303, usa.

Tudat (2012). *TU Delft Astrodynamics Toolbox*, Last Accessed on 2012-11-15. http://tudat.tudelft.nl.

Vasile, M., Ceriotti, M., Radice, G., Becerra, V., Nasuto, S., Anderson, J., and Bombardelli, C. (2007). Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Maneuvers? Final Report Ariadna Study, 06/4101c, University of Glasgow/Reading and ESA Advanced Concept Team.

Vasile, M. and de Pascale, P. (2006). Preliminary Design of Multiple Gravity-Assist Trajectories. *Journal of Spacecraft and Rockets*, 43(4):794–805.

Vasile, M., Minisci, E., and Locatelli, M. (2008). On Testing Global Optimization Algorithms for Space Trajectory Design. *AIAA/AAS Astrodynamics Specialist Conference and Exhibit*, (2008-6277).

Vasile, M., Minisci, E., and Locatelli, M. (2009). A Dynamical System Perspective on Evolutionary Heuristics Applied to Space Trajectory Optimization Problems. *Proceedings on IEEE Congress on Evolutionary Computation, CEC 2009*, pages 2340–2347.

Vasile, M., Minisci, E., and Locatelli, M. (2010). Analysis of Some Global Optimization Algorithms for Space Trajectory Design. *Journal of Spacecraft and Rockets*, 47(2):334–344.

Vasile, M., Minisci, E., and Locatelli, M. (2011). An Inflationary Differential Evolution Algorithm for Space Trajectory Optimization. *IEEE Transactions on Evolutionary Computation*, 15(2):267–281.

Vinko, T. and Izzo, D. (2008). Global Optimisation Heuristics and Test Problems for Preliminary Spacecraft Trajectory Design. Technical report, Advanced Concept Team, ESA.

Vinko, T., Izzo, D., and Bombardelli, C. (2007a). Benchmarking Different Global Optimisation Techniques for Preliminary Space Trajectory Design. *International Astronautical Federation*, 6:4181–4190.

Vinko, T., Izzo, D., and Pinna, F. (2007b). Learning the Best Combination of Solvers in a Distributed Global Optimisation Environment. Technical report, Advanced Concept Team, ESA.

Wächter, A. and Biegler, L. T. (2006). On the Implementation of a Primal-Dual Interior Point Filter Line Search Algorithm for Large-Scale Nonlinear Programming. *Mathematical Programming*, 106(1):25–57.

Wakker, K. F. (2007). *Astrodynamics I + II.* Lecture Notes AE4-874, Delft University of Technology, Delft, Netherlands.

Wales, D. J. and Doye, J. P. K. (1997). Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *Journal of Physical Chemistry A*, 101(28):5111–5116.

Wikipedia (2012). *MESSENGER*, Last Accessed on 2012-01-23,. http://en.wikipedia.org/wiki/MESSENGER.

Zazzera, F. B., Lavagna, M., Armellin, R., Di Lizia, P., Topputo, F., Bertz, M., and Vinko, T. (2007). Global Trajectory Optimisation: Can We Prune the Solution Space When Considering Deep Space Maneuvers? Final Report Ariadna Study, 06/4101b, Politecnico di Milano and ESA Advanced Concept Team.

Zhiqing, L., Dai, G., Zhan, W., and Peng, L. (2011). An Infaltionary PSwarm Algorithm for Space Trajectory Design. *International Journal of Advancements in Computing Technology*, 3(6):152–159.